Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Kalibrierung eines agentenbasierten Influenzamodells

Ausgeführt am Institut für
Analysis and Scientific Computing
der Technischen Universität Wien

unter Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker

durch
Claire Rippinger, BSc
Obermüllnerstraße 15/18
1020 Wien

Datum

Unterschrift

Inhaltsverzeichnis

1 Einleitung 1

2 Theoretische Grundlagen 3
2.1 Modellgrundlagen . . . 3
2.1.1 SIR-Modell . . . 3
2.1.2 Agentenbasierte Modellierung . . . 4
2.2 Kalibrierung . . . 5
2.3 Allgemeine Probleme bei Kalibrierung von ABM . . . 6

3 Kalibrierungsalgorithmen 9
3.1 Optimierungsalgorithmen . . . 9
3.2 Simulated Annealing . . . 10
3.3 Evolutionäre Algorithmen . . . 14

4 Anwendung auf Beispielmodelle 21
4.1 Erklärung der Beispielmodelle . . . 21
4.2 Allgemeine Vorbereitungen zur Kalibrierung . . . 22
4.2.1 Zusammenspiel Modell - Kalibrierungsalgorithmen . . . 22
4.2.2 Konfiguration der Kalibrierungsalgorithmen . . . 24
4.3 Kalibrierung des SIR-Modells . . . 26
4.3.1 Vorbereitungen . . . 26
4.3.2 Kalibrierung . . . 27
4.4 Kalibrierung des SIRS-Modells . . . 39
4.4.1 Vorbereitungen . . . 39
4.4.2 Kalibrierung . . . 40

5 Verringerung der Laufzeit 47
5.1 Vorgehensweise . . . 47
5.2 SIR-Modell . . . 50
5.3 SIR-Modell mit 3 Parametern . . . 53
5.4 SIRS-Modell . . . 54

6 Anwendung auf Influenzamodell 57
6.1 Erklärung des Modells . . . 57
6.2 Kalibrierung . . . 59
6.2.1 Vorbereitungen und bisherige Ansätze . . . 59
6.2.2 Kalibrierung mithilfe Kalibrierungsalgorithmen . . . 62

7 Zusammenfassung und Ausblick 65

Literaturverzeichnis 67

Kurzfassung

Agentenbasierte Modellierung wird in der Modellbildung und Simulation immer öfter verwendet. Demzufolge ist es wichtig, die üblichen Methoden zur Qualitätsüberprüfung eines Modells auf ihre Anwendbarkeit auf agentenbasierte Modelle zu überprüfen. Diese Arbeit beschäftigt sich mit der Kalibrierung eines agentenbasierten Modells. Dabei wird zuerst auf allgemeine Probleme eingegangen, die bei dieser Aufgabestellung auftreten. Es werden zwei Kalibrierungsalgorithmen vorgestellt, die zur Kalibrierung herangezogen werden können: das Simulated Annealing und ein evolutionärer Algorithmus. Bei beiden werden verschiedene Varianten und Konfigurationen besprochen und, soweit dies bekannt ist, auf das Konvergenzverhalten eingegangen. Die Leistung der beiden Algorithmen wird an kleinen Beispielmodellen getestet. Es handelt sich dabei um ein SIR- und um ein $SIRS$-Modell, die den Verlauf einer infektiösen Krankheit simulieren. Bei diesen Tests wird versucht eine geeignete Konfiguration der Kalibrierungsalgorithmen zu bestimmen. Es zeigt sich, dass die beiden Kalibrierungsalgorithmen nur bei einem der beiden Beispielmodelle vergleichbar gute Resultate liefern. Bei dem zweiten Modell werden nur beim evolutionären Algorithmus gute Ergebnisse erzielt. Ein Problem, das bei allen Kalibrierungsdurchläufen auftaucht, stellt die lange Rechenzeit dar. Deswegen wird eine mögliche Methode zur Reduzierung der Laufzeit vorgestellt. Sie besteht darin, die Anzahl der in der Simulation benutzten Agenten während der Kalibrierung zu verändern. Anfangs wird nur eine geringe Agentenanzahl benutzt um in kurzer Zeit mehrere Simulationen durchführen zu können. Die Simulationen mit einer geringen Agentenanzahl weisen jedoch eine größere Varianz auf, sodass die Resultate mit Unsicherheiten behaftet sind. Während der Kalibrierung wird die Agentenzahl progressiv auf ein gewünschtes Maß erhöht, um die Unsicherheit der Resultate zu verringern. Verschiedene Konfigurationen dieser Vorgehensweise werden ebenfalls an den Beispielmodellen getestet und ausgewertet. Auch bei dieser Methode ist der evolutionäre Algorithmus erfolgreicher und es ist möglich die benötigte Laufzeit um etwa die Hälfte zu reduzieren. Die erhaltenen Erkenntnisse werden schlussendlich benutzt, um ein komplexeres agentenbasiertes Modell zu kalibrieren. Hierbei handelt es sich um ein Influenzamodell, das die Ausbreitung der Grippe innerhalb der österreichischen Bevölkerung simuliert. Es wurden mehrere erfolgreiche Kalibrierungen mittels evolutionärem Algorithmus durchgeführt. Hierbei konnte zudem festgestellt werden, dass die berechneten Parameterwerte sehr unterschiedlich sind und demnach mehrere gleichwertige, lokale Minima existieren.

Abstract

Agent-based modelling has experienced increasing application in several fields since it offers many benefits over other modelling methods. Therefore it is important to verifiy whether quality assessment methods can be applied on agent-based models. This master's thesis discusses the problem of calibrating such models. Two algorithms which can be used to perform this task are presented: simulated annealing and an evolutionary algorithm. For each algorithm, different versions and configurations, as well as the convergence behaviour, are discussed. The performance of these algorithms is tested on different example models: an SIR and an SIRS model. These models are used to describe the spreading of an infectious disease. It can be observed that simulated annealing produces good results only on one of the example models, while the evolutionary algorithm performs successfull calibrations for all of them. Unfortunately, all calibrations require a lot of computing time. A possible solution to this problem is presented. It consists of varying the number of agents used in the simulation during the calibration. At the beginning, only a small amout of agents is used in order to perform many simulation runs in a short period of time. However, if a low agent count is used, the simulations exhibit a greater variability. During the course of the calibration, the agent count is progressively increased until it reaches the targeted amount. This way, the variability of the results occuring when a low agent count is used, is being reduced. Different configurations of this method are beeing tested on the example model. It shows that the evolutionary algorithm provides a better performance than simulated annealing and it is possible to cut the computing time in half. Ultimately, the findings are used to calibrate a more complex agent-based model simulating an influenza epidemic in the Austrian population. Several successful calibrations have been performed using the evolutionary algorithm. The calculated parameter sets turned out to be very diverse. Therefore it is plausible that several comparable local minima exist.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Zuerst gebührt mein Dank Herrn Prof. Dr. Felix Breitenecker, der meine Masterarbeit betreut und begutachtet hat.
Ebenfalls bedanken möchte ich mich bei den Mitarbeitern der DWH Simulation Services. Ihre Arbeit und Unterstützung hat meine Masterarbeit erst möglich gemacht. Besonders danke ich dabei Florian Miksch für die hilfreichen Anregungen und konstruktive Kritik.

Meinen Studienkollegen Andi, Irina, Stefan, Basti, Flo, Janne und Julian danke ich besonders für die Unterstützung und gute Gesellschaft während meines gesamten Mathematikstudiums. Ihre Freundschaft hat die langen Rechenabende um einiges angenehmer gemacht und immer die nötige Motivation geliefert.

Des Weiteren bedanke ich mich bei den beiden Verenas. Ich zähle mich sehr glücklich, sie nicht nur als Mitbewohnerinnen, sondern auch als Freundinnen zu haben.

Abschließend möchte ich mich bei meinen Eltern bedanken, die mir bei all meinen studienbezogenen Entscheidungen Freiraum gelassen haben und mich unterstützt haben.

Claire Rippinger

1 Einleitung

Modellbildung und Simulation bilden in der heutigen Zeit eine immer größere Rolle in der Forschung und in der Technik. In fast allen Bereichen werden Modelle benutzt, um bekannte Szenarien besser zu verstehen, diese zu optimieren oder unbekannte Szenarien vorherzusagen. Dies passiert beispielsweise in der Verkehrstheorie zur Stauprävention und in der Wettervorhersage. Ein weiteres Beispiel ist die Epidemiologie. Im Laufe des geförderten COMET K-Projektes DEXHELPP [1], dessen Ziel darin besteht, neue Hilfsmittel auszuarbeiten, um die Funktionsweise von Gesundheitssystemen zu verbessern, wurde ein Modell entwickelt, das die Ausbreitung der Influenza innerhalb der österreichischen Bevölkerung simulieren soll. Es soll dazu benutzt werden, besser zu verstehen, mit wie vielen Infizierten während einer Grippewelle zu rechnen ist und aus welchen Bevölkerungsschichten diese vorwiegend stammen. Des Weiteren soll untersucht werden, wie eine Impfung diese Ausbreitung beeinflusst.
Bei der Entwicklung eines Modells gibt es mehrere Schritte die zur Überprüfung der Qualität herangezogen werden. Zu erwähnen sind hierbei vor allem die Verifizierung, also die Frage, ob das Modell richtig implementiert wurde und es keine internen Fehler gibt, sowie die Validierung, das heißt die Frage, ob das richtige Modell implementiert wurde und es auch tatsächlich die Wirklichkeit widerspiegelt. Diese Diplomarbeit befasst sich mit einem weiteren Aspekt der Qualitätsüberprüfung eines Modells: der Kalibrierung. Sie besteht darin, einzelne, bis dahin unbekannte Modellparameter zu bestimmen, sodass das Modell ein bekanntes Referenzsystem reproduzieren kann.
Zur Beschreibung eines solchen Modells bieten sich mehrere Methoden an. Klassische Techniken benutzen dabei meistens Systeme von gewöhnlichen oder partiellen Differentialgleichungen. Das oben erwähnte Influenzamodell benutzt jedoch eine relativ neue Modellierungsmethode: die agentenbasierte Modellierung. Auch wenn die agentenbasierte Modellierung bei Anwendung in der Epidemiologie gewisse Vorteile gegenüber anderen Modellierungsarten bietet, hat sie jedoch den Nachteil, dass die bekannten Techniken zur Validierung und Kalibrierung von Modellen nicht unbedingt anwendbar sind. Außerdem wurde noch nicht hinreichend untersucht, welche alternative Methoden verwendet werden können.
Das Ziel dieser Diplomarbeit besteht darin, mögliche Methoden zur Kalibrierung eines agentenbasierten Modells vorzustellen und ihre Leistung zu testen. Es wäre wünschenswert, eine Technik zu finden, die mit möglichst wenig Information über das zu kalibrierende Modell auskommt und dementsprechend auf eine Vielzahl von agentenbasierten Modellen angewendet werden kann.

1 http://www.dexhelpp.at/

2 Theoretische Grundlagen

2.1 Modellgrundlagen

2.1.1 SIR-Modell

Das Influenzamodell, das in dieser Diplomarbeit behandelt wird, hat die Form eines SIR - Modells. Diese Modelle werden dazu benutzt, um eine große Klasse von Infektionskrankheiten vereinfacht darzustellen. Prinzipiell können sich die Personen in dieser Darstellung in einem von drei verschiedenen Zuständen befinden, nämlich S, I oder R.
S steht hierbei für die suszeptiblen Personen. Sie bilden den Teil der Bevölkerung, der bis jetzt noch nicht von der Epidemie betroffen ist und in Zukunft mit der Krankheit angesteckt werden kann. Die Personen, die bereits mit der Krankheit infiziert worden sind, befinden sich in dem Zustand I. Schlussendlich gibt es noch den Zustand R. Dieser bezeichnet all resistente Personen. Diesen Zustand nehmen alle Personen an, die von der Infektion genesen sind und somit den Zustand I verlassen haben.

Natürlich ist dieses SIR-Modell eine sehr große Vereinfachung einer Infektionskrankheit. So werden beispielsweise die Zustände “infiziert“ und “infektiös“ gleichgesetzt. Infizierte Personen sind diejenigen, die von der Krankheit angesteckt sind, während infektiöse Personen diejenigen sind, die andere Menschen mit der Krankheit anstecken können. Bei manchen Krankheiten stimmen diese beiden Phasen nicht komplett überein. Demnach existiert eine Vielzahl an Verfeinerungen und Erweiterungen dieses SIR-Modell. Beispielsweise ist es möglich, die einzelnen Zustände zusätzlich in Unterklassen einzuteilen [14]. Somit kann man die Personen nach dem Grad ihrer Symptome unterscheiden. Es kann demnach berücksichtigt werden, dass Menschen mit leichten Symptomen ihr normales Verhalten nicht ändern und dementsprechend tendenziell mehr Leute anstecken als Personen mit starken Symptomen, die sich eher zurückziehen und in Behandlung begeben.

Üblicherweise wurden SIR - Modelle mithilfe von Differentialgleichungen simuliert. [17] Hierbei wurde die Population in die verschiedenen Klassen eingeteilt und die zeitliche Veränderung dieser Klassen wurde analysiert. Dabei entstehen folgende

Differentialgleichungen:

$$\begin{aligned} \dot{S(t)} &= -\alpha \cdot S(t) \cdot I(t) & S(0) &= S_0 \\ \dot{I(t)} &= \alpha \cdot S(t) \cdot I(t) - \beta \cdot I(t) & I(0) &= I_0 = N - S_0 \\ \dot{R(t)} &= \beta \cdot I(t) & R(0) &= 0 \\ S(t) + I(t) + R(t) &= N \end{aligned} \tag{2.1}$$

Dabei bezeichnet der Parameter α den Anteil an Kontakten zwischen suszeptiblen und infizierten Personen, bei denen eine Krankheitsübertragung stattfindet. β beschreibt den Anteil an Infizierten, die in die R-Klasse übergehen. Beide Parameter sind abhängig von der betrachteten Bevölkerunggröße N.

In dieser Arbeit wird jedoch eine andere Art der Modellierung untersucht: die sogenannte agentenbasierte Modellierung (kurz ABM).

2.1.2 Agentenbasierte Modellierung

Die ABM ist eine neuere Modellierungsmethode, die erst in den 90er Jahren entwickelt wurde. Hierbei werden die Personen nicht in Gruppen zusammengefasst, sondern jedes Individuum wird einzeln durch einen sogenannten Agenten dargestellt. Die grundlegende Idee besteht nun darin, das gesamte System mithilfe dieser Agenten zu beschreiben. Sie bilden eigenständige Komponenten, die nach gewissen Regeln handeln und interagieren können. Aus diesen Aktionen ergibt sich dann das Gesamtverhalten des Systems. Es ist präzise festgelegt, welche Eigenschaften und Fähigkeiten ein Agent besitzen muss, um als solcher zu gelten. Im Laufe der Jahre wurden verschiedene Definitionen ausgearbeitet.
1997 verlangt *Wooldridge*, dass ein Agent vier Eigenschaften besitzt [29]: Autonomie, Reaktionsfähigkeit, zielorientiertes Handeln und soziale Fähigkeiten. Dies bedeutet, dass ein Agent ohne Einfluss von außen und nur abhängig von seinem Zustand handeln kann. Er kann jedoch mit seiner Umgebung und anderen Agenten interagieren und so sein Verhalten ändern, um sein Ziel zu erreichen.
Ein paar Jahre später erweitert *Jennings* diese Definition und verlangt zusätzlich, dass die einzelnen Agenten eindeutig identifizierbar und voneinander unterscheidbar sind und erkennbare Grenzen besitzen [16]. Dies wird auch in der heute gängigen Definition eines Agenten gefordert [20].

Bei einem SIR - Modell stellen die Agenten naheliegenderweise die einzelnen Personen dar. Jedoch können die Definitionen viel allgemeiner verstanden werden und die Agenten können viel abstraktere Gebilde darstellen.

Ein besonderes Merkmal der agentenbasierten Modellierung besteht darin, dass nicht versucht wird, die globalen Veränderungen im System zu untersuchen und daraus auf die einzelnen Individuen zu schließen (top-down Methode). Stattdessen werden einzelne Agenten mit einem jeweils festgelegten Verhalten erschaffen. Anschließend lässt

man die Agenten in der Simulation mit der Umgebung und untereinander agieren und lässt sich quasi von dem globalen Verhalten des Systems "überraschen" (bottom-up Methode). Dieses Phänomen nennt man *emergent behaviour* [4]. Es beschreibt die Tatsache, dass ein komplexes Systemverhalten aus relativ einfachen Handlungsregeln der einzelnen Agenten hervorgeht.
Die agentenbasierte Modellierung ermöglicht es somit, einzelne Individuen viel differenzierter darzustellen und handeln zu lassen. Dies ist insbesonders bei der Untersuchung von Epidemieausbreitungen sehr nützlich, da man in der Lage ist, auf geographische und soziale Strukturen sowie Verhaltensänderungen einzugehen. So kann man berücksichtigen, dass Personen, bei denen Krankheitsymptome auftreten, ihren normalen Tagesablauf ändern und mit einer kleineren Anzahl anderen Personen in Kontakt treten.
Außerdem werden nur Regeln für die einzelnen Individuen benötigt und es wird keine Kenntnis über das Gesamtverhalten vorausgesetzt. Dies ist sehr hilfreich bei Problemstellungen, bei denen nicht genau bekannt ist, wie sich das System als ganzes verändern wird. Ein gutes Beispiel hierfür ist das Schwarmverhalten von Fischen. Es ist bekannt, nach welchen Regeln sich der einzelne Fisch in der Gruppe orientiert, es ist jedoch schwierig vorherzusagen, wie sich der Schwarm in seiner Gesamtheit bewegt.
Offensichtlich bietet das Konzept, jedes Individuum einzeln darzustellen, eine Reihe an Vorteilen. Jedoch sollte nicht außer Acht gelassen werden, dass damit auch ein enormer Rechenaufwand verbunden ist. Besonders bei einer hohen Agentenzahl ist dies von großer Bedeutung. Als Beispiel kann man die oben erwähnte Modellierung eines SIR-Modells mithilfe Differentialgleichungen heranziehen. Hier hat die Bevölkerungsgröße keinen weiteren Einfluss auf die Komplexität des Problems. Bei der agentenbasierten Modellierung jedoch erhöht sich nicht nur die Anzahl der zu berechnenden Personen, sondern auch deren Möglichkeiten, miteinander in Kontakt zu treten. Infolgedessen steigt der Arbeitsaufwand enorm. Deshalb ist es nicht verwunderlich, dass die agentenbasierte Modellierung erst mit der Entwicklung von leistungsstarken Computern entstanden ist. Da diese immer performanter werden, wird die agentenbasierte Modellierung vermutlich immer mehr an Bedeutung gewinnen.

2.2 Kalibrierung

Ein Problem, auf das man in der Modellierung immer wieder stößt, ist, dass von verschiedenen Parametern der genaue Wert nicht bekannt ist. Bei einem Influenzamodell können das z.B. die Infektions- und die Genesungswahrscheinlichkeit sein. Nur wenn diese Parameter annähernd gut bekannt sind, liefert das Modell Resultate, die auch tatsächlich die Wirklichkeit widerspiegeln.
Ziel der Kalibrierung ist es nun, diese Parametereinstellung zu finden [15]. Dazu werden in den meisten Fällen reale, erhobene Daten herangezogen, die mit der Simulation reproduziert werden sollen. Das heißt, es sind gewünschte Modellresultate bekannt, welche mit unbekannten Parameterwerten erreicht werden sollen. Erst wenn

dies erfolgreich war, kann man das Modell benutzen, um Aussagen über die Zukunft zu treffen. Eine übliche Vorgehensweise der Kalibrierung ist das händische Ausprobieren. Dabei werden einfach ein paar plausibel erscheinende Werte der Parameter ausprobiert und überprüft, bei welchem Parameterset die simulierten Daten am besten mit den erwünschten Daten übereinstimmen. Gelegentlich hat man das Gefühl, einen kausalen Zusammenhang feststellen zu können und dreht noch ein wenig an den getesteten Parametern, um das Ergebnis zu verbessern. Bei einem oder zwei zu kalibrierenden Parametern kann diese Praxis durchaus erfolgreich sein. Steigt jedoch die Anzahl an Parametern, können sich diese untereinander auf die verschiedensten Arten beeinflussen und ein einfaches "Erraten" der richtigen Werte ist fast unmöglich.

In dieser Arbeit werden verschiedene Algorithmen ausgetestet, die versuchen, den Kalibrierungsprozess zu "automatisieren". Sie alle benutzen eine Fehlerfunktion. Diese berechnet zu gegebenen Parameterwerten den Fehler zwischen den zu reproduzierenden und den anhand des Modells berechneten Resultate. Es gibt natürlich verschiedene Möglichkeiten, diesen Fehler zu berechnen. Üblicherweise wird ein gewichteter euklidischer Abstand zwischen den bekannten Daten und den Simulationsergebnissen benutzt, wobei die Gewichte je nach Aussehen der zu reproduzierenden Daten gewählt werden. Also hat die Fehlerfunktion folgende Form:

$$F(p) = \sqrt{\sum_i w_i \cdot (s_i - r_i)^2} \tag{2.2}$$

mit

$p \in \mathbb{R}^m$ ... Parametervektor
$w \in \mathbb{R}^n$ ... Gewichte
$s \in \mathbb{R}^n$ ... Simulationsergebnisse
$r \in \mathbb{R}^n$ ... reale Daten

Die einzelnen Kalibrierungsalgorithmen unterscheiden sich jetzt dadurch, welche Methode sie anwenden, um diese Fehlerfunktion zu minimieren.

2.3 Allgemeine Probleme bei Kalibrierung von ABM

Da bei einem agentenbasierten Modell aufgrund des emergent behaviour im Voraus nichts über das globale Verhalten ausgesagt werden kann, treten bei der Kalibrierung dieser Modelle eine Reihe von Problemen auf [5].
Ein großes Problem besteht darin, dass der Zusammenhang zwischen den benutzten Parameterwerten und den simulierten Daten nicht genau bekannt ist. Es ist nicht unbedingt der Fall, dass die Resultate global stetig bezüglich Parameteränderungen sind. Stattdessen ist diese Stetigkeit oft nur auf gewissen Intervallen gegeben. Zwischen diesen Bereichen können Sprünge in den Simulationsergebnissen auftreten. Dementsprechend können auch nur kleine Modifikationen der Parameter zu grund-

verschiedenen Ergebnissen führen. So könnte beispielsweise beim Influenzamodell ein gewisses Parameterset eine populationsweite Epidemie hervorrufen, während ein leicht verändertes Parameterset dazu führt, dass die Krankheit schon nach wenigen infizierten Personen ausgelöscht ist.
Als weitere Konsequenz ist auch die Fehlerfunktion nicht in einer analytischen Form gegeben. Also können übliche Optimierungsverfahren, die beispielsweise den Gradienten der zu optimierenden Funktion benutzen, nicht angewendet werden. Stattdessen müssen sogenannte *black-box-Algorithmen* verwendet werden. Diese verwenden lediglich die einzelnen Werte der Fehlerfunktionen, die anhand eines Simulationsdurchlaufes berechnet werden. Um die Fehlerfunktion erfolgreich zu minimieren, benötigt ein black-box-Algorithmus allerdings sehr viele Funktionswerte. Das heißt, es müssen viele Modellsimulationen durchgeführt werden. Da diese in der Regel ziemlich rechenaufwendig sind, hat eine erfolgreiche Kalibrierung eine sehr lange Laufzeit.
Des Weiteren sind agentenbasierte Modelle in den seltesten Fällen deterministisch. Dadurch, dass die einzelnen Agenten autonom Entscheidungen treffen können, liefern zwei Modellaufrufe nicht exakt dieselben Daten, obwohl sie mit denselben Parameterwerten aufgerufen wurden. Diese Streuung der Resultate, die in der Natur der agentenbasierten Modellierung liegt, ist störend für die Kalibrierung. Sie führt zu einer Streuung der Werte der Fehlerfunktion und erschwert das Auffinden des Minimums. Um dem entgegenzuwirken bieten sich in der Regel zwei Methoden an. Zum einen kann man die Agentenanzahl innerhalb des Modells erhöhen. Üblicherweise glättet eine höhere Anzahl an Agenten die entstehenden stochastischen Schwankungen und die Streuung innerhalb der Resultate wird kleiner. Eine andere Möglichkeit besteht darin, das Modell für die Berechnung eines Wertes der Fehlerfunktion mehrmals aufzurufen und über die erhaltenen Resultate zu mitteln. Beide Vorgehensweisen führen allerdings dazu, dass die Laufzeit der Kalibrierung noch zusätzlich verlängert wird, sei es durch eine längere Laufzeit der einzelnen Modellaufrufen oder durch eine höhere Anzahl an benötigten Modellaufrufen.

3 Kalibrierungsalgorithmen

3.1 Optimierungsalgorithmen

Die Algorithmen, die zur Optimierung eines Black-Box-Verfahrens herangezogen werden, gehören meistens zu der Klasse der Metaheuristiken. Eine Metaheuristik ist nur eine Beschreibung einer abstrakten Vorgehensweise. Dies ermöglicht es, dass sie nicht nur auf ein spezifisches Problem, sondern auf eine Vielfalt von Optimierungsproblemen angewendet werden kann. Allerdings ist nicht garantiert, dass eine Metaheuristik eine globale optimale Lösung liefert. Im Vergleich zur Optimallösung kann die gefundene Lösung sogar beliebig schlecht sein.
Außerdem haben *Wolpert* und *Macready* in ihrem *no-free-lunch-theorem* bewiesen, dass es keinen universell besten Optimierungsalgorithmus gibt [28]. Falls ein Algorithmus auf einer Klasse von Probleme bessere Resultate erzielt, als alle anderen Optimierungsmethoden, so ist er diesen bei einer anderen Klasse von Problemen unterlegen. Deshalb ist es erforderlich, genauer zu untersuchen, welche Optimierungsmethode für die vorliegende Problemstellung am besten geeignet ist. Dabei stehen einem eine Vielzahl an verschiedenen Metaheuristiken zur Verfügung, die man gemäß ihrer generellen Vorgehensweisen und Eigenschaften klassifizieren kann [3].
Eine einfache Unterscheidung ist die zwischen einpunktigen und populationsbasierten Metaheuristiken. Ein einpunktiger Optimierungsalgorithmus betrachtet zu jedem Zeitpunkt nur einen möglichen Kandidaten für das globale Optimum und versucht, diesen in jedem Schritt durch einen besseren Punkt aus der Nachbarschaft zu ersetzen. Bei einem populationsbasierten Algorithmus werden mehrere Punkte gleichzeitig behandelt und auf verschieden Arten kombiniert, um so eine neue, vorzugsweise bessere, Population zu generieren.
Des Weiteren kann man zwischen gedächtnislosen und nicht-gedächtnislosen Algorithmen unterscheiden. Eine gedächtnislose Metaheuristik betrachtet nur die aktuellen Punkte und berücksichtigt nicht, welche Punkte im Lauf der Optimierung bereits berechnet wurden. Andere Metaheuristiken speichern diese oder andere Informationen über den bisherigen Verlauf der Berechnungen ab und benutzen diese, um die weitere Suche effizienter zu gestalten. So kann zum Beispiel vermieden werden, dass Punkte, die bereits verworfen wurden, ein weiteres Mal ausgewählt werden.
Schließlich werden, vor allem in den letzten Jahre, immer mehr Metaheuristiken entwickelt, deren Vorgehensweise von der Natur inspiriert ist. Sie versuchen vor allem, die Schwarmintelligenz zu modellieren, die beispielsweise bei Ameisen und Vögeln beobachtet werden können.

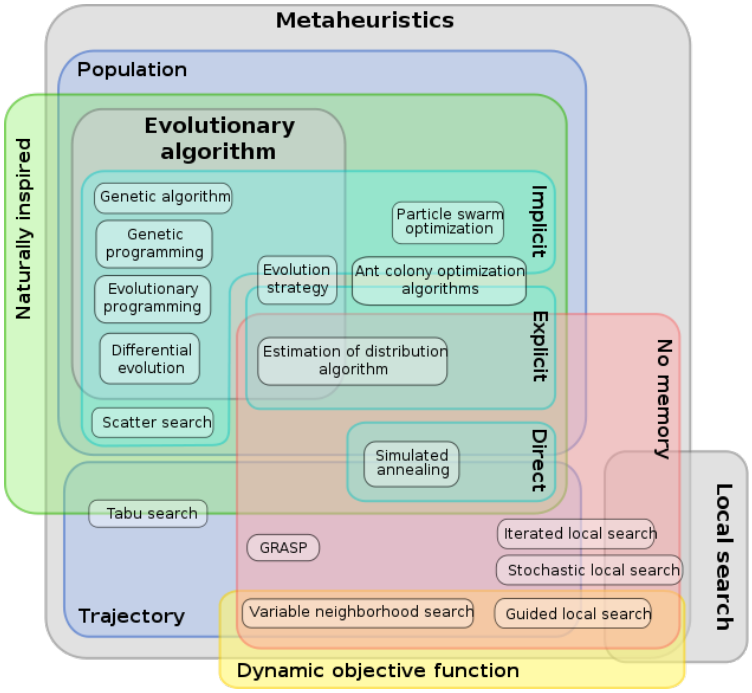

Abbildung 3.1: Klassifikation verschiedener Metaheuristiken [7]

3.2 Simulated Annealing

Das **Simulated Annealing**, im Deutschen auch als **simulierte Abkühlung** bezeichnet, ist eine einpunktige Metaheuristik. Sie wurde in den Jahren 1983 und 1985 von *Kirkpatrick et al.*[18] und *Cerny* [6] unabhängig voneinander entwickelt und ist eine Erweiterung des Metropolisalgorithmus [21].
Die Idee zu diesem Algorithmus stammt aus der Werkstoffkunde. Beim sogenannten Glühen wird ein Metall zuerst stark erhitzt und danach langsam abgekühlt. Durch diesen kontrollierten Kühlungsprozess haben die Atome Zeit, stabile Kristalle zu bilden und es wird ein niedriger, nahezu optimaler Energiezustand erreicht. Beim Simulated Annealing repräsentiert die fallende Temperatur die im Lauf der Simulation sinkende Wahrscheinlichkeit, einen energetisch schlechteren Zustand zu akzeptieren.

Konkret ergibt sich folgender Optimierungsalgorithmus: Sei F die zu minimierende Funktion, x der aktuelle Kandidat für das Minimum, sowie T die Temperatur.
Zuerst wird ein Zustand y in der Nähe von x bestimmt und der Wert $F(y)$ berechnet. Ist dieser Wert kleiner als der aktuelle Wert $F(x)$, wird y als neuer Kandidat akzeptiert. Falls gilt, dass $F(y)$ größer als $F(x)$ ist, wird der Zustand y nur mit einer Wahrscheinlichkeit von $exp\left(-\frac{F(y)-F(x)}{T}\right)$ akzeptiert. Anschließend wird die Temperatur T gesenkt.
Diese Schritte werden so oft wiederholt, bis ein Abbruchkriterium erfüllt ist.

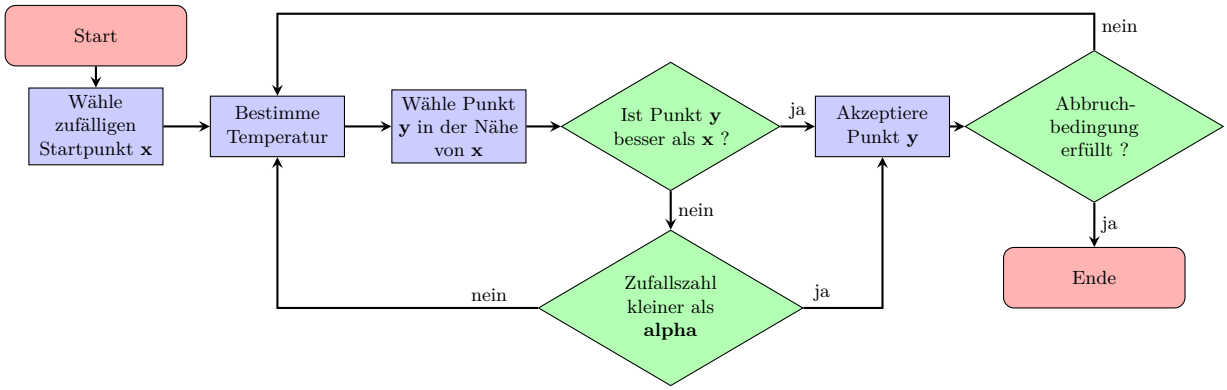

Abbildung 3.2: Flowchart des Simulated Annealing nach [19]

Der Erfolg des Simulated Annealing hängt von der Wahl von vier Komponenten ab: die Nachbarschaft, die Akzeptanzwahrscheinlichkeit, die Abkühlungsfunktion und das Abbruchkriterium. Auf diese Komponenten wird im Folgenden näher eingegangen.

Nachbarschaft

Betrachten wir zuerst den Begriff der Nachbarschaft. Inwiefern ein Zustand y sich in der Nähe von einem Zustand x befindet, hängt sehr stark von der spezifischen Problemstellung und dem benutzten Zustandsraum ab. Ein Beispiel liefert das Problem des Handelsreisenden, ein typisches Optimierungsproblem. Hier besteht der Zustandsraum aus allen möglichen Permutationen gemäß denen die Städte angefahren werden können. Bezeichnet x nun eine konkrete Reihenfolge $(x_1, x_2, \ldots, x_n)$. Dann kann man die Nachbarschaft von x definieren durch alle Reihenfolgen y, die entstanden sind, indem zwei aufeinanderfolgende Städte in x ihre Reihenfolge geändert haben. Also hat y die Form $(x_1, x_2, \ldots, x_i, x_{i-1}, \ldots, x_n)$.
Man kann die Nachbarschaft aber auch großzügiger definieren, indem man die Einschränkung weglässt, dass nur aufeinanderfolgende Städte die Reihenfolge vertauschen dürfen. Erlaubt man, dass zwei beliebige Städte x_i und x_j mit $i < j$ ihre Plätze innerhalb der Permutation tauschen, erhält man eine größere Nachbarschaft deren Elemente die Form $(x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n)$ haben.
Die Wahl der benutzten Nachbarschaft kann große Auswirkungen auf die Leistungsfähigkeit des Optimierungsalgorithmus haben und sollte deshalb bei jeder spezifischen Problemstellung sorgfältig überlegt sein.

Akzeptanzwahrscheinlichkeit

Die Methode, Punkte in der Nachbarschaft des aktuellen Kandidaten zu testen und diese zu akzeptieren, wenn sie eine Verbesserung hervorrufen, ist die Vorgehensweise

des Bergsteigeralgorithmus. [27] Sie führt allerdings dazu, dass ein lokales Minimum nie verlassen wird und somit meistens nicht ein globales Minimum erreicht wird. Dies führt uns zu einer weiteren wichtigen Komponente des Simulated Annealing, der Akzeptanzwahrscheinlichkeit. Um das Festfahren in einem lokalen Minimum zu vermeiden, werden schlechtere Werte nicht a priori abgelehnt, sondern mit einer gewissen Wahrscheinlichkeit angenommen. Das heißt, falls $F(y) > F(x)$ wird der Punkt y trotzdem mit einer Wahrscheinlichkeit $exp\left(-\frac{F(y)-F(x)}{T}\right)$ akzeptiert. Man erkennt, dass Werte y, wahrscheinlicher angenommen werden, je kleiner ihre Verschlechterung gegenüber dem aktuellen Wert x und je größer die momentane Temperatur T ist.

Kühlungsverfahren

Diese Temperatur wird im Laufe des Simulated Annealing gesenkt. Wie eingangs erwähnt, muss der Kühlungsprozess langsam verlaufen, damit das System Zeit hat, den optimalen Zustand zu erreichen. Also ist ein geeignetes Kühlungsverfahren äußerst wichtig für die Konvergenz des Algorithmus. Wie dieses optimale Kühlungsverfahren aussieht, ist allerdings noch nicht bekannt. Das liegt vor allem daran, dass nur ein Konvergenzbeweis für ein Verfahren mit unendlich vielen Schritten vorliegt. Da dies in der Praxis natürlich nicht realisierbar ist, muss man sich auf Kühlungsprozesse beschränken, die in absehbarer Zeit ein annehmbares Ergebnis liefern.
Das geometrische Abkühlungsverfahren, das von *Aarts et al.* [1] erwähnt wird, wird relativ häufig angewendet. Hier wird nach einer festgelegten Anzahl Schritten m die Temperatur T auf $\gamma \cdot T$ reduziert. Bei γ handelt es sich um einen Wert nahe bei 1 (typischerweise zwischen 0,8 und 0,99). Der genaue Wert von γ und m werden üblicherweise experimentell bestimmt.
Press et al. [23] schlägt zudem eine Abwandlung dieses Kühlungsprozesses vor. Hier nimmt die Temperatur nach jeweils m Schritten den neuen Wert $T_0\left(1-\frac{k}{K}\right)^{\alpha}$ an. Dabei bezeichnet K eine festgelegte Gesamtanzahl an Schritten, nach der das Verfahren beendet sein soll. Bei k handelt es sich um die Anzahl bereits berechneter Schritte und bei α um eine Konstante, typischerweise 1, 2, oder 4. Je höher der Wert von α, desto mehr der vorgegebenen Schritte werden bei einer niedrigen Temperatur ausgeführt.
Abramson et al. [2] stellen noch weitere, teilweise adaptive, Kühlungsverfahren vor. Eine Besonderheit dieser Verfahren ist, dass zwischendurch sogar Temperaturerhöhungen erlaubt sind. Dies soll das Auffinden eines globalen statt nur eines lokalen Optimums fördern. Allerdings verlängert sich dadurch natürlich die Laufzeit des Algorithmus und man riskiert in eine Endlosschleife zu gelangen.

Abbruchkriterium

Schlussendlich stellt sich die Frage nach dem Abbruchkriterium für das Optimierungsverfahren. Es nicht offensichtlich, wie dieses festgelegt werden soll. Wie eingangs

erwähnt, garantiert das Simulated Annealing nicht, dass ein globales Minimum gefunden wird. Dementsprechend ist es nicht zielführend, das Optimierungsverfahren erst abzubrechen, wenn das Ergebnis unterhalb einer festgelegten Schranke liegt. Dies kann unter Umständen, auch bei einer sehr langen Laufzeit, gar nicht eintreten.
Stattdessen kann man den Algorithmus nach einer festgelegten Anzahl an Schleifendurchgängen abbrechen. Da allerdings nicht sehr viel über die Konvergenzgeschwindigkeit des Simulated Annealing bekannt ist, ist es unklar, wie viele Durchgänge angebracht sind. Es kann sein, dass schon ziemlich schnell ein guter Wert gefunden wurde, der sich dann auch nicht mehr wesentlich verbessert. Es kann allerdings auch passieren, dass der Algorithmus eine längere Eingangsphase braucht, um sich dem minimalen Wert anzunähern und diesen dann nicht mehr innerhalb der vorgegebenen Zeit erreicht.
Um dies zu vermeiden, kann man das Verfahren erst dann abbrechen, wenn es stagniert. Also dann, wenn sich der aktuelle Kandidat innerhalb einer vorgegebenen Anzahl an Schleifendurchgängen nicht mehr verändert hat. Zu diesem Zeitpunkt ist zumindest ein lokales Minimum erreicht. Dieses kann nicht mehr verlassen werden, da die Temperatur, und somit die Wahrscheinlichkeit einen schlechteren Wert anzunehmen, nach vielen Schleifendurchläufen schon zu niedrig ist.

Konvergenz

Wie schon erwähnt, existieren Beweise der Konvergenz des Simulated Annealing nur für eine unendliche Folge von Temperaturwerten und unter zusätzlichen Annahmen. Einer dieser Beweise stammt von *Hajek* [12]. Er benutzt das Konzept der Markov-Ketten.
Dabei handelt es sich um einen stochastischen Prozess $(X_n)_{n\in\mathbb{N}_0}$ mit abzählbarem Zustandsraum I. Die definierende Eigenschaft einer Markov-Kette ist die Gedächtnislosigkeit, d.h es gilt für alle $n \in \mathbb{N}_0$ und alle $i_0, \ldots, i_n, i_{n+1} \in I$:

$$P(X_{n+1} = i_{n+1} \mid X_0 = i_0, \ldots, X_{n-1} = i_{n-1}, X_n = i_n) = P(X_{n+1} = i_{n+1} | X_n = i_n) \tag{3.1}$$

Dabei wird die Wahrscheinlichkeit $P(X_{n+1} = i_{n+1}|X_n = i_n) = P_n(i_n,i_{n+1})$ als Übergangswahrscheinlichkeit bezeichnet.
Eine gute Übersicht der wichtigsten Eigenschaften und Sätze zu Markov-Ketten findet sich beispielsweise in [26].
Das Simulated Annealing erfüllt die Anforderung, dass die Auswahl des nächsten Punktes y nur vom aktuellen Punkt x abhängt und kann somit als Markov-Kette aufgefasst werden. Bezeichnet man nun die Menge aller benachbarten Punkte von x mit $N(x)$, so ergibt sich für die Übergangswahrscheinlichkeit folgende Form:

$$P_n(x,y) = \begin{cases} 0 & \text{falls } y \notin N(x) \\ R(x,y) \cdot \exp\left(-\frac{[F(y)-F(x)]^+}{T_n}\right) & \text{falls } y \in N(x) \text{ und } y \neq x \\ 1 - \sum_{z\neq x} P_n(x,z) & \text{falls } y = x \end{cases} \tag{3.2}$$

$[F(y) - F(x)]^+$ bezeichnet dabei den Positivteil von $(F(y) - F(x))$. Dieser ist folgendermaßen definiert: falls $F(y) > F(x)$, also im Fall einer Verschlechterung, beträgt dieser Wert genau $F(y) - F(x)$ und es ergibt sich die oben genannte Akzeptanzwahrscheinlichkeit. Falls $F(y) < F(x)$, also im Fall einer Verbesserung, erhält der Positivteil den Wert 0 und die Akzeptanzwahrscheinlichkeit berechnet sich zu 1.
$R(x,y)$ bezeichnet die Wahrscheinlichkeit, dass der Punkt y aus der Menge der benachbarten Punkte von x ausgewählt wird. In den meisten Anwendungen gilt, dass diese unabhängig vom Punkt y ist und demnach $\frac{1}{|N(x)|}$ beträgt.
Mithilfe dieses Werkzeugs kann bewiesen werden, dass das Ergebnis des Simulated Annealing, bei einem geeigneten Abkühlungsverfahren, in der Wahrscheinlichkeit zu einem globalen Minimum konvergiert. Genauer gesagt, gilt:

$$\lim_{n\to\infty} P(X_n \in I^*) = 1 \qquad \text{falls} \quad T_n = \frac{c}{\log(1+n)} \tag{3.3}$$

Dabei bezeichnet I^* die Menge der globalen Minima und c muss gemäß dem Aussehen der lokalen Minima gewählt werden. Eine genauere Formulierung der Konvergenzaussage sowie der vollständige Beweis befinden sich in [12].

3.3 Evolutionäre Algorithmen

Als nächste Kalibrierungsmethode werden evolutionäre Algorithmen betrachtet. Diese wurden vor allem in den 70er Jahren entwickelt und sind sehr von der Natur inspiriert. Dies spiegelt sich auch in den Bezeichnungen wider: ein Parameterset wird als Genom bezeichnet und die einzelnen Parameter als Chromosomen.
Bei der Entstehungsgeschichte kann man zwischen vier Konzepten unterscheiden: genetische Algorithmen, evolutionäre Algorithmen, genetische Programmierung und evolutionäre Programmierung. Mittlerweile hat sich die Idee der einzelnen Konzepte vermischt und eine strenge Unterscheidung ist nicht mehr möglich beziehungsweise erforderlich. Demzufolge werden die Begriffe teilweise synonym verwendet. In dieser Arbeit wird eine Mischung des genetischen und des evolutionären Algorithmus betrachtet. Sie wurden von *Holland* [13], beziehungsweise *Rechenberg* [24] entwickelt.

Im Unterschied zum Simulated Annealing wird bei evolutionären Algorithmen nicht nur ein möglicher Kandidat fürs Minimum betrachtet, sondern eine ganze Population. Ähnlich wie bei einer evolutionären Entwicklung in der Natur durchläuft diese Population von Kandidaten nun mehrere Prozesse. Zuerst werden verschiedene Genome mittels eines Selektionsverfahrens für den weiteren Verlauf bestimmt. Anschließend findet eine Rekombination statt. Das heißt, die Genome fügen sich paarweise als Eltern zusammen und kombinieren ihre Chromosome derart, dass zwei neue Genome entstehen. Schlussendlich werden vereinzelte Genome ausgewählt, an deren Chromosomen eine kleine Veränderung vorgenommen wird. Nun ist eine Generation abgeschlossen. Das Verfahren von Selektion, Rekombination und Mutation wird sooft durchgeführt, bis ein Abbruchkriterium erfüllt ist.

Die einzelnen evolutionären Algorithmen, beziehungsweise die unterschiedlichen historischen Konzepte, unterscheiden sich durch die Wahl der verschiedenen Selektions-, Rekombinations- und Mutationsverfahren und der Darstellung der Genome. Bei dem von *Holland* entwickelten genetischen Algorithmus wird eine rein binäre Darstellung der Genome gewählt. Das heißt, die zu kalibrierenden Parameter müssen zuerst mithilfe der Zahlen 0 und 1 kodiert werden. Anschließend werden auf diese Bitfolge die verschiedenen Operationen angewendet. Zum Schluss wird das Ergebnis wieder auf die ursprüngliche Parameterdarstellung zurück abgebildet. Die Idee zu dieser binären Repräsentation stammt aus der Natur, da auch hier alle Genome mithilfe von nur vier Nukleinsäuren kodiert sind. Außerdem erleichtert die binäre Darstellung die Verarbeitung mittels Computern.
Rechenberg hat für seinen evolutionären Algorithmus eine direkte Parameterdarstellung gewählt, allerdings wird bei ihm die nächste Generation nur mittels Selektion und Mutation berechnet. Eine Rekombination findet nicht statt.

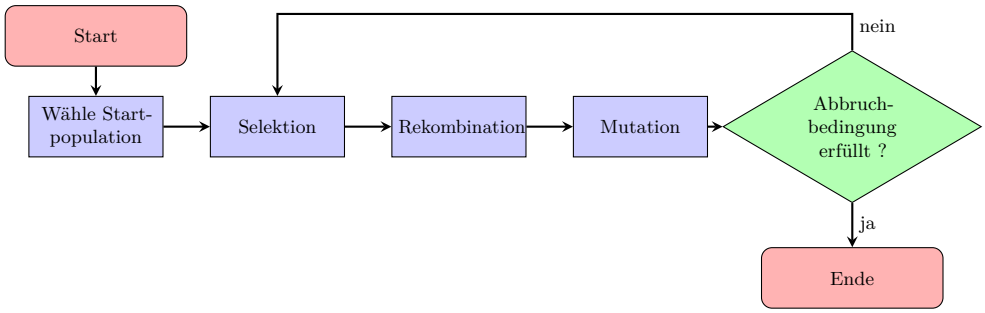

Abbildung 3.3: Flowchart des evolutionären Algorithmus nach [19]

Selektion

Bei der Wahl eines Selektionsprozesses bietet sich verschiedene Möglichkeiten. In [9] werden unter anderem die die fitnessbasierte Selektion und die Wettkampf-Selektion beschrieben.

Bei der fitnessbasierten Selektion [10] wird jedem Genom ein Fitnesswert zwischen 0 und 1 zugeordnet. Dieser soll größer sein, je kleiner der Fehlerwert $F(x)$ des Genoms ist. Zusätzlich soll die Summe der Fitnesswerte aller Genome 1 ergeben. Um diesen Wert zu berechnen gibt es mehrere Möglichkeiten. Beispielsweise kann man die n Genome nach ihrem Fehlerwert ordnen. Dem Genom mit dem größten Fehlerwert wird nun der Fitnesswert 1 zugeordnet, dem zweitschlechtesten Genom der Wert 2 und so weiter bis zum besten Genom, das den Wert n erhält. Anschließend werden alle Fitnesswerte noch durch $\sum_{i=0}^{n} i = \frac{n^2-n}{2}$ dividiert um sie zu normalisieren. Also falls x_m das m-beste Genom bezeichnet, gilt:

$$fit_{rang}(x_m) = \frac{2 \cdot (n - m + 1)}{n^2 - n} \tag{3.4}$$

Eine andere Möglichkeit die Fitness eines Genoms zu bestimmen, besteht darin, ihn direkt aus dem Fehlerwert zu berechnen. Auch hier werden am Ende die Fitnesswerte normalisiert:

$$s(x) = \frac{1}{F(x)} \qquad fit(x) = \frac{s(x)}{\sum_y s(y)} \tag{3.5}$$

Anhand der berechneten Fitness wird nun bestimmt, welche Genome wie oft für den weiteren Verlauf ausgewählt werden. Die Wahrscheinlichkeit, dass ein bestimmtes Genom selektiert wird, entspricht dabei dessen Fitnesswert. Also werden Genome mit einer hohen Fitness tendenziell öfter ausgewählt als solche mit einer niedrigeren Fitness. Will man sichergehen, dass das Genom mit dem größten Fehlerwert aus dem weiteren Verlauf ausgeschlossen wird, kann man den Fitnesswert der einzelnen Genome mithilfe dieses maximalen Fehlers $F_{max} = \max_y F(y)$ skalieren:

$$s(x) = F_{max} - F(x) \qquad fit_{skal}(x) = \frac{s(x)}{\sum_y s(y)} \tag{3.6}$$

Ein Beispiel für die fitnessbasierte Selektion mithilfe verschiedener Berechnungen der Fitness befindet sich in Abbildung 3.4.

Genom	**Fehler**	fit_{rang}	fit	fit_{skal}
a	39	0.33	0.43	0.38
b	85	0.27	0.20	0.27
c	91	0.20	0.18	0.26
d	162	0.13	0.10	0.09
e	199	0.07	0.09	0

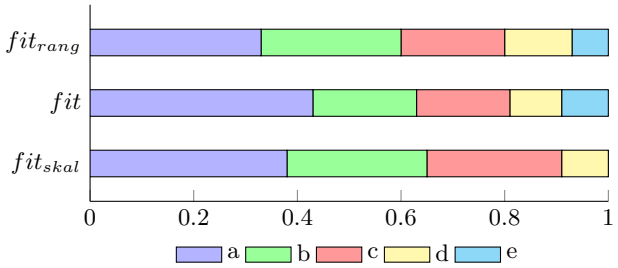

Abbildung 3.4: Verschiedene Varianten der fitnessbasierten Selektion

Bei der Wettkampf-Selektion wird kein Fitnesswert benötigt. Hier wird eine festgelegte Anzahl an Genomen zufällig, mit Zurücklegen, aus der Population gezogen und derjenige mit dem niedrigeren Fehler wird übernommen. Dies wird sooft wiederholt, bis die gewünschte Populationsgröße n erreicht ist. Üblicherweise treten nur zwei Genome gegeneinander an, größere Wettkämpfe sind jedoch auch möglich.

Rekombination

Der nächste Schritt eines evolutionären Algorithmus ist die Rekombination. Dazu wird zufällig ein gewisser Anteil p_c von Genomen ausgewählt, aus denen dann neue Genome gebildet werden. Dies kann auf verschiedene Weisen stattfinden. Üblicherweise werden jeweils zwei Elterngenome zur Rekombination bestimmt und es werden zwei Nachkommen geformt. Eine größere Menge an Eltern ist jedoch auch möglich. Allerdings ist es unklar, ob mehrere Eltern ein besseres Ergebnis liefern und die optimale Anzahl an Eltern ist vermutlich problemabhängig [8].
Die einfachste Form der Rekombination ist die im Englischen als *1-point-crossover* bezeichnete Methode. Hierbei wird zufällig eine Position innerhalb der Genome bestimmt. Die Chromosomen des ersten Nachkommen vor dieser Position werden vom Elternteil A übernommen, die dahinter von Elternteil B. Dieses Rekombinationsverfahren lässt sich einfach verallgemeinern. Anstatt eine fixe Position zu wählen, an der die Elterngenome aufgespaltet werden, kann für jedes einzelne Chromosom zufällig entschieden werden, ob es vom Elternteil A oder B übernommen wird. Die Wahrscheinlichkeit, das Chromosom von Elternteil A zu wählen, entspricht dabei $p_A = \frac{1}{2}$. Wurde bei der Selektion den einzelnen Genomen ein Fitnesswert zugeordnet, kann dieser benutzt, werden um die Auswahlwahrscheinlichkeit zu beeinflussen. In diesem Fall wäre die Wahrscheinlichkeit

$$p_A = \frac{fit(A)}{fit(A) + fit(B)} \tag{3.7}$$

Der zweite Nachfahre wird gebildet indem jene Chromosome der Elternteile gewählt werden, die nicht zur Bildung von dem ersten Nachfahren benutzt wurden.

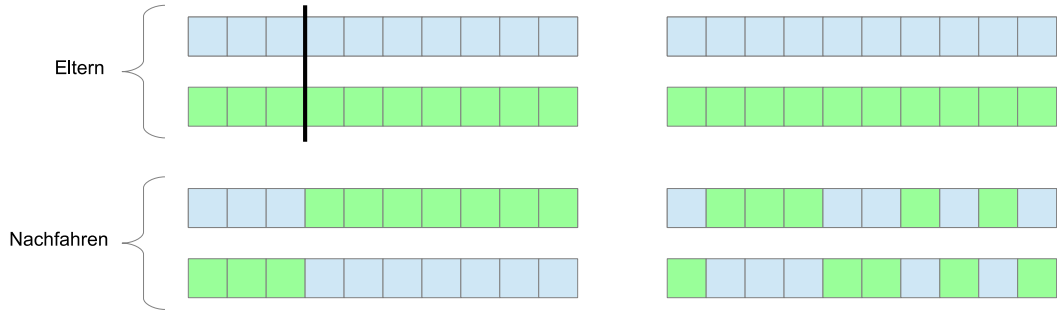

Abbildung 3.5: Verschiedene Rekombinationsmethoden

Nun muss noch entschieden werden, welche zwei der insgesamt vier Genome für den weiteren Verlauf übernommen werden. Als offensichtliche Möglichkeiten bieten sich an, die beiden Nachkommen zu behalten und die Eltern verfallen zu lassen. Alternativ wählt man aus den vier Genomen die beiden mit dem geringsten Fehlerwert aus, unabhängig davon ob es sich dabei um ein Elternteil oder einen Nachkommen handelt. Als weitere Variante kann man auch, wie in der Wettkampf-Selektion, jeweils zwei Genome gegeneinander antreten lassen und den Sieger zurückbehalten.

Diese Rekombinationsmethoden eignen sich vor allem für den Fall, dass die Parameterwerte bitkodiert sind. Eine Rekombination fand statt, indem für jede Position entschieden wurde, ob der Bitwert von Elternteil A oder B übernommen wird. Falls die Parameter mittels reeller Zahlen dargestellt werden, bietet sich eine andere Form der Rekombination an: Für jedes Chromosom können die unterschiedlichen Werte der Chromosome der Eltern kombiniert werden. Eine Möglichkeit hierfür ist, den euklidischen Mittelwert zu berechnen. Bei zwei Elternteilen gilt also, dass $C_i = \frac{1}{2}A_i + \frac{1}{2}B_i$. Falls bei der Selektion ein Fitnesswert für die einzelnen Genome berechnet wurde, kann dieser benutzt werden, um den Mittelwert zu gewichtet:

$$C_i = \frac{fit(A)}{fit(A) + fit(B)} A_i + \frac{fit(A)}{fit(A) + fit(B)} B_i \tag{3.8}$$

Bei dieser Rekombinationsmethode werden aus mehreren Elternteilen nur ein Nachfahre gebildet. Würde man für den weiteren Verlauf des evolutionären Algorithmus nur diesen beibehalten, würde also die Population schrumpfen. Deshalb wird bei dieser Methode eine Art Wettkampf-Selektion durchgeführt, so dass von den involvierten Genome alle außer das schlechteste überleben.

Mutation

Die letzte Veränderung innerhalb einer Generation stellt die Mutation dar. Hierbei wird eine gewisser Anteil p_m zufällig ausgewählter Genome durch ein Genom in seiner Nachbarschaft ersetzt. Wie diese Nachbarschaft aussieht, ist, wie beim simulated-annealing, nicht genau festgelegt und sollte problemspezifisch gewählt werden. Eine Möglichkeit würde darin bestehen, jedes Chromosom geringfügig zu verändern. Eine andere Methode wäre es, zufällig ein Chromosom auszuwählen und diesem einen komplett zufälligen Wert zuzuordnen.
Die Mutation dient nicht vorderrangig dazu, Genome mit einem geringeren Fehlerwert zu erzeugen. Vielmehr soll sie eine gewisse Vielfalt in die Population bringen und ermöglichen, dass lokale Minima verlassen werden.

Konfiguration

Offensichtlich bieten evolutionäre Algorithmen eine große Auswahl an Konfigurationen. Man kann unterschiedliche Selektions- und Rekombinationsverfahren benutzen, sowie verschiedene Populationsgrößen und Rekombinations- und Mutationsraten p_c und p_m. Dies führt einerseits dazu, dass ein evolutionärer Algorithmus sehr gut auf ein spezifisches Optimierungsproblem angepasst werden kann, andererseits erscheint die Suche nach der richtigen Konfiguration an sich wie ein eigenes Kalibrierungsproblem. Vor allem die Wahl von p_c und p_m ist sehr wichtig. Einerseits sollen sie groß genug gewählt werden, um einen möglichst großen Bereich des Parameterraums abzudecken. Andererseits sollen sie auch nicht zu groß sein, damit der Algorithmus noch immer

effizient arbeitet und nicht in eine zufällige Suche ausartet. Dementsprechend wurden schon mehrere Arbeiten in dieser Richtung verfasst. Unter anderem von *Grefenstette* [11], in der ein evolutionärer Algorithmus auf einer Metaebene agiert, um die richtige Konfiguration zu bestimmen. Ein anderer Ansatz wird von *Sriniavas und Patnaik* [25] benutzt. Hier sollen gute Rekombinations- und Mutationsraten adaptiv erarbeitet werden.

Konvergenz

Bisher existiert eine Konvergenzaussage nur für den genetischen Algorithmus von *Holland*, also für den Fall, dass die Parameter binär kodiert sind. Außerdem wird von einer fitnessbasierten Selektion, einem 1-point-crossover und einer bitweisen Mutation ausgegangen. Der Schemasatz [13] besagt, dass sich gewisse Genome mit einer überdurchnittlichen Fitness mit einer großen Wahrscheinlichkeit in den nächsten Generationen durchsetzen werden.
Um dies zu zeigen wird der Begriff des *Schemas* eingeführt. Die einzelnen Parameter werden durch Bitfolgen, bestehend aus 0 und 1 dargestellt. Für den Beweis werden N Genome der Länge l betrachtet. Bei einem Schema handelt es sich nun um eine Familie von Bitfolgen, bei der einzelne Bits fest sind. Als Beispiel dient das Schema **1*01*. Das Symbol * steht für einen beliebigen Wert. Demzufolge sind 1011011 und 0010010 Genome, die zu diesem Schema gehören. Des Weiteren werden noch die Begriffe *Durchmesser* und *Ordnung* eines Schemas benötigt. Der Durchmesser $\delta(H)$ eines Schema H berechnet sich durch den Abstand zwischen dem ersten und dem letzten fixen Bit eines Schemas. Die Ordnung $o(H)$ gibt an, aus wie vielen fixen Bits das Schema besteht. Bei dem oberen Beispiel **1*01* gilt demgemäß $\delta(H) = 4$ und $o(H) = 3$.
Um zu berechnen, wie hoch die Wahrscheinlichkeit ist, ob das Schema H in der Generation $t + 1$ noch vertreten ist, müssen die Wahrscheinlichkeiten berechnet werden, dass das Schema H die Selektion, Rekombination und Mutation innerhalb der Generation t überlebt.
Bei einer fitnessbasierten Selektion berechnet sich die Überlebenswahrscheinlichkeit folgendermaßen:

$$\begin{aligned} P_{sel} &= f(H_1) + f(H_2) + \ldots + f(H_{m(H,t)}) \\ &= m(h,t) f(H) \end{aligned} \tag{3.9}$$

Dabei bezeichnet $m(H,t)$ die Anzahl der Genome in der Generation t, die zum Schema H gehören und $f(H)$ bezeichnet die Fitness des Schemas H.
Anschließend wird die Überlebenswahrscheinlichkeit für die Rekombination betrachtet. Diese hängt von dem Aussehen der beiden Elternteile ab. Gehören beide Elternteile zum Schema H, so haben auch ihre Nachfahren diese Form. Ansonsten muss die

Wahrscheinlichkeit betrachtet werden, dass das Schema zerschnitten wird:

$$P_{cut} = \frac{\delta(H) - 1}{l - 1} \tag{3.10}$$

Also ergibt sich:

$$P_{cross} \geq P_{sel}^2 + P_{sel}(1 - P_{sel})(1 - P_{cut}) \tag{3.11}$$

Die Ungleichung entsteht, da nicht ausgeschlossen werden kann, dass der zweite Elternteil zufällig die richtige Bitkonfiguration besitzt, um das durchtrennte Schema wiederherzustellen.
Die Überlebenswahrscheinlichkeit für die Mutation ist nun abhängig von der Mutationswahrscheinlichkeit p_m und der Ordnung des Schemas. Sie hat folgende Form:

$$P_{mut} = (1 - p_m)^{o(H)} \tag{3.12}$$

Schlussendlich ergibt sich die Wahrscheinlichkeit, dass das Schema H in der nächsten Generation vertreten ist:

$$\begin{aligned} W &\geq P_{cross} \cdot P_{mut} \\ &\geq (m(h,t)f(H))^2 + m(h,t)f(H)(1 - m(h,t)f(H)) \left(1 - \frac{\delta(H) - 1}{l - 1}\right) (1 - p_m)^{o(H)} \end{aligned} \tag{3.13}$$

Man erkennt, dass sich vor allem Schemata mit einem kleinen Durchmesser und einer niedrigen Ordnung sowie einer überdurchschnittlichen Fitness durchsetzen. Es wird jedoch keine Aussage darüber getroffen, ob es sich bei einem Schema dieser Art um ein globales oder nur um ein lokales Minumum handelt.

4 Anwendung auf Beispielmodelle

4.1 Erklärung der Beispielmodelle

Bevor die in Kapitel 3 besprochenen Kalibrierungsalgorithmen auf das richtige Influenzamodell angewendet werden, werden sie zuerst anhand von zwei weniger komplexen agentenbasierten Modellen getestet.

Zunächst betrachten wir ein einfaches SIR - Modell. In diesem können sich die einzelnen Agenten in verschiedenen Räumen aufhalten. Die Anzahl der Agenten und der Räume ist jeweils festgelegt und kann sich innerhalb einer Simulation nicht verändern. Außerdem ist die Zahl der anfänglich infizierten Agenten gegeben.
Beim Start der Simulation werden die Agenten zufällig auf die verschiedenen Räume verteilt. Innerhalb jedes Zeitschrittes werden zunächst die möglichen Infektionen betrachtet. Eine Krankheitsübertragung kann genau dann stattfinden, wenn sich suszeptible und infizierte Agenten gemeinsam in einem Raum befinden. Wenn sich dabei ein Agent im Zustand S mit genau einem Agenten im Zustand I den Raum teilt, wird sich dieser Agent mit der Wahrscheinlichkeit α infizieren. Demzufolge steckt sich ein S-Agent mit einer Wahrscheinlichkeit von $1-(1-\alpha)^n$ mit der Krankheit an, wenn sich n I-Agenten im Raum befinden. Anschließend werden die möglichen Genesungen behandelt. Diese erfolgen jeweils mit einer gewissen Wahrscheinlichkeit β, unabhängig davon, wie viele Zeitschritte sich ein Agent schon im Zustand I befindet. Nachdem auch diese Zustandsänderungen erfolgt sind, werden die Agenten erneut per Zufall auf die einzelnen Räume verteilt und der nächste Zeitschritt beginnt.
In diesem Beispielmodell gibt es also zwei zu kalibrierende Parameter: die Infektionswahrscheinlichkeit α und die Genesungswahrscheinlichkeit β.

Um zu untersuchen, wie sich die Anzahl der zu kalibrierenden Parameter auf die Resultate auswirkt, wird dieses Modell anschließend auch mit einem weiteren Parameter aufgerufen. Diesmal ist auch die Anzahl anfänglich infizierter Personen I_0 variabel.

Als zweites Beispiel wird ein $SIRS$ - Modell betrachtet. Die Grundidee und Implementierung ist analog zum oberen SIR - Modell. Wieder wird eine festgelegte Anzahl Agenten auf Räume verteilt, in denen sich die suszeptiblen Agenten bei den infizierten anstecken können. Der Unterschied besteht allerdings darin, dass ein genesener Agent nicht während der restlichen Simulation in diesem Zustand bleiben muss, sondern nach d Zeitschritten wieder in den Zustand S wechselt. Dies führt dazu, dass die

Krankheit sich innerhalb einer Simulation in mehreren Wellen ausbreiten kann. Auch bei diesem Modell ist die Anzahl der Agenten, die zum Startzeitpunkt infiziert sind, variabel. Insgesamt müssen beim SIRS-Modell also vier Parameter berechnet werden: die Infektions- und Genesungswahrscheinlichkeiten α und β, sowie die Anzahl d an Tagen, während denen ein Agent resistent ist, und die Anzahl I_0 anfänglich Infizierter.

Während der Simulation eines dieser Beispielmodelle wird in jedem Zeitschritt die Anzahl suszeptibler, infizierter und resistenter Agenten abgespeichert. Diese Werte bilden die Simulationsergebnisse. Ein typisches Resultat eines Durchlaufs des SIR- und des $SIRS$-Modells ist in Abbildung 4.1 zu sehen.

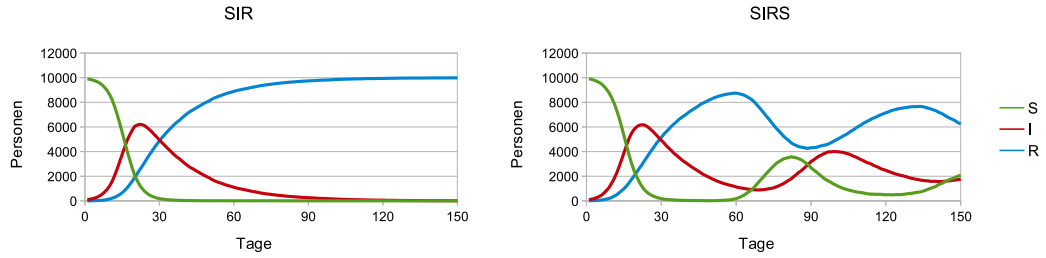

Abbildung 4.1: Resultate einer Simulation des SIR und des $SIRS$-Modells

4.2 Allgemeine Vorbereitungen zur Kalibrierung

4.2.1 Zusammenspiel Modell - Kalibrierungsalgorithmen

Das Ziel dieser Diplomarbeit besteht nicht nur darin, die Eignung verschiedener Algorithmen zur Kalibrierung agentenbasierter Modelle zu testen, sondern auch diese Kalibrierung möglichst unabhängig vom betrachteten Modell zu gestalten. Zu diesem Zweck wurde bei der Implementierung der Kalibrierungsalgorithmen in Java ein Framework geschaffen, das sich aus verschiedenen Bauteilen zusammensetzt, die unabhängig voneinander ausgetauscht werden können. Es besteht aus drei abstrakten Klassen: dem zu kalibrierenden Modell, dem Optimierungsalgorithmus und der Fehlerfunktion. Jede dieser Klassen besitzt eine Reihe von Variablen und Methoden, die das Zusammenspiel mit den anderen Klassen ermöglichen. Eine schematische Darstellung dieses Zusammenspiel ist in Abbildung 4.2 dargestellt.

Die Modellklasse enthält eine Methode *run(parameters)*. Sie ruft die Simulation mit den angegebenen Parameterwerten auf und gibt die dazugehörenden Resultate aus. Die übrigen Methoden *getRefSys()*, *getParaRange()* und *isIntiger()* dienen dazu, den anderen Klassen des Kalibrierungsframeworks wichtige Informationen zu übergeben. Die Klasse der Fehlerfunktionen besitzt eine Variable und eine Methode. Die Variable ist ein Element der Modellklasse und enthält das Modell, das kalibriert werden soll. Die

Methode *calculateError(parameters)* berechnet den Fehler, der zu den angegebenen Parameterwerten gehört. Dazu ruft sie das Modell mit den Parameterwerten auf und berechnet den Abstand der Simulationsergebnisse zum Referenzsystem. Die Daten des Referenzsystems erhält die Fehlerfunktion dabei mithilfe der Methode *getRefSys()* der Modellklasse.
Schlussendlich gibt es noch die Klasse der Kalibrierungsalgorithmen. Sie enthält drei Variablen: das zu kalibrierende Modell, die benutzte Fehlerfunktion und eine Liste von Elementen der Klasse *Point*. Letztere dient dazu, die Parameterwerte und den dazugehörenden Fehlerwert abzuspeichern. Den wichtigsten Bestandteil der abstrakten Klasse der Kalibrierungsalgorithmen bildet die Methode *run()*. Sie enthält den Code des benutzten Optimierungsalgorithmus und gibt das berechnete optimale Parameterset aus. Innerhalb dieser Methode werden die Parameterwerte variiert und es wird entschieden, welche Parametersets als nächstes ausgewertet werden sollen. Damit nur zulässige Parameterwerte an das Modell übergeben werden, muss bekannt sein, in welchem Rahmen diese Parametervariationen erlaubt sind. Der Optimierungsalgorithmus muss wissen, in welchem Bereich sich die Werte befinden dürfen und ob es sonst irgendwelche Einschränkungen gibt. Ein Beispiel dafür ist das oben beschriebene $SIRS$-Modell. Hier dürfen die Parameter d und I_0 nur ganzzahlige Werte annehmen. Diese Informationen erhält der Kalibrierungsalgorithmus vom betrachteten Modell mithilfe der Methode *getParaRange()* und *isIntiger()*. Wurde ein zu testendes Parameterset bestimmt, wird dieses an die Liste *points* übergeben und ein neues Element der Klasse *Point* wird erschaffen. Dabei wird zuerst überprüft, ob die angegebenen Parameterwerte zulässig sind, das heißt ob sie im Parameterbereich liegen und, falls dies gefordert ist, ganzzahlig sind. Ist eine dieser Bedingungen nicht erfüllt, wird der betroffene Parameter angepasst (durch das Minimum oder Maximum des Parameterbereiches ersetzt bzw. gerundet) und die neuen, zulässigen Werte werden gespeichert. Anschließend wird der Fehlerwert dieses Parametersets mithilfe der Methode *calculateError(parameters)* der Fehlerfunktion berechnet und ebenfalls abgespeichert.

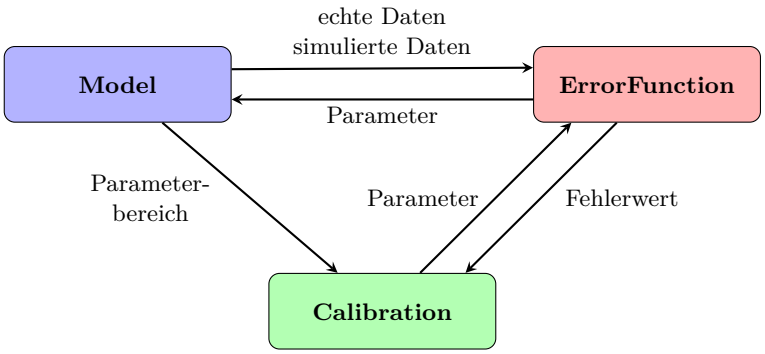

Abbildung 4.2: Zusammenspiel abstrakter Klassen

Als nächster Vorbereitungsschritt für die Kalibrierung müssen die einzelnen Elemente des Frameworks festgelegt werden.
Als Modell werden die im vorherigen Kapitel vorgestellten Beispielmodelle verwendet. Hier stellt sich allerdings die Frage, wie das Referenzsystem definiert wird. In den meisten Fällen wird das zu reproduzierende Referenzsystem durch reelle Daten gegeben. Bei den oben beschriebenen Beispielmodellen ist dies offensichtlich nicht möglich. Stattdessen werden im Voraus Parameterwerte festgelegt, mit denen das Modell aufgerufen wird. Die Ergebniskurve, die die Anzahl an Infizierten am jeweiligen Tag darstellt, wird dann als Referenzkurve gewählt. Um sie von den zufälligen Schwankungen der Simulationsresultate zu bereinigen, wird das Modell Zwanzig mal mit den festgelegten Parameterwerten aufgerufen und es wird über die erhaltenen Ergebnisse gemittelt.

Für die Fehlerfunktion wird ein gewichteter euklidischer Abstand der Form 2.2 gewählt. Dabei wird die Gewichtsfunktion abhängig von der zu reproduzierenden Datenkurve gewählt. Damit vor allem die charakteristische Form der Referenzkurve möglichst genau wiedergegeben wird, werden die lokalen Extrema besonders hoch gewichtet. Für die folgenden Kalibrierungen der Beispielmodelle wurde sich für eine stückweise lineare Gewichtsfunktion entschieden. Dabei erhalten die Punkte, an denen die Referenzkurve ein lokales Extrema hat, den Wert 40 und die Punkte in der Mitte zwischen zwei lokalen Extrema den Wert 1. Die Gewichte der übrigen Punkte werden so berechnet, dass sich insgesamt eine stückweise lineare Funktion ergibt:

$$g(x)|_{[e_i,e_{i+1}]} = 39 \cdot \left| \frac{2(x-e_i)}{e_{i+1}-e_i} - 1 \right| + 1 \qquad \text{mit } e_j \ldots \text{j-tes lokales Extremum} \quad (4.1)$$

4.2.2 Konfiguration der Kalibrierungsalgorithmen

In Kapitel 3 wurde angegeben, dass sowohl das Simulated Annealing als auch der evolutionäre Algorithmus eine große Vielfalt an möglichen Konfigurationen besitzen. Nun wird festgelegt, welche Konfiguration in den folgenden Kapiteln benutzt wird, um eine Kalibrierung der Beispielmodelle durchzuführen.

Für das Simulated Annealing müssen die Wahl der Nachbarschaft, das Abkühlungsverfahren, die Starttemperatur und die Abbruchkriterien definiert werden.
Für die Wahl der Nachbarschaft wird folgendermaßen vorgegangen: Der Parameterbereich wird in jeder Dimension in eine feste Anzahl von Bereichen eingeteilt. Anschließend wird für jede Dimension die dazu passende Schrittweite berechnet. Um einen Nachbarn eines Punktes zu berechnen, wird jetzt für jede Dimension bestimmt, ob und in welche Richtung ein Schritt zurückgelegt wird. Dementsprechend besteht die Nachbarschaft eines Punktes x aus allen Punkten y, die von x in jeder Dimension maximal einen Schritt entfernt sind. Also bedeutet eine große Schrittanzahl, dass benachbarte Punkte sehr nahe beieinander liegen, während sie bei einer niedrigen Schrittanzahl weiter entfernt sind. Die optimale Wahl der Schrittanzahl wird in den

folgenden Kapiteln experimentell bestimmt.
Als Abkühlungsverfahren wird die geometrische Abkühlung gewählt. Das heißt, es existiert ein Kühlungsfaktor γ, sodass die Temperatur T nach einer festgelegten Anzahl an Schritten auf den Wert $\gamma \cdot T$ gesenkt wird. In den folgenden Anwendungen des Simulated Annealing wird die Temperatur nach jeweils 10 Schleifendurchläufen verändert. Der geeignete Kühlungsfaktor wird experimentell bestimmt.
Um eine geeignete Anfangstemperatur zu bestimmen, wird sich an der in Kapitel 3 erwähnten Konvergenzaussage von *Hajek* [12] orientiert. Diese benutzt eine Starttemperatur c, die vom Aussehen der lokalen Minima der Fehlerfunktion abhängt. Vereinfacht ausgedrückt muss c größer sein als ein Wert $d*$, der beschreibt, wie „tief“ die Täler der lokalen Minima sind. Genauer gesagt, bezeichnet $d*$ den Wert, um den der Fehlerwert $F(x)$ eines lokalen Minimums x erhöht werden muss, damit eine Folge von benachbarten Punkten $x = x_0, x_1, \ldots x_p = y$ existiert, sodass $F(y) < F(x)$ und $F(x_k) < F(x)+d*$ für alle k. Dementsprechend wäre der Wert des globalen Maximums der Fehlerfunktion auf jeden Fall eine geeignete Anfangstemperatur. Dieser lässt sich allerdings genauso schwer bestimmen, wie das globale Minimum. Als Kompromiss wird das zu kalibrierende Modell mehrfach mit zufälligen Parameterwerten ausgeführt und eine obere Schranke der erhaltenen Fehlerwerte als Starttemperatur gewählt.
Schlussendlich werden folgende Abbruchkriterien gewählt:

- der berechnete Fehler ist klein genug;
- der aktuelle Fehlerwert hat sich während einer festgelegten Anzahl an Schleifendurchgängen nicht verbessert;
- es wurde eine maximale Anzahl an Schleifendurchgängen berechnet.

Um zu bestimmen, ab wann ein Fehlerwert als „klein genug“ gilt, wird das Modell mit den Parameterwerten aufgerufen, die zur Erstellung der Referenzkurve herangezogen wurden. Der berechnete Fehlerwert ergibt dann die Schranke, unterhalb der ein Fehler akzeptiert wird.

Die Konfigurationen, die beim evolutionären Algorithmus definiert werden müssen, sind die Selektions-, Rekombinations- und Mutationsmethoden sowie die Abbruchkriterien.
Als Selektionsverfahren wird die rangbasierte Fitness fit_{rang} benutzt. Sie hat den Vorteil, dass die einzelnen Fitnesswerte nur von der Populationsgröße abhängen und so im Laufe der Kalibrierung nicht immer neu berechnet werden müssen. Man muss lediglich die Genome nach ihrem Fehlerwert ordnen und kann ihnen dann den richtigen Fitnesswert zuweisen.
Da bei den Beispielmodellen die Parameterwerte in Form von reellen Zahlen vorliegen, wird der euklidische Mittelwert aus zwei Elterngenomen als Rekombinationsmethode gewählt. Bei einer Population von n Genomen wird dies $p_c \cdot n$ mal durchgeführt. Um ressourcenintensive Vergleiche des Fehlerwertes der beiden Eltern und des neuen Genoms zu vermeiden, werden die neuen Genome einfach zur Population hinzugefügt.
Als abschließende Mutation werden $p_m \cdot n$ Genome zufällig ausgewählt und neue Genome generiert, indem einem Chromosom ein neuer Zufallswert zugewiesen wird.

Auch diese neuen Genome werden der Population hinzugefügt.
Wenn alle Rekombinationen und Mutationen berechnet worden sind, wird die Population wieder nach ihrem Fehlerwert geordnet und die n besten Genome werden zurückbehalten, so dass die ursprüngliche Populationsgröße wieder hergestellt ist. Der Unterschied zur üblichen Wettkampf-Selektion besteht darin, dass es bei der hier verwendeten Methode passieren kann, dass nicht immer genau zwei von den in einer Rekombination involvierten Genome überleben. Außerdem wird bei einer Mutation das ursprüngliche Genom nicht zerstört. Auf diese Weise ist sichergestellt, dass der Fehlerwert nach einer abgeschlossenen Generation nicht größer ist als vorher.
Die geeigneten Werte für die Populationsgröße n sowie die Rekombinations- und Mutationsraten p_c und p_m werden experimentell bestimmt.
Beim evolutionären Algorithmus werden nur zwei Abbruchkriterien gewählt. Die Kalibrierung wird beendet, falls der berechnete Fehlerwert kleiner als der akzeptierte Fehlerwert ist oder falls eine maximale Anzahl an Modellaufrufen durchgeführt wurde. Eine Stagnation der Kalibrierung, die beim Simulated Annealing auch ein Abbruchgrund ist, wird hier nicht berücksichtigt, da es durch die Mutation theoretisch jederzeit möglich ist, ein lokales Minimum zu verlassen.

4.3 Kalibrierung des SIR-Modells

4.3.1 Vorbereitungen

Als Vorbereitung zur Kalibrierung des SIR-Modells muss zuerst das Referenzsystem festgelegt werden. Es wurde sich für einen Simulationszeitraum von 150 Tagen und für folgende Parameterwerte entschieden: $\alpha = 0.4$, $\beta = 0.05$ und $I_0 = 100$. Außerdem muss für jeden Parameter ein Wertebereich definiert werden, innerhalb dem sich die zulässigen Werte befinden. Da es sich bei α und β um eine Wahrscheinlichkeit handelt, wird das Intervall $[0, 1]$ als Parameterbereich gewählt. Der Parameter I_0 stellt eine Anzahl an Agenten dar. Dieser kann alle Werte annehmen, die kleiner sind als die Anzahl an Agenten, die für die Simulation benutzt werden. Bei diesem Modell wurde sich dafür entschieden, 10 000 Agenten zu verwenden. Diese Zahl ist so hoch gewählt, um der in Kapitel 2.3 erwähnten Streuung der Resultate entgegenzuwirken. Als zusätzliche Maßnahme wird das Modell bei jeder Fehlerauswertung zweimal aufgerufen und der Mittelwert der Ergebnisse zur Fehlerberechnung herangezogen. Trotzdem ist noch immer eine kleine Streuung der Resultate vorhanden. Berechnet man den Fehler, der entsteht, wenn das Modell mit den Parameterwerten des Referenzsystems aufgerufen wird, erhält man einen mittleren Wert von 20. Dies bildet den oben erwähnten akzeptierten Fehlerwert des SIR-Modells. Das heißt, dass die Kalibrierung frühzeitig abgebrochen wird, falls ein Parameterset gefunden wird, dessen Fehler kleiner als 20 ist.
Abschließend muss noch die für das Simulated Annealing notwendige Starttemperatur bestimmt werden. Bei Zwanzig maliger Berechnung des Fehlerwert von zufälligen

Parameterwerten ergibt sich eine obere Schranke von 1 600 für das SIR-Modell mit zwei Parametern und 1 800 für das mit drei Parametern. Die Epidemiekurve sowie die mittels (4.1) berechneten Gewichte der Fehlerfunktion sind in Abbildung 4.3 zu sehen.

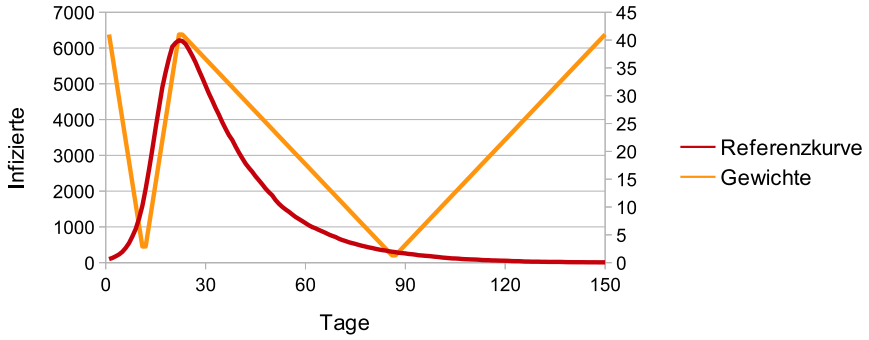

Abbildung 4.3: Epidemiekurve des SIR-Modells und zugehörige Gewichtsfunktion

4.3.2 Kalibrierung

In den folgenden Tests wird für jede Kalibrierung eine maximale Anzahl an Modellaufrufen von 1 000 Simulationen veranschlagt. Verbessert sich der Fehlerwert innerhalb 200 Modellaufrufen nicht, wird beim Simulated Annealing die Kalibrierung wegen einer Stagnation abgebrochen. Für jede Konfiguration der Optimierungsalgorithmen wird die Kalibrierung Zehn Mal berechnet. Zur Bewertung der Ergebnisse werden für jede Testreihe der Mittelwert und die Standardabweichung des Fehlers am Ende der Kalibrierung betrachtet.

Simulated Annealing

In den folgenden Testreihen wird das Simulated Annealing benutzt um das SIR-Modell zu kalibrieren. Dabei sollen geeignete Werte für die Schrittanzahl und den Kühlungsfaktor γ experimentell bestimmt werden. Für die Schrittanzahl werden die Werte $10, 20, 30, \ldots 70$ und für den Kühlungsfaktor die Werte $0.8, 0.85, 0.9, 0.95$ getestet.

Zur besseren Bewertung der Kalibrierungsergebnisse werden die einzelnen Durchläufe in verschiedene Kategorien eingeteilt, die in Tabelle 4.1 beschrieben werden. Als angemessen kleinen Fehlerwert wird für das SIR-Modell mit zwei Parametern ein Wert von 50 angenommen.

Der Mittelwert und die Standardabweichung des Fehlers werden in Abbildung 4.4 gezeigt. Tabelle 4.2 enthält den Anteil an Kalibrierungen, die in die Kategorien aus

erfolgreich	die Kalibrierung liefert einen angemessen kleinen Fehlerwert
lokales Minimum	die Kalibrierung wird frühzeitig abgebrochen, da der Algorithmus stagnierte
nicht fertig	die 1 000 Schleifendurchgänge haben nicht ausgereicht, um ein globales oder lokales Minimum zu berechnen
nicht angeschlagen	der erhaltene Fehlerwert liegt in der Nähe der Starttemperatur, demnach liefert das Ergebnis der Kalibrierung keinen besseren Fehlerwert als ein zufälliges Parameterset

Tabelle 4.1: Bewertungskategorien der Kalibrierungsdurchläufe

Tabelle 4.1 fallen.

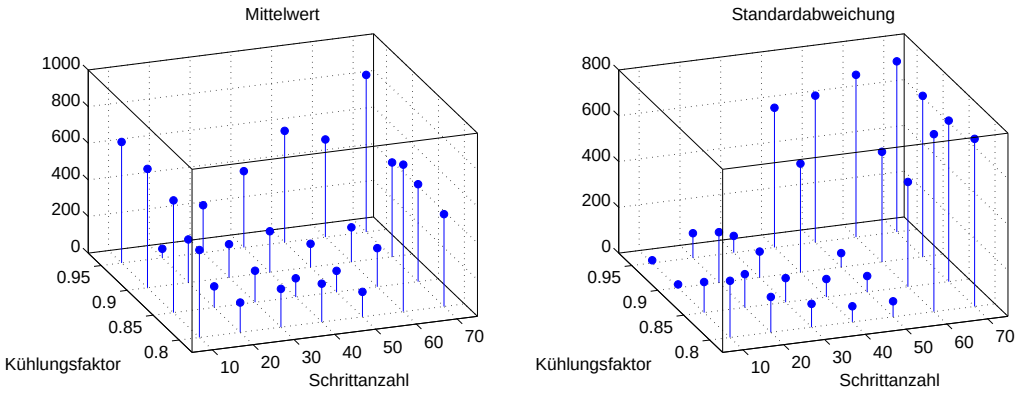

Abbildung 4.4: Mittelwert und Standardabweichung des Fehlers der Kalibrierung des SIR-Modells mittels Simulated Annealing

Anhand der Ergebnisse sind zwei Phänomene erkennbar. Zum einen wird bei einer kleinen Schrittanzahl häufig nur ein lokales Minimum berechnet. Dies liegt daran, dass das Gitter auf dem die Nachbarn berechnet werden zu grob ist. Der Algorithmus kann somit gar keinen Punkt finden, der nah genug am globalen Minimum $\alpha = 0.4$ und $\beta = 0.05$ liegt. (siehe Tabelle 4.3)

Das zweite Problem besteht darin, dass die 1000 Schleifendurchläufe oft nicht ausreichen, um das Modell vollständig zu kalibrieren. Dies ist vor allem bei einer langsamen Kühlung ($\gamma = 0.95$) und einer großen Schrittanzahl der Fall. Ein Grund hierfür ist vermutlich, dass die Schrittweite so klein ist, dass der Algorithmus nicht genug Zeit hat, sich von einem schlechten Anfangswert fortzubewegen und in die Nähe des Optimums zu gelangen. Natürlich könnte man die erlaubte Schleifenanzahl erhöhen und so den erhaltenen Fehler verkleinern. Allerdings spielt vor allem bei komplexeren Modellen deren Laufzeit eine große Rolle. Deshalb ist es wünschenswert, eine Möglichkeit zu finden, die Leistung des Kalibrierungsalgorithmus auf eine andere Weise zu verbessern.

Ein Mittel diese Probleme zu lösen, ist es die Schrittanzahl progressiv zu erhöhen.

Kühlungsfaktor	Schrittanzahl 10	20	30	40	50	60	70
0.8	0.1	0.4	0.1	0.5	0.1	0.3	0.4
	0.8	0.5	0.6	0.4	0.2	0.0	0.1
	0.1	0.1	0.3	0.1	0.7	0.2	0.2
	0.0	0.0	0.0	0.0	0.0	0.5	0.3
0.85	0.0	0.6	0.2	0.4	0.2	0.4	0.1
	0.7	0.3	0.3	0.2	0.1	0.1	0.4
	0.3	0.1	0.5	0.4	0.7	0.4	0.2
	0.0	0.0	0.0	0.0	0.0	0.1	0.3
0.9	0.0	0.4	0.2	0.5	0.1	0.6	0.4
	0.4	0.3	0.3	0.3	0.3	0.0	0.1
	0.6	0.3	0.5	0.1	0.6	0.3	0.2
	0.0	0.0	0.0	0.1	0.0	0.1	0.3
0.95	0.0	0.9	0.0	0.3	0.0	0.1	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	1.0	0.1	1.0	0.5	0.7	0.6	0.5
	0.0	0.0	0.0	0.2	0.3	0.3	0.5

erfolgreich | lokales Min. | nicht fertig | nicht angeschl.

Tabelle 4.2: Anteil an Kalibrierungen des SIR-Modell mittels Simulated Annealing, die in die Kategorien aus Tabelle 4.1 fallen

Durchlauf	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Fehler	652	656	651	651	653	655	652	659	656	608
α	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
β	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.09

Tabelle 4.3: Berechnete Parameterwerte des Simulated Annealing bei einer Schrittanzahl von 10 und einem Kühlungsfaktor von 0.9

Auf diese Weise werden zu Beginn der Kalibrierung große Schritte zurückgelegt, um in die Nähe der optimalen Parameterwerte zu gelangen. Anschließend wird mit immer kleiner werdenden Schritten versucht, sich diesen so genau wie möglich anzunähern. Die Schrittanzahl wird dementsprechend nach jeweils 50 Schleifendurchgängen mit einem Faktor multipliziert. Damit sich der Algorithmus nicht unnötigerweise festfrisst, wird eine obere Schranke für die Schrittanzahl eingeführt: ab 100 Schritten wird die Anzahl nicht weiter erhöht. Getestet werden die Kombinationen aus einer anfänglichen Schrittanzahl von 5, 10, 15 und 20 mit den Faktoren $1.1, 1.2, \ldots, 1.5$. Für den Kühlungsfaktor wird bei allen Durchläufen der Wert 0.85 gewählt. Die Ergebnisse befinden sich in Abbildung 4.5 und Tabelle 4.4.

Dieser Ansatz hat zu keiner relevanten Verbesserung der Ergebnisse geführt. Man erkennt, dass der erhaltene Fehler kleiner ist, falls der Vergrößerungsfaktor klein gewählt wird. Jedoch ist auch bei den besten Ergebnissen keine signifikanter Fortschritt gegenüber der vorherigen Vorgehensweise zu erkennen.

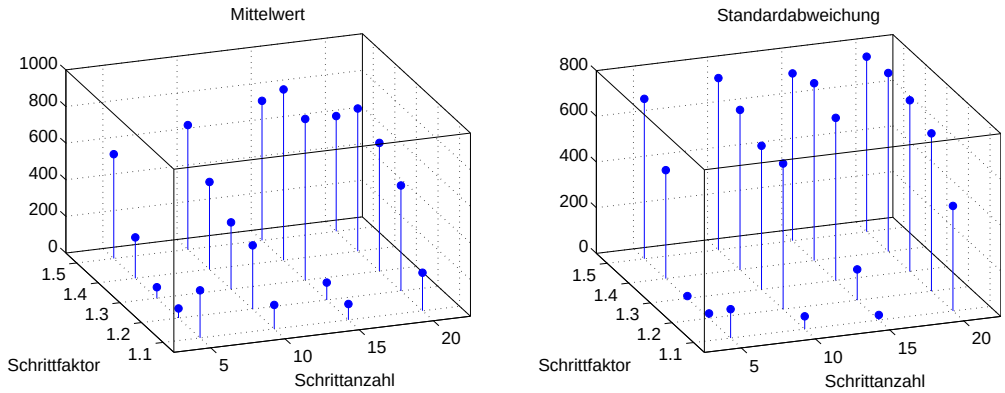

Abbildung 4.5: Mittelwert und Standardabweichung des Fehlers der Kalibrierung des SIR-Modells mittels Simulated Annealing bei progressiv erhöhter Schrittanzahl

Schrittfaktor	Schrittanzahl 5	10	15	20
1.1	0.0	0.2	0.1	0.3
	0.3	0.0	0.2	0.1
	0.7	0.8	0.7	0.5
	0.0	0.0	0.0	0.1
1.2	0.3	0.5	0.4	0.2
	0.0	0.1	0.2	0.2
	0.7	0.2	0.4	0.3
	0.0	0.2	0.0	0.3
1.3	0.2	0.0	0.1	0.3
	0.2	0.3	0.1	0.3
	0.6	0.5	0.3	0.0
	0.0	0.2	0.5	0.4
1.4	0.3	0.4	0.3	0.4
	0.1	0.2	0.1	0.1
	0.5	0.1	0.0	0.0
	0.1	0.3	0.6	0.5
1.5	0.2	0.4	0.3	0.4
	0.1	0.1	0.0	0.1
	0.4	0.1	0.3	0.1
	0.3	0.4	0.4	0.4

erfolgreich	lokales Min.	nicht fertig	nicht angeschl.

Tabelle 4.4: Anteil an Kalibrierungen des SIR-Modell mittels Simulated Annealing, die in die Kategorien aus Tabelle 4.1 fallen, unter Verwendung einer progressiv erhöhter Schrittanzahl

Eine weitere Möglichkeit die Ergebnisse des Simulated Annealing zu verbessern, ist eine andere Art der Nachbarschaft zu wählen. Bis jetzt wurde praktisch ein Gitter über den Parameterraum gelegt und als mögliche Nachbarn kamen nur Gitterpunkte in Frage. Bei der neuen Herangehensweise wird für die Berechnung eines Nachbarn von einem Punkt x für jede Dimension eine Zufallszahl bestimmt. Diese Zufallszahl ist normalverteilt, wobei der Mittelwert dem Parameterwert des Punktes x und die Standardabweichung der Schrittweite in der jeweiligen Dimension entspricht. Auch hier werden wieder verschiedene Kombinationen von Schrittanzahl und Kühlungsfaktor ausgetestet. Die Ergebnisse befinden sich in Abbildung 4.6 und Tabelle 4.5.

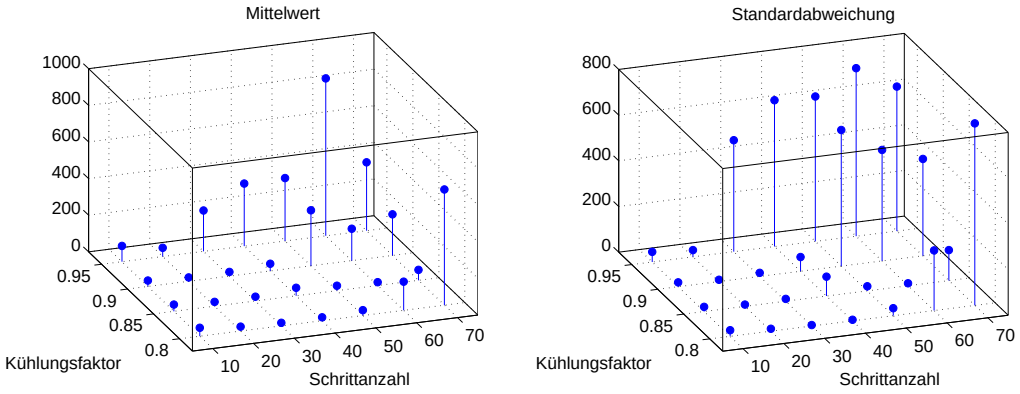

Abbildung 4.6: Mittelwert und Standardabweichung des Fehlers der Kalibrierung des SIR-Modells mittels Simulated Annealing bei einer standardnormalverteilten Nachbarschaft

Die Wahl einer standardnormalverteilten Nachbarschaft liefert deutlich bessere Resultate als die gitterbasierte Nachbarschaft. Vor allem für eine Schrittanzahl zwischen 20 und 50 produziert die Kalibrierung fast optimale Ergebnisse. Zu bemerken ist, dass auch hier höhere Werte des Kühlungsfaktors dazu führen, dass die Kalibrierung nicht in der geforderten Zeit fertig berechnet wurde und sich somit schlechtere Resultate ergeben. Abbildung 4.7 zeigt die Ergebniskurven von einer Kalibrierungskonfiguration verglichen mit der zu reproduzierenden Referenzkurve.

In einem nächsten Schritt werden diese Erkenntnisse benutzt, um das Modell mit einem zusätzlichen Parameter zu kalibrieren. Die Kalibrierungen werden mit einer Schrittanzahl von $10, 20, 30, 40, 50$ und einem Kühlungsfaktor von $0.8, 0.85, 0.9$ durchgeführt. Da davon auszugehen ist, dass die Bestimmung von drei Parametern länger dauert, wird erst ab 400 Schleifendurchgängen ohne Verbesserung von einer Stagnation ausgegangen und die Kalibrierung wird abgebrochen. Die Ergebnisse befinden sich in Abbildung 4.8 und Tabelle 4.6. Dabei wurde eine Kalibrierung als erfolgreich gewertet, wenn der erhaltene Fehler kleiner als 75 ist.

Es zeigt sich, dass der Erfolg der Kalibrierung sehr stark von der gewählten Schrittanzahl abhängt. Bei einer kleinen Schrittanzahl wird oft nur ein lokales Minimum errechnet, während bei einer großen Schrittanzahl ungefähr die Hälfte der Kalibrierungen nicht anschlagen und einen Fehler größer als 1000 liefern. Des Weiteren fällt

Kühlungsfaktor	Schrittanzahl 10	20	30	40	50	60	70
0.8	0.7	0.9	1.0	1.0	0.9	0.6	0.6
	0.3	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.1	0.0	0.0	0.1	0.4	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.4
0.85	0.7	1.0	1.0	0.9	0.9	1.0	0.9
	0.2	0.0	0.0	0.0	0.0	0.0	0.1
	0.1	0.0	0.0	0.1	0.1	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.9	1.0	0.9	0.9	0.7	0.9	0.7
	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.1	0.0	0.1	0.1	0.1	0.0	0.2
	0.0	0.0	0.0	0.0	0.2	0.1	0.1
0.95	0.1	0.6	0.7	0.6	0.6	0.2	0.6
	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.9	0.4	0.2	0.2	0.2	0.3	0.2
	0.0	0.0	0.1	0.2	0.2	0.5	0.2

erfolgreich	lokales Min.	nicht fertig	nicht angeschl.

Tabelle 4.5: Anteil an Kalibrierungen des SIR-Modell mittels Simulated Annealing, die in die Kategorien aus Tabelle 4.1 fallen, unter Verwendung einer normalverteilten Nachbarschaft

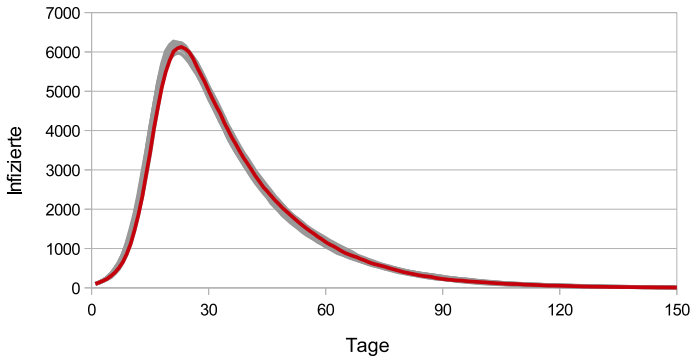

Abbildung 4.7: Vergleich der realen Epidemiekurve (rot) beim SIR-Modell und den mit einer standardnormalverteilten Nachbarschaft berechneten Epidemiekurven (grau). Schrittanzahl = 20, Kühlungsfaktor = 0.85

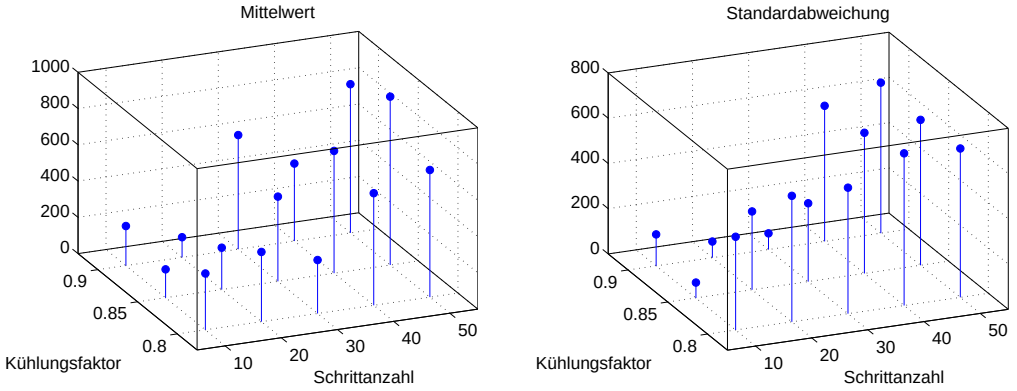

Abbildung 4.8: Mittelwert und Standardabweichung des Fehlers der Kalibrierung des SIR-Modells mittels Simulated Annealing unter Verwendung von 3 Parametern

		Schrittanzahl				
		10	20	30	40	50
Kühlungsfaktor	0.8	0.1	0.4	0.5	0.4	0.2
		0.7	0.2	0.2	0.1	0.0
		0.1	0.2	0.2	0.1	0.3
		0.1	0.2	0.1	0.4	0.5
	0.85	0.1	0.4	0.6	0.4	0.3
		0.5	0.0	0.0	0.0	0.0
		0.4	0.5	0.2	0.1	0.1
		0.0	0.1	0.3	0.5	0.6
	0.9	0.1	0.3	0.3	0.5	0.4
		0.4	0.0	0.0	0.0	0.0
		0.5	0.7	0.3	0.2	0.0
		0.0	0.0	0.4	0.3	0.6

erfolgreich | lokales Min. | nicht fertig | nicht angeschl.

Tabelle 4.6: Anteil an Kalibrierungen des SIR-Modell mit drei Parametern mittels Simulated Annealing, die in die Kategorien aus Tabelle 4.1 fallen

auf, dass bei der Bestimmung von drei Parametern, die gewählte Maximalanzahl von Schleifendurchgängen oft nicht ausreicht, um die Werte zu berechnen. Abbildung 4.9 zeigt wieder den Vergleich zwischen der zu reproduzierenden und den berechneten Epidemiekurven. Dabei ist sehr gut der Unterschied zwischen den erfolgreichen und den nicht angeschlagenen Kalbirierungen zu erkennen.

Evolutionärer Algorithmus

In den nächsten Testreihen wird das SIR-Modell mithilfe des evolutionären Algorithmus kalibriert. Hier sollen die geeigneten Werte für die Populationsgröße n sowie

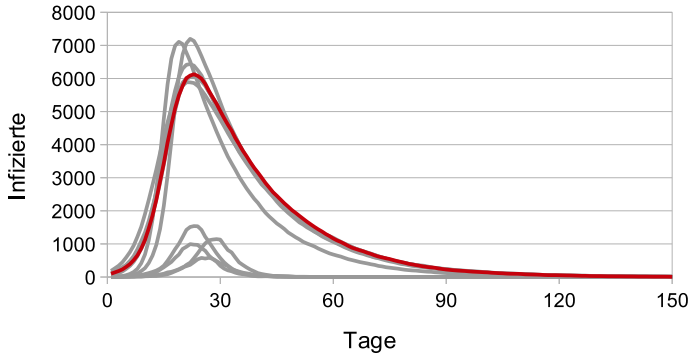

Abbildung 4.9: Vergleich der realen Epidemiekurve (rot) beim SIR-Modell mit drei Parametern und den berechneten Epidemiekurven (grau). Schrittanzahl = 40, Kühlungsfaktor = 0.8

die Rekombinations- und Mutationsraten p_c und p_m experimentell bestimmt werden. Getestet werden Kombinationen der Werte $n = 10, 20, \ldots, 50$, $p_c = 0.25, 0.5, 0.75, 1$ und $p_m = 0.1, 0.2, 0.3$. Zur Bewertung der Kalibrierungen wird nur der Anteil an erfolgreichen Kalibrierungen betrachtet, also diejenigen, die einen Fehlerwert kleiner als 50 liefern. Der Mittelwert und die Standardabweichung des Fehlers ist in den Abbildungen 4.10 bis 4.14 dargestellt. Tabelle 4.7 zeigt zusätzlich den Anteil an erfolgreichen Kalibrierungen.

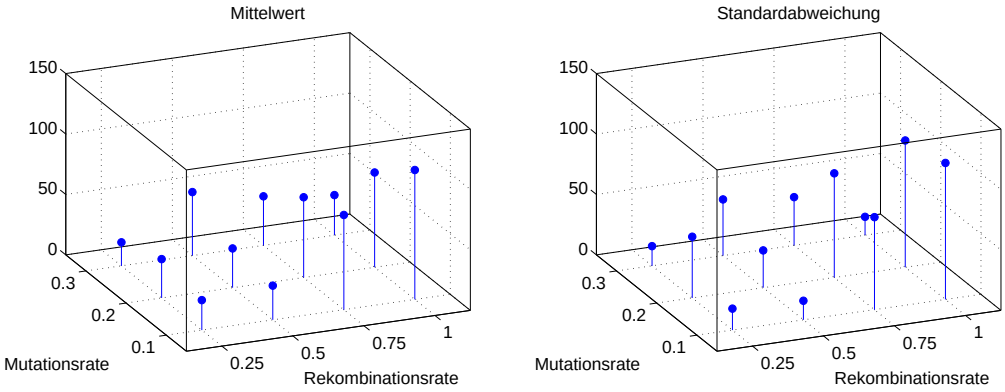

Abbildung 4.10: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des *SIR*-Modell mittles evolutionären Algorithmus und einer Population = 10

Insgesamt sind die Ergebnisse sehr zufriedenstellend und mit denen aus Abbildung 4.6 und Tabelle 4.5, die die Ergenisse der Kalibrierung mittels Simulated Annealing darstellen, vergleichbar. Es kann beobachtet werden, dass eine höhere Mutationsrate förderlich für die Kalibrierung ist, während eine zu hohe Rekombinationsrate sich negativ auf die Resultate auswirkt. Außerdem liefert eine größere Population tendenziell mehr erfolgreiche Kalibrierungen. Die Ergebniskurven der zehn Kalibrierungen einer Konfiguration sowie die Referenzkurve sind in Abbildung 4.15 zu sehen.

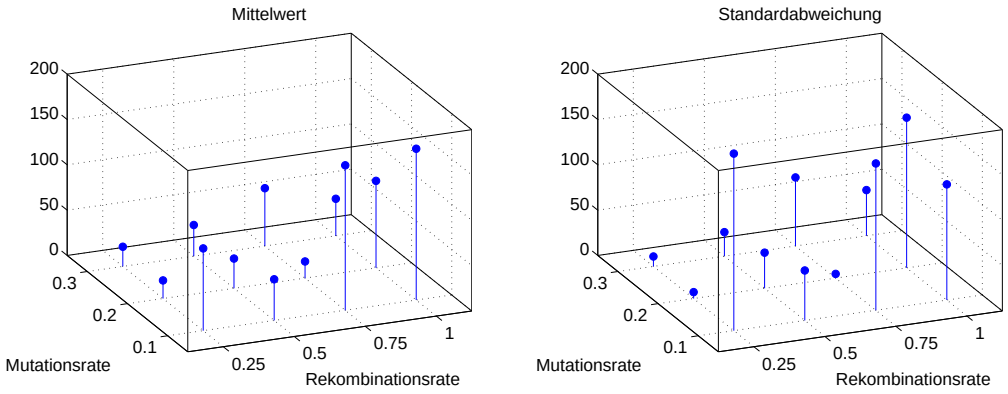

Abbildung 4.11: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des SIR-Modell mittels evolutionären Algorithmus und einer Population = 20

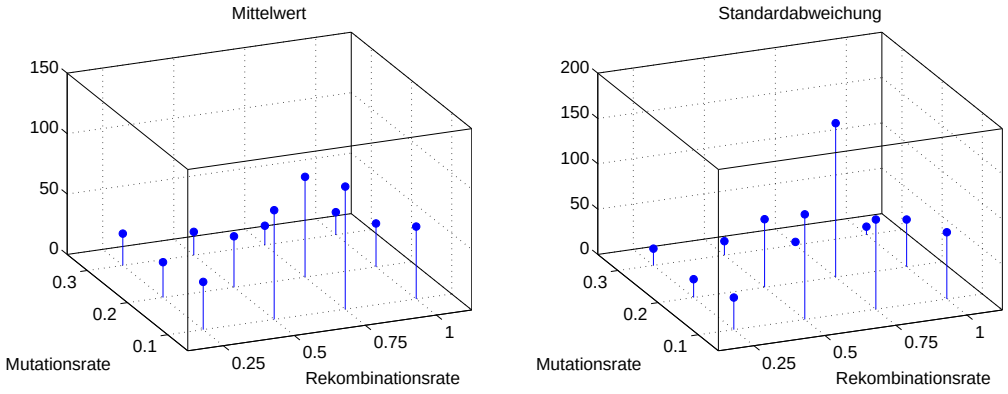

Abbildung 4.12: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des SIR-Modell mittels evolutionären Algorithmus und einer Population = 30

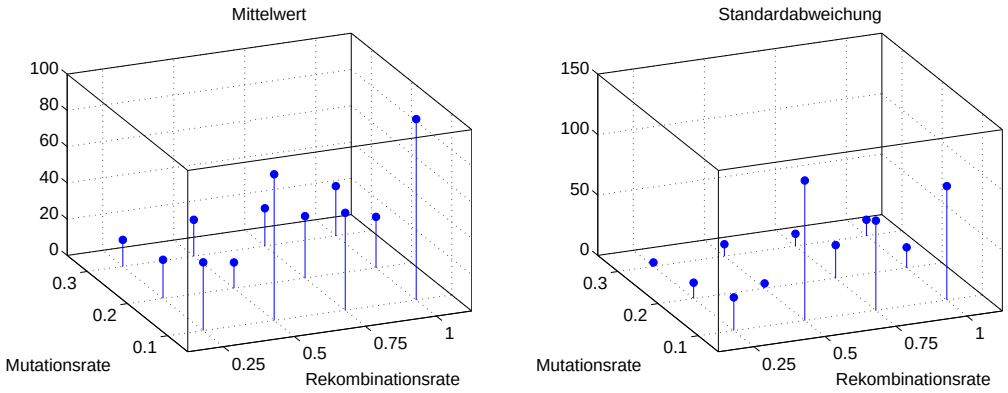

Abbildung 4.13: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des SIR-Modell mittels evolutionären Algorithmus und einer Population = 40

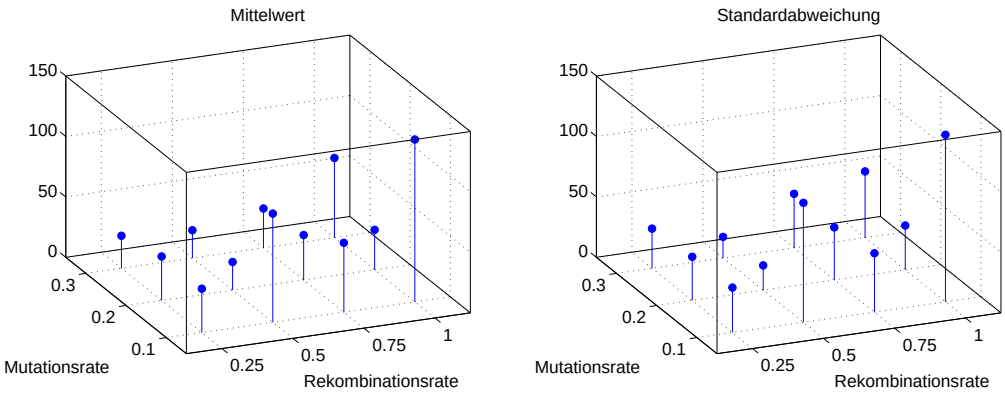

Abbildung 4.14: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des SIR-Modell mittels evolutionären Algorithmus und einer Population = 50

Mutationsrate	Rekombinationsrate: 0.25	0.5	0.75	1
0.1	0.9	0.9	0.5	0.4
	0.7	0.8	0.3	0.1
	0.7	0.4	0.4	0.7
	0.7	0.6	0.8	0.6
	0.8	0.4	0.5	0.3
0.2	0.9	0.9	0.7	0.8
	1.0	0.8	1.0	0.8
	0.8	0.9	0.7	0.9
	0.9	1.0	0.7	0.9
	0.7	0.9	0.8	0.7
0.3	0.9	0.7	0.7	0.9
	1.0	0.7	0.7	0.8
	0.9	0.9	1.0	1.0
	1.0	1.0	1.0	0.9
	0.9	0.9	0.9	0.4

Population: 10 20 30 40 50

Tabelle 4.7: Anteil erfolgreicher Kalibrierungen des SIR-Modells mittels evolutionärem Algorithmus

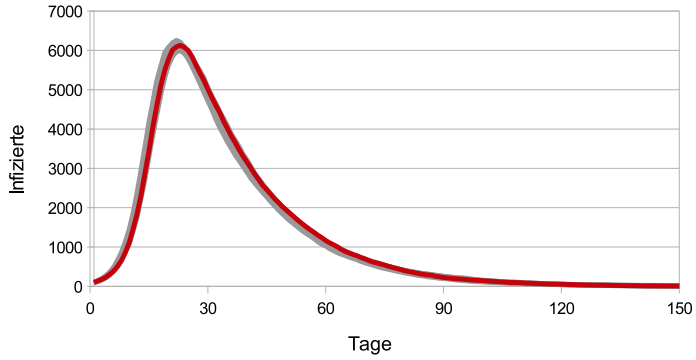

Abbildung 4.15: Vergleich der realen Epidemiekurve (rot) beim SIR-Modell und den berechneten Epidemiekurven (grau). Population = 40, Rekombinationsrate = 0.5, Mutationsrate = 0.3

In den folgenden Testreihen wird der evolutionäre Algorithmus benutzt, um das SIR-Modell mit einem zusätzlichen Parameter zu kalibrieren. Getestet werden dabei die Werte, die zuvor die besten Ergebnisse geliefert haben. Es handelt sich dabei um die Populationsgrößen $20, 30, 40, 50$, die Rekombinationsraten $0.25, 0.5, 0.75$ und die Mutationsraten $0.2, 0.3$. Abbildungen 4.16 bis 4.19 zeigen den Mittelwert und die Standardabweichung des Fehlers der einzelnen Kalibrierungskonfigurationen und Tabelle 4.8 zeigt zusätzlich den Anteil an erfolgreichen Kalibrierungen. Dabei gilt eine Kalibrierung als erfolgreich, wenn der berechnete Fehlerwert kleiner als 75 ist.

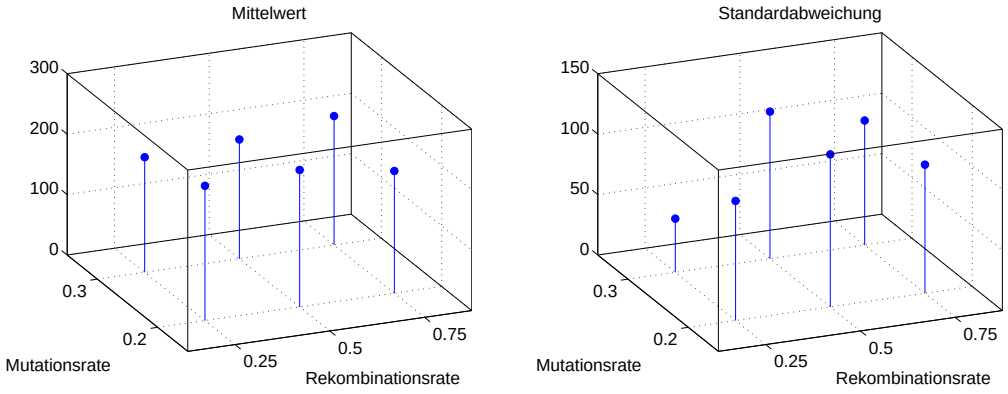

Abbildung 4.16: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des SIR-Modell mit drei Parametern mittles evolutinären Algorithmus und einer Population = 20

Im Vergleich zum SIR-Modell mit zwei Parametern werden hier nur sehr wenige erfolgreiche Kalibrierungen berechnet. Allerdings ist der Mittelwert des Fehlers generell kleiner als bei den Berechnungen, die mit Simulated Annealing durchgeführt wurden. Das liegt daran, dass es beim evolutionären Algorithmus keine Kalibrierungen gab, die nicht angeschlagen haben. Alle berechneten Parametersets liegen zumindest

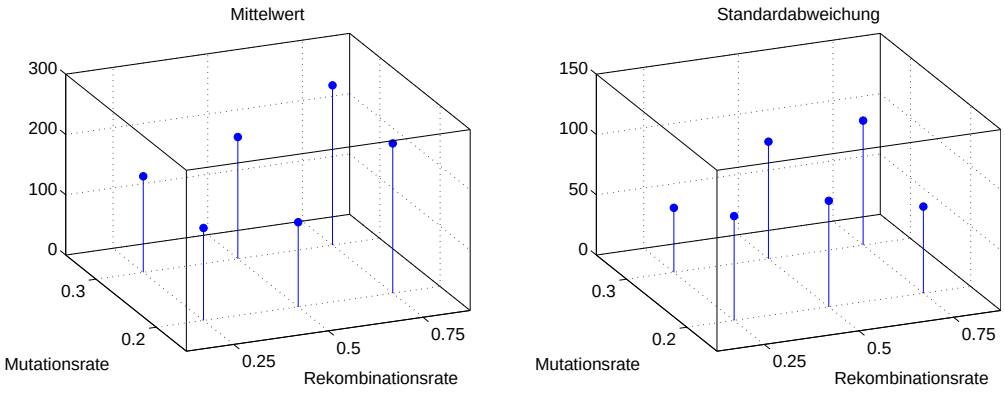

Abbildung 4.17: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des SIR-Modell mit drei Parametern mittels evolutinären Algorithmus und einer Population = 30

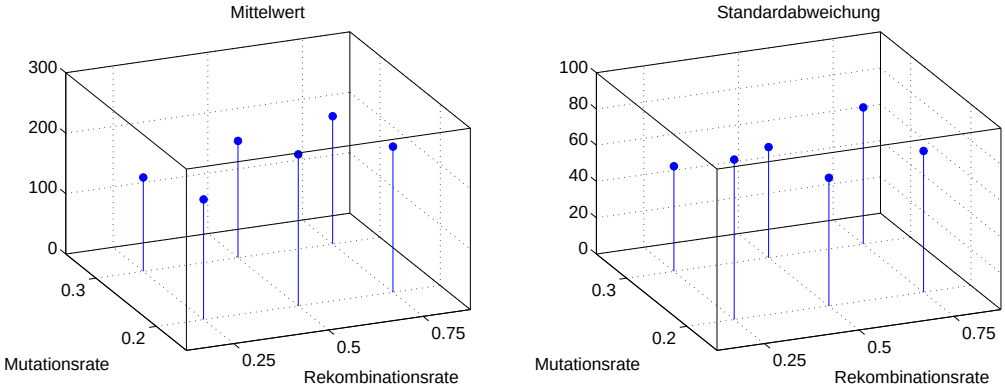

Abbildung 4.18: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des SIR-Modell mit drei Parametern mittels evolutinären Algorithmus und einer Population = 40

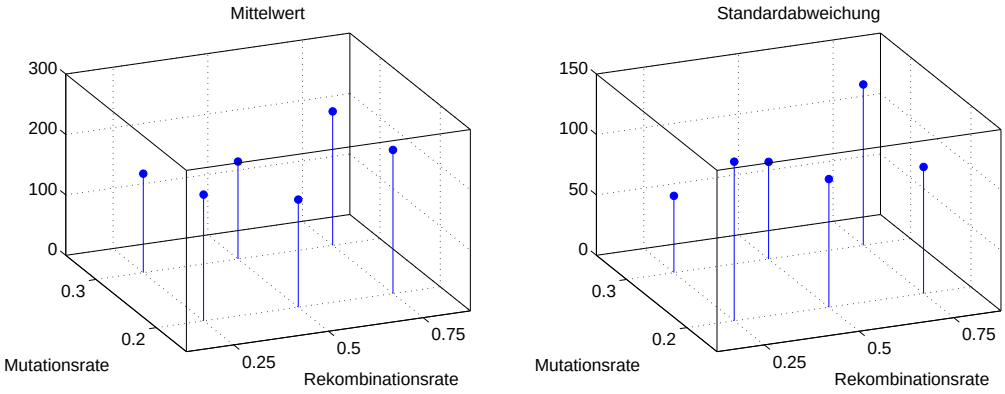

Abbildung 4.19: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des SIR-Modell mit drei Parametern mittels evolutinären Algorithmus und einer Population = 50

Mutationsrate \ Rekombinationsrate	0.25	0.5	0.75
0.2	0.0	0.0	0.1
	0.1	0.3	0.0
	0.0	0.0	0.0
	0.1	0.2	0.1
0.3	0.0	0.2	0.1
	0.0	0.2	0.0
	0.0	0.0	0.0
	0.1	0.0	0.0

Population: 20 30 40 50

Tabelle 4.8: Anzahl erfolgreicher Kalibrierungen des SIR-Modells mittels evolutionärem Algorithmus bei drei zu bestimmenden Parametern

in der Nähe der richtigen Werte. Dementsprechend ist auch hier zu erwarten, dass eine Erhöhung der Modellaufrufe die Resultate verbessern würde. Abbildung 4.20 zeigt den Vergleich der mittels Kalibrierung erhaltenen Epidemiekurven und der Epidemiekurve des Referenzsystems.

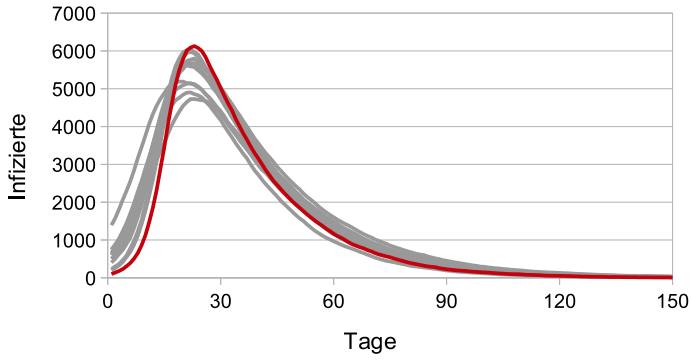

Abbildung 4.20: Vergleich der realen Epidemiekurve (rot) beim SIR-Modell mit drei Parametern und den mittels genetischem Algorithmus berechneten Epidemiekurven (grau). Population = 30, Rekombinationsrate = 0.5, Mutationsrate = 0.3

4.4 Kalibrierung des SIRS-Modells

4.4.1 Vorbereitungen

Als nächster Schritt wird versucht, das $SIRS$-Modell zu kalibrieren. Auch hier muss zuerst das Referenzsystem bestimmt werden. Wie schon beim SIR-Modell wird die Epidemie über einen Zeitraum von 150 Tagen und mithilfe von 10 000 Agenten

berechnet. Als Parameterwerte werden $\alpha = 0.4$, $\beta = 0.05$, $I_0 = 100$ und $d = 50$ gewählt. Die zulässigen Parameterbereiche für α und β sind wieder das Intervall [0,1] und für I_0 das Intervall [0,10 000]. Der Parameter d kann jede beliebige Anzahl an Tagen annehmen, die kleiner ist als der Simulationszeitraum und besitzt demzufolge in diesem Fall das Intervall $[0, 150]$ als Parameterbereich. Abbildung 4.21 zeigt die zu reproduzierende Epidemiekurve und die dazugehörenden Gewichte der Fehlerfunktion, die mittels (4.1) berechnet wurden.
Analog zu der Vorgehensweise in 4.3 wird noch der akzeptierte Fehlerwert und die Starttemperatur des Simulated Annealing bestimmt. Es ergeben sich folgende Werte: akzeptierter Fehlerwert = 30, Starttemperatur = 1900.

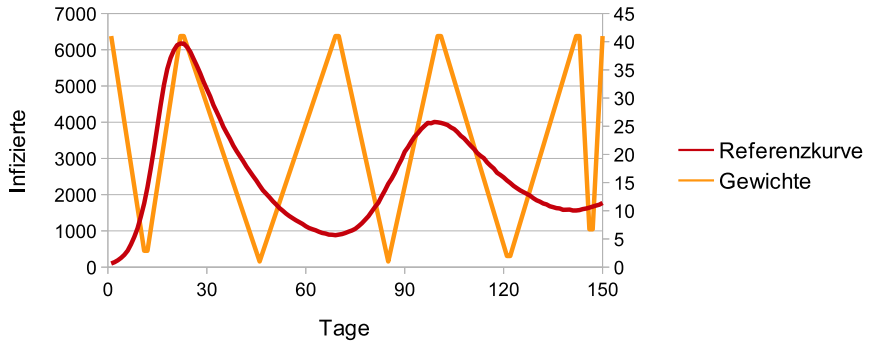

Abbildung 4.21: Epidemiekurve des SIRS-Modells und zugehörige Gewichtsfunktion

4.4.2 Kalibrierung

Wie in Kapitel 4.3 werden auch hier 10 Kalibrierungen pro Testreihe berechnet. Da die Testkalibrierungen des SIR-Modells mit drei Parametern vermuten lassen, dass eine höhere Anzahl an zu bestimmenden Parametern die benötigte Anzahl an Modellaufrufen erhöht, wird bei der Berechnung der vier Parameter des $SIRS$-Modells ein Kalibrierungsdurchlauf erst nach maximal 1 500 Modellaufrufen abgebrochen. Außerdem wird erst bei 400 Modellaufrufen ohne Verbesserung von einer Stagnation des Simulated Annealing ausgegangen.

Simulated Annealing

Bei der Anwendung des Simulated Annealings auf das $SIRS$-Modell werden Konfigurationen getestet, die bereits vorher gute Resultate geliefert haben. Das heißt, es wird eine standardnormalverteilte Nachbarschaft gewählt mit einer Schrittanzahl von $10, 20, 30, 40, 50$. Diese Schrittanzahl wird mit den Werten $0.8, 0.85, 0.9, 0.95$ für den Kühlungsfaktor kombiniert.
Zur Bewertung werden dieselben Kategorien wie in Tabelle 4.1 herangezogen. Aller-

dings werden hier Kalibrierungen als erfolgreich gewertet, die einen Fehler kleiner als 100 lieferten. Die Ergebnisse sind in Abbildung 4.22 und Tabelle 4.9 zu sehen.

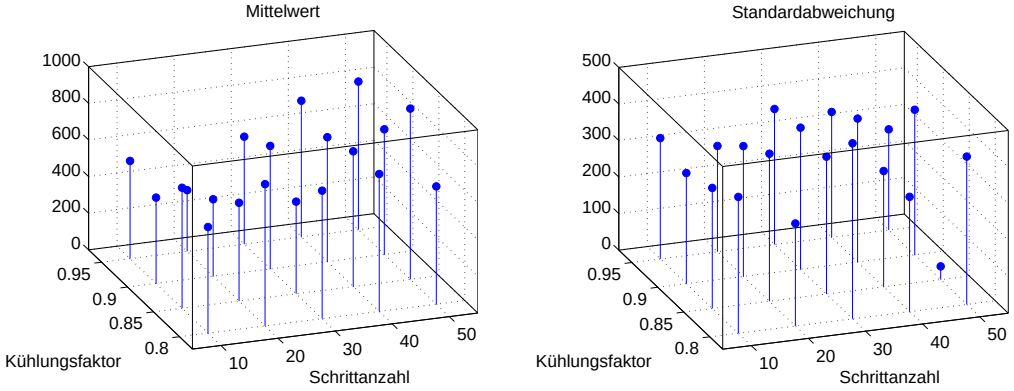

Abbildung 4.22: Mittelwert und Standardabweichung des Fehlers der Kalibrierung des SIRS-Modells mittels Simulated Annealing

Kühlungsfaktor	Schrittanzahl 10	20	30	40	50
0.8	0.1	0.1	0.1	0.1	0.3
	0.8	0.9	0.7	0.7	0.7
	0.1	0.0	0.1	0.2	0.0
	0.0	0.0	0.1	0.0	0.0
0.85	0.0	0.2	0.1	0.1	0.0
	0.9	0.7	0.9	0.4	0.9
	0.1	0.1	0.0	0.5	0.1
	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.1	0.1	0.2	0.2
	0.8	0.6	0.7	0.6	0.6
	0.2	0.3	0.2	0.2	0.2
	0.0	0.0	0.0	0.0	0.0
0.95	0.1	0.2	0.0	0.1	0.1
	0.6	0.2	0.4	0.2	0.1
	0.3	0.6	0.6	0.7	0.8
	0.0	0.0	0.0	0.0	0.0

erfolgreich | lokales Min. | nicht fertig | nicht angeschl.

Tabelle 4.9: Anteil an Kalibrierungen des SIRS-Modell mittels Simulated Annealing, die in die Kategorien aus Tabelle 4.1 fallen.

Auch wenn das Simulated Annealing mit einer standardnormalverteilten Nachbarschaft für das SIR-Modell gute Ergebnisse geliefert hat, so sind die Resultate für das $SIRS$-Modell jedoch sehr schlecht. In sehr wenigen Fällen wurde eine erfolgreiche Kalibrierung durchgeführt. Stattdessen bleibt der Algorithmus fast immer in einem lokalen Minimum hängen. In Tabelle 4.10 sind exemplarisch die Resultate für eine Schrittanzahl von 50 und einem Kühlungsfaktor von 0.8 dargestellt. Man erkennt,

dass bei fast allen berechneten Parametersets d einen Wert in der Nähe von 0, statt in der Nähe von 50 annimmt. Die Werte der restlichen Parameter sind dabei über ihren gesamten Parameterbereich verstreut. Es scheint, als ob für den Wert $d = 0$ eine "minimale Hyperebene" mit einem Fehler in der Nähe von 900 existiert, aus der der Kalibrierungsalgorithmus nicht mehr herausfindet, da lediglich eine Änderung des Wertes von d eine Verbesserung liefern würde. Alle Modifikationen der restlichen Parameter liefern keine signifikante Änderung des Fehlerwertes. Tabelle 4.11 zeigt den Anteil an Kalibrierungen, die einen Wert von d in der Nähe von 0 erzeugen. Eine Erhöhung der erlaubten Schleifendurchgänge würde keine Verbesserung der Resultate hervorrufen, da die Kalibrierung in den meisten Fällen eh schon vorzeitig abgebrochen wurde.
Ein Vergleich der mittels Kalibrierung berechneten Epidemiekurven und der Epidemiekurve des Referenzsystems ist in Abbildung 4.23 zu sehen.

Durchlauf	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Fehler	905	913	951	927	906	17	83	89	901	740
α	0.76	0.53	0.58	0.43	0.65	0.40	0.37	0.37	0.94	0.42
β	0.49	0.30	0.38	0.26	0.35	0.05	0.05	0.05	0.6	0.08
d	0	1	0	1	1	50	49	48	0	23
I_0	443	942	2290	1191	426	99	184	129	368	99

Tabelle 4.10: Berechnete Parameterwerte des Simulated Annealing bei einer Schrittanzahl von 50 und einem Kühlungsfaktor von 0.8

	Schrittanzahl				
Kühlung	10	20	30	40	50
0.8	0.4	0.8	0.5	0.8	0.6
0.85	0.6	0.4	0.3	0.8	1.0
0.9	0.1	0.3	0.7	0.7	0.7
0.95	0.3	0.2	0.5	0.8	0.9

Tabelle 4.11: Anteil an Kalibrierungen des SIRS-Modells mittels Simulated Annealing, die einen Wert von d in der Nähe von 0 erzeugen

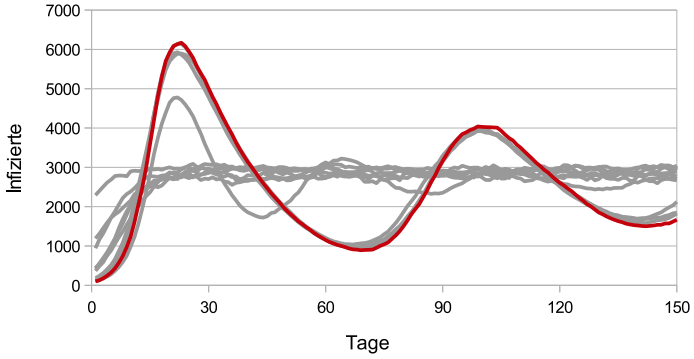

Abbildung 4.23: Vergleich der realen Epidemiekurve (rot) beim SIRS-Modell und den berechneten Epidemiekurven (grau). Schrittanzahl = 50, Kühlungsfaktor = 0.8

Evolutionärer Algorithmus

Als Nächstes wird der evolutionäre Algorithmus auf das $SIRS$-Modell angewendet. Getestet werden dabei die Konfigurationen, die bereits beim SIR-Modell gute Ergebnisse geliefert haben. Das heißt, eine hohe Bevölkerungszahl wird mit einer moderaten Rekombinationsrate und einer hohen Mutationsrate kombiniert. Da aus den Kalbrierungsversuchen mittels Simulated Annealing bekannt ist, dass ein hohes Risiko besteht, lediglich ein lokales Minimum zu berechnen, werden für die Mutationsraten noch höhere Werte ausgetestet, als dies beim SIR-Modell der Fall war. Konkret werden folgende Werte getestet: $n = 30, 40, 50$, $p_c = 0.25, 0.5, 0.75$ und $p_m = 0.3, 0.4, 0.5$. Die Resultate sind in Abbildung 4.24 bis 4.26 zu sehen. Tabelle 4.12 zeigt wieder den Anteil an erfolgreichen Kalibrierungen. Das sind in diesem Fall diejenigen Kalibigrierungen, die einen Fehlerwert kleiner als 100 liefern.

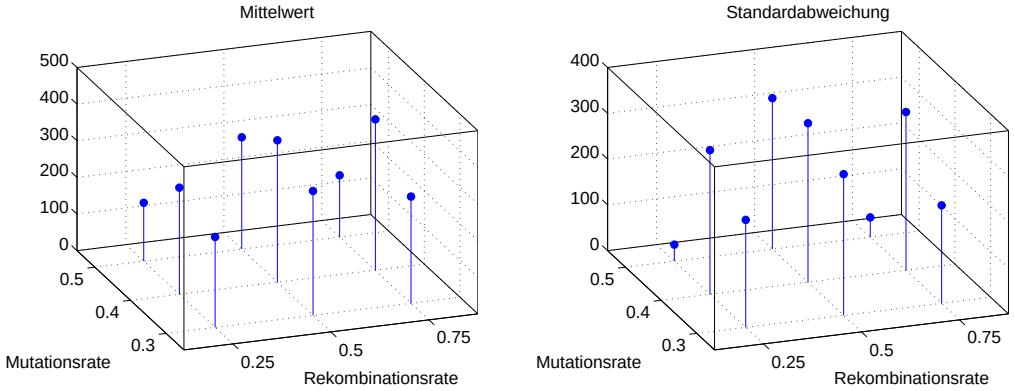

Abbildung 4.24: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des $SIRS$-Modell mittles evolutinären Algorithmus und einer Population = 30

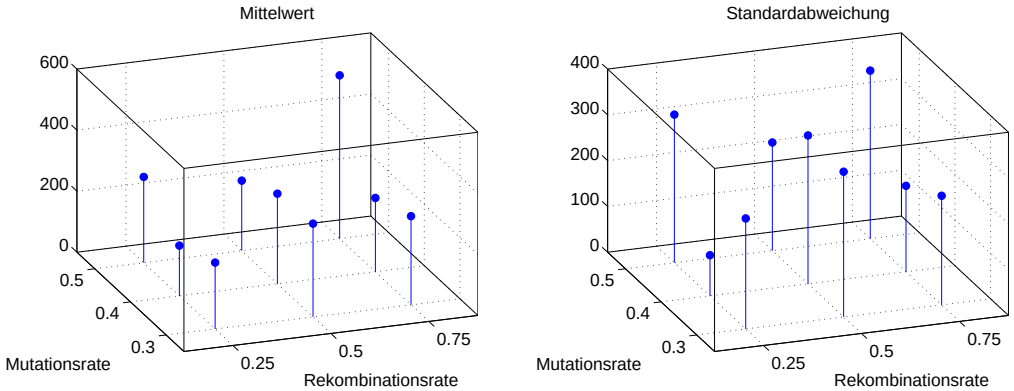

Abbildung 4.25: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des $SIRS$-Modell mittels evolutinären Algorithmus und einer Population = 40

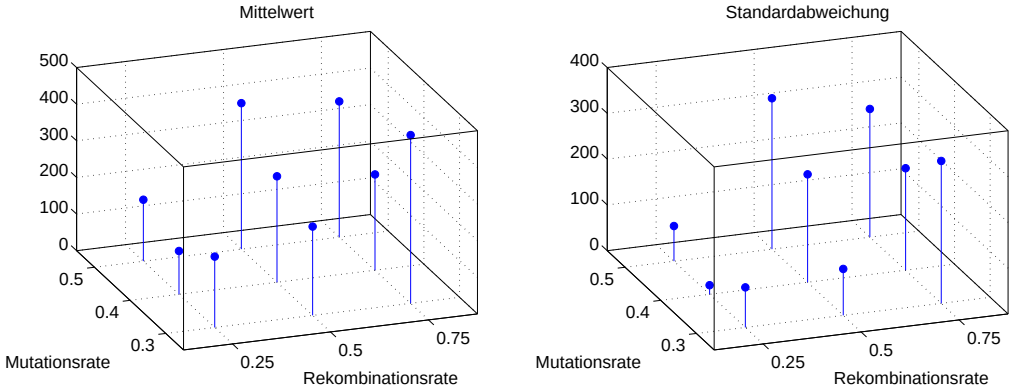

Abbildung 4.26: Mittelwert und Standardabweichung des Fehlers der Kalbrierung des $SIRS$-Modell mittels evolutinären Algorithmus und einer Population = 50

Mutationsrate	Rekombinationsrate 0.25	0.5	0.75
0.3	0.1	0.1	0.1
	0.3	0.1	0.1
	0.2	0.1	0.0
0.4	0.2	0.1	0.0
	0.2	0.2	0.3
	0.2	0.1	0.1
0.5	0.1	0.1	0.0
	0.2	0.1	0.0
	0.1	0.0	0.0

Population:	30	40	50

Tabelle 4.12: Anteil an erfolgreichen Kalibrierungen des SIRS-Modells mittels evolutionärem Algorithmus

Insgesamt sind die Ergebnisse beim evolutionären Algorithmus um einiges besser als beim Simulated Annealing. Zwar gibt es hier auch nicht viele Durchläufe, die ein optimales Resultat liefern, doch fast alle berechneten Parametersets befinden sich in der Nähe der reellen Werte. Es gibt nur wenige Kalibrierungsversuche, die in den lokalen Minimum bei $d = 0$ hängen bleiben (siehe Tabelle 4.13).
In Abbildung 4.27 ist wieder ein Vergleich der zu reproduzierenden und den berechneten Epidemiekurven des $SIRS$-Modells zu sehen.

Mutationsrate \ Rekombinationsrate	0.25	0.5	0.75
0.3	0.1	0.2	0.1
	0.1	0.2	0.1
	0.0	0.0	0.3
0.4	0.2	0.3	0.3
	0.0	0.2	0.0
	0.0	0.1	0.1
0.5	0.0	0.2	0.0
	0.2	0.0	0.5
	0.0	0.2	0.1

Population:	30	40	50

Tabelle 4.13: Anteil an Kalibrierungen des SIRS-Modells mittels evolutionärem Algorithmus, die einen Wert von d in der Nähe von 0 erzeugen

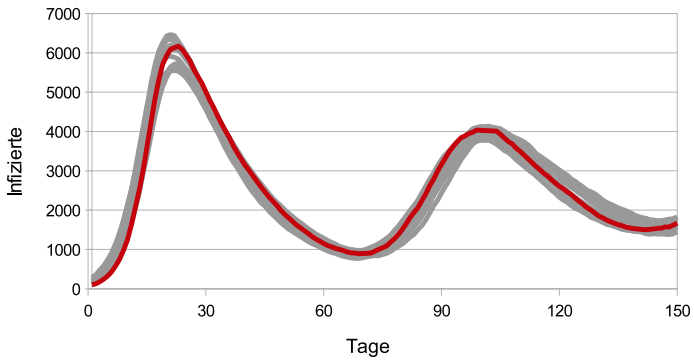

Abbildung 4.27: Vergleich der realen Epidemiekurve (rot) beim SIRS-Modell und den berechneten Epidemiekurven (grau). Population = 50, Rekombinationsrate = 0.25, Mutationsrate = 0.4

5 Verringerung der Laufzeit

5.1 Vorgehensweise

Die in Kapitel 4.3 und 4.4 erzielten Resultate sind sehr zufriedenstellend. Ein Problem bilden jedoch die Laufzeiten der einzelnen Kalibrierungen. Da die Berechnungen immer mit 10 000 Agenten durchgeführt worden sind, um die Streuung der Simulationsergebnisse gering zu halten, dauert ein Simulationsaufruf, und demzufolge ein Kalibrierungsdurchlauf, relativ lange. So beträgt die durchschnittlich benötigte Zeit für eine Kalibrierung des SIR-Modells rund 8.5 Minuten. Die Kalibrierung des $SIRS$-Modells benötigt sogar fast 16 Minuten. Dabei ist die Laufzeit eines einzelnen Aufrufs des $SIRS$-Modells mit 0.4 Sekunden noch vergleichsweise gering. Bei komplexeren Modellen kann solch ein Simulationsdurchlauf schon mehrere Minuten dauern und somit wird eine Kalibrierung in einer angemessenen Zeit nicht möglich sein. Natürlich ermöglichen es leistungsstarke Computer, die Laufzeiten zu verringern. Vor allem bei den evolutionären Algorithmen kann mittels paralleler Programmierung eine erhebliche Reduktion der Laufzeit erreicht werden. Trotzdem wäre es wünschenswert, einen anderen Weg zu finden, um die Kalibrierungen innerhalb einer kürzeren Zeit durchzuführen.

Eine Möglichkeit wäre, zumindest am Anfang der Kalibrierung, auf die Exaktheit der Simulationsergebnisse zu verzichten und erst am Ende des Kalibrierungsprozesses mit genauen Resultaten zu arbeiten. Demzufolge würde bei Beginn einer Kalibrierung die Simulation mit einer niedrigen Agentenanzahl aufgerufen werden, was es ermöglicht, viele Simulationen in einer geringen Zeit durchzuführen. Im Laufe des Kalibrierungsvorgangs wird die Agentenzahl dann progressiv erhöht. Dadurch erhöht sich auch die Laufzeit der einzelnen Simulationsaufrufe, jedoch verringert sich die Streuung der Ergebnisse und die erhaltenen Fehlerwerte sind exakter. Die Idee bei dieser Vorgehensweise liegt darin, dass die anfangs benutzten, ungenauen Simulationsergebnisse die Suche nach den richtigen Parametern trotzdem in die richtige Richtung führen und erst zur abschließenden Feineinstellung die aufwendig berechneten, genauen Fehlerwerte benötigt werden.

Diese Vorgehensweise wurde anhand der Beispielmodelle ausgetestet. Dabei wurden verschiedene Strategien und Konfigurationen angewendet. Die Kalibrierung wurde mit unterschiedlichen Agentenzahlen gestartet. Außerdem wurde diese Agentenzahl auf zwei verschiedene Weisen aktualisiert: mittels linearem und geometrischem Wachstum.

Insgesamt werden folgende Informationen benötigt:

- Agentenanzahl N_i bei Start der Kalibrierung
- Agentenanzahl N_f bei Ende der Kalibrierung
- Art des Wachstums
- Anzahl n_u an vorgenommenen Modellupdates
- Gesamtanzahl n_r an Modellaufrufen

Die ersten beiden Informationen sind modellspezifisch und müssen demzufolge in der Java-Klasse *Model* zur Verfügung gestellt werden. Die restlichen Informationen beziehen sich auf den Kalibrierungsalgorithmus und müssen entsprechend in die Klasse *Calibration* eingefügt werden. Die Gesamtanzahl an Modellaufrufen war schon vorher bekannt.
Am Anfang einer Kalibrierung werden die Updateparameter berechnet. Bei einem linearen Wachstum beträgt dieser

$$u_{lin} = \frac{N_f - N_i}{n_u} \tag{5.1}$$

und beim geometrischen Wachstum

$$u_{geom} = \sqrt[n_u]{\frac{N_f}{N_i}} \tag{5.2}$$

Nach $\frac{n_r}{n_u+1}$ Modellaufrufen wird die aktuelle Agentenanzahl N durch $N + u_{lin}$, bzw. $N \cdot u_{geom}$ ersetzt. Auf diese Weise wird die gewünschte Zielanzahl an Agenten bis zum Ende der Kalibrierung erreicht.

Es ist zu beachten, dass, vor allem bei den evolutionären Algorithmen, verschiedene Parameterkandidaten über einen längeren Zeitraum erhalten bleiben. Bisher wurde deren Fehlerwert einmalig berechnet und dann für den weiteren Verlauf der Kalibrierung als gegeben betrachtet. Nun kann es jedoch vorkommen, dass dieser Fehlerwert nicht mehr zutrifft, da sich die Agentenanzahl mittlerweile erhöht hat. Demzufolge muss dieser Fehlerwert neu berechnet werden. Beim Simulated Annealing wäre dies nur ein Kandidat, der aktualisiert werden müsste, bei den evolutionären Algorithmen hingegen die gesamte Population. Da jedoch zu erwarten ist, dass sich die innerhalb einer Generation neu entstandenen Genome immer näher am globalen Minimum befinden, werden die alten Genome voraussichtlich nach und nach aus der Population verschwinden. Demnach scheint es überflüssig, bei jedem Aktualisierungsvorgang die Fehlerwerte der gesamten Population neu zu berechnen. Andererseits kann ein Fehlerwert, der mit einer niedrigen Agentenzahl berechnet wurde, einen unbegründet niedrigen Wert annehmen und der zugehörige Parameterkandidat würde fälschlicherweise lange in der Population überleben. Als Kompromiss wurde folgende Vorgehensweise gewählt: Nach einer Erhöhung der Agentenzahl werden alle neu generierten Kandidaten mit der neuen Agentenzahl evaluiert. Die restlichen Kandidaten ändern ihren Fehlerwert vorerst nicht. Erst unmittelbar bevor die nächste Erhöhung

der Agentenanzahl durchgeführt wird, werden die alten Fehlerwerte aktualisiert. Bei der letzten Erhöhung wird jedoch anders vorgegangen. Hier wird zuerst die Agentenzahl auf N_f erhöht und anschließend alle Fehler mithilfe dieser Agentenzahl aktualisiert. Dies stellt sicher, dass das Ergebnis der Kalibrierung nicht mit einer zu niedrigen Agentenzahl berechnet wurde. Eine schematische Darstellung dieser Vorgehensweise befindet sich in Abbildung 5.1.

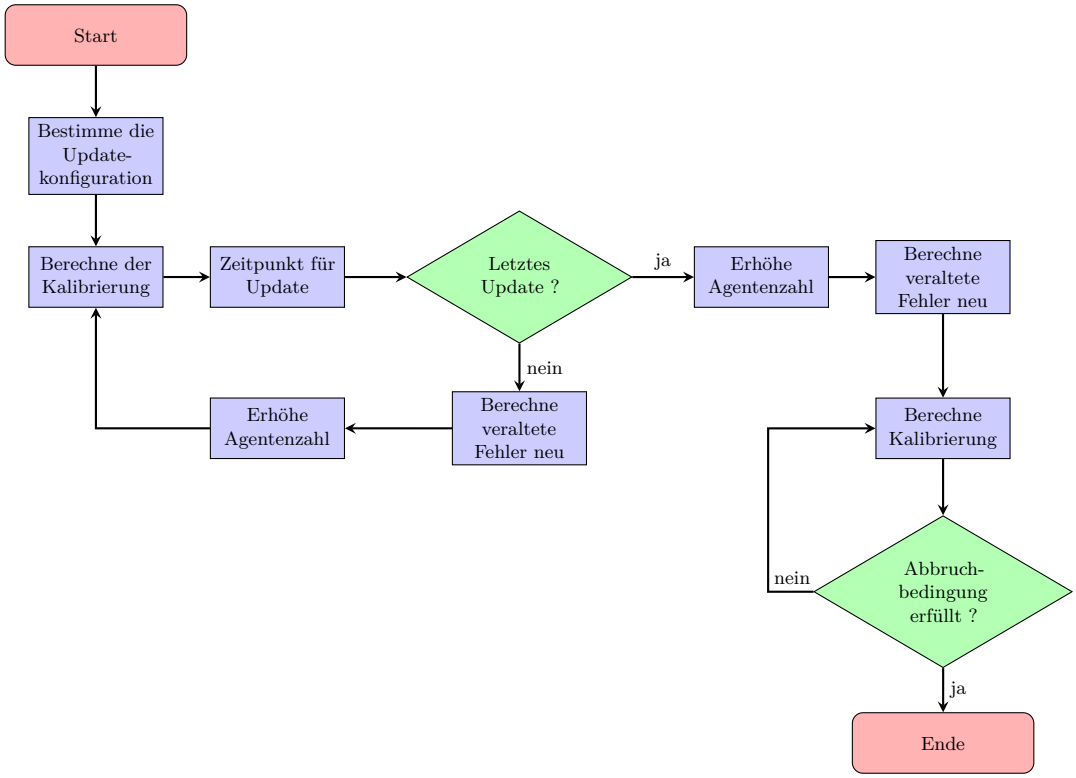

Abbildung 5.1: Flowchart für eine Kalibrierung mittels Agentenupdate

Abschließend müssen noch kleine Modifikationen der Beispielmodelle durchgeführt werden, damit eine Aktualisierung der Agentenanzahl realisierbar ist. So ist es erforderlich, die pseudo-reellen Daten anzupassen. Diese werden bei der Initialisierung des Modells mithilfe von 10 000 Agenten berechnet. Bei einer Änderung der Agentenzahl im Laufe des Kalibrierungsvorgangs werden die Daten gemäß der aktuellen Agentenanzahl skaliert, damit ihre Größenordnung mit den berechneten Simulationswerten übereinstimmt. Außerdem muss die Funktion des Parameters I_0 innerhalb des $SIRS$-Modells geändert werden. Bislang stellte er die Absolutzahl anfangs infizierter Personen dar. Dies ist jedoch nur bei einer gleichbleibenden Agentenanzahl sinnvoll. Wird die Agentenanzahl variiert, ist es zweckmäßiger, diesen Parameter als Anteil infizierter Personen innerhalb der Gesamtbevölkerung zu betrachten und ihn somit unabhängig von der Populationsgröße zu definieren. Weiters müssen die Abbruchbedingungen verändert werden. In Kapitel 4 wurde die Kalibrierung frühzeitig abgebrochen, wenn der berechnete Fehler unterhalb einer gewissen Schranke lag. Nun besteht allerdings das Problem, dass dieser Wert mittels einer niedrigen

Agentenzahl berechnet wurde und nicht unbedingt gültig ist. Dementsprechend muss die Kalibrierung weiterlaufen, bis das letzte Update durchgeführt worden ist und die gewünschte hohe Agentenzahl erreicht worden ist.

Alle Berechnungen wurden auf einem Laptop, ausgestattet mit dem Betriebssystem Windows 8.1 und einem Intel$^{(R)}$ Core$^{(TM)}$ i5-4200U Prozessor mit 8,00 GB RAM durchgeführt.

5.2 SIR-Modell

Als Erstes wird die Methode auf das SIR-Modell angewendet. Getestet werden Kombinationen der beiden Wachstumsarten mit den Werten $N_i = 1000, 3000$ und $n_u = 2, 4, 8$. Als Endanzahl an Agenten wird in allen Fällen der Wert 10 000 gewählt, und die Gesamtanzahl an Modellaufrufen beträgt immer 1000.
Zu Vergleichszwecken werden zusätzliche Tests durchgeführt, bei denen eine konstante Agentenanzahl $N_c = 1000, 3000$ benutzt wird. Nach Abschluss einer solchen Kalibrierung werden die berechneten Parameterwerte erneut dem Modell übergeben und der Fehler unter Benutzung von 10 000 Agenten berechnet. Dieser Fehlerwert wird dann als Ergebnis der Kalibrierung angesehen. Insgesamt werden folgende Konfigurationen getestet:

	N_i	**Art**	n_u		N_i	**Art**	n_u		N_i	**Art**	n_u
1	1000	-	0	**6**	1000	geom.	8	**11**	1000	linear	4
2	3000	-	0	**7**	3000	geom.	2	**12**	1000	linear	8
3	10000	-	0	**8**	3000	geom.	4	**13**	3000	linear	2
4	1000	geom.	2	**9**	3000	geom.	8	**14**	3000	linear	4
5	1000	geom.	4	**11**	1000	linear.	2	**15**	3000	linear	8

Tabelle 5.1: Tabelle der getesteten Updatekonfigurationen

Für jede Updatekonfiguration werden die mittleren Resultate aus jeweils 10 Kalibrierungen mit drei verschiedenen Konfigurationen genommen, die im Kapitel 4.3 gute Resultate geliefert haben. Für das Simulated Annealing wird eine Schrittweite von 20 gewählt, kombiniert mit den Werten $0.8, 0.85$ und 0.9 für den Kühlungsfaktor. Beim evolutionären Algorithmus wird eine Populationsgröße von 40 und eine Rekombinationsrate von 0.25 gewählt, kombiniert mit den Mutationsraten $0.2, 0.3$ und 0.4. Der mittlere Fehlerwert sowie die durchschnittlich benötigte Rechenzeit dieser Kalibrierungen sind in Abbildung 5.2 zu sehen. Die Updatekonfiguration mit der Nummer 3 entspricht dabei dem Standard, der in Kapitel 4.3 angewendet wurde.

Auffallend ist, dass die Ergebnisse des evolutionären Algorithmus, insgesamt besser sind als die Resultate, die mittels Simulated Annealing berechnet wurden. Bis auf einige Ausnahmen ist der mittlere Fehler sogar kleiner als 20. Bei diesem Wert wurde in Kapitel 4.3 die Kalibrierung vorzeitig abgebrochen, da der Fehler klein genug war.

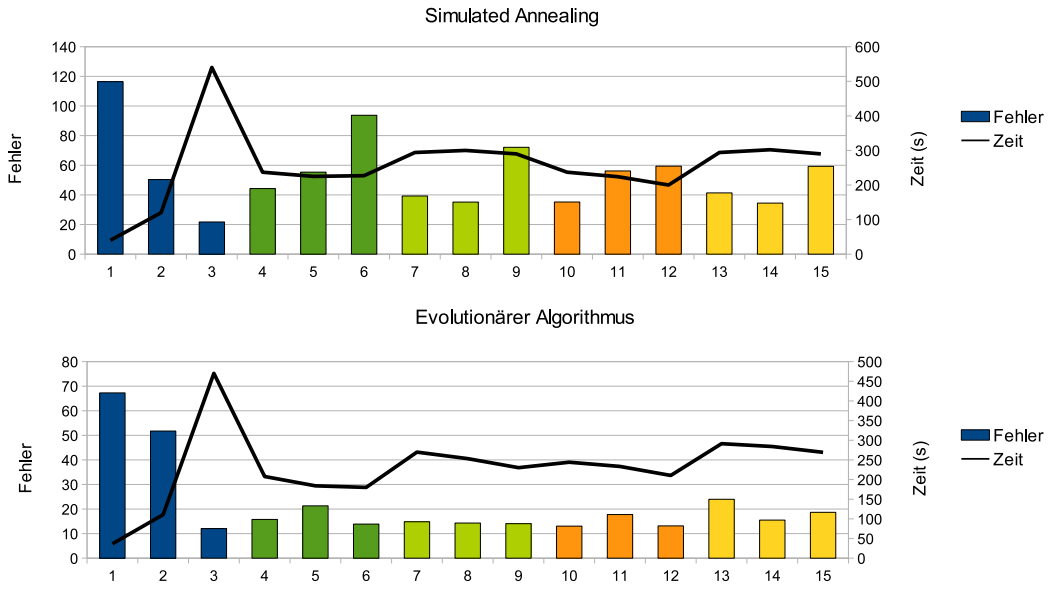

Abbildung 5.2: Resultate der Kalibrierung des SIR-Modells mittels Simulated Annealing und evolutionärem Algorithmus unter Anwendung der Updatekonfigurationen aus Tabelle 5.1

Dementsprechend sind fast alle Kalibrierungen, die mittels evolutionärem Algorithmus und Agentenerhöhung berechnet werden, erfolgreich.
Beim Simulated Annealing bietet sich ein anderes Bild. Hier sind die Ergebnisse, die mittels Agentenerhöhung berechnet wurden, nur in wenigen Fällen gleichwertig mit den im vorherigen Kapitel erhaltenen Werten. Besonders bei einer hohen Anzahl an Updates sind die Fehlerwerte vergleichsweise hoch.
Wie erwartet, ist die Laufzeit für Kalibrierungen, die mittels Agentenerhöhungen berechnet wurden, um einiges geringer als die für die Standardkalibrierung. Generell gilt, dass die Rechenzeit auf etwa die Hälfte reduziert werden kann. Außerdem gilt, dass die Laufzeit bei der Anwendung eines geometrischen Wachstums und bei einer niedrigen Anfangszahl an Agenten am geringsten ist. Des Weiteren sinkt sie mit der Zahl der vollzogenen Erhöhungen der Agentenzahl. Es ist jedoch zu erwarten, dass dies kein anhaltender Trend ist, da ab einem gewissen Punkt die Arbeit der Re-Evaluierung der aktuellen Population bei jedem Update gegenüber dem Vorteil des Rechnens mit weniger Agenten überwiegt. Um dies zu überprüfen, sind weitere Kalibrierungen mit einer Vielzahl an Updates durchgeführt worden. Die neue Testreihe umfasst Kalibrierungen mit geometrischem Wachstum und einer Startanzahl von 1000 Agenten. Sie setzt demnach die Konfigurationen **4**, **5** und **6** aus Tabelle 5.1 fort. Abbildung 5.3 zeigt den mittleren Fehler und die Laufzeit dieser Kalibrierungen, abhängig von der Anzahl an durchgeführten Agentenerhöhungen. Man erkennt, dass die Laufzeit in der Tat wieder ab einer gewissen Updateanzahl steigt, ohne dass die berechneten Fehlerwerte signifikant besser werden.

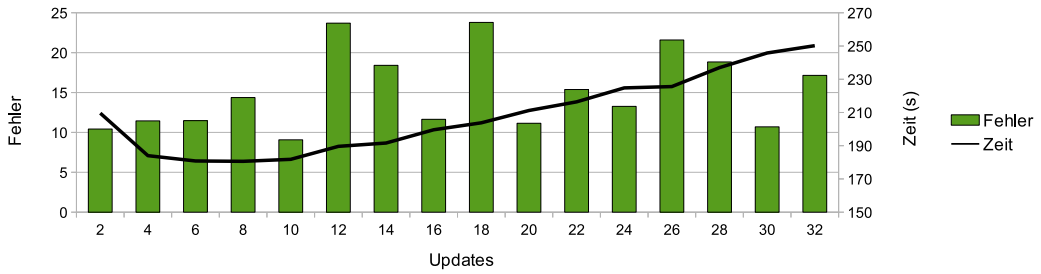

Abbildung 5.3: Fehler und benötigte Zeit für eine Kalibrierung mittels evoutionärem Algorithmus abhängig von der Anzahl an durchgeführten Agentenupdates

Bei der Diskussion von Abbildung 5.2 wurde erwähnt, dass der mittlere Fehler beim Simulated Annealing mit einer wachsenden Anzahl an Agentenerhöhungen steigt. Zu dieser Beobachtung wurden weitere Untersuchungen durchgeführt und folgendes Phänomen hat sich gezeigt: Die Entwicklung des Fehlers innerhalb eines Kalibrierungsdurchlaufes verläuft auf eine von zwei verschiedenen Arten. In den meisten Fällen wird schon mithilfe der Simulationsdurchläufe mit einer niedrigen Agentenanzahl ein relativ kleiner Fehler angenommen und der restliche Kalibrierungsvorgang dient nur noch dazu, die Feineinstellungen des gefundenen Parametersets vorzunehmen. Falls dies jedoch nicht der Fall ist und der Fehler nach der Anfangsphase des Kalibrierungsprozesses noch immer vergleichsweise hoch ist, wird das aktuelle Parameterset auch im weiteren Verlauf lange nicht durch einen besseren Kandidaten ersetzt. Durch die Reevaluierung des Fehlerwertes bei einem Update erhöht sich dieser Fehler meistens nur noch. Erst wenn der Kalibrierungsvorgang so weit fortgeschritten ist, dass die Simulationen mit der anvisierten hohen Agentenzahl durchgeführt werden, greift der Kalibrierungsalgorithmus und der Fehler wird verkleinert. Abbildung 5.4 zeigt diese beiden typischen Verhaltensweisen des Fehlerwertes anhand von Kalibrierungen, die mittels 4 bzw. 8 Updates durchgeführt wurden. Hier ist auch zu erkennen, dass bei einer hohen Anzahl an Updates die Zahl der Simulationsaufrufe, die mit der Endanzahl an Agenten durchgeführt werden, nicht ausreicht um rechtzeitig einen niedrigen Fehlerwert zu berechnen. Der Grund für dieses Verhalten ist nicht offensichtlich. Es ist jedoch möglich, dass diesem Phänomen mit einem geeigneteren Kühlungsverlauf entgegengewirkt werden kann.

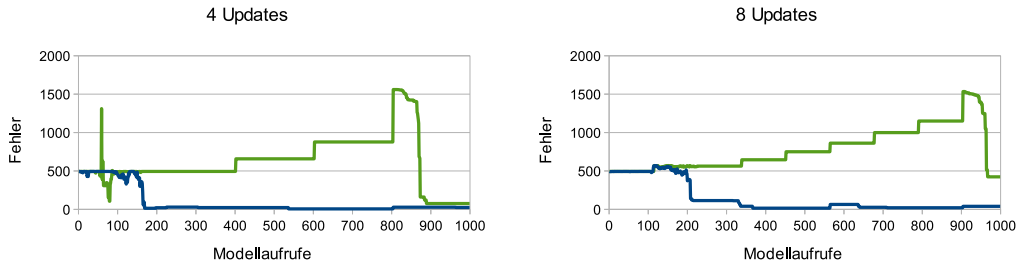

Abbildung 5.4: Verschiedene Entwicklungen des Fehlers im Laufe einer Kalibrierung mittels Simulated Annealing und Erhöhung der Agentenzahl

5.3 SIR-Modell mit 3 Parametern

Da das Simulated Annealing weder bei der Standardkalibrierung des *SIR*-Modells mit 3 Parametern noch bei der Kalibrierung des *SIR*-Modells mittels Steigerung der Agentenzahl gute Resultate erzielen konnte, wird hier auf eine Anwendung dieses Algorithmus verzichtet. Stattdessen werden nur Kalibrierungen mithilfe des evolutionären Algorithmus durchgeführt.
Die gewählten Kalibrierungskonfigurationen benutzen eine Populationsgröße von $30, 40, 50$ kombiniert mit einer Rekombinationsrate von 0.25 und einer Mutationsrate von 0.3. Der Fehler und die benötigte Rechenzeit der Kalibrierungen sind in Abbildung 5.5 zu sehen.

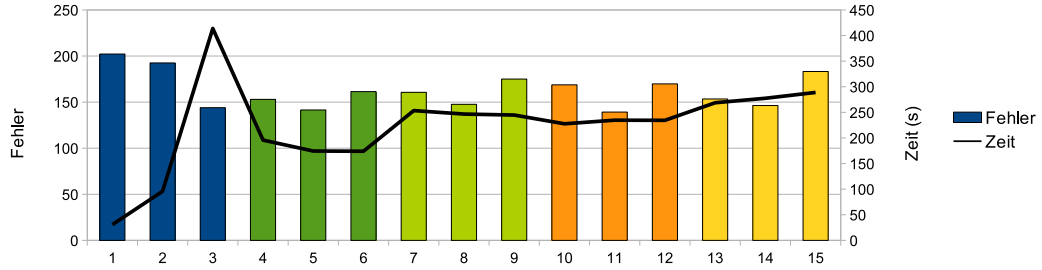

Abbildung 5.5: Resultate der Kalibrierung des SIR-Modells mit drei Parametern mittels evolutionärem Algorithmus unter Anwendung der Updatekonfigurationen aus Tabelle 5.1

Wie schon in Kapitel 4.3 sind auch hier die Fehlerwerte insgesamt größer als beim *SIR*-Modell mit zwei Parametern. Des Weiteren ist es auffallend, dass der Unterschied zwischen einer Kalibrierung mit konstant 1000 und 3000 Agenten und der Standardkalibrierung nicht mehr so groß ist, wie dies für das einfache *SIR*-Modell der Fall war. Die mittels Agentenerhöhung erhaltenen Ergebnisse befinden sich jedoch weiterhin in derselben Größenordnung wie die der Standardkalibrierung. Dabei scheint kein signifikanter Unterschied zwischen den angewendeten Wachstumsarten zu bestehen. Lediglich die Anwendung von vier Agentenupdates liefert durchwegs

bessere Ergebnisse als die anderen Konfigurationen.
Durchschnittlich benötigen die Kalibrierungen mittels Agentenerhöhung auch hier etwas mehr als die Hälfte der Rechenzeit einer Standardkalibrierung.

5.4 SIRS-Modell

Als Nächstes wird das *SIRS*-Modell kalibriert. Wie beim *SIR*-Modell mit drei Parametern, werden hier nur Kalibrierungen mit dem evolutionärem Algorithmus durchgeführt. Dafür wird eine Populationsgröße von 50 und eine Rekombinationsrate von 0.25 gewählt, kombiniert mit den Werten $0.3, 0.4$ und 0.5 als Mutationsrate. Diese Konfigurationen werden jeweils mit den unterschiedlichen Updatemethoden getestet. Wie bei der Kalibrierung in Kapitel 4.4, werden auch hier 1500 Modellaufrufe innerhalb einer Kalibrierung durchgeführt. Die Resultate sind in Abbildung 5.6 zu sehen.

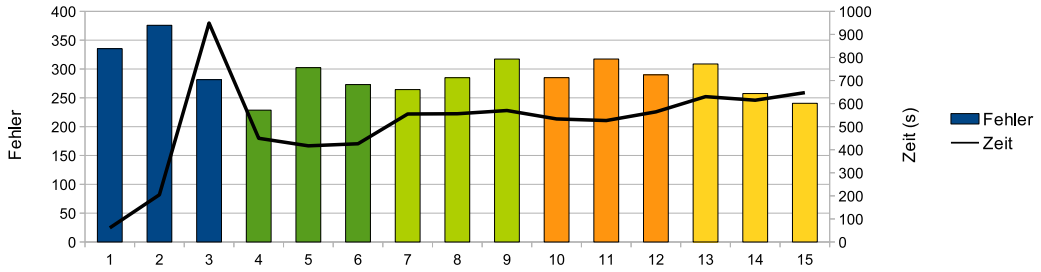

Abbildung 5.6: Resultate der Kalibrierung des SIRS-Modells mittels evolutionärem Algorithmus unter Anwendung der Updatekonfigurationen aus Tabelle 5.1

Wie bei der Kalibrierung des *SIR*-Modells mit drei Parametern, sind auch hier die Fehlerwerte durchgehend größer. Die Erhöhung der Agentenzahl während der Kalibrierung verschlechtert die Ergebnisse jedoch nur gering gegenüber der Standardkalibrierung und es ist nicht offensichtlich, welche Wachstumsart bzw. Anzahl an Updates vorzuziehen ist. Interessant ist, dass sowohl die Anzahl erfolgreicher Kalibrierungen und die Anzahl an Kalibrierungen, die im lokalen Minimum bei $d = 0$ hängen bleiben, scheinbar nicht von der gewählten Updatekonfiguration abhängt. Die Standardkalibrierung liefert hier keine besseren Ergebnisse als Kalibrierungen mit einer Erhöhung der Agentenzahl. Die Werte sind in Tabelle 5.2 zu sehen.
Des Weiteren unterliegt die Entwicklung des Fehlers keinem unerwarteten Verhalten, wie das in Kapitel 5.2 der Fall war. Einige typischer Fehlerverläufe sind in Abbildung 5.7 zu sehen.

Dementsprechend sind die höheren Fehlerwerte also nur darauf zurückzuführen, dass die Kalibrierung innerhalb der gewählten Zeit noch nicht abgeschlossen war. Zu dieser Überlegung wurden weitere Tests durchgeführt. Hierbei wurde die Anzahl der Modellaufrufe auf 2000 erhöht. Die Ergebnisse sind in Abbildung 5.8 darge-

Konfig.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
erfolg.	0	1	3	5	0	2	1	2	3	3	0	1	0	2	2
d=0	3	4	1	1	1	3	2	1	3	2	2	2	3	0	1

Tabelle 5.2: Anzahl erfolgreicher Kalibrierungen und Kalibrierung mit d=0 des SIRS-Modells mittels evolutionärem Algorithmus unter Anwendung der Updatekonfigurationen aus Tabelle 5.1

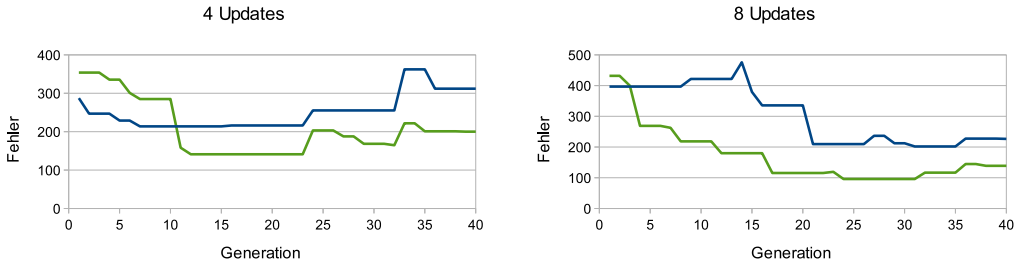

Abbildung 5.7: Verschiedene Entwicklungen des Fehlers im Laufe einer Kalibrierung mittels evoutionärem Algorithmus und Erhöhung der Agentenzahl

stellt. Die Updatekonfiguration **3** entspricht dabei der Standardkalibrierung mit 1500 Modellaufrufen.

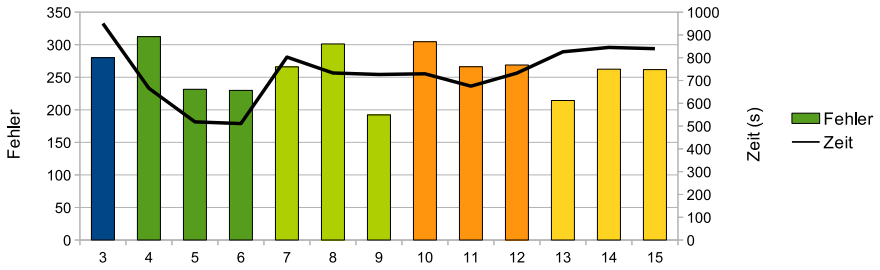

Abbildung 5.8: Resultate der Kalibrierung des SIRS-Modells mittels evolutionärem Algorithmus unter Anwendung der Updatekonfigurationen aus Tabelle 5.1 und 2000 Modellaufrufen

Die Ergebnisse sind etwas besser als die in Abbildung 5.2. In den meisten Fällen ist der Fehler, der mittels Agentenerhöhung berechnet wurde, kleiner als der Fehler der Standardkalibrierung. Allerdings ist erwartungsgemäß auch die benötigte Rechenzeit gestiegen, so dass der Zeitgewinn durch die Agentenerhöhung nicht mehr so signifikant ist. Dementsprechend muss abgeschätzt werden, ob die kleine Verbesserung des Fehlers diesen Mehraufwand Wert ist.

6 Anwendung auf Influenzamodell

6.1 Erklärung des Modells

Bei dem Influenzamodell, das in dieser Arbeit kalibriert werden soll, handelt es sich um ein komplexes agentenbasiertes Modell, das die Grippeausbreitung innerhalb der österreichischen Bevölkerung simulieren soll. Eine sehr ausführliche Beschreibung dieses Modells befindet sich in [22]. Um die Handhabung des Modells zu vereinfachen, wurde die Programmierung in mehrere Module aufgeteilt: Bevölkerung, Kontakte, Krankheit und Auswertung.

Das Bevölkerungsmodul generiert die Agenten, die die einzelnen Personen darstellen. Jedem Agenten wird dabei ein Alter und ein Geschlecht zugeordnet und es wird abhängig vom Alter entschieden, ob es sich um ein Kleinkind, ein Schulkind, einen Erwachsenen oder einen Pensionisten handelt. Bei Erwachsenen wird zusätzlich noch entschieden, ob der Agent erwerbstätig oder arbeitslos ist. Die Verteilung dieser Werte orientiert sich an der realen demographischen Verteilung der österreichischen Bevölkerung. Die dazu benötigten Daten wurden von der *Statistik Austria* bezogen. Außerdem ermöglicht das Bevölkerungsmodul den Agenten, zusätzliche Informationen vom Kontakt- und vom Krankheitsmodul zu beziehen. So erhält der Agent eine Beschreibung seines sozialen Verhaltens und seines Krankheitszustandes.
Im Rahmen des Kontaktmoduls werden Orte erschaffen, an denen die Agenten sich aufhalten können. Dabei wird zwischen vier verschiedenen Möglichkeiten der Aufenthaltsorte unterschieden: Haushalt, Schule, Arbeitsstätte und Freizeit. Unabhängig vom Alter wird jeder Person ein Haushalt zugeordnet, sodass die Haushaltszusammenstellungen die bekannten statistischen Werte widerspiegeln. Es wird allerdings darauf geachtet, dass jedem Haushalt mit einem Kind auch mindestens eine erwachsene Person zugehört. Anschließend wird den Personen gemäß ihrem Alter und ihrer Erwerbstätigkeit eine Schule oder eine Arbeitsstätte zugewiesen. Pensionierten oder arbeitslosen Personen sowie Kleinkindern wird dabei kein Ort zugeordnet. Bei der Freizeit handelt es sich um einen einzigen Ort, der nicht weiter unterteilt wird und in dem sich alle Agenten aufhalten können.
Zusätzlich zur Erschaffung der Orte, generiert das Kontaktmodul Informationen, wie die Personen innerhalb dieser Orte miteinander interagieren. Für jede Ortsart gibt es eine Anzahl an Kontakten, die Personen innerhalb dieser Umgebung pro Tag durchschnittlich erfahren. Die Auswahl der miteinander in Kontakt tretenden Personen kann dabei altersabhängig erfolgen. So haben beispielsweise in der Freizeit eher Personen miteinander Kontakt, die sich in der selben Alterskategorie befinden.

Alle Informationen, die den Infektionsprozess betreffen, werden im Krankheitsmodul behandelt. Hier ist abgespeichert, wie viele Personen anfänglich infiziert oder natürlich immun sind und welcher Anteil der Bevölkerung geimpft wird. Außerdem teilt es den einzelnen Agenten mit, in welchem Krankheitszustand sie sich gerade befinden. Dabei wird zwischen den Klassen suszeptibel, infektiös und resistent unterschieden. Zusätzlich kann festgelegt werden, ob Symptome auftreten, wie stark diese sind und wie lange sie andauern werden. Abhängig davon wird entschieden, ob die Person zuhause bleibt oder nicht. Bei starken Symptomen bleibt die Person in jedem Fall der Arbeit oder der Schule fern, bei milderen Symptomen eher selten. Des Weiteren legt das Krankheitsmodul die Stärke der Infektiosität für die einzelnen Personen fest. Dies geschieht altersabhängig, da beispielsweise bei Kindern die Infektionswahrscheinlichkeit bei einem Kontakt mit einer anderen Person höher ist als bei einem Erwachsenen.
Schlussendlich gibt es noch das Auswertungsmodul. Seine Aufgabe ist das Erzeugen und Darstellen der berechneten Daten. Dabei kann eine Vielfalt an Informationen abgespeichert werden. Zum einen kann die Anzahl an Agenten betrachtet werden, die einem gegebenen Zeitpunkt suszeptibel, infektiös oder resistent sind, sowie jene, die milde oder starke Symptome vorweisen. Zum anderen kann man auch für jeden Tag die Anzahl an Neuzugängen zu diesen Gruppen betrachten. Außerdem ist es oft nützlich, die erwähnten Daten nicht tageweise, sondern wochenweise gruppiert darzustellen. Des Weiteren kann das Auswertungsmodul eine Epidemie erkennen. Dazu wird das Maximum der Ergebniskurve bestimmt und nur noch Daten in einer gewissen Zeitspanne um das Maximum herum betrachtet. Diese Zeitspanne kann im Vorhinein festgelegt worden sein. Sie kann aber auch abhängig von den Daten bestimmt werden, je nachdem wann die berechneten Werte eine gewisse Schranke unterschreiten.

Die Daten, bezüglich denen das Influenzamodell kalibriert werden soll, sind in Abbildung 6.1 dargestellt. Sie stammen von der Grippewelle des Jahres 2007. Die Kurve zeigt die Neuinfizierten pro Woche. Nun stellt sich die Frage, welche der vom Modell erzeugten Resultate an diese Kurve angepasst werden sollen. Da die Daten zur Grippewelle von Ärzten erhoben wurden, erscheint es sinnvoll, nur diejenigen Personen zu betrachten, die starke Symptome verspüren, da nur diese einen Arzt aufsuchten und in der Statistik auftauchten.

Abbildung 6.1: Neuinfizierte pro Woche während der Grippewelle des Jahres 2007

6.2 Kalibrierung

6.2.1 Vorbereitungen und bisherige Ansätze

Bei diesem Modell sollen acht verschiedene Parameter durch die Kalibrierung bestimmt werden:

- die Anzahl an Kontakten im Haushalt
- die Anzahl an Kontakten in der Schule
- die Anzahl an Kontakten in der Arbeit
- die Anzahl an Kontakten in der Freizeit
- der Anteil der Bevölkerung an natürlich immunen Personen
- die Infektionswahrscheinlichkeit
- der Anteil an allen Erkrankungen, die infektiös und ohne Symptome verlaufen
- der Anteil an allen Erkrankungen, die infektiös und mit Symptomen verlaufen

Hierbei ist zu beachten, dass bei den letzten beiden Parametern die Einschränkung gilt, dass die Summe dieser Werte kleiner als 1 sein muss. Die fehlende Differenz beschreibt dann den Anteil an Erkrankungen, die nicht infektiös sind.
Der Verdacht liegt nahe, dass der Epidemieverlauf nur von der Gesamtanzahl an Kontakten abhängt und nicht von der spezifischen Verteilung der Kontakte auf die verschiedenen Orte. Deshalb werden die Parameter für die Kontaktwerte zusammengefasst. Das heißt, für jeden Ort wird eine Anzahl an Kontakten festgelegt und mittels Kalibrierung wird ein Faktor bestimmt, mit dem dieser Wert multipliziert wird. Dies reduziert die Zahl der zu bestimmenden Parameter auf fünf.

Vor dem Verfassen dieser Diplomarbeit wurden noch keine Algorithmen benutzt, um dieses Modell zu kalibrieren. Stattdessen wurde versucht, die Simulationsergebnisse händisch an die Datenkurve anzupassen. Um diese Arbeit zu erleichtern, wurde versucht, die passenden Parameterwerte mithilfe verschiedener Studien und Statistiken zu bestimmen [22]. Die resultierenden Werte stehen in Tabelle 6.1. Ruft man nun die

Simulation mit diesen Parameterwerten auf und variiert die Infektionswahrscheinlichkeit, ergeben sich die Kurve, aus Abbildung 6.2.

Kontakte im Haushalt	2.50
Kontakte in der Arbeit	5.28
Kontakte in der Schule	4.64
Kontakte in der Freizeit	6.11
Anteil natürlich immuner Personen	0.15
Anteil der infektiösen Erkrankungen ohne Symptome	0.29
Anteil der infektiösen Erkrankungen mit Symptomen	0.62

Tabelle 6.1: Mögliche Parameterwerte, die mittels verschiedener Studien ermittelt wurden

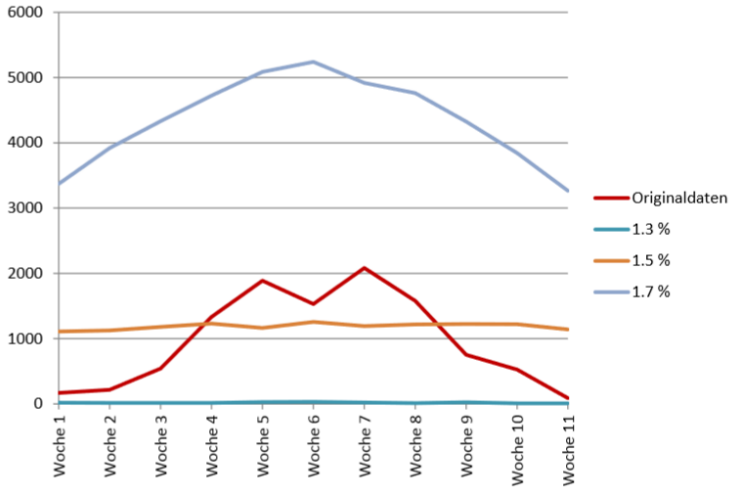

Abbildung 6.2: Anzahl an Neuinfizierten mit schweren Symptomen bei Variation der Infektionswahrscheinlichkeiten

Man erkennt, dass die Ergebnisse nicht zufriedenstellend sind. Entweder werden zu viele oder zu wenige Personen infiziert oder die Kurve der Neuinfizierten hat nicht die typische Form einer Epidemie, welche in der Mitte einen Höhepunkt hat und am Anfang und am Ende abflacht.

Um diese Simulationsergebnisse besser zu erklären, wurden weitere Parameterstudien angestellt um den Einfluss der einzelnen Parameter besser zu verstehen. Dabei wurden jeweils die Werte für die Anzahl der natürlich immunen Personen, die Infektionswahrscheinlichkeit und den Anteil an infektiösen, symptomlosen Erkrankungen unabhängig voneinander variiert. Es ergaben sich die Kurven aus Abbildung 6.3 bis 6.5.

Zusammenfassend ergibt sich, dass eine Erhöhung der natürlich immunen Personen dazu führt, dass die Epidemie länger, jedoch flacher verläuft, während eine Erhöhung der Infektionswahrscheinlichkeit und der infektiösen, symptomlosen Erkrankungen eine kürzere, jedoch steilere Epidemiekurve hervorruft. Ausgehend von den Werten

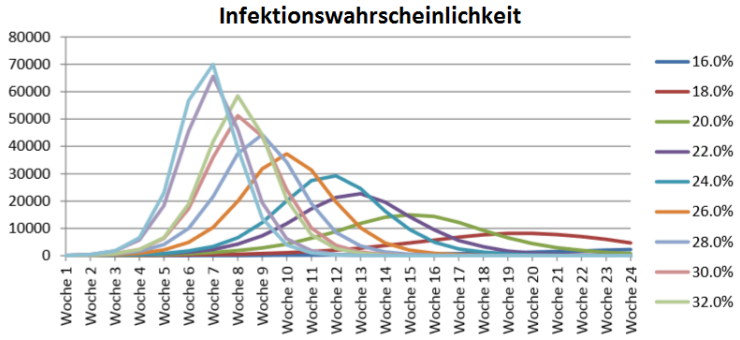

Abbildung 6.3: Anzahl der Infizierten mit schweren Symptomen bei Variation der Infektionswahrscheinlichkeit

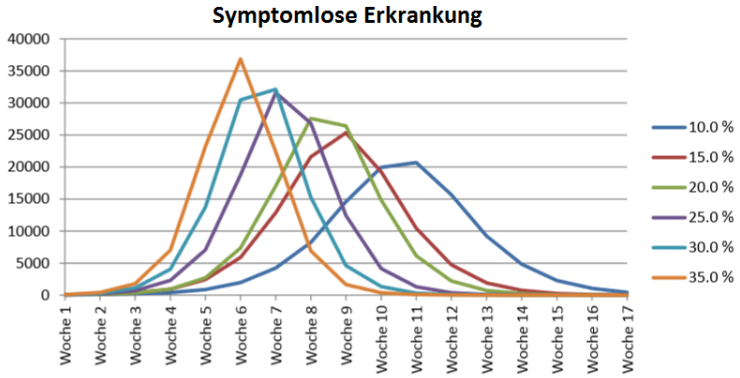

Abbildung 6.4: Anzahl der Infizierten mit schweren Symptomen bei Variation des Anteil an symptomatischen Erkrankungen ohne Symptome

in Tabelle fehlt, wurde anfangs versucht diese Parameter händisch zu variieren, um die Simulationsergebnisse an die echte Daten anzugleichen. Dies war jedoch nicht erfolgreich und somit war klar, dass ein Kalibrierungsalgorithmus benutzt werden muss.

Abschließend muss noch eine Fehlerfunktion bestimmt werden. Diese hat die Form aus 2.2. Dabei werden die Gewichte so gewählt, dass der Anfang und das Ende der Epidemie nicht so stark bewertet werden. Außerdem erscheint es unnatürlich, dass in der Mitte des Epidemieverlaufs ein kurzfristiger Rückgang an Neuinfizierten stattfindet und dann wieder hohe Werte erzielt werden. Deshalb handelt es sich bei den Zahlen der sechsten Woche vermutlich um einen Fehler in den Daten. Also wird auch dieser Punkt sehr gering bewertet. Es ergeben sich die Gewichte aus Tabelle 6.2.

Woche	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Gewicht	1	4	8	16	42	4	42	16	8	4	1

Tabelle 6.2: Werte der Gewichtsfunktion

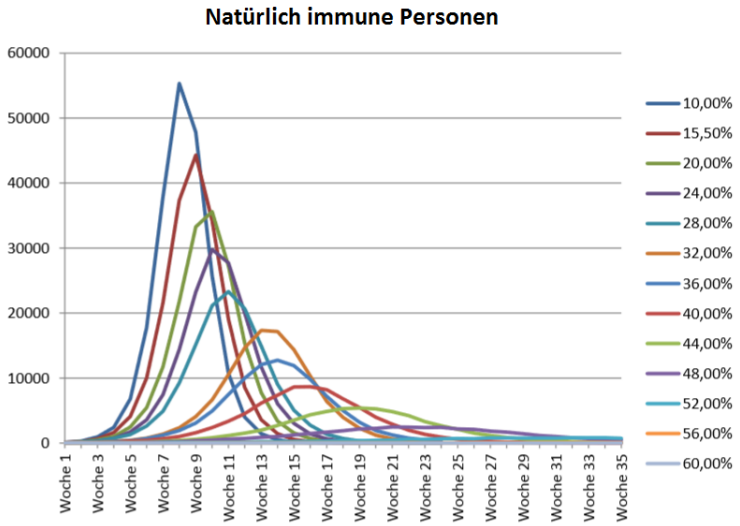

Abbildung 6.5: Anzahl der Infizierten mit schweren Symptomen bei Variation des Anteils an natürlich immunen Personen

6.2.2 Kalibrierung mithilfe Kalibrierungsalgorithmen

Nun wird dieses Influenzamodell mit den Methoden aus den vorherigen Kapiteln kalibriert. Um die Zuverlässigkeit der Resultate besser bewerten zu können, werden mehrere Kalibrierungen durchgeführt. Als Kalibrierungsalgorithmus wird ein evolutionärer Algorithmus mit 40 Genomen, einer Rekombinationsrate von 0.25 und einer Mutationsrate von 0.5 gewählt. Gestartet wird mit 50 000 Agenten, die im Lauf der Kalibrierung geometrisch auf 800 000 Agenten erhöht werden. Die eigentlich erwünschte Agentenzahl beträgt 8 300 000 Agenten, also der Größe der österreichischen Bevölkerung. Aus Gründen der Simulationsdauer wird jedoch auf diese hohe Agentenzahl verzichtet. Die Daten aus Abbildung 6.1 werden dementsprechend umgerechnet. Des Weiteren wird auch der Fehlerwert der einzelnen Simulationsdurchläufe der benutzten Agentenzahl entsprechend skaliert. Als Skalierungsfaktor wird hier die skalierte Gesamtanzahl der Neuinfizierten während der Grippewelle des Jahres 2007 benutzt. Das heißt: bei 8 300 000 Personen gab es insgesamt 107 043 Neuinfizierungen. Also wird bei beispielsweise 100 000 Agenten der Fehler mit einem Faktor $\frac{107043 \cdot 100000}{8300000} \approx 1290$ skaliert. Eine ähnliche Vorgehensweise wurde bereits in [22] gewählt.

Insgesamt wurden sechs Kalibrierungen berechnet, von denen jeweils zwei dieselbe Konfiguration benutzten. Tabelle 6.3 beschreibt die durchgeführten Kalibrierungen. Es sind jeweils die Anzahl an Agentenupdates angegeben, der Fehler der Kalibrierung und die berechneten Werte für die Parameter. Abbildung 6.6 zeigt zusätzlich den Vergleich der berechneten Epidemiekurve mit der zu reproduzierenden Datenkurve.

	n_u	Fehler	Kontakt-faktor	natürlich immun	Infektions-wahrsch.	infektiös ohne Symptome	infektiös mit Symptome
1	4	17.15	0.80	0.84	0.64	0.15	0.07
2	4	21.88	0.08	0.59	0.45	0.73	0.08
3	6	25.09	0.23	0.55	0.15	0.65	0.09
4	6	31.80	1.39	0.89	0.45	0.23	0.11
5	8	18.46	1.83	0.90	0.29	0.28	0.22
6	8	18.51	0.19	0.82	0.58	0.64	0.23

Tabelle 6.3: Fehler und Parameterwerte der Kalibrierungen des Influenzamodells

Laut Abbildung 6.6 kann die Kalibrierung die Datenkurve sehr gut reproduzieren. Die Probleme einer generell zu niedrigen oder zu hohen Anzahl an Infizerten, die in Abbildung 6.2 dargestellt sind, treten nicht auf. Demnach kann die Kalibrierung als erfolgreich angesehen werden. Betrachtet man jedoch die Tabelle 6.3, so erkennt man, dass die berechneten Parameterwerte sehr unterschiedlich sind. Also scheinen mehrere gleichwertige, lokale Minima zu existieren.
Vergleicht man die berechneten Parameterwerte mit den als plausibel erscheinenden Werten aus Tabelle 6.1, so fällt Folgendes auf: der berechnete Werte für den Anteil an natürlich immuner Personen ist durchwegs signifikant größer als der durch Studien bestimmte Wert, während der durch Kalibrierung bestimmte Anteil an infektiösen Erkrankungen ohne Symptome generell kleiner ist.

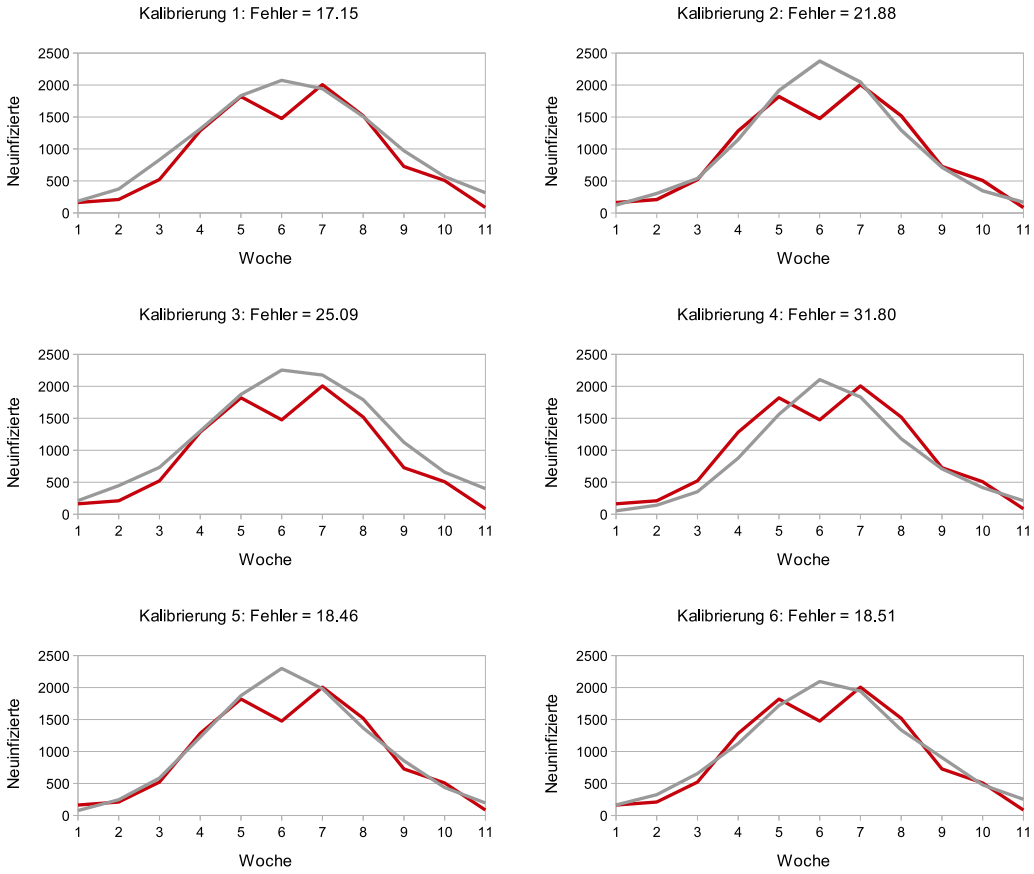

Abbildung 6.6: Vergleich der reellen Daten (rot) mit der berechneten Epidemiekurve (grau) mit den Kalibrierungen aus Tabelle 6.3

7 Zusammenfassung und Ausblick

In dieser Arbeit wurden die Grundzüge der Modellkalibrierung präsentiert. Es wurde erklärt, dass ein Kalibrierungsproblem als Optimierungsproblem angesehen werden kann und es von Vorteil ist, Algorithmen für diese Aufgabe zu verwenden. Des Weiteren wurde auf die Probleme eingegangen, die speziell bei der Kalibrierung agentenbasierter Modelle auftreten.
Anschließend wurden zwei Algorithmen vorgestellt, die zur Kalibrierung ebendieser Modelle herangezogen werden können: das Simulated Annealing und ein evolutionärer Algorithmus. Die einzelnen Komponenten dieser Algorithmen wurden beschrieben und, soweit möglich, das Konvergenzverhalten diskutiert. Die Leistung der beiden Algorithmen wurde an zwei verschiedenen, einfachen agentenbasierten Modellen getestet. Es wurden jeweils verschiedene Konfigurationen angewendet und mithilfe des erzielten Fehlerwertes miteinander verglichen.
In einem nächsten Schritt wurde eine Strategie zur Verkürzung der Laufzeit vorgestellt. Die Idee besteht darin, die Kalibrierung mit einer geringen Anzahl an Agenten zu starten und sie im weiteren Verlauf progressiv zu erhöhen, bis die gewünschte Agentenzahl erreicht ist. Dies führt dazu, dass am Anfang der Kalibrierung eine größere Variabilität der Simulationsergebnisse vorliegt, diese Resultate jedoch auch mit einer erheblich geringeren Laufzeit berechnet werden. Es wurden verschiedene Updatekonfigurationen an den Beispielmodellen getestet und anhand der benötigten Laufzeit und des erhaltenen Fehlerwertes beurteilt.
Schlussendlich wurde eine gute Konfiguration des evolutionären Algorithmus benutzt, um ein komplexeres agentenbasiertes Modell zu kalibrieren.

Es hat sich gezeigt, dass die beiden vorgestellten Kalibrierungsalgorithmen beim einfachen SIR-Modell zufriedenstellende Ergebnisse liefern. Beim Simulated Annealing fiel jedoch auf, dass die Konvergenz sehr viel von der gewählten Nachbarschaft abhängt. Nur eine passende Wahl hat zu guten Resultaten geführt. Die Kalibrierungsresultate, die beim SIR-Modell mit drei Parametern mithilfe vom Simulated Annealing berechnet wurden, waren ebenfalls schlechter als diejenigen, die vom evolutionären Algorithmus geliefert wurden. Auch bei der Kalibrierung des $SIRS$-Modells hat sich herausgestellt, dass die Funktionsweise des evolutionären Algorithmus vorteilhafter ist. Dadurch, dass dieser mit mehreren Kandidaten gleichzeitig arbeitet, war es ihm eher möglich, das lokale Minimum zu verlassen und einen geringeren Fehlerwert zu erreichen.
Auch bei der progressiven Erhöhung der Agentenzahl zur Verringerung der Laufzeit lieferte das Simulated Annealing insgesamt schlechtere Ergebnisse. Die Hoffnung, dass auch eine niedrige Agentenzahl die Kalibrierung in die richtige Richtung weisen

würde, hat sich leider nicht immer erfüllt und die Fehlerwerte der Standardkalibrierung konnten nicht reproduziert werden. Der evolutionäre Algorithmus lieferte bessere Resultate. Beim SIR-Modell wurden genauso niedrige Fehlerwerte erzielt, wie bei einer Standardkalibrierung. Die benötigte Rechenzeit betrug dabei nur etwa die Hälfte. Für die Kalibrierung des SIR-Modells mit drei Parametern ergab sich beim evolutionären Algorithmus ein ähnliches Bild. Auch hier wurden in etwa der halben Laufzeit Fehlerwerte in derselben Größenordnung wie bei einer Standardkalibrierung berechnet. Bei der Kalibrierung des $SIRS$-Modells waren die Resultate leider nicht ganz so gut. Um vergleichsweise Fehlerwerte wie bei der Standardkalibrierung zu erzielen, konnte die Laufzeit nur um 25% gesenkt werden.
Bei der Kalibrierung des Influenzamodells mithilfe des evolutionären Algorithmus hat sich gezeigt, dass diese Vorgehensweise deutlich bessere Ergebnisse liefert als diejenige, die bisher bei einer händischen Kalibrierung erzielt wurden. Bei mehreren Durchläufen wurden jeweils erfolgreich die Simulationsergebnisse an die reellen Daten angepasst. Bei der Betrachtung der berechneten Parameterwerte hat sich jedoch gezeigt, dass mehrere lokale Minima existieren. Außerdem unterscheiden sich einige Parameter stark von den Werten, die zuvor anhand verschiedener Studien und Statistiken als plausibel erachtet wurden.

Insgesamt konnte festgestellt werden, dass die Verwendung von Algorithmen eine geeignete Methode ist, um ein agentenbasiertes Modell zu kalibrieren. Der langen Rechenzeit kann dabei durch die Erhöhung der Agentenzahl im Verlauf der Kalibrierung entgegengewirkt werden. Diese Vorgehensweise scheint großes Potential zu haben und liefert noch einige offene Fragen, die es sich lohnt, zu untersuchen. So besteht vermutlich die Möglichkeit, dass die Leistung des Simulated Annealing durch ein anderes Kühlungsverfahren verbessert werden kann. Des Weiteren wäre es interessant, andere Updatekonfigurationen zu testen. So könnten beispielsweise die Zeitpunkte des Modellupdates anders festgelegt werden. Es wäre sicherlich von Vorteil, diese Zeitpunkte abhängig vom Verhalten des Fehlers zu bestimmen. In diesem Zusammenhang wäre es auch wünschenswert, geeignetere Abbruchkriterien zu bestimmen und so zusätzlich Rechenzeit einzusparen. Allerdings stellt sich bei einem adaptiven Verfahren die Frage, wie sichergestellt werden kann, dass die gewünschte Zielanzahl an Agenten genau erreicht wird.
Darüber hinaus ist es erforderlich, dieses Verfahren noch an anderen Modellen zu testen. Diese sollten sich im Aufbau von den in dieser Arbeit verwendeten Modellen unterscheiden. Dabei soll untersucht werden, ob diese Kalibrierungsmethode auch bei diesen Modellen funktioniert und welche Modelleigenschaften für eine gute Leistung förderlich sind.
Generell gilt, dass ein größeres Vorwissen über das Modell die Kalibrierung erleichtern würde. Da sich gezeigt hat, dass eine größere Anzahl an zu kalibrierenden Parametern die Leistung der Kalibrierungsalgorithmen verschlechtert, wäre es wünschenswert, geeignete Methoden der Sensitivitätsanalyse zu finden, die auf agentenbasierte Modelle angewendet werden können. Auf diese Weise könnte bestimmt werden, welche Parameter einen großen Einfluss auf das Ergebnis der Simulation haben. Dementsprechend müssten nur diese Parameter während einer Kalibrierung bestimmt werden.

Literaturverzeichnis

[1] E. Aarts, J. Korst, and W. Michiels, *Simulated annealing*, Search methodologies, Springer, 2005, pp. 187–210.

[2] D. Abramson, M. Krishnamoorthy, and H. Dang, *Simulated annealing cooling schedules for the school timetabling problem*, 1997.

[3] C. Blum and A. Roli, *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*, ACM COMPUTING SURVEYS (2003), 268–308.

[4] E. Bonabeau, *Agent-based modeling: Methods and techniques for simulating human systems*, Proceedings of the National Academy of Sciences **99** (2002), no. suppl 3, 7280–7287.

[5] B. Calvez and G. Hutzler, *Automatic tuning of agent-based models using genetic algorithms*, Multi-agent-based simulation VI, Springer, 2005, pp. 41–57.

[6] V. Cerny, *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*, Journal of Optimisation Theory Application **45** (1985), no. 1, 41–51.

[7] J. Dreo, *Different classifications of metaheuristics*, `https://upload.wikimedia.org/wikipedia/commons/9/94/Metaheuristics_classification_fr.svg`, 2007, [Online; accessed 9-August-2016].

[8] A. E. Eiben, P.-E. Raue, and Zs. Ruttkay, *Genetic algorithms with multi-parent recombination*, Parallel Problem Solving from Nature-PPSN III, Springer, 1994, pp. 78–87.

[9] D. E. Goldberg and K. Deb, *A comparative analysis of selection schemes used in genetic algorithms*, Foundations of genetic algorithms **1** (1991), 69–93.

[10] D. E. Goldberg et al., *Genetic algorithms in search optimization and machine learning*, vol. 412, Addison-wesley Reading Menlo Park, 1989.

[11] J. J. Grefenstette, *Optimization of control parameters for genetic algorithms*, Systems, Man and Cybernetics, IEEE Transactions on **16** (1986), no. 1, 122–128.

[12] B. Hajek, *Cooling schedules for optimal annealing*, Mathematics of operations research **13** (1988), no. 2, 311–329.

[13] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press, 1975.

[14] J. M. Hyman and J. Li, *Epidemic models with differential susceptibility and staged progression and their dynamics*, Mathematical Biosciences and Engineering **6** (2009), no. 2, 321–332.

[15] PHM Janssen and PSC Heuberger, *Calibration of process-oriented models*, Ecological Modelling **83** (1995), no. 1, 55–66.

[16] N. R. Jennings and M. Wooldridge, *On agent-based software engineering*, Artificial Intelligence **117** (2000), 277–296.

[17] W. O. Kermack and Ag McKendrick, *A contribution to the mathematical theory of epidemics*, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character **115** (1927), no. 772, 700–721.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, SCIENCE **220** (1983), no. 4598, 671–680.

[19] C. Y. Kong, P. M. McMahon, and G. S. Gazelle, *Calibration of disease simulation model using an engineering approach*, Value in Health **12** (2009), no. 4, 521–529.

[20] C. Macal and M. North, *Tutorial on agent-based modeling and simulation*, Proceedings of the 2005 Winter Simulation Conference, 2005, pp. 2–15.

[21] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, The Journal of Chemical Physics **21** (1953), no. 6, 1087–1092.

[22] F. Miksch, G. Zauner, P. Pichler, C. Urach, and N. Popper, *Influenza - modellierung und simulation von influenza-epidemien*, Tech. report, dwh simulation services, 2013.

[23] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes 3rd edition: The art of scientific computing*, 3 ed., Cambridge University Press, New York, NY, USA, 2007.

[24] I. Rechenberg, *Optimierung technischer systeme nach prinzipien der biologischen evolution*, 1970.

[25] M. Srinivas and L. M. Patnaik, *Adaptive probabilities of crossover and mutation in genetic algorithms*, Systems, Man and Cybernetics, IEEE Transactions on **24** (1994), no. 4, 656–667.

[26] K.-H. Waldmann and U. M. Stocker, *Stochastische modelle: Eine anwendungsorientierte einführung*, Springer-Verlag, 2012.

[27] T. Weise, *Global optimization algorithms - theory and application*, 2008.

[28] D. H. Wolpert and W. G. Macready, *No free lunch theorems for optimization*, Evolutionary Computation, IEEE Transactions on **1** (1997), no. 1, 67–82.

[29] M. Wooldridge, *Agent-based software engineering*, IEE Proceedings on Software Engineering, 1997, pp. 26–37.