Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Alena Störmer

Eine Analyse politikwissenschaftlicher Modellbildung: Parametervariation am Beispiel des Emergent Polarity Model

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Alena Störmer

Eine Analyse politikwissenschaftlicher Modellbildung: Parametervariation am Beispiel des Emergent Polarity Model

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Michael Köhler-Bußmeier
Zweitgutachter: Prof. Dr. Julia Padberg

Eingereicht am: 22. März 2018

Alena Störmer

Thema der Arbeit

Eine Analyse politikwissenschaftlicher Modellbildung: Parametervariation am Beispiel des Emergent Polarity Model

Stichworte

Komplexe Adaptive Systeme, Agenten, Emergent Polarity Model, Simulation

Kurzzusammenfassung

In politikwissenschaftlichen Simulationen werden komplexe Sachverhalte basierend auf formalisierten Theorien modelliert. Im Schritt von der Theorie zum programmierten Modell können Fehler und Artefakte entstehen, die die Aussagekraft des Ergebnisses schwächen. Ein Beispiel für solche Artefakte ist Parametersensibilität: Die Ergebnisse des Modells ändern sich abhängig davon welche Werte dem System übergeben werden, oder das Modell funktioniert nur mit wenigen Werten. Diese Problematik wird beispielhaft am von Cederman (1997) entwickelten Emergent Polarity Model untersucht. Dieses Modell vom Entstehen und Zerfallen von Staaten wird nachvollzogen und mit zufälligen Parameterwerten initialisiert um zu prüfen, welche Ergebnisse entstehen wenn das Modell nicht nur mit optimalen Werten arbeitet.

Alena Störmer

Title of the paper

An Analysis of Modelling in Political Science: Variation of Parameters Using the Example of the Emergent Polarity Model

Keywords

Complex adaptive systems, agents, emergent polarity model, simulation

Abstract

Simulations in the political sciences model complex issues, based on formalised theories. When converting such theories into programmes, errors or artifacts can manifest and distort the simulation's results. One example of such artifacts is argument sensitivity: The model's results change with different values passed to the system, or the model only works with few argument values. This issue is examined on the Emergent Polarity Model, developed by Cederman (1997), which models emergence and collapse of states. This model is recreated and initialised with random values to investigate the model's results when using suboptimal argument values.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1. Einleitung

1.1. Einleitung

1.1.1. Kontext

Experimente sind meist ein grundlegender Bestandteil wissenschaftlichen Erkenntnisgewinns. In den Sozialwissenschaften wird experimentelle Forschung jedoch nur selten betrieben. Dies ist unter anderem der Tatsache geschuldet, dass die Zielobjekte nur schwer unter isolierten Bedingungen betrachtet und beeinflusst werden können. Dennoch können Experimente auch in den Sozialwissenschaften zu neuen Erkenntnissen führen, zum Beispiel über Ursache-Wirkungs-Beziehungen (Gilbert und Troitzsch, 2005). Soll eine sozialwissenschaftliche Theorie experimentell erforscht werden, bietet sich ein Agentensystem zur Modellierung an (Wooldridge, 2009, S.22). In dem Schritt von der Theorie zum Modell muss diese formalisiert werden, wobei zwangsläufig willkürliche Entscheidungen auftreten (Gilbert und Troitzsch, 2005, S.24). Diese können zum Beispiel numerische Werte betreffen, die notwendig sind damit das Modell sinnvolle Ergebnisse erzielt. Ist der Definitionsbereich dieser Werte zu stark eingeschränkt, bedeutet dies häufig auch eine beschränkte Funktionalität und Aussagekraft des Modells.

1.1.2. Fragestellung

Diese Ad-Hoc-Entscheidungen bei der Modellierung führen zur Frage nach der Robustheit des Modells: Nach welchen Kriterien werden diese willkürlichen Werte ausgewählt, und wird durch sie die Aussagekraft des Modells geschwächt? Sollen aus dem Modell weitere Theorien abgeleitet, oder Aussagen in die reale Welt übertragen werden, ist es außerdem relevant, welche Werte die Realität am genauesten abbilden (Lazer, 2001).

Die Relevanz dieser Problematik begründet sich vor allem im wachsenden Interesse der Sozialwissenschaften an Computersimulation (vgl. Gilbert und Troitzsch (2005), S.6ff). In der untersuchten Literatur wird zudem die Frage nach dem Einfluss von willkürlichen Werten auf Simulationsergebnisse nur unzureichend behandelt und soll daher genauer untersucht werden. Nicht zu vernachlässigen ist auch der Mehrwert von Simulationen für das Fachgebiet

der Informatik. Daher ist es auch hier wichtig, diese möglichst aussagekräftig modellieren zu können.

Gängige Ansätze um die Robustheit und Aussagekraft eines Modells zu maximieren und notwendige konkrete Werte möglichst sinnvoll auszuwählen, sind unter anderem die iterative Anpassung von Modellparametern und Verhaltensregeln für Agenten (Sayama, 2015, S.458).

Durch eben dieses Vorgehen soll die Frage nach der Robustheit an einem Beispiel dargestellt und exemplarisch geprüft werden. In Cederman (1997) wird das Entstehen und Zerfallen von Staaten diskutiert. Hierfür erstellt er das Emergent Polarity Model (EPM), ein Komplexes Adaptives System (CAS). Dieses wurde zwar für seine Formalisierung der ursprünglichen Theorie gelobt, jedoch stellte sich auch hier die Frage nach der Parametersensibilität (Lazer, 2001).

> „The general question that this story raises regarding modeling is one of robustness - might slight changes to the model of no apparent substantive importance greatly change the observed dynamics? This is particularly a concern where there are assumptions embedded in operational decisions regarding the simulation. [...] [I]t is a call to producing extensive robustness tests of propositions that are produced with simulations " (ebd.).

Aus dieser Kritik wird die Frage abgeleitet, wie das EPM auf schon geringe Änderungen reagiert und ob sich durch sie die Ergebnisse drastisch verändern. Dies würde auf eine fehlende *„robustness"* hindeuten. Bisher wurde das EPM zwar kritisiert, jedoch wurde diese Kritik nie am Modell auf ihren Gehalt überprüft.

1.1.3. Vorgehen und Aufbau der Arbeit

Zur Untersuchung der Fragestellung wird zunächst der aktuelle Stand der Formalisierung von sozialwissenschaftlichen Theorien in Agentensystemen geprüft. Dabei wird der Schwerpunkt auf die Frage der Parametersensibilität von Modellen gelegt. Anhand dieser Recherche wird das EPM kritisch betrachtet, und anschließend auf willkürliche Werte in der Modellierung untersucht. Geeignete Werte werden für den nächsten Schritt ausgesucht, diese Eignung bezieht sich vor allem auf den vermuteten Einfluss des Wertes auf das Gesamtmodell. Je stärker beeinflussend ein solcher Wert für das EPM ist, desto wichtiger ist auch seine Robustheit, und desto mehr Auswirkungen könnte eine Variation dieses Wertes auf das Gesamtergebnis haben.

Zunächst wird das EPM so exakt wie möglich nachgebildet. Dabei werden die im EPM fest implementierten, willkürlichen Werte im neuen Modell variabel modelliert. Diese Modell wird für den Versuchsaufbau im nächsten Schritt verwendet. Es wird für alle variablen Parameter ein

Wertebereich mit Schrittweite festgelegt. Aus diesem wird für jeden Parameter ein zufälliger Wert ausgewählt, und mit dieser Konfiguration die Simulation durchlaufen. Dies wird mehrfach wiederholt, um durch die Verwendung verschiedener Konfigurationen den Definitionsbereich der Simulationsergebnisse sichtbar zu machen. Diese Ergebnisse werden für die anschließende Auswertung verwendet.

Die Ergebnisse der Durchläufe werden mit den Resultaten des EPM verglichen. Ergeben sich zum Beispiel in der Variation der Werte sehr große Unterschiede im Ergebnis, oder weichen sämtliche Ergebnisse von denen des EPM stark ab, deutet dies auf eine Parametersensibilität des EPM hin.

Im letzten Schritt werden die Ergebnisse auf ihre Gültigkeit für Verallgemeinerungen, und Schlussfolgerungen für die generelle Formalisierung von sozialwissenschaftlichen Theorien in Agentensystemen geprüft.

Da im Rahmen dieser Arbeit nur ein Überblick über den aktuellen Stand der Forschung gegeben werden kann, besteht kein Anspruch auf Vollständigkeit. Auch können aufgrund des Umfangs der Arbeit nicht alle Willkürlichkeiten des Modells variiert werden. In einer Diskussion werden aber die Werte für das Modell ausgewählt, die als sinnvoll eingeschätzt werden. Durch diese Entscheidungen kann trotz der Einschränkungen eine gültige Aussage über die Auswirkungen der Änderungen des EPM getroffen werden.

1.1.4. Zielsetzung

Zielsetzung dieser Arbeit ist die genaue Untersuchung des EPM und der Ausweitung der gewonnenen Erkenntnisse auf politikwissenschaftliche Modelle im Allgemeinen.

Die Arbeit wird als erfolgreich gewertet, wenn sich aus dem variablen Modell eine kritische Einschätzung des EPM ableiten lässt. Diese kann entweder das ursprüngliche Modell für seine Parametersensibilität kritisieren, oder dessen Aussage unterstützen. Beides würde einen Aufschluss über die Vorgehensweise der Formalisierung des ursprünglichen Theorie geben, aus dem sich weitere Schlüsse auf die übergeordnete Frage nach der Parametersensibilität von Modellen ziehen lassen.

2. Einordnung

Bevor eine Analyse des Modells und Entwicklung eines eigenen Prototypen stattfinden kann, müssen zunächst einige Begrifflichkeiten definiert und der aktuelle Forschungsstand aufgearbeitet werden. Welche Vorgehensweisen gibt es, um aus politikwissenschaftlichen Theorien Modelle zur Simulation zu erzeugen? Hierbei können aus Gründen der Begrenzung dieser Arbeit nicht alle einschlägigen Werke untersucht werden. Durch umfassende Recherche in Sammelbänden und Hauptwerken zum Thema CAS in Sozialwissenschaften wird jedoch ein Überblick über die wichtigsten Blickpunkte des aktuellen Forschungsstands gegeben.

2.1. Begriffsdefinition

Vor Beginn der Analyse des aktuellen Forschungsstands sollen zunächst einige wichtige Begriffe definiert werden, insbesondere die im Folgenden verwendeten Fachausdrücke CAS, Agent und Parametersensibilität.

Cederman (1997) definiert ein CAS als ein „adaptive network exhibiting aggregate properties that emerge from the local interaction among many agents mutually constituting their own environment“ (S.50), was er weiter präzisiert:

> "1. A CAS exhibits emergent properties [...]
> 2. A CAS presupposes local interaction among agents rather than globally managed behaviour [...]
> 3. A CAS features a large number of agents [...]
> 4. A CAS is adaptive because of the microlevel agents' capacity to change their behaviour or because of ecological effects that influence the development of the entire system [...]"

Diese Beschreibung wird im Folgenden als Definition eines CAS übernommen.

Ein Agent eines CAS ist definiert als ein „computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its

delegated objectives“ (Wooldridge, 2009, S.29). Da im EPM die Agenten des Systems semantisch den Staaten entsprechen, werden beide Begriffe zur Beschreibung der Agenten verwendet.

Um Parametersensibilität zu definieren, muss als erstes Robustheit (engl: „*robustness*“) definiert werden: „[R]obustness of a model is how insensitive the model's prediction is to minor variations of model assumptions and/or parameter settings“ (Sayama, 2015, S.21). Ein Fehlen dieser „*robustness*“ wird als Parametersensibilität des Modells bezeichnet.

2.2. Aktueller Forschungsstand

2.2.1. Simulationen in den Sozialwissenschaften

Die Verwendung von Simulationen in den Sozialwissenschaften ist ein relativ junger Forschungsbereich:

> „Although there are isolated earlier examples, the first developments in computer simulation in the social sciences coincided with the first use of computers in university research in the early 1960s [...]. They mainly consisted of discrete event simulations or simulations based on system dynamics. [...]
>
> One approach that did blossom for some years became known as 'Simulmatics' [...]. Another approach that has thrived for more than two decades, impelled by policy concerns, is rather misleadingly called 'microsimulation' [...].
>
> Apart from microsimulation, little was heard about simulation during the 1980s, in marked contrast to the situation in the natural sciences where simulation is now a basic methodological tool. However, in the early 1990s the situation changed radically, mainly as a result of the development of multi-agent models which offered the promise of simulating autonomous individuals and the interactions between them. These opportunities came from techniques imported from the study of nonlinear dynamics and from artificial intelligence research. [...]
>
> Another approach that has been influenced by ideas from physics is multilevel modeling [...] which has taken its inspiration from the theory of synergetics, originally developed for application to condensed matter physics. [...]“ (Gilbert und Troitzsch, 2005, S.6ff)

Dieser kurze Überblick über die Entwicklung von Simulationstechniken ermöglicht eine Einordnung von CAS in den größeren wissenschaftlichen Kontext. Weitere Quellen geben Aufschluss über die Entwicklung von CAS aus theoretischen Modellen.

2.2.2. Modellierung und Formalisierung von Theorien

Durch die hier aufgeworfene Fragestellung besteht besonderes Interesse an der Überführung von Theorien in CAS und Computersimulationen. Gilbert (2008) und Gilbert und Troitzsch (2005) bieten Anleitungen für die Überführung einer sozialwissenschaftlichen Theorie in ein Agentenmodell. Hier werden beispielhaft Agentensysteme in Frameworks wie *NetLogo* erzeugt. Sayama (2015) liefert eine verhältnismäßig anschauliche Anleitung:

> „A typical cycle of rule-based modeling effort goes through the following steps (which are similar to the cycle of scientific discoveries we discussed above):
>
> 1. Observe the system of your interest.
> 2. Reflect on the possible rules that might cause the system's characteristics that were seen in the observation.
> 3. Derive predictions from those rules and compare them with reality.
> 4. Repeat the above steps to modify the rules until you are satisfied with the model (or you run out of time or funding)" (S.15).

Er erkennt jedoch an, dass dieses Vorgehen nicht einfach umzusetzen sei:

> „Of course, each of the four steps has its own unique challenges, but as an educator who has been teaching complex systems modeling for many years, I find that the second step [...] is particularly challenging to modelers. This is because this step is so deeply interwoven with the modeler's knowledge, experience, and everyday cognitive processes. It is based on who you are, what you know, and how you see the world—it is, ultimately, a personal thinking process, which is very difficult to teach or to learn in a structured way." (Sayama, 2015, S.15)

Die vorgeschlagenen Ansätze lassen sich darin unterscheiden, ob sie, wie bei Sayama (2015) ein generelles Vorgehen der Modellierung, oder eine Modellierung auf Basis eines Frameworks vorschlagen: „Over the years, it has become clear that many models involve the same or similar building blocks with only small variations. Rather than continually reinventing the wheel, commonly used elements have been assembled into libraries or frameworks that can be linked into an agent-based program." (Gilbert, 2008, S.47)

Auch Cederman erkennt die Wichtigkeit von Frameworks für Agentensimulationen an:

> „The scientific success of agent-based modeling depends crucially on the availability of reliable and easy-to-use tools. While the field still has a long way to go

> until it matches the software available for statistical processing, steady progress has been made in the last few years. The first generations of scholars had to program their models from scratch in general-purpose computer languages, but more recently specialized software libraries have started to appear. Such packages provide a set of programming tools that relieve applied scientists from having to code housekeeping routines, including control panels, graphs, graphical displays, interaction spaces, and measurement routines“ (Cederman, 2001, S.19).

Aufgrund der Komplexität der Theorien, aus denen die Modelle entwickelt werden sollen, besteht keine konkrete oder allgemeine Vorgehensweise für die Modellierung einer sozialwissenschaftlichen Theorie.

In der untersuchten Fachliteratur werden meist mehrere Schritte von der Theorie zum Programm aufgeführt. Galán u. a. (2013) definieren diese Schritte nach Drogoul u. a. (2003) anhand der Personen, durch die sie ausgeführt werden. Diese sind: „the *thematician* (domain expert), the *modeller*, and the *computer scientist*“ (S.102)

> „The role of the thematician is undertaken by experts in the target domain. They are the ones that better understand the target system, and therefore the ones who carry out the abstraction process that is meant to produce the first conceptualisation of the target system. [...] The next stage in the modelling process is carried out by the role of the modeller.
>
> The modeller's task is to transform the non-formal model that the thematician aims to explore into the (formal) requirement specifications that the computer scientist – the third role – needs to formulate the (formal) executable model. [...]
>
> [Drogoul's] third role is the computer scientist. Here we distinguish between computer scientist and programmer. [...] The job of the computer scientist consists in finding a suitable (formal) approximation to the modeller's formal model that can be executed in a computer (or in several computers) given the available technology. [...]
>
> Finally, the role of the programmer is to implement the computer scientist's executable model. “ (Galán u. a., 2013, S.103ff)

Die Schritte von der Theorie zum Modell zum Programm entsprechen auch denen von Gilbert und Troitzsch (2005) und Sayama (2015). Beide berufen sich für die Überführung des Modells in ein Programm auf Frameworks:

> „Once the research questions, the theoretical approach and the assumptions have been clearly specified, it is time to begin to design the simulation. There is a sequence of issues that need to be considered for almost all simulations, and it is helpful to deal with these systematically and in order. Nevertheless, design is more of an art than a science and there is no 'right' or 'wrong' design so long as the model is useful in addressing the research question."(Gilbert und Troitzsch, 2005, S.203)

Außerhalb der Nutzung von Frameworks bestehen in der untersuchten Literatur also keine, oder nur wenige konkrete Hinweise für die Überführung des Modells in ein Programm. Eines davon liefert Sayama (2015):

> „Design tasks you need to do when you implement an ABM
>
> 1. Design the data structure to store the attributes of the agents.
> 2. Design the data structure to store the states of the environment.
> 3. Describe the rules for how the environment behaves on its own.
> 4. Describe the rules for how agents interact with the environment.
> 5. Describe the rules for how agents behave on their own.
> 6. Describe the rules for how agents interact with each other." (S.432)

Unabhängig davon wie die Modellierung erfolgt, ist es wichtig, dass bei der Überführung von der Theorie in ein Agentenmodell keine Fehler unterlaufen welche dazu führen, dass das Modell unbemerkt keine oder nur wenig Aussagekraft für die Fragestellung der Theorie hat. Diese Problematik wird hauptsächlich in Bezug auf den Schritt von einer sozialwissenschaftlichen Theorie zu logischen Regeln für ein Modell aufgegriffen. Bereits hier könnten Fehler oder Ungenauigkeiten dafür sorgen, dass das fertige Modell nur wenig Aufschluss für die ursprüngliche Fragestellung liefere. Diese Fehler oder Ungenauigkeiten entständen vor allem durch zufällig gewählte Werte.

> „[T]here are always errors when we create assumptions about, or measure parameter values from, the real world. If the prediction made by your model is sensitive to their minor variations, then the conclusion derived from it is probably not reliable. But if your model is robust, the conclusion will hold under minor variations of model assumptions and parameters, therefore it will more likely apply to reality, and we can put more trust in it."(Sayama, 2015, S.21f)

Unter Berücksichtigung dieser Schwierigkeiten wird deutlich, dass diese Aspekte bei der Modellierung beachtet werden müssen, um ein robustes Modell zu erhalten. Auf dies soll im nächsten Abschnitt genauer eingegangen werden.

2.2.3. Fehler und Parameterensibilität

In allen Schritten der Modellierung können Fehler auftreten. Hier unterscheiden Galán u. a. (2013) zwischen *error* und *artefact*: „Errors appear when a model does not comply with the requirement specifications self- imposed by its own developer. In simple words, an error is a mismatch between what the developer thinks the model is, and what it actually is. [...]

[A]rtefacts relate to situations where there is no mismatch between what the developer thinks a model is and what it actually is. Here the mismatch is between the set of assumptions in the model that the developer thinks are producing a certain phenomenon, and the assumptions that are the actual cause of such phenomenon." (Galán u. a., 2013)

Diese Fehler und Artefakte könnten an verschiedenen Stellen in der Programmierung auftreten. Sie entständen aus schlecht umgesetzten Rollen oder fehlender Kommunikation zwischen den Beteiligten:

> „The role of the computer scientist could introduce artefacts in the process. This would be the case if, for instance, his specifications require the use of a particular pseudo-random number generator, he believes that this choice will not have any influence in the results obtained, but it turns out that it does. Similar examples could involve the arbitrary selection of an operating system or a specific floating-point arithmetic that had a significant effect on the output of the model.
>
> Errors can quite easily appear in between the role of the computer scientist and the role of the programmer. Note that in our framework any mismatch between the computer scientist's specifications and the executable model received by the programmer is considered an error. In particular, if the computer scientist's specifications are not executable, then there is an error. This could be, for instance, because the computer scientist's specifications stipulate requirements that cannot be executed with present-day computers (e.g. real arithmetic), or because it does not specify all the necessary information to be run in a computer in an unequivocal way (e.g. it does not specify a particular pseudo-random number generator). The error then may affect the validity of the model significantly, or may not.
>
> Note from the previous examples that if the computer scientist does not provide a fully executable set of requirement specifications, then he is introducing an

> error, since in that case the computer program (which is executable) would be necessarily different from his specifications. On the other hand, if he does provide an executable model but in doing so he makes an arbitrary accessory assumption that turns out to be significant, then he is introducing an artefact.“ (Galán u. a., 2013, S.109f)

Entständen Fehler oder Artefakte, müssten diese gefunden und behoben werden um eine Verzerrung der Ergebnisse zu verhindern.

> „The most important strategy for finding bugs is to create test cases for which the output is known or predictable, and to run these after every change until all the test cases yield the expected results. Even this will not necessarily remove all bugs and modellers should always be aware of the possibility that their results are merely artefacts generated by their programs.
>
> Another kind of test is to compare the results from the model with data from the target (that is, from the ‘real world’ being modelled). While such comparisons are highly desirable, it is not often that they can be achieved. Often, the target is itself neither well understood nor easy to access (that this is so is one reason for building a simulation, rather than observing the target directly). In addition, the behaviour of both the target and the model may be stochastic (influenced by random events) and very sensitive to the conditions or parameters at the start [...]. If the latter is the case, even a perfect model could be expected to differ in its behaviour from the behaviour of the target. It may be possible to run the model many times to obtain a stable statistical distribution of the output, but normally it is not possible to ‘run the real world’ many times. As a result, the best one can do is to test that there is a reasonable likelihood that the observed behaviour of the target could be drawn from the distribution of outputs from the model – which is rather a weak test.
>
> The most thorough way of verifying a model (of ensuring that the output does reflect the underlying model and is not a consequence of bugs [...]) is to re-implement the model using a different programming language and, ideally, a different implementer.“(Gilbert und Troitzsch, 2005, S.212)

Hier berufen sich Gilbert und Troitzsch (2005) auf Hales u. a.:

> „It is now clear that MABS [multi-agent based simulation, Anm.] has more in common, methodologically, with the natural sciences and engineering disciplines

> than deductive logics or mathematics – it is closer to an experimental science than a formal one. With this in mind, it is important that simulations be replicated before they are accepted as correct. That is results from simulations cannot be proved but only inductively analyzed. This indicates that the same kinds of methods used within other inductive sciences will be applicable. In its simplest form a result that is reproduced many times by different modellers, re-implemented on several platforms in different places, should be more reliable. Although never attaining the status of a proof we can become more confident over time as to the veracity of the results"

Auch Cederman spricht die Problematik der Parametersensibilität im Rahmen der Herausforderungen, die ein CAS-Modell mit sich bringt, an:

> „No methodology is without its problems. Unsurprisingly, the CAS approach does not escape this rule. Because of their heavy reliance on simulation, models belonging to the CAS family often come under attack from skeptics who prefer purely deductive frameworks. While sometimes overdrawn, this critique is worth taking seriously because it does point to a series of important shortcomings. The attacks usually center on four related points:
>
> 1. Ad hoc assumptions [...]
> 2. Failure to yield unique predictions [...]
> 3. Fragility of results [...]
> 4. Lack of cumulation [...]
>
> [...] *Fragility of results.* Even when simulation models yield unique results, there is no guarantee that these are not an artifact of specific parameter configurations. Closed-form mathematical models generate findings that are verifiable for the entire parameter space. Moreover, it is often possible to study the effect of each parameter with comparative statistics, an option that is absent in simulation research since analytically differentiable expressions are lacking [...] (Cederman, 1997, S.62)"

Ob dieses Bewusstsein das EPM von Sensibilität bezüglich der Parameter schützt, soll in dieser Arbeit untersucht werden.

2.2.4. Kritik am aktuellen Forschungsstand

An dieser Stelle soll eine kurze Kritik des aktuellen Forschungsstands stattfinden. Die Unterscheidung zwischen freier und Framework-Modellierung ist deshalb wichtig, weil sie das weitere Vorgehen entscheidend beeinflussen. Wird ein Modell außerhalb eines Frameworks erzeugt, müssen mehr eigene Entscheidungen getroffen werden. Wird das Modell mit Hilfe eines Frameworks erstellt, besteht allerdings die Möglichkeit, dass Modellierungsentscheidungen nicht aus Basis der zugrundeliegenden Theorie getroffen werden, sondern durch vom Frameworks auferlegte Constraints oder Vorlagen.

Zudem geht keine der untersuchten Quellen in der Tiefe auf die Fragestellung dieser Arbeit ein, also wie eine Parametersensibilität von Modellen im Schritt vom theoretischen Modell zur programmieren Simulation entstehen kann. Dieser Mangel an *„robustness“* entsteht zum Beispiel durch ein *„Overfitting“*, also die fehlende Generalisierbarkeit des Modells, oder durch das Anpassen von Parametern bis zum gewünschten Ergebnis, ohne die das Modell inhaltlich keine sinnvollen Ergebnisse erzeugt.

Sämtliche aufgezeigten Ansätze für die Überprüfung der Parameterensibilität bzw. Robustheit der programmierten Modelle sind iterativ und explorativ. Dies ist zwar einleuchtend, kann aber nicht sicherstellen, dass sämtliche Ungenauigkeiten erkannt und behoben werden können.

Diese und andere Problematiken werden in der Literatur nur fragmentarisch besprochen, sodass bei der Frage nach der Sensibilität eine Ausgewogenheit des Forschungsstands fehlt.

3. Das Emergent Polarity Model

3.1. Emergent Actors in World Politics

3.1.1. Überblick

Das von Cederman (1997) entwickelte Emergent Polarity Modell ist ein CAS zur Modellierung des Entstehens und Zerfallens von Staaten. Das EPM wird hier zunächst in den historischen und politikwissenschaftlichen Kontext eingeordnet und in mehreren Schritten entwickelt. Hierbei wird zunächst ein grundlegendes Modell entwickelt, das im nächsten Schritt überarbeitet und erweitert wird.

Dieser Kontext und die Entwicklung des Modells werden im Folgenden vorgestellt. Auch wird die Begründung dafür, warum für das Modell ein CAS verwendet wird, erläutert. Dabei werden die politik- und sozialwissenschaftlichen Begründungen der Modellierungsentscheidungen nicht ausgeführt, können aber in Cederman (1997) nachvollzogen werden.

3.1.2. Emergente Akteure in internationaler Politik

Cederman (1997) stellt die Frage nach dem Entstehen und Zerfallen von Staaten im Zusammenhang mit dem Ende des kalten Krieges:

> „The fall of the Berlin Wall in 1989 triggered an astonishing series of epochal events that led to the dissolution of multiethnic communist states and the creation of dozens of new sovereign units. While the collapse of communism accelerated the splintering of the Soviet Union, Yugoslavia, and Czechoslovakia, other parts of Europe experienced a trend toward unprecedented levels of political integration. Germany reunited, and despite certain delays associated with the ratification of the Maastricht Treaty, Western European integration progressed toward a deeper and a wider union“ (S.3f).

Die daraus resultierende Frage, die Cederman (1997) aus dieser Entwicklung ableitet, ist, wie ein solcher gleichzeitiger Trend zur Integration und Desintegration erklärt werden kann.

Dies kann durch gängige Theorien nur unzureichend erklärt werden (S.3). Ziel seiner Studie zu emergenten Akteuren sei daher,

> „to develop new heuristic devices that deal directly with complexity and change in world politics. [...] To this end, I present a series of models that represent both states and nations as inherently history-dependent actors. In addition to stressing their emergence, the proposed formal frameworks also distinguishes clearly between states and nations as actor categories“ (Cederman, 1997, S.5).

Anzumerken ist die begriffliche Trennung von Staat und Nation, auf die im Folgenden eingegangen wird.

3.1.3. Staat und Nation

Die Entscheidung zur Unterscheidung zwischen Staaten und Nationen begründet Cederman (1997) unter Bezug auf Arnold Wolfers:

> „If the nation-states are seen as the sole actors, moving or moved like a set of chess figures in a highly and abstract game, one may lose sight of the human beings for whom and by whom the game is supposed to be played. If, on the other hand, one sees only the mass of individual human beings of whom mankind is composed, the power game of states tends to appear as an inhuman interference with the lives of ordinary people“ (Wolfers, 1962, S.3).

Werde also nur der Staat oder das menschliche Individuum als Akteur betrachtet, führe dies zu paradoxen Situationen: Eine Betrachtung nur mit Fokus auf Staaten führe dazu, dass der Bevölkerung nur eine untergeordnete Rolle in den Entscheidungen einer politischen Elite zugesprochen wird. Werde andererseits nur das Individuum betrachtet, würden kollektivistische Einflüsse auf die Meinung der Bevölkerung und politische Entscheidungen ignoriert (vgl. Cederman (1997), S.14f).

Um diese Problematik zu lösen unterscheidet Cederman (1997) zwischen Staat und Nation, indem er sich des Weber'schen Definition dieser Begriffe bedient. Weber definiert Staat als „diejenige menschliche Gemeinschaft, welche innerhalb eines bestimmten Gebietes – dies: das ‚Gebiet', gehört zum Merkmal – das Monopol legitimer physischer Gewaltsamkeit für sich (mit Erfolg) beansprucht“ (Weber, 1946, S.821).

Eine Nation ist nach Weber etwas, das „nicht nach empirischen gemeinsamen Qualitäten der ihr Zugerechneten definiert werden“ könne, aber „daß gewissen Menschengruppen ein spezifisches Solidaritätsempfinden anderen gegenüber zuzumuten sei“ (Weber, 1946, S.528).

Diese Nationen manifestierten sich aber in Staaten und produzierten oft eigene Staaten (vgl. Cederman (1997), S.18).

Die Unterscheidung zwischen Staat und Nation bezieht sich auf geografische Territorien und Gewaltmonopol einerseits und eine weniger klar eingegrenzte emotionale Gemeinschaft andererseits. Dabei müsse aber nicht jede Gemeinschaft eine eigene Nation bilden, sondern sie zeichne sich durch das Streben nach, oder dem Bestehen eines eigenen Staats dieser Nation aus (ebd.).

Veränderten sich diese Staaten und Nationen im historischen Verlauf, entsteht durch Staatenbildung und Nationalismus Emergenz (vgl. Cederman (1997), S.22). Das im Folgenden erläuterte EPM bezieht sich auf diese Definition von Staat.

3.1.4. Entscheidung für ein CAS zur Simulation

Klassischerweise werden Experimente in Geisteswissenschaften eher als Gedankenexperimente betrieben, statt einem klaren Ablauf wie in den Naturwissenschaften zu folgen. Die gerade im Jahr 1997 ungewöhnliche Entscheidung, ein CAS zur Modellierung zu verwenden, begründet Cederman (1997) damit, dass es hilfreich sei, zufällige Einflüsse zu berücksichtigen:

> „Ignoring the problems of path-dependence, or concentrating on simpler, linear problems, will not provide a solution to this dilemma. [...] Instead of excluding accidental influences by assumption, [...] the modeling framework ought to let accidents generate processes along dramatically divergent historical paths, should positive feedback be present. A modeling test bench of the CAS type allows for replication of such experiments many times. If historical contingencies turn out to be unimportant, they will wash out. If not, we will learn about the limitations of postulated structural laws“ (S.44).

Im Vergleich zu der klassischen Modellierung bietet laut Cederman (1997) ein CAS auch den Vorteil der Nachvollziehbarkeit und Wiederholung:

> „With the possibility of ‚rerunning the tape' [...], one gets a more direct feeling of the complex and unanticipated consequences of policy choices than is offered by simpler frameworks that rule out historical contingency and path dependencies by assumption“ (S.10).

Es bestünden Vorteile gegenüber klassischer Modellierung und Naturwissenschaften Forschung:

> „In essence, the CAS method differs from conventional modeling techniques in two respects. First, while relying on computer-guided deduction to generate results, it uses induction rather than deduction as the main method of exploration. [...] The second major difference from conventional modeling approaches pertains to the working style of CAS researchers. Instead of emulating the analytical method of natural scientists, which aims at separating and isolating carious partes of the complex systems, students of CAS stress synthesis and engineering [...]“ (Cederman, 1997, S.53f).

Zusammenfassend kommt Cederman (1997) zu dem Schluss, dass:

> „the CAS modeling tradition opens a third route between highly formalized, deductive frameworks and rich verbal descriptions. This combination of rigor and flexibility can be exploited to circumvent the dilemma of historical contingency and methodological individualism“ (S.62).

3.2. Kritik am CAS

Trotz der oben genannten Vorteile, ist sich Cederman (1997) der Einschränkungen und Nachteile eines CAS bewusst. Gängige Kritikpunkt an diesen Systemen seien:

> „1. Ad hoc assumptions [...]
> 2. Failure to yield unique predictions [...]
> 3. Fragility of results [...]
> 4. Lack of cumulation [...]” (S.62f)

Diese Punkte nehmen die am Buch geübte Kritik bereits vorweg. Insbesondere 1. bezieht sich auf die Tatsache, dass Ad-Hoc-Entscheidungen getroffen werden müssten, um ein CAS zu entwickeln. Dieser Kritikpunkt ist laut Cederman (1997) gerechtfertigt. Ihm solle auf zwei Wege entgegengewirkt werden: Die Entscheidungen über Mechanismen und Annahmen sollten so gut wie möglich auf Grundlage von theoretisch Konzepten getroffen werden, und das Modell werde schrittweise mit regelmäßiger Überprüfung erstellt (vgl. S.63f).

Als Reaktion auf die Kritik, dass mit einem CAS keine neu- oder einzigartigen Vorhersagen getroffen werden könnten (2.) argumentiert Cederman (1997), dass dies nicht zwangsläufig Ziel eines solchen Modells sein solle, sondern es eher um Heuristiken als um Vorhersagen gehe: Das Ziel eines Modells liege in seiner Konstruktion und Exploration, und als deren Resultat

um eine verbesserte Intuition über Verhalten, Aspekte und Anfälligkeiten des Systems, also eher eine Erforschung und Erkundung als konkrete Vorhersagen (vgl. S.64).

Die „Fragilität“ (3.) des Systems sei bei jeglicher Modellierung ein Problem, es bestehe „no guarantee that [unique results] are not an artifact of specific parameter configuration“((Cederman, 1997, S.62)). Dieses Problem lasse sich laut Cederman (1997) nicht gänzlich umgehen:

> „Parameter sensitivity poses the most serious threat to CAS-based research. [...] [T]he Achilles' heel of fragile results remains, but its consequences can fortunately be mitigated. First, if the goal is to establish possibility of certain outcomes, rather than their probability or certainty, the critique loses its bite [...]. Second, game-theoretical models often fall short of the robustness that their creators and supporters call for. [...] Third, it is not always the case that we need to know everything about the entire parameter space. [...] Fourth, the contribution of 'intuitive' calibration should not be underestimated. Since CAS models allow the analyst to observe the system as it unfolds, it becomes possible to check whether the results make qualitative sense. In addition, artificial parameter randomization for each replication helps improve the robustness of the results“ (S.64f).

Dem 4. Kritikpunkt entgegnet Cederman (1997), dass zwar die größere Komplexität eines CAS die Replikation von Ergebnissen erschwere, diese aber nicht unmöglich mache (S.65).

Diese Punkte sollen wieder aufgegriffen, und das EPM an ihnen gemessen und diskutiert werden.

3.3. Modellierungsvorgehen für das EPM

Unter Beachtung dieser Fallstricke entwickelt Cederman (1997) sein Modell. Da das EPM eine komplexe politikwissenschaftliche Theorie modelliere, bringe dies einige Schwierigkeiten mit sich:

> „The crux is to find a way of modeling collective identities without assuming the existence of a metaphysical ‚super-mind' existing above and beyond individuals. [...] Clearly a dynamic concept that links individual identity with collective structures is needed“ (Cederman, 1997, S.45).

Zur Vereinfachung wird daher das Modell in zwei Schritten entwickelt. Das grundlegende EPM modelliere Machtkämpfe als historischen Prozess. Dies setze voraus, dass „some states disintegrate, other states grow territorially, and system structure, especially poliarity, emerges

from what actors do“ (Cederman, 1997, S.72). Polarität („*polarity*“) bezeichnet die Anzahl der aktuell existierenden Staaten im Modell: Unipolarität bedeutet, dass genau ein Staat existiert, Bipolarität zwei Staaten usw. .

3.4. Emergent Polarity Model

3.4.1. Mechanismen des EPM

Machtkämpfe, Zerfall und Wachstum bilden die Grundlage des EPM. Die „Spielregeln“ des EPM werden im Folgenden im Detail beschrieben. Das vorgestellte EPM

> „1. provides an explicit spatial representation of the international system
> 2. involves a large number of actors
> 3. endogenizes the outer boundaries of these actors as well as the polarity structure of both the regional and global structure
> 4. endows these agents with a bounded and historically contingent decision scope” (Cederman, 1997, S.72f).

Basierend auf diesen Überlegungen wird ein konkretes Modell entwickelt.

> „The Emergent Polarity Model contains three independent variables. First, the *proportion of predators* refers to the density of predator states at the outset of the analysis. Second, *defense dominance* stands for how difficult predators judge attacking to be. [...] Finally, the presence of *defensive regional alliances* entails the possibility of external balancing against immediate threats from neighboring states.“ (Cederman, 1997, S.78)

Die Akteure des EPM werden auf einem schachbrettartigen Raster der Größe 10×10 dargestellt, sodass jeder der Akteure räumlich festgelegt und anfangs von bis zu acht Nachbarn umgeben ist. Kommunikation findet nur mit diesen direkten Nachbarn statt. Die Akteure haben kein Wissen über die restlichen Staaten. Ein Akteur ist entweder als Räuber („*predator*“) oder Beute („*prey*“) definiert, wobei sich beide Typen an das Prinzip der Wechselseitigkeit im Bezug auf Angriffe verhalten, die *Predator*-Staaten aber manchmal unprovoziert angreifen wenn sie sich als überlegen einschätzen (vgl. Cederman (1997), S79).

Das Prinzip der Wechselseitigkeit führt dazu, dass jeder Angriff zu einem kostspieligen Krieg zwischen Angreifer und Angegriffenem führt. Bei einem Sieg durch den ressourcenstärkeren Staat findet eine Eroberung des schwächeren Staats statt. War der Verlierer Teil eines größeren,

schon bestehendes Staates, verliert dieser den besiegten Quadranten (seine „Provinz“) an den Sieger (vgl. Cederman (1997), S.80).

> „These rules imply three things: First, the number of states is going to decrease as the predators absorb their victims. Second, as a consequence of conquest, the predatory actors increase in size, their territory expanding into the conquered areas. Third, the surviving units will be predominantly predators, the more peaceful prey states having been eliminated in the selection process“ (ebd).

Zu Beginn der Simulation erhält jeder Staat eine Menge von Ressourcen, die aus dem Mittelwert 50 plus/minus einer zufällig gewählten Anzahl als Variation (im Modell ± 20) pro Staat zufällig verteilt wird. Durch eine Eroberung wird dem siegreichen Staat eine neue Provinz hinzugefügt und untergeordnet. Es existieren zwei Hierarchiestufen: Die Hauptstadt des Staates und Provinzen (vgl. Cederman (1997), S.84). Die Hauptstadt befindet sich auf dem Feld, das zu Beginn alleine den Staat ausmachte und von dem aus die Eroberungen begannen.

Eine Simulation erstreckt sich über eine feste Anzahl von Zeiteinheiten. Jeder Zeitabschnitt ist in drei Phasen aufgeteilt: Zuerst treffen alle Staaten Entscheidungen. Im zweiten Schritt werden die Konsequenzen der aktuellen Ressourcenverteilung berechnet. Dabei findet eine Einschätzung der eigenen Über- oder Unterlegenheit im Vergleich zu den gesamten Ressourcen der Nachbarn statt. Ab einem bestimmten Ressourcenverhältnis, im EPM Faktor 3,0, wenn also die eigenen Ressourcen das dreifache der gegnerischen betragen oder diesen Grenzwert überschreiten, schätzt sich der Akteur als überlegen ein (vgl. Cederman (1997), S.127). Zuletzt finden aufgrund dieser Einschätzungen Eroberungen statt, die das System transformieren (Cederman, 1997, S.84). In jeder Zeiteinheit wird allen Akteuren eine Menge von Ressourcen als Ernte zugeteilt, die nach dem gleichen Prinzip wie die initial verteilten Ressourcen in gewissem Maß variieren.

Der Entscheidungsprozess findet jeweils zwischen zwei Staaten statt, wobei die Entscheidung zwischen Kooperation („*cooperation*“) und Angriff („*defect*“) getroffen werden kann. *Prey*-Staaten halten sich dabei an Wechselseitigkeit und reagieren mit der Entscheidung, die der Gegenspieler in der letzten Runde getroffen hat. Sie greifen also nie von sich aus an (vgl. Cederman (1997), S.85).

Diesen Regeln folgen zunächst auch die *Predators*, diese greifen aber in manchen Runden ohne Provokation an, wenn zwei Bedingungen erfüllt sind:

> „First, a predator only considers an unprovoked aattack if it is not already involved in warfare on another front. This [...] avoids initiation constly two-front wars that might endanger the survival of the state. Second, revisionist states only attack if the

> relative power balance between them and their potential victim exceeds a certain ratio that determines the offense-defense orientation of the system. Should there be many potential victims, the aggressor selects the weakest of them“ (Cederman, 1997, S.85).

Ist die Grenze zwischen einem Angreifer und einem Verteidiger mehr als ein Feld groß, wählt der Angreifer zufällig eine Provinz aus und greift diese an.

Zum Ende der Entscheidungsphase werden die Ressourcen gleichmäßig an den Grenzen zu den direkten Nachbarn des jeweiligen Staates verteilt. Je nach Ressourcenlage findet in allen Staatenpaaren die Entscheidung zwischen *Cooperate* („*C*“) und *Defect* („*D*“) statt. Die Entscheidung hat also die möglichen Ergebnisse „*CC*“, „*CD*“, „*DC*“ und „*DD*“. Jeder Konflikt hat dabei einen Einfluss auf die Ressourcen der beiden Akteure, da in den Konflikten Ressourcen vernichtet werden („*destruction rate*“) (vgl. Cederman (1997), S.87).

In der Simulation werden bei einem angegriffenen Akteur fünf Prozent der an der betroffenen Grenze allokierten Ressourcen vernichtet. Haben beide Akteure die Option „*D*“ gewählt, werden auf beiden Seiten Ressourcen zerstört. Ist das Ressourcen-Gefälle zwischen den Staaten groß genug, erzielt der Stärkere der beiden Staaten einen Sieg. Dabei werden, anders als bei der Entscheidung zum Angriff, nicht die gesamten Ressourcen des Staates betrachtet, sondern nur die an der betroffenen Front (vgl. Cederman (1997), S.88).

Es existieren drei Szenarien für eine Eroberung:

> „1. If it is an independent primitive actor, this unit is subordinated under the head of the conquering state. [...]
> 2. If the target is a province of a corporate [‚*corporate*‘ = sich über mehr als ein Feld erstreckender Staat, Anm.] actor, the subordinate unit is transferred to the attacker. [...]
> 3. If the target is the capital of a corporate actor, the entire corporate unit collapses into its primitive units when the capital falls” (Cederman, 1997, S.89).

Werden bei einer Eroberung Provinzen so von ihrem Staat abgeschnitten, dass sie keine direkte Landverbindung mehr zu ihrem Staat haben, werden sie zu einem eigenständigen Staat um Enklaven zu verhindern (vgl. Cederman (1997), S.89).

Zerfällt ein Staat in seine einzelnen Felder und somit in eigenständige Staaten, werden die Rohstoffe des ehemaligen Staats gleichmäßig zwischen allen neuen Akteuren aufgeteilt und alle Staaten erhalten ihre ursprüngliche *Prey-Predator*-Orientierung zurück (vgl. Cederman (1997), S.89).

3.4.2. Verteidigungsbündnisse

Für alle Staaten besteht die Möglichkeit, sich zur Verteidigung mit anderen Staaten zu verbünden. Cederman (1997) begründet die Notwendigkeit von Bündnissen („*alliances*“) wie folgt:

> “1. [Alliances] serve mainly to counter external security threats
> 2. form as a matter of expediency rather than principle
> 3. are ad hoc arrangements rather than long-lasting coalitions - if the threat disappears, so does the alliance” (S.90).

Zu diesem Zweck wird ein Modell der Bedrohung eingeführt, bei dem früheres Verhalten von Staaten in Betracht gezogen wird. Greift ein Staat über längere Zeit nicht grundlos an, wird er als vertrauenswürdig eingeschätzt. Dieser „*trust level*“ wird vor der Entscheidungsphase berechnet.

Anschließend wählt jeder Staat den aggressivsten Nachbarn aus, außer wenn er nur von friedlichen Staaten umgeben ist.

Am Ende einer Zeiteinheit gibt jeder Staat der sich als bedroht einschätzt bekannt, ob er gegen den Aggressor vorgehen will. Nennen zwei oder mehr Staaten den gleichen *Predator*, bilden sie automatisch ein Bündnis. Dieses Bündnis hält so lange, wie es mindestens zwei Staaten gibt die sich vom gleichen Aggressor bedroht fühlen, danach löst es sich auf (vgl. Cederman (1997), S.91).

Bündnisse haben eine abschreckende Wirkung auf Angreifer: Ein potenzieller Angreifer kalkuliert jetzt nicht nur die Ressourcen, die sein Ziel an der Grenze platziert hat, sondern die Ressourcen aller Bündnispartner des Ziels. Ist ein *Predator* selbst Teil eines Bündnisses, bekommt er nur dann Unterstützung von dessen Mitglieder, wenn sie das gleiche Ziel angreifen wollen (vgl. Cederman (1997), S.90).

Wird einer der Bündnisstaaten angegriffen, sind alle anderen Mitglieder des Bündnisses verpflichtet ihm zu helfen, indem sie den Angreifer bekämpfen. Sind Angreifer und *Prey* Mitglieder im gleichen Bündnis, wird der Angreifer automatisch ausgeschlossen, „to make it impossible for predators to exploit the alliance from within“ (Cederman, 1997, S.90).

3.4.3. Explorative Anpassung des EPM

Nach Vollendung der Modellierung werden einige der Ad-Hoc-Entscheidungen des entwickelten Modells getestet. Cederman (1997) prüft zunächst den Anteil der *Predator*-Staaten zu Beginn der Simulation, wobei der Parameterraum (0-100%) mit den Werten 0, 5, 10, 20, 40, 60,

80 und 100 abgedeckt werden soll. Auch wird überprüft, ab welchem Ressourcenverhältnis zwischen *Predator* und *Prey* ein Angriff durch den *Predator* erfolgt. Dabei wird ein zwei- und dreifaches Ressourcenverhältnis getestet. Alle Belegungen der Parameter werden, wegen der zufälligen Elemente der Simulation wie z.B. der Verteilung der *Predators*, in mehreren Durchläufen geprüft. Ein Durchlauf endet entweder nach 1000 Zeiteinheiten, oder wenn nur noch ein Staat sämtliche Felder beherrscht (vgl. Cederman (1997), S.92).

Cederman (1997) kommt zu dem Ergebnis, dass Simulationen mit höherer Dichte an *Predators* als Ergebnis eine höhere Polarität haben:

> „The greater the competition, and the more even the growth rate of the most aggressive states, the more likely a balance of power becomes. By contrast, a more defensively oriented system widens the window of opportunity for potential hegemons [‚*hegemon*' = Alleinherrscher, hier: Unipolarität, Anm.] by thwarting the attempts of predatory rivals to catch up" (S.96).

Cederman (1997) prüft auch die Auswirkungen von Verteidigungsbündnissen auf Machtkämpfe und Eroberungen („*power politics*") im Vergleich zu einer bündnislosen Simulation.

> „[T]he alliance mechanism does offer some protection at least for low predation frequency. [...] A similar, but weaker tendency is present in the offence-dominiated world. [...] Compared to the non-aligned world [...] [it] shows a remarkable dominance of unipolarity at the expense of power politics. From having been the most common outcome in the offense-dominated system, power politics becomes less likely than universal empire. In the defense-oriented runs, power politics almost does not appear at all except for predator rates above 80 percent" (S.98ff).

Cederman (1997) zieht aus diesen Tests folgenden Schluss für das EPM:

> „[T]he emphasis on heuristic insights [of CAS, Anm.] does not exclude the need for more rigorous sensitivity testing. [...] Although I randomized some factors such as the initial resource distribution for each replication, many other parameters and rules are hard-wired into the model. Therefore, the exact results should be interpreted with caution" (S.107f)

Aus diesem Grund erweitert Cederman (1997) das Modell im nächsten Schritt.

3.5. Das Erweiterte Emergent Polarity Model

Die Erweiterung des EPM begründet Cederman (1997) mit der Notwendigkeit „to extend and check the robustness of the Emergent Polarity Model. [...] [W]e emphasize the importance of

sensitivity and robustness-checking in simulation work. The main idea is thus to investigate whether the results hold up when different, more realistic, mechanisms are added“ (S.109).

3.5.1. Vereinfachte Verteidigungsbündnisse

Das Ergebnis der Bündnisse im EPM widerspreche den Erwartungen: „Instead of generating bi- or multipolar systems, unipolarity or large polarity followed when defensive alliances were added. [...] How sensitive is the finding to changes in the particular specification of alliance formation?“ (Cederman, 1997, S.109).

Im bisherigen Modell wurden Bündnisse auf Basis eines *„trust level“* geformt, der sich aus Nicht-Angriffen über Zeit berechnet. In der Erweiterung berechnet jeder Akteur das Verhältnis der eigenen Ressourcen im Vergleich zu denen aller Nachbarn. Den Akteuren ist die *Predator-Prey*-Orientierung aller Nachbarn bekannt, sodass nur gegen *Predator*-Staaten eine Verbündung in Betracht gezogen wird. Alle Akteure, unabhängig von der eigenen Orientierung, berechnen mögliche Bedrohungen und wählen die Größte anhand des Ressourcen-Verhältnisses aus. Hierbei wird ein gerade stattfindender Angriff immer als die größte Bedrohung gewertet, unabhängig von den Ressourcen anderer Nachbarn.

Ist die gewählte Bedrohung groß genug, gibt der Akteur sein Interesse an einem Bündnis bekannt. Ab diesem Punkt funktioniert die Verbündung wie im ursprünglichen EPM (vgl. Cederman (1997), S.111).

3.5.2. Erweiterte Agentenstrategie

Bisher wurde angenommen, dass Staaten ihre strategische Orientierung (*Predator*, *Prey*) für die Dauer einer Simulation beibehalten.

> „This behavioral invariance is, of course, both unrealistic and theoretically impoverished, for states anticipate future interactions and adapt their strategies to there expectations. [...]
>
> Suppose that instead of forming an alliance as a response to threats, states change their own strategies in order to prepare for the worst. This requires a third strategic category, conditional predation [...]. [P]rey states may switch to conditional predators should they feel threatened. Moreover, unthreatened conditional predators sometimes switch back to prey behaviour“ (Cederman, 1997, S.112ff).

Es werden zwei Werte eingeführt: Die Wahrscheinlichkeit pro Zeiteinheit, mit der ein bedrohter *Prey*-Staat zu einem bedingten *Predator* wird, und die Wahrscheinlichkeit der Rückwandlung (Cederman, 1997, S.114).

3.5.3. Ressourcenverteilung

Auch die Aufteilung der Ressourcen eines Akteurs wurde im neuen Modell überarbeitet.

> „In the basic version of the Emergent Polarity Model, resource allocation follows a particularly simple logic. Each state divides its total resources by the number of neighboring states and allocates an equal share to each front regardless of their neighbors' behaviour or resource endowment. [...] [T]his is not a terribly realistic rule of resource allocation. [...] [W]e need a more refined mechanism of Proportional Resource Allocation (PRA). According to this scheme, states earmark forces proportional to their neighbor's mobilization while paying close attention to each front's status" (Cederman, 1997, S.113f).

Zu diesem Zweck wird nun zwischen aktiven und passiven Grenzen unterschieden: Fand in der letzten Zeiteinheit an einer Grenze eine Schlacht statt ist sie aktiv; ansonsten ist sie passiv. Für jede Zeiteinheit berechnet ein Staat, wie viele Ressourcen er an eine Grenzen entrichten könnte, sollte ein Angriff stattfinden und sollte sich der Status der anderen Fronten im Bezug auf Angriffe nicht verändern.

Zu aktiven Fronten sendet der Staat eine Menge an Ressourcen, die sich aus den gesamten Ressourcen des Staates, der Ressourcenverteilung der Nachbarstaaten und den Ressourcen, die aktuell gegen den Angreifer der aktiven Front berechnet. Sie ermöglichen einerseits die Einschätzung des Staats durch den Gegner, andererseits werden sie in Konflikten verbraucht.

Für passive Fronten werden die Ressourcen berechnet die allokiert werden können, sollten sie zu aktiven Fronten werden. Die Berechnung erfolgt analog zur Berechnung für aktive Fronten.

3.5.4. Zweistufige Konflikte

Bisher fand Zerfall nur durch externe Eroberungen statt. Dem werden im erweiterten Modell zusätzliche Zerfallsmechanismen hinzugefügt:

> „The current formulation of the Emergent Polarity Model presumes that conquest automatically produces great gains for the conqueror and that only external pressure can undermine states. [...] If the center cannot rely on automatic support from the periphery, revolts become a serious possibility [...]. Here we extend the Emergent Polarity Model by adding the possibility of ‚domestic' collective action leading to secession. [...] First, the decision-making mechanism must be modified in order to allow subordinated provinces to stage revolts within states. Second,

> in order to model taxation and the consequences of civil war, resource transfer between the center and provinces need to be modeled explicitly“ (Cederman, 1997, S.121f).

Um dies zu erzielen prüft jede Hauptstadt in jeder Zeiteinheit nicht nur die Liste der Nachbarn, sondern auch all ihre Provinzen, falls der Staat aus mehr als einem Feld besteht. Jede Provinz kann sich, analog zu den *Predator*-Staaten, entscheiden ihre Hauptstadt anzugreifen, während die Hauptstadt nie unprovoziert ihre Provinzen angreift. Diese Konflikte verlaufen wie bilaterale Machtkämpfe, mit dem Unterschied dass ein Sieg der Provinz eine Abspaltung von der Hauptstadt, und ein Verlust die Erhaltung des Status Quo bedeutet. Auch werden die internen Fronten nur je mit den lokalen Ressourcen versorgt, statt sich aus den gesamten Ressourcen des Staates zusammenzusetzen.

Dies kann in manchen Situationen dazu führen, dass *Predator*-Staaten ihren Angriff aufgrund interner Kämpfe vertagen müssen. Eine Provinz entscheidet sich, wie ein *Predator*-Staat, dann zum Angriff, wenn sie eine günstige Ressourcenlage vorfindet (vgl. Cederman (1997), S.123).

Die Besteuerung der Ressourcen durch die Hauptstadt erfolgt ab dem Zeitpunkt, an dem eine neue Provinz von der Hauptstadt erobert wird. Im ursprünglichen EPM wurden sämtliche Ressourcen der eroberten Provinz an die Hauptstadt übertragen. Nun behalten die Provinzen ihre Ressourcen, müssen aber einen Teil als Steuer an die Hauptstadt zahlen. Diese Steuern müssen in jedem Zug als fester Anteil der jeweiligen „Ernte“ an die Hauptstadt gezahlt werden.

Cederman (1997) testet unterschiedliche Steuerraten in Simulationen, in der 100% der Staaten *Predators* sind. „The main finding is that power politics appear to depend positively on the tax rate. In systems characterized by less pronounced ‚fiscal‘ centralization, unipolarity becomes more frequent. As the tax rate decreases, massively multipolar outcomes also gain in importance“ (Cederman, 1997, S.125). Bei Steuerraten unter 20% finden keine Machtkämpfe mehr statt, und unterhalb von zehn Prozent finden keinerlei Eroberungen mehr statt.

Dennoch: „Because of the deterministic decision criterion, secession almost never takes place. Moreover, taxation is unrealistically uniform, regardless of the distance between the centers and their provinces“ (Cederman, 1997, S.126). Um dies zu beheben wird das Ressourcenverhältnis, bei dem sich ein *Predator* oder eine Provinz zu einem Angriff entscheidet, von einem festen Wert in eine Kurve umgewandelt. Mit steigender Überlegenheit steigt jetzt die Wahrscheinlichkeit eines Angriffs (vgl. Cederman (1997), S.127).

Auch werden Steuern jetzt in Abhängigkeit von der Entfernung der Provinz zur Hauptstadt erhoben. „For every unit’s distance from the center, the tax contribution declines by a given fraction. The simulation runs [...] feature a discount rate of 0.7, which means that provinces

at one unit's distance from the center pay 70 percent of the nominal tax rate, and those at two units' distance pay $0.7 \times 0.7 = 49$ percent, etc." (Cederman, 1997, S.128).

Mit dieser Erweiterung schließt Cederman (1997) die Entwicklung des EPM ab und beginnt mit der Exploration der Ergebnisse.

3.6. Schlussfolgerungen aus dem EPM

Das EPM wurde vor allem zur Exploration und Experimentation entwickelt, mit dem Ziel neue Erkenntnisse zu gewinnen. Cederman (1997) schließt:

> „[T]he Emergent Polarity Model helped us to understand that power politics and state formation are two sides of the same coin. [...] In addition to addressing neglected questions, the emergent-actor perspective generates new, counterintuitive insights about established IR [international relations, Anm.] theories. [...] [T]he Emergent Polarity Model showed that defensive technology and alliances may sometimes increase the chances for unipolarity rather than contribute to stability. [...] Modeling power politics as a complex adaptive system (CAS) can alert us to potential inconsistencies in existing theories" (S.213ff).

Dennoch merkt Cederman (1997) auch kritische Punkte an der Modellierung an:

> „[C]autionary words against exaggerated confidence in the robustness of the results are also in order. While we have investigated numbers of extensions and alternative mechanisms, we have by no means exhausted the space of theoretically relevant variations. For example, despite the introductions of strategic adaptions, the basic prey and predator strategies of the states have been constant. Moreover, the harvest mechanism is wildly unrealistic and the initial conditions were never varied [...]. Only continued research featuring important deviations from the current framework, guided by thorough empirical analysis, is likely to yield more reliable knowledge of the complex patterns of power politics (S.134f)."

Dieser und weitere Kritikpunkte werden im Verlauf dieser Arbeit erneut aufgegriffen.

4. Kontingente Modellierungsentscheidungen im EPM

Wie oben erörtert, müssen für die Modellierung einer Theorie Entscheidungen getroffen werden die die Formalisierung ermöglichen, beginnend mit der Wahl des Modells. Viele Modellierungsentscheidung sind kontingent, weshalb durch das Treffen der Entscheidung andere Optionen ausgeschlossen werden.

Bei der Formalisierung des EPM stellt Cederman dies selbst fest: „As the model building proceeds, several simplifying assumptions become inevitable“ (Cederman, 1997, S.73).

Wenn aus dem Modell und dessen Ergebnissen Schlüsse für die ursprüngliche Theorie gezogen werden sollen, ist es notwendig sicherzustellen, dass das Modell die Theorie adäquat wiedergibt. Ansonsten ist es möglich, dass durch Fehler im Modell falsche Rückschlüsse auf die Theorie gezogen werden. Deshalb ist es wichtig, die Alternativen jeder getroffenen Entscheidung zu berücksichtigen und zu prüfen, welche von ihnen die Theorie am besten repräsentiert. Im Folgenden sollen die getroffenen Entscheidungen bei der Formalisierung des EPM herausgearbeitet und auf ihre Alternativen untersucht werden. Danach werden diese Alternativen auf ihre vermutete Auswirkung auf die Gültigkeit des Gesamtmodell geprüft, und schließlich einige von ihnen für die weitere Untersuchung mittels des hier entwickelten Modells ausgewählt.

4.1. Modellierungsentscheidungen im EPM

4.1.1. Entscheidung für ein CAS

Die erste Entscheidung die Cederman für das EPM trifft, ist, ein Programm zur Simulation zu benutzen. Dies war und ist in den Sozialwissenschaften ungewöhnlich, besonders im Jahr 1997. Häufiger wurden Gedankenexperimente oder eine statistische Auswertungen schon geschehener Ereignisse genutzt, um mögliche Muster zu erkennen (vgl. Cederman (1997), S.30).

Ausgehend von der Entscheidung, eine Simulation mittels Programmierung zu erzeugen, ist die für das EPM grundlegende Entscheidung, eine Formalisierung durch ein CAS zu wählen. Alternativen wären beispielsweise ein Netzwerk oder Graph.

Mit der Entscheidung, einen Mechanismus für den Zerfall größerer Staaten schon in das Modell einzubauen setzt sich Cederman zwangsläufig der Kritik aus, dass es eine wenig überraschende Erkenntnis ist, wenn im Modell ein Zerfall stattfindet. Zerfall als emergenter Teil des Modells wäre ein größerer Erkenntnisgewinn für Cedermans Theorie, als wenn dieser im Modell von Beginn an vorgesehen ist.

Eine gleichsam grundlegende Entscheidung Cedermans ist die Modellierung des „Spielfelds“ als Schachbrett. Dies erfordert die Einschränkung, dass ein Staat immer nur genau drei, fünf oder acht direkte Nachbarn haben kann. Wäre das System beispielsweise als Graph angelegt, wäre die Zahl der Nachbarn variabel und für jeden einzelnen Fall entscheidbar. Alternativ wäre denkbar, dass Akteure nicht nur mit ihren direkten Nachbarn interagieren können, sondern auch mit Nachbarn von Nachbarn, oder allen Akteuren auf dem Spielfeld.

4.1.2. Allgemeine Entscheidungen im EPM

Die Entscheidung zu einer eingeschränkten Kommunikation zwischen Agenten ist die erste von mehreren kontingenten Entscheidungen für die Modellierung von Agenten im EPM. Weiterhin könnten die Agenten beispielsweise ein generelles Wissen über den Zustand der anderen Akteure für die Entscheidungsphase der Staaten besitzen, wie in der Realität Staaten über Wissen über andere Staaten verfügen und ihre Entscheidungen mit diesem Wissen abwägen.

Im Modell existieren innerhalb eines Staats nur zwei Hierarchiestufen: Hauptstadt, und Provinz. Hier gibt es keine Abstufungen, was aber als Alternative denkbar wäre. Dies könnte Auswirkungen auf die Ressourcenverteilung haben. Diese erfolgt im EPM anhand der Grenzen und Bedrohungen, und im erweiterten EPM zusätzlich aufgrund des aktuellen Zustands einer Grenze.

Im EPM existieren nur zwei Arten von Akteuren: *Prey* und *Predator*. Hier könnte anhand der realen Inspiration für das Modell, internationaler Politik, drei oder mehr Arten von Akteuren eingeführt werden, beispielsweise neutrale Staaten die sich nicht in Konflikte verwickeln lassen, oder aggressive Staaten die, anders als *Predators* in (fast) jeder Runde angreifen.

Ebenfalls könnten die Regeln für die Umwandlung von *Prey*- in *Predator*-Staaten angepasst werden. Diese Umwandlung geschieht im erweiterten EPM als Reaktion auf eine konkrete Bedrohung, oder deren Ausbleiben über längere Zeit. Diese Logik könnte auf andere Politikstile ausgeweitet werden, oder durch andere Szenarien ausgelöst werden, so wie sich in der Realität beispielsweise die Außenpolitik eines Staates nach einem Regierungswechsel ändern kann.

Andererseits könnten die Regeln für Angriffe zwischen den beiden *Policies* der Akteure anders modelliert werden. Im EPM beruht die Entscheidung über Angriffe auf Ressourcen, Vorgeschichte und Zufall. Ebenso denkbar wären Angriffe aufgrund anderer Kriterien, inspiriert von internationaler Politik, zum Beispiel wegen unterschiedlicher Weltansichten oder multilateraler Kommunikation, aus der Provokationen entstehen könnten.

Interessant wäre auch die Einführung multilateraler Konflikte. Im EPM sind Kämpfe nur zwischen zwei Akteuren möglich. Realistischer wären Konflikte von mehreren Akteuren, die in kriegerische Auseinandersetzungen meist mehrerer Länder verwickelt sind.

Im EPM führt jeder Angriff zu einem militärischen Konflikt. Es könnten aber Konflikte durch Diplomatie gelöst werden, statt in jedem Fall zu eskalieren.

Andererseits wäre es denkbar, andere Arten der Interaktion zwischen Akteuren zu ermöglichen, zum Beispiel Ressourcenhandel. Dieser könnte, angelehnt an realen Handel, als Anreiz gesehen werden, Konflikte zwischen Handelspartnern zu vermeiden. Gemeinsam mit der bisherigen Logik der Verteidigungsallianzen wäre dies sinnvoll, da Allianzen durch Handel noch gestärkt würden.

In diesen Allianzen werden außerdem sämtliche Nachbarn des bedrohlichsten Akteurs zusammengefasst. Eine Alternative dazu ist, dass Staaten nur Allianzen mit Akteuren eingehen, denen sie vertrauen. Ebenfalls möglich wären positive Allianzen, aus denen allen beteiligten ein Gewinn z.B. von Ressourcen entsteht, statt Bündnisse nur wegen einer Bedrohung einzugehen.

Allgemein vergrößert sich im EPM das Vertrauen zwischen Staaten über die Zeit, auch wenn es in der Vergangenheit Konfrontationen zwischen ihnen gab. Es wäre denkbar dass Akteure existieren, die nach einer Auseinandersetzung für längere Zeit, oder für den restlichen Verlauf der Simulation, Feinde bleiben.

Gewinnt ein Staat einen Konflikt, erobert er automatisch eine Provinz des angegriffenen Akteurs. Hier könnte es eine Phase der Verhandlung zwischen den Konfliktpartnern geben, in denen über die Zukunft der Provinz entschieden wird, oder in der die Provinz die Möglichkeit hat, sich gegen die Übernahme aufzulehnen und selbst über ihre Zukunft zu entscheiden.

Spaltet sich eine Provinz wegen geografischer Abgeschiedenheit oder als Rebellion von ihrer Hauptstadt ab, erhält sie einen Anteil derer Ressourcen. Würde dies nicht stattfinden, könnte das für die Provinzen ein Anreiz sein, weiterhin der Hauptstadt zugehörig zu sein. Allgemein könnte, ohne die Möglichkeit der Abspaltung, eine geografisch abgelegene Provinz als Enklave der Hauptstadt erhalten bleiben. Zudem könnte die Hauptstadt die Möglichkeit haben, Rebellionen zu unterdrücken statt dass zwangsläufig eine Abspaltung stattfindet. Analog zu möglichen Verhandlungen mit anderen Staaten wäre hier eine diplomatische Auseinandersetzung mit den eigenen Provinzen denkbar.

Das Modell der Besteuerung wird aktuell dazu genutzt, Ressourcen für Auseinandersetzungen mit Nachbarn zu sammeln. Diese und die jede Runde als Ernte erhaltenen Ressourcen könnten andererseits genutzt werden, um andere Projekte innerhalb eines Staates zu finanzieren. Gäbe es beispielsweise Forschungsprogramme, könnte ein Akteur mit fortgeschrittener Forschung dies zu seiner Überlegenheit im nächsten Konflikt nutzen. Dies führt erneut zu der Überlegung, dass der Sieg eines Konflikts nicht nur von der Ressourcenverteilung abhängen könnte. Auch gäbe es mögliche Auswirkungen auf das Steuersystem. Steuern werden bisher gegen den Willen der Provinzen erhoben und können zu deren Rebellion führen. Gäbe es positive Anreize oder Belohnungen für Steuerzahlungen, könnte dies den Zusammenhalt innerhalb eines Staates stärken statt ihn zu verringern.

Zudem werden Ressourcen derzeit in jeder Runde zufällig an die Akteure verteilt. In der Realität ist es unwahrscheinlich dass ein Staat, der in der letzten Saison hohe Staatseinnahmen hatte, in der nächsten Saison nur noch sehr geringe Einnahmen erwirtschaftet. Dies ist beispielsweise dadurch zu erklären, dass Einnahmen aus natürlichen Ressourcen des Staates entstehen, die zwar zum Teil endlich sind, aber selten von einem zum nächsten Moment versiegen. Daher würde ein Staat mit Bodenschätzen meist über längere Zeit Gewinne erwirtschaften können. Auch würde ein Staat mit guter Infrastruktur, der seine Einnahmen mittels dieser erwirtschaftet, diese nicht innerhalb kurzer Zeit verlieren. Generell ist es in der Realität wahrscheinlicher, dass große Einnahmen dazu verwendet werden, im nächsten Schritt noch mehr Einnahmen machen zu können. Daher entspricht ein zufälliges Verteilen der Ressourcen nicht der wirtschaftlichen Realität der Staaten.

Dies sind die wichtigsten kontingenten Modellierungsentscheidungen die für das EPM getroffen wurden. Zusätzlich werden die aus diesen Entscheidungen resultierenden Parameter mit Ad-Hoc-Werten belegt, was im Folgenden untersucht wird.

4.1.3. Parameterwerte

Cederman benutzt zum Teil ad hoc ausgewählte Werte, z.B. für die Ressourcenverteilung, Anteil von *Predator*-Akteuren und Kosten für Konflikte. Eine Variation dieser Werte, die eher eine zufällige Entscheidung als eine Repräsentation der Realität darstellen, könnte die Ergebnisse und damit die Aussagekraft des Modells verändern und verringern.

Es wird bei den meisten dieser Werte weder eine sinnvolle Begründung für einen möglichen Wertebereich gegeben, noch wird der Einfluss des gewählten Wertes auf das Ergebnis der Simulation überprüft. Stattdessen begründet Cederman (1997) seine Entscheidungen zum Teil mit intuitiven Entscheidungen (vgl. Abschnitt 3.2).

Zu diesen Werten gehören ebenfalls die anfängliche Menge von Ressourcen, die jeder Staat erhält, die finanzielle Überlegenheit, die notwendig ist um einen Konflikt zu gewinnen, die Wahrscheinlichkeit, mit der ein *Predator*-Staat angreift, die Steuerrate für Provinzen und die Ressourcen, die in einem Konflikt vernichtet werden (siehe Abschnitt 3.4).

Erzeugt das Modell nur mit diesen zufälligen Werten sinnvolle Ergebnisse, spricht das gegen dessen Aussagekraft.

4.2. Mögliche alternative Modellierungen

Es werden nun die Merkmale ausgewählt, welche für die Prüfung der Parametersensibilität des EPM in der Nachbildung variiert werden. Die Entscheidung, für die Simulation ein Programm zu verwenden, wird von Cederman selbst begründet: „The computer methodology allows for systematic exploration of artificially created complex networks featuring large numbers of locally interacting agents“ (Cederman, 1997, S.7). Diese Begründung erscheint gegenüber Gedankenexperiementen oder Datenauswertungen als Alternative schlüssig. Zudem ist es aufgrund der Fragestellung dieser Arbeit sinnvoll, ebenfalls ein programmierbares Modell zu benutzen. Diese Alternativen sind sinnvolle Werkzeuge zur weiteren Überprüfung, werden aber für diese Arbeit als nicht zielführend eingeschätzt.

Ebenfalls scheint es logisch, ein CAS als Grundlage der Formalisierung beizubehalten. Einerseits könnte so gezeigt werden, dass schon kleine Variationen in einem ähnlich aufgebauten Modell zu großen Schwankungen im Ergebnis führen können. Andererseits ist für eine Formalisierung des EPM ein CAS sinnvoll, wie Cederman (1997) selbst sinnvoll begründet: „[T]he CAS approach offers an alternative to quasi-experimental analysis of the real world and single case thought experiment by providing the means to replicate systematic and controlled thought experiment in artificial computer worlds“ (S.45). Er spricht Alternativen wie Netzwerke an:

> „[S]tandard network theory suffers from a number of drawbacks. First, since network approaches usually stress structure [...], it is not clear who is acting [...]. Second, the structural bias often includes dynamic analysis, thus making the method unsuited for capturing the inherently dynamic nature of figurational processes“(Cederman, 1997, S.46).

Diese Begründung wird als ausreichend empfunden, um ein CAS als Grundlage des Modells beizubehalten.

Die allgemeinen Entscheidungen Cedermans sollen für das Modell ebenfalls übernommen werden. Diese können zwar kritisiert werden (siehe Abschnitt 4.1.2), da es sich beim EPM

aber um ein Modell handelt, sind Vereinfachungen notwendig. Fraglich ist, ob diese Vereinfachungen im Kern die zugrunde liegende Theorie abbilden, oder ob wichtige Aspekte vernachlässigt wurden. Es kann argumentiert werden, dass beispielsweise die Beschränkung auf bilaterale Konflikte eine unrealistische Verkürzung ist, da historisch betrachtet kriegerische Auseinandersetzungen selten nur zwei Staaten betreffen.

Ungenauigkeit könnte man Cederman bei verschiedenen Entscheidungen vorwerfen. Die Analyse dieser Vereinfachungen und Erforschung alternativer Modellierungen stellt eine interessante Fragestellung für weiterführende Arbeiten dar, ist aber im Rahmen der hier zu beantwortenden Fragestellung zu umfangreich.

Anders verhält es sich mit den Ad-Hoc-Entscheidungen für die numerischen Werte des Modells. Einige der gewählten Werte auf denen das ursprüngliche Modell beruht, werden auf ihre Validität geprüft (siehe Abschnitt 3.4.3). Werden konkrete Zahlen in das Programm übernommen, sind diese häufig, aber nicht immer auch durch Replikation und Experimente auf ihre Wirkung für die Resultate des Programms geprüft. In Kombination werden die verschiedenen Werte allerdings nie getestet. Wie in Abschnitt 2.2.3 erörtert, ist ausführliches Testen ein unerlässliches Werkzeug bei der Entwicklung von Modellen, dessen Wichtigkeit von Cederman (1997) selbst angemerkt wird. Gerade deshalb ist es interessant zu untersuchen, welche Auswirkung eine Variation der verschiedenen Werte kombiniert hat.

Es soll für die Analyse der Parametersensibilität die Auswirkung der Variation dieser Werte untersucht werden. Der dazu notwendige Versuchsaufbau basiert auf dem im Folgenden entwickelten Modell.

5. Nachbildung des EPM

5.1. Nachbildung des EPM

Das Modell, mit dem der Versuch durchgeführt werden soll, erfolgt auf Basis der ursprünglichen Modellierung des EPM. Die oben beschriebenen Funktionalitäten sollen so exakt wie möglich nachgebildet werden. Dies dient als Grundlage des Versuchsaufbaus, um Schlüsse aus dessen Ergebnissen auf das ursprünglich EPM übertragen zu können. Damit anhand des Versuchs Aussagen über die Gültigkeit des EPM gemacht werden können, muss das nachgebildete Modell dem Original so ähnlich wie möglich sein. Zu diesem Zweck wurde der ursprüngliche Code freundlicherweise von Herrn Cederman zur Verfügung gestellt (vgl. Anhang A.3). Leider ist dieser stellenweise unvollständig, sodass eigene Modellierungsentscheidungen getroffen werden, um diese Lücken zu füllen. Auch ist der Code, bedingt durch die Entwicklung ab 1993, in der Programmiersprache *Pascal* geschrieben.

5.1.1. Programmiersprache

Veröffentlicht im Jahr 1970, entspricht *Pascal* heute nicht mehr den Anforderungen an eine Programmiersprache. Stattdessen soll hier für die Modellierung *Python* verwendet werden. Diese Wahl wurde getroffen, weil *Python* eine universelle, gut lesbare Sprache mit kompaktem Code ist, deren umfangreiche Standardbibliotheken und offene Entwicklergemeinschaft eine schnelle Entwicklung von übersichtlichen Programmen ermöglichen. Weitere Kriterien waren *Pythons* Objektorientierung, die sich für das Modell von Vorteil ist, und persönliche Präferenz und Fähigkeiten.

5.1.2. Entwicklung

Da der originale Code nicht direkt verwendet werden kann und nur teilweise zur Verfügung gestellt werden konnte, soll das neue Modell mithilfe des zur Verfügung gestellten Originalcodes in moderner Form nachgebildet werden. Da hierfür einige Modellierungsentscheidungen getroffen werden müssen, wurde das Modell nach Prinzipien des modernen Software Enginee-

ring entwickelt, um die Qualität des Ergebnisses sicherzustellen und einen Leitfaden für eine gezielte Entwicklung und Orientierung zu bieten.

Da das Modell in Einzelarbeit statt im Team entwickelt wurde, entfällt bei der Auswahl des Modells der Aspekt der Teamplanung. Daher wurde ein generisches iteratives Modell der Softwareentwicklung gewählt (vgl. Larman und Basili (2003)), das auf die kontinuierliche Entwicklung des Modells bis zur Lauffähigkeit abzielt. In diesem Sinne wird der Code in den Phasen:

- Requirements/Planung
- Design/Implementierung
- Testing
- Evaluation

in mehreren Iterationen erzeugt. Alternative Vorgehen zur Softwareentwicklung wie z.B. das Wasserfallmodell oder agile Methoden wurden als zu umfangreich, oder als zu sehr auf Teamarbeit ausgerichtet empfunden.

Die Requirements des Codes sind durch das EPM vorgegeben und orientieren sich an den oben beschriebenen Funktionalitäten. Diese werden aufgrund der ausführlichen Beschreibung ab Abschnitt 3.4.1 hier nicht erneut aufgeführt.

5.1.3. Design und Implementierung

Für das Design des Modells mussten einige Entscheidungen getroffen werden, mit denen die Logik des *Pascal*-Modells in modernen Code umgewandelt werden konnten. Zu diesem Zweck wurde ein objektorientiertes Modell gewählt. Dies ist insofern sinnvoll, als dass eine Instanz des Systems erzeugt werden kann, die wiederum so viele Instanzen von Staaten erzeugt wie notwendig sind, um das Spielfeld zu füllen. Außerdem ist von Vorteil, dass das gesamte Modell über Aufruf der System-Instanz erzeugt und mit Werten initiiert werden kann. Dies ist für den Versuchsaufbau hilfreich, um der Simulation verschiedene Werte als Parameter übergeben zu können. Die übergebenen Werte müssen so nur einmal bei der Initiierung an das System übergeben werden, und werden von der übergeordneten Klasse *System* dort, wo sie die einzelnen Akteure benötigen, direkt an sie weitergegeben. Die agentenorientierte Programmierung wird so als Abstraktion der Objektorientierung nachgebildet, indem sich die Staaten als Agenten an der Handlungslogik klassischer Agenten orientieren, die nach dem „*sense - reason - act*“-Modell miteinander interagieren und auf ihre Umwelt reagieren. Die „*Beliefs*“ der Agenten

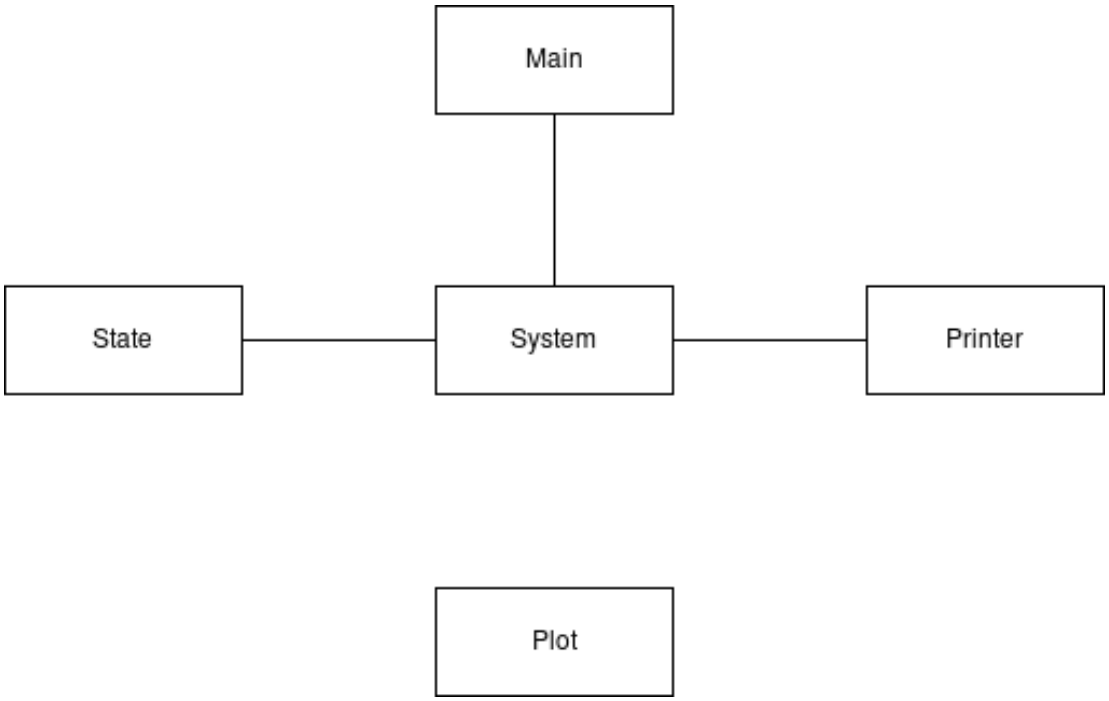


Abbildung 5.1.: UML-Diagramm des Modells

werden durch Variablen wie beispielsweise der *Prey/Predator*-Orientierung repräsentiert (vlg. Wooldridge und Jennings (1995)). Insofern wurde hier die Agentenlogik als in den Objekten ausreichend repräsentiert empfunden. Aus diesem Grund wurde auch die Objektorientierung anderen Architekturstilen wie funktionaler oder prozeduraler Programmierung vorgezogen. Das System wurde analog zu Abbildung 5.1 entwickelt und soll die im Diagramm abgebildeten Funktionalität anbieten. *System* übernimmt die Rolle des Koordinators des Modells, wird selbst initialisiert und initialisiert wiederum *State*, wo die Agenten mit ihrer *„sense - reason - act“*-Logik modelliert werden. *Main* initialisiert die Simulation und automatisiert den Versuchsablauf. *Printer* wird von *System* angesprochen, um die Ergebnisse in eine CSV-Datei zu schreiben. *Plot* ist für die grafische Darstellung der Ergebnisse zuständig. Das Modell wurde dabei nah an der Logik des EPM entwickelt, um ein möglichst ähnliches Verhalten und damit möglichst verallgemeinerbare Ergebnisse zu erhalten.

Hierzu fand eine Abwägung zwischen Annäherung an das Original und Performanz des Codes statt. Durch die Weiterentwicklung von Programmiersprachen und Technologie ist die Implementierung des EPM fast vollständig überholt und deshalb deutlich aufwendiger und weniger performant, als es eine heutige Implementierung wäre. Eine vollständige Neuimple-

mentierung der Logik des Programms bedeutet daher eine deutliche Verbesserung zum Beispiel von Laufzeit und Ressourcennutzung des Programms. Dies würde erfordern, ein Programm mit einer völlig neuen Struktur anhand der Requirements zu erzeugen. Die Gefahr bei diesem Ansatz besteht darin, die Requirements zwar zu erfüllen, sich durch die strukturellen Änderungen aber so weit vom Original zu entfernen, dass mögliche Artefakte in der Programmierung und den Ergebnissen im Vergleich nicht auffallen, sondern durch die Neuimplementierung unerkannt bleiben. Dies würde unbemerkt die Aussagekraft der Ergebnisse schwächen und so die Rückschlüsse, die vom neuen Modell auf das EPM gezogen werden, schwächen.

Daher wurde für diese Arbeit eine Implementierung gewählt, die sich in der Struktur und den Algorithmen am EPM orientiert. Die Verschlechterung der Laufzeit und Ressourcennutzung wurde in Kauf genommen, da es sich um ein abgeschlossenes System handelt welches nur in der Versuchsumgebung genutzt wird. Im Fall beispielsweise einer Applikation die in einem Unternehmen eingesetzt wird und daher performant sein muss, müsste diese Entscheidung anders abgewogen werden. In diesem Fall wurde es aber als sinnvoll eingeschätzt, die Ergebnisse möglichst aussagekräftig zu machen und dafür eine Verschlechterung der Performance hinzunehmen.

Daher wurde das EPM nicht eins zu eins übernommen, aber dessen Algorithmen und Methodenaufrufe als Leitfaden für die Entwicklung genutzt.

Mit den Requirements und dem Leitfaden des EPM wurde das System entwickelt. Dazu wurde zunächst ein erster Entwurf erzeugt, in dem Versuch die Requirements vollständig abzudecken.

5.1.4. Testing und Evaluation

Mittels Unit-Tests wurde dieser erste Entwurf im nächsten Schritt getestet. Es wurde zunächst jede Funktion des Codes auf Positiv-, Negativ- und Grenzfälle getestet. Hier wurde sowohl auf semantischer als auch auf syntaktischer Ebene der Code darauf geprüft, ob er die Requirements erfüllt. Dies wurde iterativ so lange wiederholt, bis der Code als logisch und inhaltlich stimmig eingeschätzt wurde.

Die Ergebnisse wurden mit denen des EPM verglichen. Nach einigen kleinen Anpassungen, vor allem in Bezug auf Parameter, die von Cederman (1997) nicht konkret benannt wurden, konnten diese Werte mit einigen geringen Abweichungen reproduziert werden. Als konkrete Ergebnisse des EPM wird ausschließlich die Anzahl der am Ende eines Simulationslauf bestehenden Staaten dokumentiert. Anhand dieser Metrik wurden die Ergebnisses verglichen. Kleine Abweichungen in den Ergebnissen werden darauf zurückgeführt, dass dank der eher geisteswissenschaftlichen als technischen Beschreibung in Cederman (1997) und des unvollständigen Originalcodes ein gewisses Maß an „Reverse Engineering“ des Modells stattfinden

musste. Diese Abweichungen sind nicht von der Hand zu weisen, befinden sich aber einerseits in einem vertretbaren Rahmen, andererseits schwankt das Maß der Annäherung je nach Parameterkonfiguration des Modells, sodass davon ausgegangen werden kann, dass die beiden Modelle mit gleicher Konfiguration die gleichen Ergebnisse erzielen würden.

Das allgemeine Verhalten des EPM wird so reproduziert, dass das Modell als aussagekräftig genug eingeschätzt wird, um Schlüsse die aus ihm gezogen werden mit gewisser Vorsicht und unter Beachtung dieser Abweichungen auf das EPM übertragen zu können.

5.1.5. Das Modell

Die Felder und Methoden der Klassen *State* und *System* sind in Abbildung 5.2 detailliert aufgeführt, um die innere Logik der Klassen zu verdeutlichen. Dabei wurden möglichst sprechende Namen gewählt, um das Modell intuitiv verständlich zu machen. Während es zu aufwendig wäre, die gesamte Implementierung des Modells ausführlich darzustellen, soll hier auf einige entscheidenden Stellen im Code eingegangen werden.

State

```
class State():
  def make_decision_prey(self):
    if self.involved_in_warfare:
      self.next_action = 'd'
    elif self.system_has_alliances:
      for ally in self.allies:
        if ally.involved_in_warfare:
          self.in_warfare(ally.involved_in_warfare_with)
          break
        if not self.involved_in_warfare:
          self.not_in_warfare()
          for neighbor in self.neighboring_states:
            if self.history_with_neighbor[neighbor] == 'd':
              self.in_warfare(neighbor)
              break
```

Listing 5.1: Code Entscheidungsverlauf *Prey*

In Listing 5.1 ist der Entscheidungsablauf eines *Prey*-Akteurs dargestellt: Die nächste Aktion, entweder Angriff oder friedliches Verhalten, wird unter Berücksichtigung der letzten Aktion, der Verbündeten die möglicherweise Hilfe benötigen und der Angriffe durch Nachbarn getroffen.

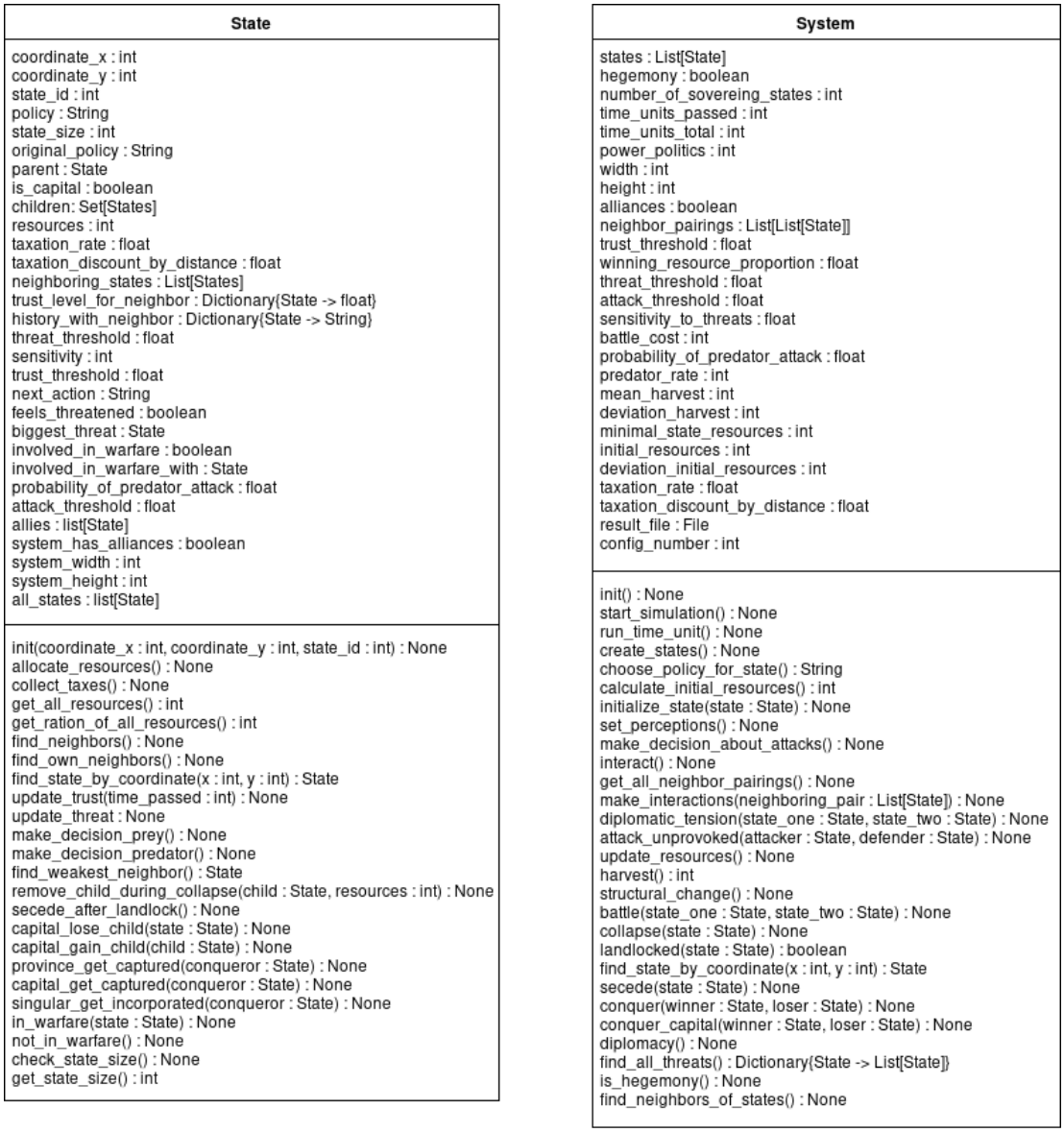


Abbildung 5.2.: Die Klassen *State* und *System*

Die Entscheidung von *Predator*-Akteuren verläuft entsprechend, mit einem Zusatz, dargestellt in Listing 5.2.

```
def make_decision_predator(self):
  . . .
  rand_int = randint(1, 101)
    if rand_int <= self.probability_of_predator_attack:
      weakeast_neighbor = self.find_weakest_neighbor()
      if weakeast_neighbor.resources > 0:
        if (self.resources / weakeast_neighbor.resources) < self.
            attack_threshold:
          self.in_warfare(weakeast_neighbor)
      else: self.in_warfare(weakeast_neighbor)
```

Listing 5.2: Code Entscheidungsverlauf *Predator*

Es wird zusätzlich ein Zufallswert erzeugt. Fällt dieser in den Wertebereich der Wahrscheinlichkeit eines unbegründeten Angriffs, greift der Akteur seinen ressourcenschwächsten Nachbarn an, sofern das Ressourcengefälle zwischen beiden ausreichend groß ist.

System

Das *System* initialisiert die einzelnen Staaten und organisiert den Ablauf jeder einzelnen Zeiteinheit.

Die Ressourcen, die sowohl bei der Initialisierung als auch bei der Ernte in jeder Zeitrunde verteilt werden, werden als zufällige Abweichung von einem festgelegten Mittelwert errechnet, dargestellt in Listing 5.3.

```
class System():
  def harvest(self):
    rand_harvest = randint((-1 * self.deviation_harvest), self.
        deviation_harvest)
    return self.mean_harvest + rand_harvest
```

Listing 5.3: Code zur zufälligen Errechnung der Ernte

In jeder Zeiteinheit werden die in Listing 5.4 dargestellten Schritte durchlaufen.

```
def run_time_unit(self):
  self.set_perceptions()
  self.make_decisions_about_attacks()
  self.interact()
  self.update_resources()
  self.structural_change()
```

```
7   if self.alliances:
8       self.diplomacy()
9   self.is_hegemony()
```

Listing 5.4: Code Ablauf einer Zeiteinheit

Diese Schritte orientieren sich stark am Ablauf der Zeiteinheiten im EPM. Im Schritt *set_perceptions()* orientiert sich jeder Staat daran, welches Vertrauen er in dieser Zeiteinheit seinen Nachbarn entgegenbringt bzw. wie bedroht er sich von ihnen fühlt. Danach folgen Entscheidungen über Angriffe, was die weiter oben beschriebenen Entscheidungsmethoden abhängig von der *Policy* des Akteurs aufruft. Die Interaktionen im nächsten Schritt finden jeweils zwischen zwei Akteuren statt, die ihre Entscheidung über Konflikt oder Nicht-Angriff abgleichen. Abhängig davon, ob ein Konflikt stattfindet, werden daraufhin in *update_resources()* die Ressourcen an den Fronten verteilt. Nachdem die Konflikte ausgetragen wurden, werden in *structural_change()* die Gebiete dementsprechend neu verteilt und es finden Eroberungen und Auflösungen statt.

Wurde zu Beginn der Simulation entschieden, dass es Bündnisse zwischen Akteuren geben kann, werden diese im nächsten Schritt geschlossen. Eine Allianz zwischen zwei oder mehr Akteuren entsteht, wenn sich beide von einem dritten Akteur in gewissem Maß bedroht fühlen.

Im letzten Schritt wird geprüft, ob das System hegemoniell ist, also nur noch aus einem Akteur besteht, der alle anderen Staaten erobert hat. In diesem Fall endet die Simulation vorzeitig, da keine weiteren Eroberungen stattfinden können.

Ein Konflikt läuft ab wie in Listing 5.5 dargestellt.

```
1  def battle(self, state_one, state_two):
2    winner = None
3    loser = None
4    if (state_one.state_size > 0) & (state_two.state_size > 0):
5      average_resources_one = (state_one.resources / state_one.
           state_size)
6      average_resources_two = (state_two.resources / state_two.
           state_size)
7    else:
8      average_resources_two = state_two.resources
9      average_resources_one = state_one.resources
10     if (average_resources_one != 0) &
11     (average_resources_two != 0):
12       if average_resources_one / average_resources_two >= self.
             winning_resource_proportion:
13         winner = state_one
14         loser = state_two
```

```
15      elif average_resources_two / average_resources_one >= self.
            winning_resource_proportion:
16        winner = state_two
17        loser = state_one
18      if winner != None:
19        self.conquer(winner, loser)
```

Listing 5.5: Code Konfliktverlauf

Hier werden Gewinner und Verlierer des Konflikts über die Ressourcen ermittelt. Im Fall einer Eroberung nimmt der Gewinner das umkämpfte Gebiet ein, dargestellt in Listing 5.6

```
1 def conquer(self, winner, loser):
2   if loser.state_size > 1:
3     if not loser.is_capital:
4       loser.province_get_captured(winner)
5     else:
6       self.conquer_capital(winner, loser)
7   else:
8     loser.singular_get_incorporated(loser)
9     self.number_of_sovereign_states -= 1
```

Listing 5.6: Code Eroberung

Hier wird zwischen mehreren Szenarien der Eroberung unterschieden: Entweder es handelt sich bei dem Besiegten um eine Hauptstadt oder Provinz, und um einen singulären oder zusammengesetzten Staat. Diese Fälle werden unterschiedlich behandelt (vlg. Abschnitt 3.4.1).

Dieses Modell dient als Basis des Versuchsaufbaus, in dem sämtliche numerischen Parameter, welche für die Simulation notwendig sind, variiert werden können. Der vollständige Code kann in Anhang A.1 eingesehen werden.

5.2. Modellierung der Parametervariation

Die im EPM verwendeten numerischen Werte wurden als Parameter der Klasse *System* modelliert. Für jeden Simulationslauf kann für jeden dieser Parameter ein neuer Wert eingesetzt werden. Dies ermöglicht eine automatische Variation der Werte in verschiedenen Simulationsläufen, die für den Versuch notwendig sind.

Das Modell wurde so strukturiert, dass sämtliche Parameter für die Durchführung des Experiments möglichst leicht und automatisiert einzufügen sind. Daher bietet die Klasse *System* eine Schnittstelle für *Main* an, mittels derer alle numerischen Werte für das System

mit einem Aufruf übergeben werden können, der im nächsten Schritt einen vollständigen Durchlauf der Simulation startet (siehe Abschnitt 5.1.3).

Es ist anzumerken, dass das Modell unter der Annahme verwendet wird, dass es das gleiche Verhalten zeigt wie das EPM. Dies ist nicht abschließend zu beweisen. Kleinere Abweichungen in den Ergebnissen zeigen, dass Schlüsse die von diesem Modell auf das EPM gezogen werden, mit einer gewissen Vorsicht zu behandeln sind. Da Cederman seine Parameterkonfiguration für seinen Versuch nicht vollständig beschreibt, sind Abweichungen nicht endgültig festzustellen. Da das Modell sämtliche Requirements erfüllt und ausführlich auf seine Funktionalität getestet wurde, wird im Folgenden von einer gleichen Funktionalität der beiden Modelle ausgegangen.

6. Versuch

In diesem Abschnitt wird der Aufbau des in Abschnitt 6.3 durchgeführten Versuchs zunächst erläutert. Dies beinhaltet die allgemeine Beschreibung des Vorgehens, des Aufbaus und der erwarteten Ergebnisse.

6.1. Versuchsablauf

Der Versuch wird durchgeführt, indem zunächst ein Wertebereich für jeden der zu variierenden Parameter festgelegt wird. Diese Parameter sind:

- Die Schwelle, die überschritten werden muss, damit Vertrauen zwischen zwei Akteuren besteht
- Das Verhältnis, das zwischen den Ressourcen zweier kämpfender Akteure bestehen muss, damit einer der Akteure den Konflikt gewinnt
- Die Schwelle, die überschritten werden muss damit sich ein Akteur von einem anderen bedroht fühlt
- Die prozentuale Wahrscheinlichkeit, mit der ein *Predator* angreift
- Die Kosten eines Konflikts für beide Seiten
- Die Verhältnis der Ressourcen eines *Predators* zu denen eines Nachbarn, das erreicht werden muss damit dieser angreift
- Der Anteil von *Predators* in der Gesamtmenge der Staaten
- Der Mittelwert der Ernte pro Zeiteinheit
- Die mögliche maximale Abweichung von diesem Mittelwert
- Der Mittelwert der initial vergebenen Ressourcen
- Die mögliche maximale Abweichung von diesem Mittelwert

- Der Steuersatz für die Provinzen in Prozent
- Die Ermäßigung dieses Steuersatzes mit steigender Entfernung.

Jedem dieser Parameter wird ein zufälliger Wert innerhalb seines Definitionsbereiches zugeordnet. Diese Konfiguration wird an das *System* übergeben und die Simulation durchlaufen. Das System enthält durch die zufällige Verteilung von Ressourcen zu Beginn der Simulation und zufällige Einteilung in *Prey* und *Predator* zwangsläufig einige Zufallswerte. Um einer Verzerrung der Ergebnisse durch zufällig entstandene Ausreißer entgegenzuwirken, wird die Simulation in jeder Konfiguration mehrfach durchgeführt. Die mögliche Anzahl an Wiederholungen liegt dabei in einem unendlichen Wertebereich. Es wurden zunächst zehn Wiederholungen als eine sinnvolle Anzahl gewählt, da sie groß genug erscheint, um Schwankungen der Zufallswerte auszugleichen. Sollte sich im Versuchsverlauf diese Anzahl von Wiederholungen als zu gering oder zu hoch herausstellen, ist sie erneut zu evaluieren.

Ebenso muss die Anzahl an Zeiteinheiten, die eine Simulation maximal dauern kann festgelegt werden. Hier wurde zunächst 600 als Wert gewählt, der sich nach einigen Testläufen bewährt hat. Es zeigte sich, dass Hegemonie oder eine dauerhafte Verringerung der Anzahl von souveränen Staaten in fast allen Fällen in den ersten 500 Zeiteinheiten stattfindet, nach welchen sich das Ergebnis kaum noch veränderte. Zur Sicherheit werden hier 600 Zeiteinheiten als Maximum gewählt, um sicherzustellen dass signifikante Änderungen stattfinden können bevor die Simulation endet.

Das Ergebnis eines Durchlaufs setzt sich aus der Anzahl der am Ende bestehenden Staaten, und der Anzahl der Eroberungen und Zerfälle von Staaten zusammen.

Es werden nun 100 dieser Konfigurationen erzeugt und jeweils zehn Simulationen mit ihnen durchgeführt. Die Ergebnisse werden zusammen mit den Kennzahlen der Konfiguration in einer CSV-Datei gespeichert, um die Auswertung zu ermöglichen.

6.2. Versuchsaufbau

6.2.1. Definitionsbereiche

Einige Definitionsbereiche lassen sich schnell eingrenzen: Parameter, die prozentuale Anteile beschreiben haben notwendigerweise einen Definitionsbereich D = [0, 100]. Dies betrifft die Wahrscheinlichkeit eines Angriffs, den Anteil von *Predators*, den Steuersatz und die Steuerermäßigung. Eine Schrittweite von eins wird hier als ausreichend genau empfunden und erscheint bei Prozentzahlen sinnvoll.

Für die Schwellen für ein Vertrauen bzw. Bedrohungsgefühl der Akteure wird von Cederman kein konkreter Wert genannt. In Testläufen führte ein Wert von zehn bei beiden Parametern zu einem dem EPM ähnlichen Verhalten. Daher wird zehn als ein sinnvoller Ausgangspunkt betrachtet. Es wird ein Definitionsbereich von D = (1,2,...,25) gewählt. Dies ergibt auch in Bezug auf die Logik des Modells Sinn. Beide Werte werden unter anderem in Bezug auf die Anzahl der bisher vergangenen Zeitrunden berechnet und während der Berechnung mit dieser Anzahl multipliziert. Bei einem Ausgangswert von zehn und maximal 600 Zeiteinheiten ist beispielsweise ein Wert von eins eine genügend große Abweichung, um einen Effekt auf das Gesamtergebnis zu haben. Gleiches gilt für einen maximalen Wert von 25. Auch hier wird eine Schrittweite von eins als sinnvoll betrachtet.

Das Verhältnis von Ressourcen zwischen zwei Akteuren, sowohl für einen Angriff als auch für einen Sieg in einem Konflikt, wird im EPM mit dem Wert drei belegt, ein Akteur muss also die dreifache Menge an Ressourcen seines Nachbarn haben, um diesen besiegen zu können. Daher wird der Wert drei als Ausgangspunkt betrachtet. Das Verhältnis muss größer als eins sein, da in jedem Fall eine Ressourcenüberlegenheit existieren soll. Als realistischer Definitionsbereich wird hier D = [1,1; 5] betrachtet. Fünf stellt die Obergrenze dessen dar, was mit der Ernte pro Runde an Überlegenheit der Akteure realistisch erreicht werden kann. Die Schrittweite beträgt 0,1, was als ausreichend genaue Abdeckung des Parameterraums betrachtet wird, ohne durch zu geringe Schrittweite in verschiedenen Variationen ein sehr ähnliches Ergebnis zu erzielen.

Alle anderen Parameter beziehen sich auf die verteilten Ressourcen. Der Mittelwert der initialen Menge zu Beginn der Simulation beträgt im EPM 50, mit einer maximal möglichen Abweichung von ± 20. Ausgehend von diesem Wert wird für den Mittelwert ein Definitionsbereich D = [5, 200] mit einer Schrittweite von eins verwendet. Die Untergrenze ergibt sich aus der Notwendigkeit, eine gewisse Varianz schon zu Beginn der Simulation zu ermöglichen. Daher muss der Wert groß genug sein, damit Nachbarn zufällig verschieden große Mengen an Ressourcen zugesprochen werden können. Das Vierfache der von Cederman genutzten Ressourcenmenge erscheint als ausreichend groß, um aussagekräftige aber realistische Ergebnisse zu erhalten. Zu beachten ist, dass dabei zufällig ein größere Menge an Ressourcen vergeben werden kann, da diese mit einer zufälligen Abweichung vergeben werden, was den Wertebereich zusätzlich vergrößert.

Die mögliche Variation der Menge der Ressourcen ergibt sich aus der Menge der vergebenen Ressourcen und soll bei der zufälligen Verteilung bei einem Mittelwert M im Extremfall im Bereich [-M, M] möglich sein. Daher ist der Definitionsbereich für die Abweichung [0, M], da positive oder negative Abweichung im System für jeden Akteur einzeln festgelegt werden (siehe Listing 5.3). Im EPM ist die sich daraus ergebende Menge an Ressourcen stets größer eins,

kann aber hier auch null betragen. Dies ist dem Simulationsverlauf nicht abträglich, sondern kann von einem Akteur mit großer Wahrscheinlichkeit bereits in der nächsten Zeiteinheit durch die erhaltene Ernte ausgeglichen werden.

Ausgehend von den initialen Ressourcen wird der Definitionsbereich der Ernte pro Runde festgelegt. Dieser wird mit gleicher Begründung mit D = [5, 200] mit Schrittweite eins, und einer Abweichung von bis zu ± 100% der Erntemenge gewählt.

Der letzte festzulegende Wert beschreibt die Kosten für einen Konflikt. Dieser wird im EPM mit fünf festgelegt, was 1/5 der durchschnittlichen anfänglichen Ressourcen beträgt. Um ein realistisches Verhältnis der Ressourcen beizubehalten, wird der Definitionsbereich im Modell im Verhältnis zu den initialen Ressourcen modelliert, und kann dabei zwischen 1% und 100% dieser Ressourcen betragen.

Für alle Parameter wird für jede Konfiguration ein zufälliger Wert innerhalb ihres Wertebereichs erzeugt und das System damit instantiiert.

6.2.2. Kenngrößen

Die gewählten Kenngrößen zur Messung sind die Anzahl der Staaten, die zu Ende der Simulation existieren, und die Anzahl der „Machtkämpfe“, also Eroberungen und Zerfall von Staaten.

Die erste Metrik wurde unter anderem zur Vergleichbarkeit mit dem EPM gewählt. Dort wertet Cederman seine Ergebnisse hauptsächlich nach der final bestehenden Anzahl von Staaten aus (vgl. Cederman (1997), S.99). Zudem ist die Frage, wie häufig alle Staaten bestehen bleiben, wie häufig Hegemonie erreicht wird und wie häufig das Ergebnis dazwischen liegt ein geeigneter Indikator für die Funktionsweise des Modells, weil hier die ursprüngliche Frage Cedermans nach dem Zerfallen und Entstehen von Staaten besonders verdeutlicht wird.

Die zweite Kenngröße, die Anzahl an Machtkämpfen, verdeutlicht weniger das finale Ergebnis als die innere Arbeitsweise des Modells. Es ist möglich und nicht unwahrscheinlich, dass in einer Simulation am Ende zwar noch die maximale Anzahl an einzelnen Staaten existiert, diese aber nicht kampflos, sondern durch häufiges Erobern und Zerfallen entstanden ist. Um diese inneren Mechanismen der Simulation zu verdeutlichen, wurden die Machtkämpfe als zweite Metrik gewählt.

Beispielhaft sieht der Code für die Durchführung des Versuchs für zwei der Parameter aus wie in Listing 6.1 dargestellt.

```
1 class Main()
2   . . .
3   percentage_low = 0
4   percentage_high = 100
```

```
5   . . .
6   def configuration()
7     . . .
8     self.predator_rate = randint(self.percentage_low, self.
          percentage_high)
9     self.probability_of_predator_attack = randint(self.
          percentage_low, self.percentage_high)
10    . . .
11  def run_simulation()
12    sys = system.System()
13    . . .
14    sys.probability_of_predator_attack = self.
          probability_of_predator_attack
15    sys.predator_rate = self.predator_rate
16    . . .
17  def run_experiment(self):
18    self.configuration()
19    for i in range(1, 11):
20      self.run_simulation()
21      i += 1
```

Listing 6.1: Ausschnitt Code Versuchsaufbau

6.2.3. Erwartete Ergebnisse

Erwartet wird als Ergebnis ein Zusammenhang zwischen den beiden Kenngrößen. Wie stark oder schwach dieser ausfällt, lässt sich schwierig einschätzen, da zwar inhaltlich ein Zusammenhang zwischen beiden besteht, das System aber einige zufällige Kennzahlen enthält.

Idealerweise lassen sich in den Ergebnissen distinkte Punktwolken erkennen, was darauf hindeuten würde, dass sich zwei oder mehr völlig unterschiedliche Ergebnisgruppen aus dem gleichen Modell erzeugen lassen. Dies würde die Vermutung unterstützen, dass beim Modell eine Parametersensibilität existiert.

Es ist auf Basis des Versuchsaufbaus schwer einschätzen, wie wahrscheinlich dieses Ergebnis ist. In diesem und allen anderen Fällen müssen die Ergebnisse sorgfältig ausgewertet und auf ihre möglichen Aussagen über das EPM untersucht werden.

6.3. Versuchsdurchführung

Da durch die erforderlichen Iterationen und Wiederholungen verhältnismäßig viel an Rechenleistung notwendig ist, wurde für die Durchführung der Berechnungen ein externer Server angemietet, um eine zuverlässige Durchführung sicherzustellen. Um eine volle Auslastung zu gewähren, wurden jeweils acht Simulationen parallel durchgeführt, was auf einem Ubuntu-Server mit dem Befehl in Listing 6.2 (Tange, 2011) erreicht wurde.

```
1 parallel -n0 -j8 --linebuffer python3 -u ./main.py ::: {1..100}
```

Listing 6.2: Befehl zum Starten des Versuchs

Die Ergebnisse der Durchführung werden im Folgenden dargestellt und ausgewertet.

7. Versuchsauswertung

7.1. Ergebnisse des Versuchs

Ergebnis des Versuchs sind nach 100 Konfigurationen, die jeweils zehn mal an die Simulation übergeben wurden, 1000 Datensätze. Hinzu kommen zehn Konfigurationen, die verwendet wurden um die Lauffähigkeit des Versuchsaufbaus zu testen. Da diese unter den gleichen Voraussetzungen erzeugt wurden wie die Simulationen des Versuchs, können sie zu den Versuchsergebnissen hinzugefügt werden.

Jeder Konfiguration können also zehn Werte pro Kenngröße zugeordnet werden. Um vergleichbare Werte zu erhalten werden aus diesen Wertemengen Mittelwert und Standardabweichung berechnet (vgl. A.2). Grafisch dargestellt in Bezug auf Machtkämpfe und Staaten am Ende der Simulation entsteht das in 7.1 dargestellte Ergebnis.

Für bessere Übersichtlichkeit werden in Abbildung 7.2 nur die Mittelwerte ohne Standardabweichung und Konfigurations-IDs dargestellt.

Vor allem in Abbildung 7.2 wird sichtbar, dass es zwischen den beiden Kenngrößen einen schwachen negativen Zusammenhang gibt: Hohe Werte auf einer Achse korrelieren mit niedrigen Werten auf der andern Achse und umgekehrt. Ebenso fällt eine deutliche Häufung um den Punkt (0|100) auf. Circa ¾ der Ergebnisse befinden sich auf einer geringen Fläche in diesem Bereich. Um dies stärker zu verdeutlichen wird in Abbildung 7.3 die Raumdichte der Ergebnisse dargestellt. Dies entspricht nicht ganz den erwarteten bzw. erhofften Ergebnissen. Ein Zusammenhang ist vorhanden, aber nicht stark und es existieren keine klar unterscheidbaren Punktwolken. Dennoch sind die Ergebnisse für die Fragestellung von Interesse.

Was diese Ergebnisse für das hier aufgestellte Modell bedeuten und welche Schlüsse aus ihnen für die Fragestellung gezogen werden können, soll im Folgenden diskutiert werden.

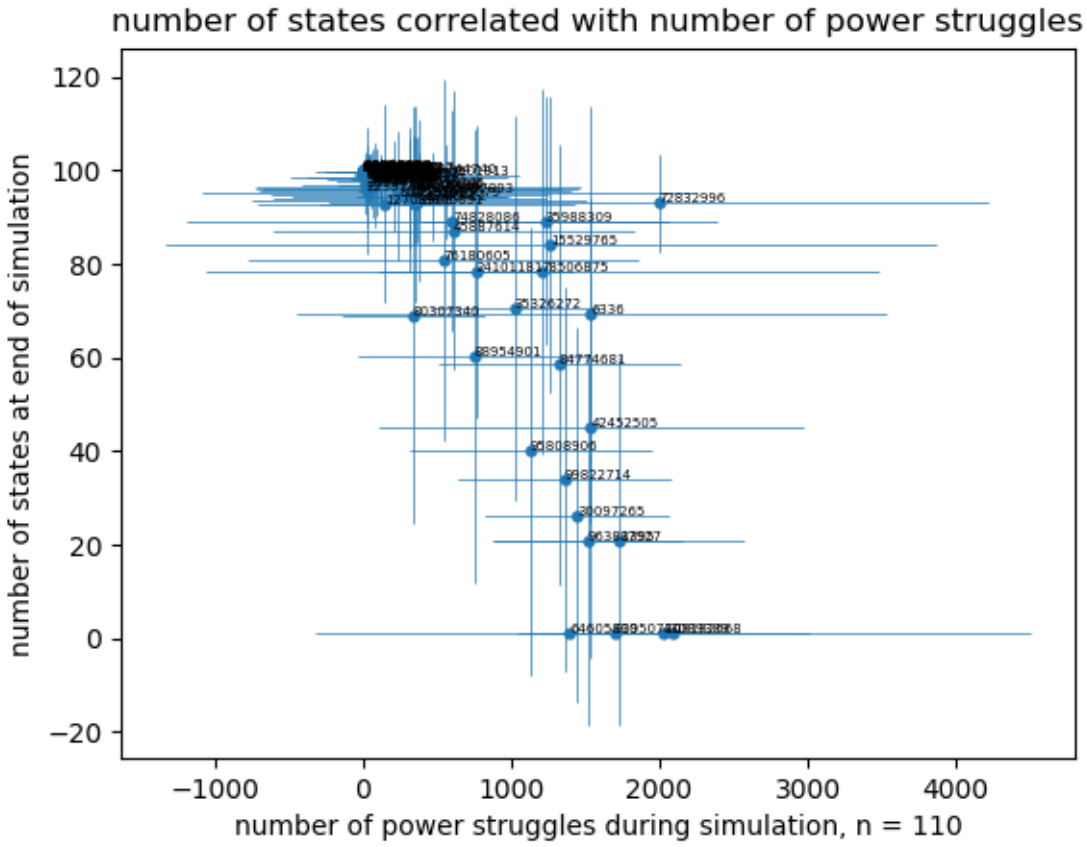


Abbildung 7.1.: Mittelwerte und Standardabweichungen der Ergebnisse der Konfigurationen

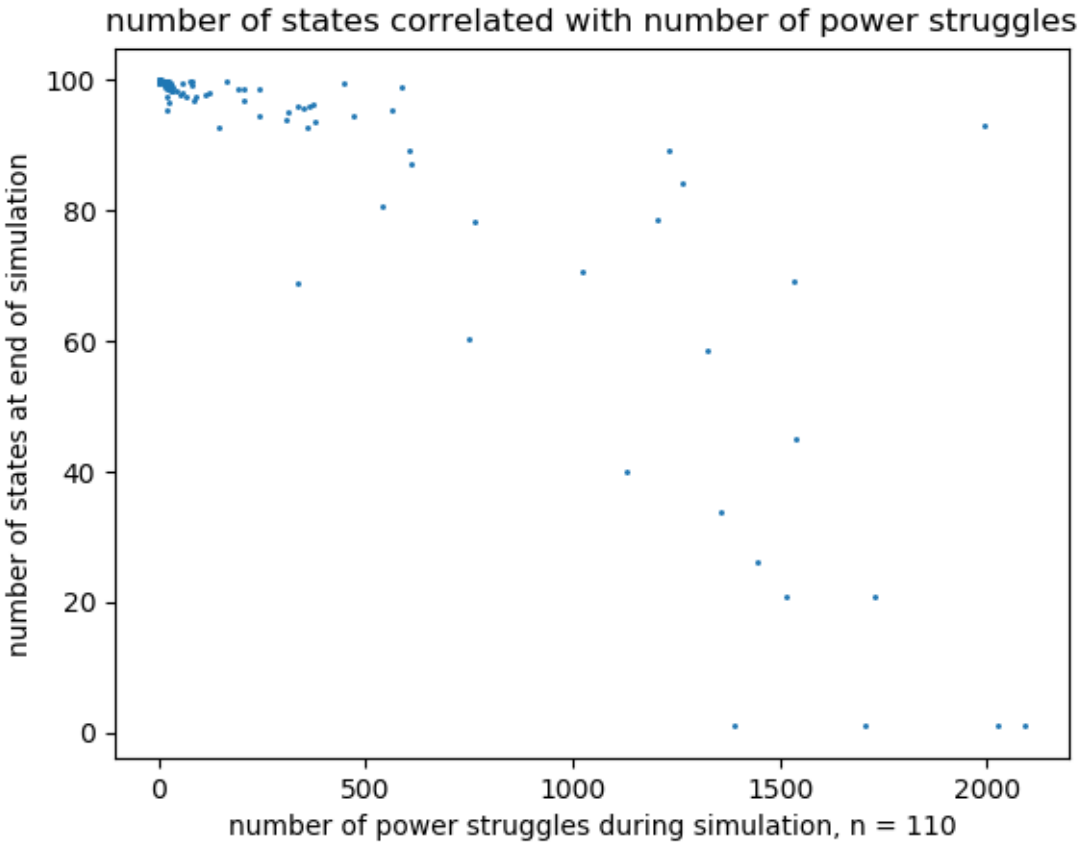


Abbildung 7.2.: Mittelwerte der Ergebnisse

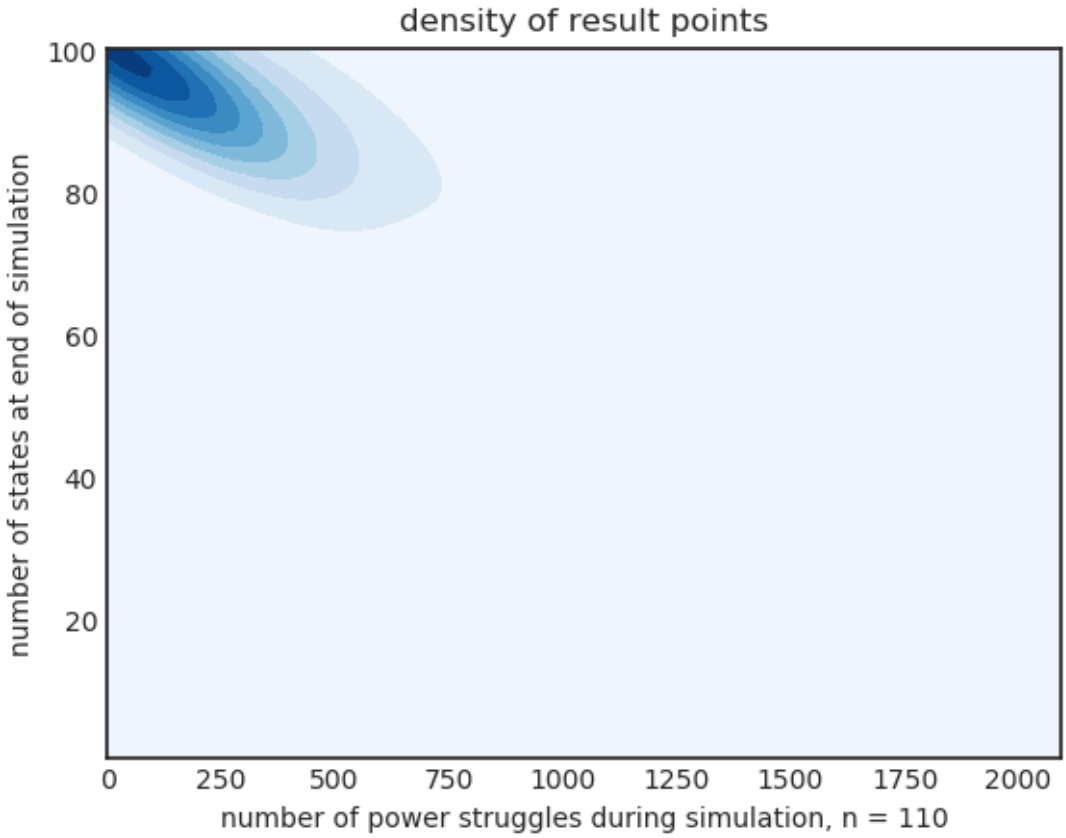


Abbildung 7.3.: Raumdichte der Ergebnisse

7.2. Schlussfolgerungen

7.2.1. Interpretation der Ergebnisse

Der schwache negative Zusammenhang zwischen der Anzahl von Staaten und der Anzahl von Machtkämpfen lässt sich aus des Logik des Modells erklären. Finden keine Machtkämpfe statt, verändert sich die Anzahl der Staaten nicht; gibt es viele Machtkämpfe ist die Wahrscheinlichkeit groß dass sich die Anzahl der Staaten verringert oder Hegemonie erreicht wird.

Dass kein starker, sondern nur ein schwacher Zusammenhang besteht, erklärt sich durch die Möglichkeit, dass trotz vieler Machtkämpfe dennoch die ursprüngliche Anzahl von Staaten bestehen bleibt, weil beispielsweise durch eine hohe Dichte an *Predators* zwar häufig Provinzen erobert werden, diese aber schon in der nächsten Zeiteinheit von einem anderen *Predator* erobert werden, sodass keine größeren zusammengesetzten Staaten entstehen können. Umgekehrt besteht auch die Möglichkeit, mit wenigen Machtkämpfen Hegemonie zu erreichen. Ein einziger starker *Predator* reicht aus, um in 99 Zeiteinheiten sämtliche anderen Staaten einzunehmen. All diese Faktoren begründen den schwachen Zusammenhang zwischen der Anzahl von Staaten und der Anzahl von Machtkämpfen. Da viele Werte der Simulation zufällig erzeugt werden und so auch innerhalb einer Konfiguration größere Schwankungen in den Werten existieren können, ist es unwahrscheinlich einen starken Zusammenhang zu erreichen.

Interessanter für die Auswertung ist die hohe Wertedichte um den Punkt (0|100). Wenn sich ¾ der Ergebnisse in diesem Bereich befinden, bietet dies Anlass für eine genauere Untersuchung.

Inhaltlich bedeutet jeder Punkt der sich genau auf (0|100) befindet, dass während der Simulation keine Veränderung des Startzustands stattfand: Es gab keine Machtkämpfe und keine Änderung in der Anzahl der Staaten. Während dies für einige zufällige Konfigurationen zu erwarten ist, etwa durch sehr geringe Anzahlen an *Predators* oder eine unerreichbar hohe Angriffsschwelle, ist eine derart starke Häufung eher als Symptom eines Grundproblems des Modells zu interpretieren. Wenn ein Großteil der Werte im Definitionsbereich dazu führt, dass die Simulation effektiv nicht mehr funktioniert, entspricht dies einem Problem des Modells. Dieses erzeugt in diesem Fall nur unter sehr speziellen Umständen, also nur für einige Konfigurationen, sinnvolle Ergebnisse.

Auffällig ist, dass es bei den Konfigurationen, die zu Simulationen mit Machtkämpfen und Änderungen in der Anzahl der Staaten führen, keine offensichtlichen Regelmäßigkeiten in den zufällig erzeugten Parametern gibt (vgl. Anhang A.2.3). Sowohl in den funktionalen als auch den nicht-funktionalen Konfigurationen existiert eine Streuung aller Werte über den gesamten Definitionsbereich. Unter der Annahme, dass durch die Wiederholung und

Mittlung der Ergebnisse die zufälligen Einflüsse der Simulationsläufe vernachlässigt werden können, lässt dies mehrere Schlussfolgerungen zu. Zum einen verhindert kein Parameter, der mit dem minimalen oder maximalen Wert seines Definitionsbereichs belegt ist, den Ablauf der Simulation. Die Definitionsbereiche scheinen also sinnvoll gewählt. Außerdem ist kein einzelner Parameter für den Erfolg oder Misserfolg der Simulation verantwortlich, sondern es gibt verschiedene Kombinationen von Werten, die sinnvolle Ergebnisse erzeugen. Wenn der Erfolg der Simulation nicht von einem einzelnen Parameter abhängt, sondern von der Kombination sämtlicher Werte, und wenn sämtliche Werte innerhalb der Definitionsbereiche zu einer erfolgreichen Konfigurationen gehören können, ist die logische Schlussfolgerung, dass es am Modell selbst liegen muss dass viele der Konfigurationen keine Änderungen in dessen Zustand bewirken.

Somit ist eine Parametersensibilität offengelegt. Das Modell erzeugt nur unter bestimmten Umständen semantisch sinnvolle Ergebnisse. Mögliche Gründe hierfür sind Fehler oder Artefakte im Modell selbst sowie eine erhöhte Parametersensibilität des Modells. Da Fehler und Artefakte durch Tests während der Entwicklung so weit wie möglich ausgeschlossen wurden, muss davon ausgegangen werden, dass das Modell entsprechend der Requirements funktioniert. Daraus folgt, dass die fehlerhafte Funktionalität des Modells aus dem EPM und dessen Modelllogik und damit aus der ursprünglichen Modellierung von Cederman (1997) stammt.

7.2.2. Bedeutung für das EPM

Wird davon ausgegangen, dass das hier entwickelte Modell die gleiche Funktionalität bereitstellt wie das EPM, können die Schlüsse aus dem Modell auf das EPM übertragen werden. In diesem Fall gilt also, dass es auch im EPM einen schwachen negativen Zusammenhang zwischen der Anzahl der Staaten und Anzahl der Machtkämpfe gibt, und dass eine Parametersensibilität existiert.

Cederman (1997) selbst bringt die Kritik an, dass bei CAS die Gefahr der Parametersensibilität von Ergebnissen besteht (vgl. Abschnitt 3.2), die mit Tests so weit wie möglich umgangen werden kann. Allerdings wird nur für wenige der Parameter angegeben, dass sie ausführlich auf ihre Auswirkungen am Gesamtergebnis getestet wurden. Den Erwiderungen aus Abschnitt 3.2, dass es nur um die Etablierung der Möglichkeit verschiedener Ergebnisse gehe und dass intuitive Konfigurationen ein wertvolles Mittel seien, ist schwer zu widersprechen. Wenn es aber um die Wahrscheinlichkeit sinnvoller Ergebnisse geht, zeigt das EPM im hier beschriebenen Versuch Schwächen, da mit großer Wahrscheinlichkeit ein gleichförmiges, wenig aussagekräftige Ergebnis erzielt wird. Gerade hier zeigt sich die Schwäche des Einwands, man müsse

nicht alle Details des Parameterraums kennen (vgl. Abschnitt 3.2). Wird nicht der komplette Parameterraum untersucht, lassen sich zwar mit einigen Konfigurationen sinnvolle, inhaltlich aussagekräftige Ergebnisse erzielen. Die Aussagekraft des gesamten EPM wird aber durch die Möglichkeit, den Ablauf der Simulation durch bloße Parameterkonfiguration zu stören, deutlich geschwächt. Hier kann der Vorwurf angebracht werden, einem nicht vollständig sinnvollen Modell die gewünschten Ergebnisse mittels der passenden Parameter zu entlocken.

Unter Beachtung dieser Ergebnisse stellt sich ebenfalls die Frage nach dem Mehrwert der Erkenntnisse durch das EPM. Da die Ergebnisse keine Überraschenden Sprünge, Kipppunkte o.ä. enthalten, können möglicherweise keine wirklich überraschenden Erkenntnisse aus dem Modell gezogen werden. Stattdessen könnte Cederman (1997) der Vorwurf gemacht werden, dass die Ergebnisse nur die Modellierungsentscheidungen und eigenen Überlegungen widerspiegeln.

Daraus lässt sich die Frage ableiten, ob es sich bei diesem System wirklich um ein CAS handelt, da der Aspekt der Komplexität sich in den eher linearen und teils gehäuften Ergebnissen nicht unbedingt widerspiegelt.

Die Bedeutung dieser Erkenntnisse soll im Folgenden auf die ursprüngliche Fragestellung übertragen werden.

8. Fazit

8.1. Schlussfolgerungen für die Fragestellung

Zu Beginn dieser Arbeit wurde die Frage nach der Parametersensibilität politikwissenschaftlicher Modellbildung gestellt. Hierfür wurden der aktuelle Forschungsstand und das EPM untersucht und anhand des EPM ein eigenes Modell entwickelt. Bei zufälliger Konfiguration dieses Modells ergab sich eine deutliche Häufung von Konfigurationen, die zu Simulationen ohne Machtkämpfe und Änderungen in der Anzahl der Staaten führten. Dies unterscheidet sich zwar von den erhofften Ergebnissen, in denen klare Punktwolken anzeigen, dass das Modell mehrere verschiedene Funktionsweisen hat. Dennoch sind diese Häufungen für die Fragestellung relevant: Wie wird die Aussagekraft eines Modells durch die willkürliche Auswahl von Werten geschwächt?

Ist davon auszugehen, dass ein Großteil der Konfigurationen aus dem begründet ausgewählten Definitionsbereich der Parameter dazu führt, dass das Modell effektiv nicht mehr funktioniert, ist das Modell weniger aussagekräftig als in Cederman (1997) dargestellt. Wie in Abschnitt 2.2.3 erörtert, müssen die Ad-Hoc-Entscheidungen, die bei der Entwicklung eines solchen Modells getroffen werden, ausreichend getestet werden. Dies fand bei Cederman (1997) nur für wenige Werte statt, sodass in der Nachbildung des EPM die Parametersensibilität ein deutliches Problem für die Ergebnisse darstellt.

Wie von Cederman (1997) selbst angesprochen (vgl. Abschnitt 3.2), ist auch bei diesem CAS die Parametersensibilität bzw. „Fragilität“ des Modells ein Problem. Seine Entgegnung, dies durch „artificial parameter randomization for each replication“ (S.65) mitigieren zu können, scheint für das EPM aber nicht, oder nur in sehr geringem Umfang stattgefunden zu haben, da sich die Probleme des Modells durch genau dieses Vorgehen offenlegen lassen.

Dies wirft ultimativ die Frage auf, ob es sich beim EPM nach der Definition aus Abschnitt 2.1 um ein CAS handelt. Diese stammt von Cederman (1997) selbst, und setzt für ein CAS Emergenz, Interaktion zwischen Agenten, eine große Menge von Agenten und deren Fähigkeit sich anzupassen, oder Änderungen des Systems durch Umwelteinflüsse voraus. Einerseits ist der Aspekt der Emergenz fraglich. Wie in Abschnitt 4.1.1 diskutiert, wird der Mechanismus des

Zerfalls von Staaten, den Cederman (1997) erforschen möchte, bereits im Modell angelegt und ist damit möglicherweise nicht aus dem Modell emergent. Wenn zudem für ein CAS notwendig ist, dass äußere Einflüsse das Verhalten des Systems beeinflussen und wenn davon ausgegangen wird, dass die Parameter, mit denen das System initialisiert wird, als solche Einflüsse zu werten sind, ist dies für einen weiten Teil des möglichen Parameterraums des EPM nicht der Fall. Dieses reagiert nur auf einen bestimmten Bereich möglicher Umwelteinflüsse mit Änderungen im Systemzustand.

Zur übergeordneten Fragestellung nach möglichen Fehlern bei der Übertragung politikwissenschaftlicher Theorien in Computersimulationen oder -modelle kann auf Grundlage dieser Erkenntnisse geschlossen werden, dass wie vermutet Modellierungsfehler auftreten können. Bei der Modellierung des EPM wurden Ad-Hoc-Entscheidungen getroffen, die dessen Aussagekraft bei genauerer Untersuchung schwächen. Diese wurden aber entweder nicht genauer untersucht, oder nicht als Artefakte erkannt. In beiden Fällen bedeutet dies, dass die Schlüsse, die aus dem Modell auf die Theorie gezogen werden weniger aussagekräftig sind als von Cederman (1997) angenommen. Wie in Abschnitt 2.2.3 diskutiert, ist diese Problematik der Modellierung bekannt, aber nicht umfassend erforscht oder dokumentiert. Das Ergebnis dieser Arbeit stimmt mit dem Forschungsstand darin überein, dass diese Fehler auftreten können und oft unbemerkt bleiben.

8.2. Kritik des Modells

Die hier gezogenen Schlüsse basieren auf einigen Annahmen und Vereinfachungen. Die wichtigste dieser Annahmen ist die, dass das entwickelte Modell die gleiche Funktionalität wie das EPM hat. Zudem wurden nur einige der Entscheidungen der Entwicklung des EPM genauer untersucht. Auch wurden die Definitionsbereiche der Parameter basierend auf Annahmen festgelegt. Dennoch wurden sämtlichen Entscheidungen ausführlich begründet und geprüft. Daher ist diese Kritik zwar angebracht, dennoch besteht Vertrauen in die gewonnenen Erkenntnisse.

Da nur ein konkretes Beispiel untersucht wurde, ist zudem die Verallgemeinerbarkeit der Ergebnisse begrenzt. Da die Problematik der Parametersensibilität in der untersuchten Fachliteratur aber nicht, oder nur unzureichend diskutiert wurde, ist davon auszugehen dass sich diese Problematik in weiteren politikwissenschaftlichen Modellen finden lässt.

Es muss angemerkt werden, dass die Ergebnisse des hier entwickelten Modells die Modellierungsentscheidungen und Vorüberlegungen widerspiegeln. Dennoch werden die Ergebnisse als aussagekräftig eingeschätzt, da durch die Reflexion der Annahmen bei der Modellierung, bei der deren Auswirkungen diskutiert wurden, diese bei der Auswertung berücksichtigen werden

konnten. Durch diese Reflexion und sorgfältige Vorüberlegungen zur Modellierung lässt sich zwar nicht verhindern, dass die Ergebnisse die Summe der Modellierungsentscheidungen sind, sie machen aber die Ergebnisse als Antwort auf die Fragestellung dennoch gültig.

8.3. Fazit

Es konnte gezeigt werden, dass die Kritik am EPM aus Lazer (2001) durchaus angebracht ist. Das EPM zeigt deutliche Parametersensibilitäten. Übertragen auf die Fragestellung nach der Modellierung politikwissenschaftlicher Theorien deutet dies auf Schwächen in der Vorgehensweise der Modellierung hin.

Auch wenn die Ergebnisse mit gewisser Vorsicht zu betrachten sind (siehe Abschnitt 8.2), wird diese Arbeit nach den in Abschnitt 1.1.4 gesetzten Zielen als Erfolg gewertet.

8.4. Ausblick

Für weitergehende Untersuchungen gibt es mehrere Ansatzpunkte. Im Bezug auf das EPM gibt es vielfältige Modellierungsentscheidungen die in Frage gestellt und untersucht werden können.

Allgemeiner ist die Fragestellung nach der Parametersensibilität in politikwissenschaftlichen Modellen nicht abschließend beantwortet. Die Untersuchung weiterer Modelle kann mehr Aufschluss geben.

Übergeordnet ist die Vorgehensweise bei der Modellierung politikwissenschaftlicher Fragestellungen in Simulationen aus theoretischen Modellen nicht umfassend beschrieben. Gerade die Problematik von Fehlern im Schritt von der Theorie zur Simulation, die teilweise interdisziplinäre Zusammenarbeit erfordert, stellt weiterhin eine methodische Bruchstelle dar und bietet damit Ansätze zur weiteren Forschung.

A. Anhang

A.1. Sourcecode

A.1.1. Main

```
1 import system
2 from random import randint
3
4
5 class Main():
6     config_number = 0
7     trust_threshold = 0
8     winning_resource_proportion = 0
9     attack_threshold = 0
10    threat_threshold = 0
11    battle_cost = 0
12    probability_of_predator_attack = 0
13    predator_rate = 0
14    mean_harvest = 0
15    deviation_harvest = 0
16    initial_resources = 0
17    deviation_initial_resources = 0
18    taxation_rate = 0
19    taxation_discount_by_distance = 0
20
21    percentage_low = 0
22    percentage_high = 100
23
24    harvest_low = 5
25    harvest_high = 200
26
27    threshold_low = 1
28    threshold_high = 25
```

```
29
30      proportion_low = 11
31      proportion_high = 50
32
33      def main(self):
34          self.run_experiment()
35
36      def run_experiment(self):
37          self.configuration()
38          for i in range(1, 11):
39              print('Running␣simulation␣' + str(self.config_number) + ',␣run
                    ␣' + str(i))
40              self.run_simulation()
41              i += 1
42          print('Simulation␣' + str(self.config_number) + '␣finished␣
                normally')
43
44      def configuration(self):
45          self.config_number = randint(1, 100000000)
46          self.trust_threshold = randint(self.threshold_low, self.
                threshold_high)
47          self.threat_threshold = randint(self.threshold_low, self.
                threshold_high)
48
49          self.winning_resource_proportion = (randint(self.proportion_low,
                self.proportion_high) / 10)
50          self.attack_threshold = randint(self.harvest_low, self.
                harvest_high)
51
52          self.initial_resources = randint(self.harvest_low, self.
                harvest_high)
53          self.mean_harvest = randint(self.harvest_low, self.harvest_high)
54
55          self.deviation_initial_resources = randint(0, self.
                initial_resources)
56          self.deviation_harvest = randint(0, self.mean_harvest)
57          self.battle_cost = randint(0, self.initial_resources)
58
59          self.predator_rate = randint(self.percentage_low, self.
                percentage_high)
```

```
60          self.probability_of_predator_attack = randint(self.percentage_low,
                self.percentage_high)
61          self.taxation_rate = (randint(self.percentage_low, self.
               percentage_high) / 100)
62          self.taxation_discount_by_distance = (randint(self.percentage_low,
                self.percentage_high) / 100)
63
64      def run_simulation(self):
65          sys = system.System()
66
67          # time
68          sys.time_units_total = 1000
69
70          # system
71          sys.width = 10
72          sys.height = 10
73
74          # relations
75          sys.alliances = True
76
77          # proportions
78          sys.trust_threshold = self.trust_threshold
79          sys.winning_resource_proportion = self.winning_resource_proportion
80          sys.threat_threshold = self.threat_threshold
81          sys.attack_threshold = self.attack_threshold
82
83          # warfare
84          sys.battle_cost = self.battle_cost
85          sys.probability_of_predator_attack = self.
               probability_of_predator_attack
86          sys.predator_rate = self.predator_rate
87
88          # resources
89          sys.mean_harvest = self.mean_harvest
90          sys.deviation_harvest = self.deviation_harvest
91          sys.initial_resources = self.initial_resources
92          sys.deviation_initial_resources = self.deviation_harvest
93          sys.taxation_rate = self.taxation_rate
94          sys.taxation_discount_by_distance = self.
               taxation_discount_by_distance
```

```
95
96          sys.config_number = self.config_number
97
98          # nicht variiert
99          sys.minimal_state_resources = 0
100         sys.sensitivity_to_threats = 7
101
102         sys.config_number = self.config_number
103
104         # output
105         sys.result_file = 'result.csv'
106
107         sys.start_simulation()
108
109
110 if __name__ == '__main__':
111     m = Main()
112     m.main()
```

A.1.2. System

```
1 import state
2 import printer
3 from random import randint
4
5
6 class System():
7     # states
8     states = []
9     hegemony = False
10    number_of_sovereign_states = 0
11
12    # time
13    time_units_passed = 0
14    time_units_total = 0
15    power_politics = 0
16
17    # system
18    width = 0
19    height = 0
20
```

```
21      # relations
22      alliances = False
23      neighbor_pairings = []
24
25      # proportions
26      trust_threshold = 0
27      winning_resource_proportion = 0
28      threat_threshold = 0
29      attack_threshold = 0
30
31      # warfare
32      sensitivity_to_threats = 0
33      battle_cost = 0
34      probability_of_predator_attack = 0
35      predator_rate = 0
36
37      # resources
38      mean_harvest = 0
39      deviation_harvest = 0
40      minimal_state_resources = 0
41      initial_resources = 0
42      deviation_initial_resources = 0
43      taxation_rate = 0
44      taxation_discount_by_distance = 0
45
46      # output
47      result_file = None
48      config_number = 0
49
50      ############## init, run simulation ##############
51
52      def __init__(self):
53          return None
54
55      # initialize system and run simulation
56      def start_simulation(self):
57          self.create_states()
58          for state in self.states:
59              state.all_states = self.states
60              self.initialize_state(state)
```

```
61          while (self.time_units_passed < self.time_units_total) & (not self
                .hegemony):
62              self.run_time_unit()
63              self.time_units_passed += 1
64              print('time␣unit␣' + str(self.time_units_passed) + '␣passed␣
                    successfully')
65          printer.print_to_csv_with_configs(self.config_number,self.
                trust_threshold, self.threat_threshold, self.
                winning_resource_proportion, self.attack_threshold, self.
                initial_resources, self.mean_harvest, self.
                deviation_initial_resources, self.deviation_harvest, self.
                battle_cost, self.predator_rate, self.
                probability_of_predator_attack, self.taxation_rate, self.
                taxation_discount_by_distance, self.number_of_sovereign_states,
                 self.time_units_passed, self.power_politics, self.result_file)
66
67      # tasks for each separate time unit
68      def run_time_unit(self):
69          self.set_perceptions()
70          self.make_decisions_about_attacks()
71          self.interact()
72          self.update_resources()
73          self.structural_change()
74          if self.alliances:
75              self.diplomacy()
76          self.is_hegemony()
77
78      ############## create states ##############
79
80      # initialize individual states
81      def create_states(self):
82          state_id = 1
83          for i in range(1, self.width + 1):
84              for j in range(1, self.height + 1):
85                  s = state.State(i, j, state_id)
86                  s.policy = self.choose_policy_for_state()
87                  s.resources = self.calculate_initial_resources()
88                  s.is_capital = True
89                  s.system_height = self.height
90                  s.system_width = self.width
```

```
91                  s.threat_threshold = self.threat_threshold
92                  s.trust_threshold = self.trust_threshold
93                  s.sensitivity = self.sensitivity_to_threats
94                  s.system_has_alliances = self.alliances
95                  s.original_policy = s.policy  # for conditional predator
                        system, conquering
96                  s.taxation_rate = self.taxation_rate
97                  s.taxation_discount_by_distance = self.
                        taxation_discount_by_distance
98                  s.probability_of_predator_attack = self.
                        probability_of_predator_attack
99                  s.attack_threshold = self.attack_threshold
100                 self.number_of_sovereign_states += 1
101                 self.states.append(s)
102                 state_id += 1
103
104     # return prey or predator based on probability
105     def choose_policy_for_state(self):
106         rand_int = randint(1, 101)
107         if rand_int > self.predator_rate:
108             return 'prey'
109         else:
110             return 'predator'
111
112     # initial resources +/- part of a standard deviation, based on chance
113     def calculate_initial_resources(self):
114         rand_deviation = randint((-1 * self.deviation_initial_resources),
                self.deviation_initial_resources)
115         return self.initial_resources + rand_deviation
116
117     # initial neighbors, resource distribution
118     def initialize_state(self, state):
119         state.find_neighbors()
120         state.allocate_resources()
121         state.next_action = 'c'
122         for neighbor in state.neighboring_states:
123             state.trust_level_for_neighbor[neighbor] = 0
124             state.history_with_neighbor[neighbor] = 'c'
125
126     ############## perceptions ##############
```

```
127
128     # make every state update their perception
129     def set_perceptions(self):
130         self.find_neighbors_of_states()
131         for state in self.states:
132             state.check_state_size()
133             state.update_trust(self.time_units_passed)
134             state.update_threat()
135
136     ############## decisions ##############
137
138     # make every state decide next move based on policy
139     def make_decisions_about_attacks(self):
140         for state in self.states:
141             if state.policy == 'prey':
142                 state.make_decision_prey()
143             else:
144                 state.make_decision_predator()
145
146     ############## interactions ##############
147
148     # all states interact with their neighbors
149     def interact(self):
150         self.get_all_neighbor_pairings()
151         for pair in self.neighbor_pairings:
152             self.make_interactions(pair)
153
154     # collect all pairs of neighboring states
155     def get_all_neighbor_pairings(self):
156         for state in self.states:
157             for neighbor in state.neighboring_states:
158                 if ([neighbor, state] not in self.neighbor_pairings)
159                         & ([state, neighbor] not in self.neighbor_pairings
                            ):
160                     self.neighbor_pairings.append([state, neighbor])
161
162     # update trust, resources, fight battle if one or both want to attack
163     def make_interactions(self, neighboring_pair):
164         state_one = neighboring_pair[0]
165         state_two = neighboring_pair[1]
```

```
166         if (state_one.next_action == 'c') & (state_two.next_action == 'c')
                :
167             state_one.trust_level_for_neighbor[state_two] = self.
                    trust_threshold
168             state_two.trust_level_for_neighbor[state_one] = self.
                    trust_threshold
169             state_one.history_with_neighbor[state_two] = 'c'
170             state_two.history_with_neighbor[state_one] = 'c'
171         elif ((state_one.next_action == 'd') & (state_one.
                involved_in_warfare_with == state_two)) | ((state_two.
                next_action == 'd') & (state_two.involved_in_warfare_with ==
                state_one)):
172             self.diplomatic_tension(state_one, state_two)
173
174     # in case of attack, update resources and trust
175     def diplomatic_tension(self, state_one, state_two):
176         if (state_one.next_action != state_two.next_action):
177             if state_one.next_action == 'd':
178                 self.attack_unprovoked(state_one, state_two)
179             else:
180                 self.attack_unprovoked(state_two, state_one)
181         else:
182             state_one.trust_level_for_neighbor[state_two] = self.
                    trust_threshold * -1
183             state_two.trust_level_for_neighbor[state_one] = self.
                    trust_threshold * -1
184         state_one.resources -= self.battle_cost
185         state_two.resources -= self.battle_cost
186
187     # change history, trust, alliance in case of attack
188     def attack_unprovoked(self, attacker, defender):
189         defender.history_with_neighbor[attacker] = 'd'
190         defender.trust_level_for_neighbor[attacker] = -1 * self.
                trust_threshold
191         if attacker in defender.allies:
192             defender.allies.remove(attacker)
193
194     ############## update resources ##############
195
196     # add harvest, taxes for each state, allocate resources at front
```

```
197     def update_resources(self):
198         for state in self.states:
199             state.resources += self.harvest()
200             if state.is_capital:
201                 state.collect_taxes()
202                 state.allocate_resources()
203 
204     def harvest(self):
205         rand_harvest = randint((-1 * self.deviation_harvest), self.
                deviation_harvest)
206         return self.mean_harvest + rand_harvest
207 
208     ############## structural changes ##############
209 
210     def structural_change(self):
211         for pair in self.neighbor_pairings:
212             state_one = pair[0]
213             state_two = pair[1]
214             if (state_one.involved_in_warfare_with == state_two) | (
                    state_two.involved_in_warfare_with == state_one):
215                 self.battle(state_one, state_two)
216         for state in self.states:
217             if state.is_capital:
218                 if state.resources / state.state_size < self.
                        minimal_state_resources:
219                     self.collapse(state)
220             elif self.landlocked(state):
221                 self.secede(state)
222         self.find_neighbors_of_states()
223 
224     def battle(self, state_one, state_two):
225         winner = None
226         loser = None
227         if (state_one.state_size > 0) & (state_two.state_size > 0):
228             average_resources_one = (state_one.resources / state_one.
                    state_size)
229             average_resources_two = (state_two.resources / state_two.
                    state_size)
230         else:
231             average_resources_two = state_two.resources
```

```
232             average_resources_one = state_one.resources
233         if (average_resources_one != 0) & (average_resources_two != 0):
234             if average_resources_one / average_resources_two >= self.
                    winning_resource_proportion:
235                 winner = state_one
236                 loser = state_two
237             elif average_resources_two / average_resources_one >= self.
                    winning_resource_proportion:
238                 winner = state_two
239                 loser = state_one
240             if winner != None:
241                 self.conquer(winner, loser)
242 
243     def collapse(self, state):
244         children = state.children
245         copy_of_children = children.copy()
246         ratio = state.get_ratio_of_all_resources()
247         for child in copy_of_children:
248             state.remove_child_during_collapse(child, ratio)
249             self.number_of_sovereign_states += 1
250             self.power_politics += 1
251         state.resources = ratio
252         self.find_neighbors_of_states()
253 
254     def landlocked(self, state):
255         landlocked = False
256         number_of_foreign_neighbors = 0
257         x_coord = state.coordinate_x
258         y_coord = state.coordinate_y
259         if not state.is_capital:
260             for i in range(x_coord - 1, x_coord + 2):
261                 for j in range(y_coord - 1, y_coord + 2):
262                     if (i > 0) & (i <= self.width) & (j > 0) & (j <= self.
                            height):
263                         other = self.find_state_by_coordinate(i, j)
264                         if (state not in other.children) & (other != state
                                ):
265                             number_of_foreign_neighbors += 1
266         if number_of_foreign_neighbors >= 8:
267             landlocked = True
```

```
268         return landlocked
269
270     def find_state_by_coordinate(self, x, y):
271         for state in self.states:
272             if (state.coordinate_x == x) & (state.coordinate_y == y):
273                 return state
274
275     # when a landlocked province leaves the state
276     def secede(self, state):
277         state.secede_after_landlock()
278         self.number_of_sovereign_states += 1
279         self.power_politics += 1
280
281     # when a province is conquered by a neighbor (only taking the province
            , not the whole state)
282     def conquer(self, winner, loser):
283         if loser.state_size > 1:
284             if not loser.is_capital:
285                 loser.province_get_captured(winner)
286             else:
287                 self.conquer_capital(winner, loser)
288         else:
289             loser.singular_get_incorporated(loser)
290             self.number_of_sovereign_states -= 1
291             self.power_politics += 1
292
293     def conquer_capital(self, winner, loser):
294         self.number_of_sovereign_states += len(loser.children)
295         loser.capital_get_captured(winner)
296         self.number_of_sovereign_states -= 1
297         self.power_politics += 1
298
299     ############## diplomacy ##############
300
301     def diplomacy(self):
302         alliances = []
303         threatened_states_by_threat = self.find_all_threats()
304         for key in threatened_states_by_threat:
305             value = threatened_states_by_threat[key]
306             if len(value) >= 2:
```

```
307                 alliances.append([value])
308                 for state in value:
309                     state.allies = value
310                     state.allies.remove(state)
311
312     def find_all_threats(self):
313         threatened_states_by_threat = {}
314         for state in self.states:
315             if state.feels_threatened:
316                 biggest_threat = state.biggest_threat
317                 if biggest_threat in threatened_states_by_threat:
318                     threatened_states_by_threat[biggest_threat].append(
                            state)
319                 else:
320                     threatened_states_by_threat[biggest_threat] = [state]
321         return threatened_states_by_threat
322
323     ############## helper methods ##############
324
325     # more than one state left in system?
326     def is_hegemony(self):
327         if self.number_of_sovereign_states <= 1:
328             self.hegemony = True
329
330     # find all neighbors for every state
331     def find_neighbors_of_states(self):
332         for state in self.states:
333             state.find_neighbors()
```

A.1.3. State

```
1 from random import randint
2
3
4 class State():
5     # state
6     coordinate_x = 0
7     coordinate_y = 0
8     state_id = 0
9     policy = None
10    state_size = 1  # number of children +1
```

```
11      original_policy = None
12      parent = None
13
14      # children
15      is_capital = True
16      children = set([])
17
18      # resources
19      resources = 0
20      taxation_rate = 0
21      taxation_discount_by_distance = 0
22
23      # neighbors
24      neighboring_states = []
25      trust_level_for_neighbor = {}
26      history_with_neighbor = {}
27      threat_threshold = 0
28      sensitivity = 0
29      trust_threshold = 0
30
31      # warfare
32      next_action = None
33      feels_threatened = False
34      biggest_threat = None
35      involved_in_warfare = False
36      involved_in_warfare_with = None
37      probability_of_predator_attack = 0
38      attack_threshold = 0
39
40      # allies
41      allies = []
42      system_has_alliances = False
43
44      # system
45      system_width = 0
46      system_height = 0
47      all_states = []
48
49      ############## init state ##############
50
```

```
51      def __init__(self, coordinate_x, coordinate_y, state_id):
52          self.coordinate_x = coordinate_x
53          self.coordinate_y = coordinate_y
54          self.state_id = state_id
55
56      ############## resources ##############
57
58      def allocate_resources(self):
59          ratio = self.get_ratio_of_all_resources()
60          for child in self.children:
61              child.resources = ratio
62          self.resources = ratio
63
64      def collect_taxes(self):
65          if self.children:
66              for child in self.children:
67                  # calculate biggest distance on x or y-axis
68                  x = abs(self.coordinate_x - child.coordinate_x)
69                  y = abs(self.coordinate_y - child.coordinate_y)
70                  distance = min(x, y)
71                  child_resources = child.resources
72                  # taxes = resources * tax percentage * (discount ^
                        distance)
73                  discount = pow(self.taxation_discount_by_distance,
                        distance)
74                  taxes = child_resources * self.taxation_rate * discount
75                  child.resources -= taxes
76                  self.resources += taxes
77
78      def get_all_resources(self):
79          total_resources = self.resources
80          if self.children:
81              for child in self.children:
82                  total_resources += child.resources
83          return total_resources
84
85      def get_ratio_of_all_resources(self):
86          total = self.get_all_resources()
87          if self.state_size > 0:
88              ratio = (total / self.state_size)
```

```
89          else:
90              ratio = self.resources
91          return ratio
92
93      ############## neighbors ##############
94
95      def find_neighbors(self):
96          self.neighboring_states = []
97          self.find_own_neighbors()
98
99      def find_own_neighbors(self):
100         for i in range(self.coordinate_x - 1, self.coordinate_x + 2):
101             for j in range(self.coordinate_y - 1, self.coordinate_y + 2):
102                 if (i > 0) & (i <= self.system_width) & (j > 0) & (j <=
                        self.system_height):
103                     state = self.find_state_by_coordinate(i, j)
104                     if (state not in self.children) & (self != state) & (
                            state not in self.neighboring_states) & (state !=
                            None):
105                         self.neighboring_states.append(state)
106
107     def find_state_by_coordinate(self, x, y):
108         for state in self.all_states:
109             if (state.coordinate_x == x) & (state.coordinate_y == y):
110                 return state
111
112     ############## perceptions ##############
113
114     def update_trust(self, time_passed):
115         for neighbor in self.neighboring_states:
116             trust = (
117                 (1 - self.sensitivity) * self.trust_level_for_neighbor.get
                        (neighbor) + (self.sensitivity * time_passed))
118             if neighbor in self.trust_level_for_neighbor:
119                 if trust > self.trust_threshold:
120                     self.trust_level_for_neighbor[neighbor] = self.
                            trust_threshold
121                 elif trust < (-1 * self.trust_threshold):
122                     self.trust_level_for_neighbor[neighbor] = (-1 * self.
                            trust_threshold)
```

```
123             else:
124                 self.trust_level_for_neighbor[neighbor] = self.sensitivity
                        * time_passed
125
126     def update_threat(self):
127         balance = 0
128         for neighbor in self.neighboring_states:
129             if self.resources > 0:
130                 new_balance = neighbor.resources / self.resources
131             else:
132                 new_balance = 0
133             if (neighbor.policy == 'predator') & (self.
                    trust_level_for_neighbor[neighbor] < 0):
134                 if new_balance >= balance:
135                     balance = new_balance
136                     if new_balance < self.threat_threshold:
137                         self.feels_threatened = True
138                         self.biggest_threat = neighbor
139
140     ############## decisions ##############
141
142     def make_decision_prey(self):
143         if self.involved_in_warfare:
144             self.next_action = 'd'
145         elif self.system_has_alliances:
146             for ally in self.allies:
147                 if ally.involved_in_warfare:
148                     self.in_warfare(ally.involved_in_warfare_with)
149                     break
150         if not self.involved_in_warfare:
151             self.not_in_warfare()
152             for neighbor in self.neighboring_states:
153                 if self.history_with_neighbor[neighbor] == 'd':
154                     self.in_warfare(neighbor)
155                     break
156
157     def make_decision_predator(self):
158         if self.involved_in_warfare:
159             self.next_action = 'd'
160         elif self.system_has_alliances:
```

```
161             for ally in self.allies:
162                 if ally.involved_in_warfare:
163                     self.in_warfare(ally.involved_in_warfare_with)
164                     break
165         if not self.involved_in_warfare:
166             self.not_in_warfare()
167             rand_int = randint(1, 101)
168             if rand_int <= self.probability_of_predator_attack:
169                 weakeast_neighbor = self.find_weakest_neighbor()
170                 if weakeast_neighbor.resources > 0:
171                     if (self.resources / weakeast_neighbor.resources) <
                            self.attack_threshold:
172                         self.in_warfare(weakeast_neighbor)
173                 else:
174                     self.in_warfare(weakeast_neighbor)
175 
176     def find_weakest_neighbor(self):
177         weakest = self.neighboring_states[0]
178         for neighbor in self.neighboring_states:
179             if neighbor.resources <= weakest.resources:
180                 weakest = neighbor
181         return weakest
182 
183     ############## structural changes ##############
184 
185     def remove_child_during_collapse(self, child, resources):
186         child.policy = child.original_policy
187         child.is_capital = True
188         child.state_size = 1
189         child.resources = resources
190         child.parent = None
191         self.children.remove(child)
192         self.state_size -= 1
193 
194     # province is lost through landlock
195     def secede_after_landlock(self):
196         self.resources = self.get_ratio_of_all_resources()
197         self.policy = self.original_policy
198         self.is_capital = True
199         self.state_size = 1
```

```
200         self.children = set([])
201         self.parent.capital_lose_child(self)
202         self.parent = None
203 
204     def capital_lose_child(self, state):
205         self.state_size -= 1
206         self.children.remove(state)
207 
208     def capital_gain_child(self, child):
209         self.state_size += 1
210         self.children.add(child)
211         child.policy = self.policy
212         child.parent = self
213         child.is_capital = False
214 
215     # wenn ein child erobert wird und aus meinem staat entfernt wird
216     def province_get_captured(self, conqueror):
217         self.parent.capital_lose_child(self)
218         conqueror.capital_gain_child(self)
219         conqueror.not_in_warfare()
220 
221     # capital is captured, children become sov
222     def capital_get_captured(self, conqueror):
223         ratio = self.get_ratio_of_all_resources()
224         for child in self.children.copy():
225             self.remove_child_during_collapse(child, ratio)
226         self.resources = ratio
227         self.not_in_warfare()
228         conqueror.not_in_warfare()
229         conqueror.capital_gain_child(self)
230 
231     # empire consists of only one state, gets conquered
232     def singular_get_incorporated(self, conqueror):
233         self.not_in_warfare()
234         conqueror.not_in_warfare()
235         conqueror.capital_gain_child(self)
236 
237     ############## warfare ##############
238 
239     def in_warfare(self, state):
```

```
240         self.next_action = 'd'
241         self.involved_in_warfare = True
242         self.involved_in_warfare_with = state
243 
244     def not_in_warfare(self):
245         self.next_action = 'c'
246         self.involved_in_warfare = False
247         self.involved_in_warfare_with = None
248 
249     ############## helper methods ##############
250 
251     def check_state_size(self):
252         self.state_size = (len(self.children) + 1)
253 
254     def get_state_size(self):
255         return self.state_size
256 
257     def __str__(self):
258         return 'state id: ' + str(self.state_id)
```

A.1.4. TestSystem

```
1 import unittest
2 import system
3 import state
4 
5 
6 class TestStringMethods(unittest.TestCase):
7     sys = system.System()
8     sys.initial_resources = 50
9     sys.deviation_initial_resources = 20
10     sys.mean_harvest = 20
11     sys.deviation_harvest = 10
12     sys.predator_rate = 30
13     sys.trust_threshold = 3
14     sys.height = 10
15     sys.width = 10
16 
17     s111 = state.State(1, 1, 1)
18     s123 = state.State(1, 2, 3)
19     s222 = state.State(2, 2, 2)
```

```
20      s444 = state.State(4, 4, 4)
21      all_states = [s111, s123, s222, s444]
22      sys.states = all_states
23      s111.all_states = all_states
24      s123.all_states = all_states
25      s222.all_states = all_states
26      s444.all_states = all_states
27
28      def test_choose_policy_for_state(self):
29          self.sys.predator_rate = 30
30          policy = self.sys.choose_policy_for_state()
31          self.assertTrue((policy == 'predator') | (policy == 'prey'))
32
33      def test_calculate_initial_resources(self):
34          res = self.sys.calculate_initial_resources()
35          self.assertTrue((res >= (-1 * self.sys.initial_resources)) | (res
                <= self.sys.initial_resources))
36
37      def test_get_all_neighbor_pairings(self):
38          self.s111.neighboring_states = [self.s123, self.s222]
39          self.s123.neighboring_states = [self.s111]
40          self.s222.neighboring_states = [self.s111]
41          self.sys.get_all_neighbor_pairings()
42          self.assertEqual(len(self.sys.neighbor_pairings), 2)
43
44      def test_make_interactions(self):
45          self.s111.next_action = 'c'
46          self.s222.next_action = 'c'
47          self.sys.make_interactions([self.s111, self.s222])
48          self.assertEqual(self.s111.trust_level_for_neighbor[self.s222],
                self.sys.trust_threshold)
49          self.assertEqual(self.s222.trust_level_for_neighbor[self.s111],
                self.sys.trust_threshold)
50
51          self.s111.next_action = 'd'
52          self.s222.next_action = 'c'
53          self.s111.trust_level_for_neighbor[self.s222] = 0
54          self.s222.trust_level_for_neighbor[self.s111] = 0
55          self.sys.make_interactions([self.s111, self.s222])
56          self.assertEqual(self.s111.trust_level_for_neighbor[self.s222], 0)
```

```
57          self.assertEqual(self.s222.trust_level_for_neighbor[self.s111], 0)
58
59          self.s111.next_action = 'd'
60          self.s222.next_action = 'd'
61          self.s111.trust_level_for_neighbor[self.s222] = 0
62          self.s222.trust_level_for_neighbor[self.s111] = 0
63          self.sys.make_interactions([self.s111, self.s222])
64          self.assertEqual(self.s111.trust_level_for_neighbor[self.s222], 0)
65          self.assertEqual(self.s222.trust_level_for_neighbor[self.s111], 0)
66
67      def test_attack_unprovoked(self):
68          self.s111.history_with_neighbor[self.s222] = 'c'
69          self.s111.trust_level_for_neighbor[self.s222] = 10
70          self.sys.attack_unprovoked(self.s222, self.s111)
71          self.assertEqual(self.s111.history_with_neighbor[self.s222], 'd')
72          self.assertNotEqual(self.s111.trust_level_for_neighbor[self.s222],
                10)
73
74      def test_harvest(self):
75          res = self.sys.harvest()
76          self.assertTrue(res >= (-1 * self.sys.mean_harvest) | res <= self.
               sys.mean_harvest)
77
78      def test_landlocked(self):
79          s55 = state.State(5, 5, 5)
80          s56 = state.State(5, 6, 6)
81          s57 = state.State(5, 7, 7)
82          s65 = state.State(6, 5, 8)
83          s66 = state.State(6, 6, 9)
84          s67 = state.State(6, 7, 10)
85          s75 = state.State(7, 5, 11)
86          s76 = state.State(7, 6, 12)
87          s77 = state.State(7, 7, 13)
88          s66.is_capital = False
89          self.sys.states.extend([s55, s56, s57, s65, s66, s67, s75, s76,
               s77])
90          self.assertTrue(self.sys.landlocked(s66))
91
92          s66.is_capital = True
93          self.assertFalse(self.sys.landlocked(s66))
```

```
94          self.assertFalse(self.sys.landlocked(s55))
95
96      def test_find_state_by_coordinates(self):
97          self.assertEqual(self.sys.find_state_by_coordinate(1, 1), self.
                s111)
98          self.assertNotEqual(self.sys.find_state_by_coordinate(1, 1), self.
                s222)
99
100     def test_find_all_threats(self):
101         self.s111.feels_threatened = True
102         self.s111.biggest_threat = self.s222
103         self.s222.feels_threatened = True
104         self.s222.biggest_threat = self.s123
105         self.assertTrue(len(self.sys.find_all_threats()), 2)
106
107         self.s123.feels_threatened = True
108         self.s123.biggest_threat = self.s222
109         self.assertTrue(len(self.sys.find_all_threats()), 2)
110
111     def test_is_hegemony(self):
112         self.sys.number_of_sovereign_states = 2
113         self.assertFalse(self.sys.is_hegemony())
114         self.sys.number_of_sovereign_states = 1
115         self.assertFalse(self.sys.is_hegemony())
116
117
118 if __name__ == '__main__':
119     unittest.main()
```

A.1.5. TestState

```
1 import unittest
2 import state
3
4
5 class TestStringMethods(unittest.TestCase):
6     state_list = []
7
8     s111 = state.State(1, 1, 1)
9     s122 = state.State(1, 2, 2)
10
```

```
11      state_list.append(s111)
12      state_list.append(s122)
13
14      s111.all_states = state_list
15      s122.all_states = state_list
16      s111.system_height = 10
17      s111.system_width = 10
18      s122.system_width = 10
19      s122.system_height = 10
20      s111.taxation_discount_by_distance = 0.7
21      s122.taxation_discount_by_distance = 0.7
22      s111.taxation_rate = 0.7
23      s122.taxation_rate = 0.7
24
25      def test_allocate_resources(self):
26          self.s111.resources = 1
27          self.s111.allocate_resources()
28          self.assertEqual(self.s111.resources, 1)
29
30          self.s111.children.add(self.s122)
31          self.s111.state_size = 2
32          self.s111.allocate_resources()
33          self.assertEqual(self.s111.resources, 0.5)
34          self.assertEqual(self.s122.resources, 0.5)
35
36      def test_collect_taxes(self):
37          self.s111.children.add(self.s122)
38          self.s111.resources = 0
39          self.s122.resources = 1
40          self.s111.collect_taxes()
41
42          self.assertEqual(self.s111.resources, 0.7)
43          self.assertTrue(self.s122.resources < 1)
44
45      def test_get_all_resources(self):
46          self.s111.children.add(self.s122)
47          self.s111.resources = 1
48          self.s122.resources = 1
49          self.assertEqual(self.s111.get_all_resources(), 2)
50
```

```
51      def test_get_ratio_of_all_resources(self):
52          self.s111.children.clear()
53          self.s111.children.add(self.s122)
54          self.s111.resources = 1
55          self.s122.resources = 1
56          self.s111.state_size = 2
57          self.assertEqual(self.s111.get_ratio_of_all_resources(), 1)
58
59      def test_find_own_neighbors(self):
60          self.s111.children.clear()
61          self.s111.find_own_neighbors()
62          self.assertEqual(self.s111.neighboring_states, [self.s122])
63
64          s222 = state.State(2, 2, 3)
65          self.s111.all_states.append(s222)
66          self.s111.find_own_neighbors()
67          self.assertEqual(len(self.s111.neighboring_states), 2)
68
69      def test_find_state_by_coordinate(self):
70          self.assertEqual(self.s111.find_state_by_coordinate(1, 2), self.
                s122)
71          self.assertNotEqual(self.s111.find_state_by_coordinate(1, 1), self
                .s122)
72
73      def test_make_decision_prey(self):
74          self.s111.in_warfare = True
75          self.s111.next_action = 'c'
76          self.s111.make_decision_prey()
77          self.assertEqual(self.s111.next_action, 'd')
78
79      def test_make_decision_predator(self):
80          self.s111.in_warfare = True
81          self.s111.next_action = 'c'
82          self.s111.make_decision_predator()
83          self.assertEqual(self.s111.next_action, 'd')
84
85      def test_find_weakest_neighbor(self):
86          s222 = state.State(2, 2, 3)
87          self.s111.neighboring_states = [self.s122, s222]
88          self.s122.resources = 3
```

```
89          s222.resources = 2
90          self.assertEqual(self.s111.find_weakest_neighbor(), s222)
91
92      def test_remove_child_during_collapse(self):
93          self.s111.children.add(self.s122)
94          self.s122.parent = self.s111
95          self.s122.original_policy = 'x'
96          self.s122.policy = 'y'
97          self.s122.is_capital = False
98          self.s111.remove_child_during_collapse(self.s122, 1)
99
100         self.assertEqual(self.s122.policy, 'x')
101         self.assertTrue(self.s122.is_capital)
102         self.assertEqual(self.s122.state_size, 1)
103         self.assertEqual(self.s122.resources, 1)
104         self.assertEqual(self.s122.parent, None)
105         self.assertEqual(self.s111.state_size, 1)
106
107     def test_secede_after_landlock(self):
108         self.s122.children.add(self.s111)
109         self.s111.original_policy = 'x'
110         self.s111.policy = 'y'
111         self.s111.is_capital = False
112         self.s111.state_size = 2
113         self.s111.parent = self.s122
114         self.s111.secede_after_landlock()
115
116         self.assertEqual(self.s111.policy, 'x')
117         self.assertTrue(self.s111.is_capital)
118         self.assertEqual(self.s111.state_size, 1)
119         self.assertEqual(self.s111.parent, None)
120         self.assertEqual(len(self.s111.children), 0)
121
122     def test_capital_lose_child(self):
123         self.s111.children.clear()
124         self.s111.children.add(self.s122)
125         self.s111.state_size = 2
126         self.s111.capital_lose_child(self.s122)
127
128         self.assertEqual(self.s111.state_size, 1)
```

```
129         self.assertEqual(len(self.s111.children), 0)
130
131     def test_capital_gain_child(self):
132         self.s111.state_size = 1
133         self.s111.children.clear()
134         self.s111.policy = 'x'
135         self.s122.policy = 'y'
136         self.s122.parent = None
137         self.s122.is_capital = True
138         self.s111.capital_gain_child(self.s122)
139
140         self.assertEqual(self.s111.state_size, 2)
141         self.assertEqual(len(self.s111.children), 1)
142         self.assertEqual(self.s122.policy, 'x')
143         self.assertEqual(self.s122.parent, self.s111)
144         self.assertFalse(self.s122.is_capital)
145
146     def test_capital_get_captured(self):
147         self.s111.children.clear()
148         self.s111.children.add(self.s122)
149         self.s111.involved_in_warfare = True
150         self.s111.is_capital = True
151         self.s111.capital_get_captured(self.s122)
152         self.assertFalse(self.s111.involved_in_warfare)
153         self.assertFalse(self.s111.is_capital)
154         self.assertFalse(self.s122.involved_in_warfare)
155
156     def test_in_warfare(self):
157         self.s111.next_action = 'c'
158         self.s111.involved_in_warfare = False
159         self.s111.in_warfare(self.s122)
160         self.assertEqual(self.s111.next_action, 'd')
161         self.assertTrue(self.s111.involved_in_warfare)
162         self.assertEqual(self.s111.involved_in_warfare_with, self.s122)
163
164     def test_not_in_warfare(self):
165         self.s111.next_action = 'd'
166         self.s111.involved_in_warfare = True
167         self.s111.involved_in_warfare_with = self.s122
168         self.s111.not_in_warfare()
```

```
169         self.assertEqual(self.s111.next_action, 'c')
170         self.assertFalse(self.s111.involved_in_warfare)
171         self.assertEqual(self.s111.involved_in_warfare_with, None)
172
173     def test_check_state_size(self):
174         self.s111.children.clear()
175         self.s111.children.add(self.s122)
176         self.s111.check_state_size()
177         self.assertEqual(self.s111.get_state_size(), 2)
178
179
180 if __name__ == '__main__':
181     unittest.main()
```

A.1.6. Printer

```
1 import csv
2
3
4 def print_to_csv_with_configs(config_number, trust_threshold,
      threat_threshold, winning_resource_proportion, attack_threshold,
      initial_resources, mean_harvest, deviation_initial_resources,
      deviation_harvest, battle_cost, predator_rate,
      probability_of_predator_attack, taxation_rate,
      taxation_discount_by_distance, number_of_states, time_units_passed,
      power_politics, result_file):
5     with open(result_file, 'a', newline='') as csvfile:
6         writer = csv.writer(csvfile, delimiter=';')
7         data = [config_number, trust_threshold, threat_threshold,
              winning_resource_proportion, attack_threshold,
              initial_resources, mean_harvest, deviation_initial_resources,
              deviation_harvest, battle_cost, predator_rate,
              probability_of_predator_attack, taxation_rate,
              taxation_discount_by_distance, number_of_states,
              time_units_passed, power_politics]
8         writer.writerow(data)
```

A.1.7. Plot

```
1 import matplotlib.pyplot as plt
2 import csv
```

```
3 import statistics
4 import seaborn as sns
5
6
7 class Plot():
8     # number of states at end of simulation
9     x_values = []
10    x_err = []
11
12    # number of power struggles
13    y_values = []
14    y_err = []
15
16    labels = []
17
18
19    def main(self):
20        sorted_data = self.extract_data()
21        self.calculate(sorted_data)
22        self.print()
23        self.density_plot()
24        # self.plot()
25
26    def plot(self):
27        plt.figure()
28        plt.scatter(self.x_values, self.y_values, s=1.2)
29        i = 0
30        for txt in self.labels:
31            plt.annotate(txt, (self.x_values[i], self.y_values[i]),
                  fontsize=5)
32            i += 1
33        plt.errorbar(self.x_values, self.y_values, xerr=self.x_err, yerr=
              self.y_err, fmt='.', markersize=7,
34                     elinewidth=0.5)
35        plt.title("number␣of␣states␣correlated␣with␣number␣of␣power␣
              struggles")
36        plt.ylabel('number␣of␣states␣at␣end␣of␣simulation')
37        plt.xlabel('number␣of␣power␣struggles␣during␣simulation,␣n␣=␣' +
              str(len(self.x_values)))
38        plt.show()
```

```
39
40      def density_plot(self):
41          sns.set_style("white")
42          sns.kdeplot(self.x_values, self.y_values, cmap="Blues", shade=True
                , bw=.15)
43          plt.title("density of result points")
44          plt.ylabel('number of states at end of simulation')
45          plt.xlabel('number of power struggles during simulation, n = ' +
                str(len(self.x_values)))
46          plt.show()
47
48      def extract_data(self):
49          data = []
50          data_split = {}
51          with open('results_clean.csv', newline='') as csvfile:
52              reader = csv.reader(csvfile, delimiter=';')
53              for row in reader:
54                  array = []
55                  array.append(row[0])
56                  array.append(row[14])
57                  array.append(row[15])
58                  array.append(row[16])
59                  data.append(array)
60          # remove column labels
61          del data[0]
62
63          for entry in data:
64              id = entry[0]
65              if id in data_split:
66                  value = data_split[id]
67                  value[0].append(entry[1])
68                  value[1].append(entry[2])
69                  value[2].append(entry[3])
70              else:
71                  data_split[id] = [[entry[1]], [entry[2]], [entry[3]]]
72          return data_split
73
74      def calculate(self, data):
75          for key, value in data.items():
76              x = value[2]
```

```
77              x_int = list(map(int, x))
78              std_x = statistics.pstdev(x_int)
79              mean_x = statistics.mean(x_int)
80
81              y = value[0]
82              y_int = list(map(int, y))
83              std_y = statistics.pstdev(y_int)
84              mean_y = statistics.mean(y_int)
85
86              self.labels.append(key)
87              self.x_values.append(mean_x)
88              self.x_err.append(std_x)
89              self.y_values.append(mean_y)
90              self.y_err.append(std_y)
91
92      def print(self):
93          with open('means_stdev.csv', 'a', newline='') as csvfile:
94              writer = csv.writer(csvfile, delimiter=';')
95              for i in range(0, len(self.x_values)):
96                  data = [self.labels[i], self.x_values[i], self.x_err[i],
                        self.y_values[i], self.y_err[i]]
97                  writer.writerow(data)
98
99
100 if __name__ == '__main__':
101     p = Plot()
102     p.main()
```

A.2. Versuchsergebnisse

A.2.1. Konfigurationen

Config ID	Trust Threshold	Threat Threshold	Winning Resource Proportions	Attack Threshold	Initial Resources	Mean Harvest	Deviation Initial Resources	Deviation Harvest	Battle Cost
3033645	15	4	1,9	85	112	173	66	66	77
13114340	7	23	3,7	179	38	161	57	57	13
17368091	5	14	1,9	139	94	160	55	55	68
25068343	10	20	2,6	194	30	136	75	75	29
25160289	14	12	2,4	19	31	118	49	49	6
52099747	11	11	3,4	189	138	112	32	32	105
58375463	2	12	2,8	167	167	149	93	93	140
96532932	25	3	4,8	66	9	29	25	25	8
2235186	16	11	2,4	117	50	180	150	150	44
2703527	7	14	4,2	5	21	47	46	46	16
6847933	21	7	1,5	113	179	24	13	13	20
7501913	17	15	3,6	180	143	67	6	6	94
11127317	1	22	2,9	162	18	77	77	77	2
13204756	8	20	1,1	81	70	195	78	78	23
15529765	12	17	1,3	30	158	25	6	6	109
18182668	12	14	2,6	20	56	15	12	12	34
21385412	18	25	3,8	73	14	74	36	36	1
21446496	5	2	3,4	89	49	95	38	38	23
21905818	19	24	4,8	102	191	154	123	123	135
22152530	16	23	2,3	121	41	197	21	21	26
26994715	23	22	4,2	134	132	171	127	127	57
29983117	25	6	4,1	160	47	88	24	24	38
34389050	17	14	3,3	57	188	161	45	45	148
35168284	18	23	4,8	142	112	143	102	102	109
35326272	21	12	4,4	40	99	17	1	1	59
35988309	22	22	3,6	176	112	58	14	14	106
42781241	10	9	1,5	171	42	43	41	41	20
45589820	4	1	1,9	22	63	149	36	36	14
45887614	7	11	2,2	114	57	71	54	54	2
48455345	23	2	2,1	46	113	173	97	97	29
49301962	4	25	3,9	92	68	133	127	127	30
51454329	22	2	4,4	159	195	51	22	22	36
52973079	5	16	4,6	177	155	32	1	1	114
57493782	23	18	3,8	132	35	35	31	31	14
59546349	11	22	3,8	88	100	152	115	115	15
61166650	21	3	3,1	28	169	165	126	126	45
64295372	10	3	4,6	33	160	181	55	55	68
64605810	5	7	1,8	33	169	19	0	0	167
65578893	9	3	3,5	102	133	11	1	1	130
68097703	19	4	4,7	49	128	134	132	132	107
68313227	21	6	3,3	158	115	112	27	27	85
68498230	9	3	5	153	159	193	137	137	18
69316974	20	6	1,7	50	40	161	9	9	32
71080252	11	21	4	163	19	159	124	124	9

Config ID	Trust Threshold	Threat Threshold	Winning Resource Proportions	Attack Threshold	Initial Resources	Mean Harvest	Deviation Initial Resources	Deviation Harvest	Battle Cost
71328624	23	12	1,4	103	72	58	45	45	11
72832996	23	19	1,1	70	173	34	13	13	23
78506875	7	11	2,4	167	95	25	12	12	0
80258701	18	16	4,3	50	54	170	98	98	46
80307340	21	17	1,7	120	140	120	80	80	67
80744709	11	11	2	20	137	147	67	67	102
88116460	24	7	5	40	42	148	26	26	17
88954901	7	3	4,1	141	195	58	20	20	166
89653457	18	24	1,4	137	89	52	25	25	38
96388395	12	22	2,8	58	183	59	29	29	130
98416016	16	16	1,9	89	60	13	10	10	4
98633644	2	16	4,3	103	27	176	71	71	5
99788682	21	21	4,4	170	41	102	64	64	38
99822714	3	25	3,9	128	91	6	2	2	38
93950730	16	19	1,1	207	53	7	7	3	9
42452505	19	35	1,1	172	50	7	12	3	6
6336	3	17	1,4	73	83	15	7	7	28
17927	2	2	3,7	43	189	18	5	5	151
4332162	16	8	3,7	118	82	188	86	86	13
4385590	9	21	3,9	162	120	141	138	138	46
4971300	12	5	4	61	197	122	0	0	116
5414059	25	1	1,7	8	141	77	24	24	2
9880226	18	10	4,1	20	21	13	5	5	7
10074575	17	14	1,9	117	18	75	54	54	16
11738705	2	5	2,9	88	38	112	50	50	37
12708906	23	19	4,6	80	121	119	48	48	11
16626008	8	5	3	142	147	155	152	152	27
16988164	20	9	2	114	96	143	28	28	45
18744740	16	14	3,3	129	183	84	79	79	81
21902052	23	3	3,4	146	70	96	25	25	16
23945656	9	6	3,7	109	130	133	56	56	88
24056253	13	5	2,5	60	100	113	15	15	55
24101181	19	4	4,8	130	200	5	4	4	172
25797893	12	14	2,1	162	35	151	88	88	35
29206814	5	20	3,2	107	5	65	24	24	2
30097265	5	1	3,3	154	39	16	15	15	30
32449581	24	11	1,1	50	54	161	53	53	50
33805831	12	6	2,6	13	99	99	43	43	17
35919653	25	18	4,3	68	90	189	87	87	65
37881981	11	11	3	111	67	150	92	92	11
42391997	23	11	3,4	75	178	189	56	56	118
42669851	23	10	1,3	49	22	108	108	108	8
43089389	14	23	2,2	190	116	44	3	3	110
43905044	23	14	2,1	180	161	90	36	36	11

Config ID	Trust Threshold	Threat Threshold	Winning Resource Proportions	Attack Threshold	Initial Resources	Mean Harvest	Deviation Initial Resources	Deviation Harvest	Battle Cost
48517823	7	6	1,2	85	129	99	63	63	29
49329311	8	20	4,3	143	134	135	34	34	81
57560597	15	12	1,6	167	27	65	20	20	19
58443989	17	9	2,9	30	61	116	78	78	26
64459891	17	7	3,4	135	168	90	51	51	43
65891098	24	18	1,3	58	188	200	197	197	62
67610875	13	16	1,5	63	159	123	8	8	83
68298104	4	10	2,7	24	37	93	18	18	29
74505617	24	9	3,8	103	96	69	6	6	53
74828086	4	16	2,3	30	20	63	42	42	8
76180605	12	21	2,9	108	15	165	33	33	10
78819455	23	10	4	110	43	200	116	116	38
84774681	4	9	4,4	23	171	21	14	14	110
85822476	21	19	4	105	107	82	18	18	50
88400455	13	22	4,4	158	62	57	2	2	56
88603508	20	16	3,2	113	62	150	76	76	29
94256855	8	5	2,1	58	19	65	20	20	3
94746535	8	9	1,6	119	16	180	66	66	8
95808906	23	8	2,2	68	69	38	27	27	37
96223697	8	2	3,7	192	21	119	1	1	4
97327008	20	4	3,8	96	8	142	51	51	4
98949596	2	17	1,4	111	110	197	145	145	93

Config ID	Predator Rate	Probability of Attack	Taxation Rate	Taxation Discount
3033645	84	18	0,42	0,64
13114340	92	27	0,85	0,19
17368091	65	6	0,2	0,7
25068343	28	28	0,98	0,49
25160289	14	87	0,46	0,93
52099747	49	5	0,79	0,23
58375463	67	52	0,47	0,9
96532932	8	22	0,92	0,22
2235186	78	56	0,6	0,86
2703527	32	13	0,96	0,64
6847933	62	95	0,98	0,05
7501913	49	8	0,46	0,8
11127317	98	22	0,99	0,66
13204756	50	63	0,23	0,37
15529765	68	78	0,41	0,86
18182668	43	71	0,48	0,47
21385412	92	83	0,58	0,37
21446496	69	35	0,21	0,85
21905818	36	40	0,9	0,31
22152530	11	87	0,73	0,83
26994715	30	78	0,96	0,25
29983117	56	88	0,14	0,75
34389050	16	24	0,36	0,84
35168284	87	4	0,7	0,01
35326272	98	50	0,08	0,19
35988309	99	16	0,62	0,19
42781241	43	68	0,48	0,51
45589820	4	28	0,71	0,73
45887614	4	36	0,11	0,36
48455345	92	75	0,05	0,27
49301962	38	24	0,11	1
51454329	68	95	0,75	0,42
52973079	23	77	0,72	0,78
57493782	82	81	0,72	0,13
59546349	89	99	0,26	0,96
61166650	36	79	0,98	0,86
64295372	76	39	0,24	0,17
64605810	30	57	0,81	0,6
65578893	8	47	0,84	0,72
68097703	71	9	0,09	0,23
68313227	18	54	0,98	0,57
68498230	74	24	0,32	0,95
69316974	86	5	0,5	0,08
71080252	40	78	0,44	0,78

Config ID	Predator Rate	Probability of Attack	Taxation Rate	Taxation Discount
71328624	15	56	0,57	0,54
72832996	58	71	0,18	0,35
78506875	11	73	0,22	0,84
80258701	95	23	0,37	0,37
80307340	3	75	0,6	0,66
80744709	96	73	0,51	0,87
88116460	65	16	0,93	0,1
88954901	21	92	0,85	0,81
89653457	48	27	0,55	0,11
96388395	67	70	0,49	0,46
98416016	4	50	0,7	0,56
98633644	62	31	0,67	0,33
99788682	23	7	0,21	0,98
99822714	61	90	0,61	0,31
93950730	5	70	0,63	0,72
42452505	7	68	0,71	0,69
6336	73	15	0,31	0,13
17927	41	26	0,91	0,73
4332162	98	57	0,17	0,69
4385590	33	60	0,75	0,67
4971300	43	64	0,17	0,44
5414059	93	10	0,14	0,4
9880226	31	27	0,31	0,58
10074575	100	66	0,54	0,51
11738705	91	81	0,15	0,37
12708906	15	63	0,08	0,23
16626008	29	0	0,22	0,47
16988164	0	88	0,55	0,71
18744740	80	50	0,67	0,16
21902052	4	82	0	0,88
23945656	93	46	0,1	0,98
24056253	87	0	0,16	0,8
24101181	85	98	0,63	0,15
25797893	83	89	0,89	0,41
29206814	60	63	0,06	0,85
30097265	52	33	0,8	0,27
32449581	77	29	0,7	0,01
33805831	38	34	0,23	0,42
35919653	51	8	0,69	0,27
37881981	62	24	0,19	0,39
42391997	12	9	0,87	0,26
42669851	63	35	0,89	0,68
43089389	51	74	0,96	0,83
43905044	69	94	0,28	0,79

Config ID	Predator Rate	Probability of Attack	Taxation Rate	Taxation Discount
48517823	68	2	0,96	0,7
49329311	57	61	0,14	0,77
57560597	6	61	0,69	0,15
58443989	24	55	0,85	0,53
64459891	30	9	0,95	0,33
65891098	43	78	0,08	0,81
67610875	26	89	0,42	0,1
68298104	32	16	0,6	0,83
74505617	57	26	0,61	0,46
74828086	62	87	0,82	0,19
76180605	18	57	0,98	0,94
78819455	50	89	0,15	0,44
84774681	71	45	0,72	0,97
85822476	73	75	0,17	0,36
88400455	16	19	0,4	0,04
88603508	64	29	0,29	0,54
94256855	86	34	0,84	0,76
94746535	74	71	0,85	0,97
95808906	82	47	0,55	0,81
96223697	56	60	0,82	0,88
97327008	93	89	0,56	0,4
98949596	52	97	0,68	0,25

A.2.2. Mittelwerte und Abweichungen der Konfigurationen

Config ID	Mean Number of Power Struggles	Standard Deviation Number of Power Struggles	Mean Number Of States	Standard Deviation Number of States
6336	1536	1994.453509109701	69.2	44.7097304845377
17927	1729.8	840.0380705658524	20.8	39.6
2235186	21.3	63.89999999999999	95.5	13.5
2703527	29.3	12.915494570476191	98.7	2.193171219946131
3033645	0.8947368421052632	3.7960469305804128	99.94736842105263	0.22329687826943606
4332162	0.1	0.30000000000000004	99.9	0.30000000000000004
4385590	65.1	81.44501212474586	97.4	7.472616676907762
4971300	7	9.219544457292887	100	0.0
5414059	121.5	364.5	98.2	5.4
6847933	114.1	181.05438409494533	97.9	4.657252408878007
7501913	586	467.21986259147843	99	1.7888543819998317
9880226	21.5	22.953213282675698	98.5	2.692582403567252
10074575	33.4	100.2	99	3.0
11127317	162.9	480.42448938412787	99.7	0.9
11738705	54	78.36836096282734	97.8	5.96322060635023
12708906	147.6	442.8	92.9	21.3
13114340	59.578947368421055	252.77206620100162	99.57894736842105	1.7863750261554885
13204756	28.8	86.4	99.5	1.5
15529765	1263.4	2604.447242698535	84.1	31.807074684730125
16626008	0	0.0	100	0.0
16988164	0	0.0	100	0.0
17368091	0.6842105263157895	2.9028594175026687	99.94736842105263	0.22329687826943606
18182668	2092.9	2417.977272432477	1	0.0
18744740	447.1	250.15453223957385	99.4	1.2806248474865698
21385412	378	1132.334314590881	93.7	17.29768770674277
21446496	335	1004.333510344049	95.9	11.648605066702192
21902052	0.1	0.30000000000000004	99.9	0.30000000000000004
21905818	19.3	12.868954891520913	99.5	1.2041594578792296
22152530	0.4	1.2000000000000002	99.6	1.2
23945656	14.3	25.31027459353217	99.7	0.458257569495584
24056253	0	0.0	100	0.0
24101181	761.1	660.4374989353648	78.4	31.417192745374308
25068343	81.26315789473684	343.593677122337	99.73684210526316	0.90856192113853
25160289	0.05263157894736842	0.22329687826943606	99.94736842105263	0.22329687826943606
25797893	563.2	1653.5860304199477	95.3	10.139526616169022
26994715	26	48.52422075623677	96.6	8.912911982062877
29206814	364	1091.3334962329343	96	11.349008767288886
29983117	20.9	42.07719097088112	98.9	3.3000000000000003
30097265	1443.4	620.7345970702777	26.3	40.17474330969645
32449581	77.7	233.1	99.7	0.9
33805831	358.1	1073.633498918509	92.8	20.94182418033348
34389050	18.8	13.622040963086258	97.4	6.499230723708768
35168284	16.3	22.65413869472861	99.9	0.30000000000000004
35326272	1024.1	572.8936114148944	70.7	41.07322728980522
35919653	0.1	0.30000000000000004	99.9	0.30000000000000004
35988309	1233.888888888889	836.6874425633564	89.22222222222223	26.632385372273433
37881981	0.2	0.6000000000000001	99.8	0.6
42391997	1	3.0	99.9	0.30000000000000004
42452505	1536.7777777777778	1432.768243473512	45	49.193495504995376
42669851	20.3	60.9	99.2	2.4
42781241	29.1	87.3	98.3	5.1000000000000005
43089389	2023.6	989.8628389832604	1	0.0
43905044	244.6	731.4696439360966	98.5	3.0083217912982647
45589820	0.2	0.6000000000000001	99.8	0.6
45887614	610.9	1221.2389979033587	87.2	29.99266577015121

Config ID	Mean Number of Power Struggles	Standard Deviation Number of Power Struggles	Mean Number Of States	Standard Deviation Number of States
48455345	373.1	1101.0887748042844	96.3	7.40337760755184
48517823	6.3	12.930970574554719	99.9	0.30000000000000004
49301962	310.3	918.9626815056201	93.8	15.289211882893113
49329311	18.1	27.18253115513712	99.4	1.4966629547095764
51454329	21.2	38.55593339552293	99.4	1.8
52099747	11.4	16.43492216795281	99.8	0.5416025603090641
52973079	472.9	517.9348318080181	94.5	9.436630754670864
57493782	19.7	41.68464945276618	99.9	0.30000000000000004
57560597	3.8	11.4	99.8	0.6
58375463	204.5	170.89368039807675	98.7	2.0024984394500787
58443989	1.7	5.1000000000000005	99.9	0.30000000000000004
59546349	314.6	932.8518853494375	95	11.7983049630021
61166650	1.9	5.7	99.9	0.30000000000000004
64295372	0.3	0.9	99.8	0.6
64459891	0.1	0.30000000000000004	99.9	0.30000000000000004
64605810	1390.8	322.8540846884239	1	0.0
65578893	80.5	119.6973266201046	99.2	2.4
65891098	34.6	103.8	98.3	5.1000000000000005
67610875	42.4	127.2	98.3	5.1000000000000005
68097703	24.1	20.709659581943882	99.5	0.806225774829855
68298104	0.3	0.9	99.7	0.9
68313227	17.5	15.350895739337167	99.5	0.6708203932499369
68498230	0.1	0.30000000000000004	99.9	0.30000000000000004
69316974	0.2	0.6000000000000001	99.8	0.6
71080252	192.9	574.0485084032533	98.6	4.2
71328624	3.7	11.1	99.9	0.30000000000000004
72832996	1994.1	2226.2925436698565	93	10.392304845413264
74505617	13.7	21.133149315707772	99.6	0.916515138991168
74828086	603.2	1790.6806415438796	89.2	23.553343711668624
76180605	542.3	1318.0088049781762	80.7	38.616188315264885
78506875	1205.9	2267.41418580726	78.5	39.07492802296634
78819455	3.6	10.8	99.4	1.8
80258701	241.6	721.4728269311327	94.6	13.741906709041508
80307340	336.5	476.81322339045926	69	44.55782759515998
80744709	18.1	36.139867182932484	99.9	0.30000000000000004
84774681	1323.1	814.8164762693499	58.5	47.04519104010526
85822476	32.2	68.3663659996639	98.9	3.3000000000000003
88116460	88.7	266.09999999999997	97.6	7.2
88400455	16	6.6932802122726045	99	1.0
88603508	0.9	2.7	99.9	0.30000000000000004
88954901	751.2	788.614075451358	60.3	48.41910779847146
89653457	30	83.45657553482529	98.8	3.6
93950730	1703.3	393.4122646791785	1	0.0
94256855	0.1	0.30000000000000004	99.9	0.30000000000000004
94746535	4.3	12.9	99.6	1.2
95808906	1131.8	814.1816504933037	40.1	47.898747374018036
96223697	25.5	76.5	99.9	0.30000000000000004
96388395	1513.9	643.1275845429117	20.8	39.6
96532932	57.416666666666664	35.768837305987766	97.91666666666667	3.839668677836091
97327008	0.1	0.30000000000000004	99.9	0.30000000000000004
98416016	352.3	1055.2343862858147	95.7	11.331813623599711
98633644	205.8	617.4	96.8	9.6
98949596	85.5	256.5	97	9.0
99788682	0.1	0.30000000000000004	99.9	0.30000000000000004
99822714	1358.1	718.3715542809306	33.9	41.17147070484609

A.2.3. Erfolgreiche Konfigurationen

Config ID	Trust Threshold	Threat Threshold	Winning Resource Proportions	Attack Threshold	Initial Resources	Mean Harvest	Deviation Initial Resources	Deviation Harvest	Battle Cost	Predator Rate	Probability of Attack	Taxation Rate	Taxation Discount
6336	3	17	1,4	73	83	15	7	7	28	73	15	0,31	0,13
17927	2	2	3,7	43	189	18	5	5	151	41	26	0,91	0,73
15529765	12	17	1,3	30	158	25	6	6	109	68	78	0,41	0,86
18182668	12	14	2,6	20	56	15	12	12	34	43	71	0,48	0,47
24101181	19	4	4,8	130	200	5	4	4	172	85	98	0,63	0,15
30097265	5	1	3,3	154	39	16	15	15	30	52	33	0,8	0,27
35326272	21	12	4,4	40	99	17	1	1	59	98	50	0,08	0,19
35988309	22	22	3,6	176	112	58	14	14	106	99	16	0,62	0,19
42452505	19	35	1,1	172	50	7	12	3	6	7	68	0,71	0,69
43089389	14	23	2,2	190	116	44	3	3	110	51	74	0,96	0,83
45887614	7	11	2,2	114	57	71	54	54	2	4	36	0,11	0,36
64605810	5	7	1,8	33	169	19	0	0	167	30	57	0,81	0,6
72832996	23	19	1,1	70	173	34	13	13	23	58	71	0,18	0,35
74828086	4	16	2,3	30	20	63	42	42	8	62	87	0,82	0,19
76180605	12	21	2,9	108	15	165	33	33	10	18	57	0,98	0,94
78506875	7	11	2,4	167	95	25	12	12	0	11	73	0,22	0,84
80307340	21	17	1,7	120	140	120	80	80	67	3	75	0,6	0,66
84774681	4	9	4,4	23	171	21	14	14	110	71	45	0,72	0,97
88954901	7	3	4,1	141	195	58	20	20	166	21	92	0,85	0,81
93950730	16	19	1,1	207	53	7	7	3	9	5	70	0,63	0,72
95808906	23	8	2,2	68	69	38	27	27	37	82	47	0,55	0,81
96388395	12	22	2,8	58	183	59	29	29	130	67	70	0,49	0,46
99822714	3	25	3,9	128	91	6	2	2	38	61	90	0,61	0,31

A.3. Code des EPM

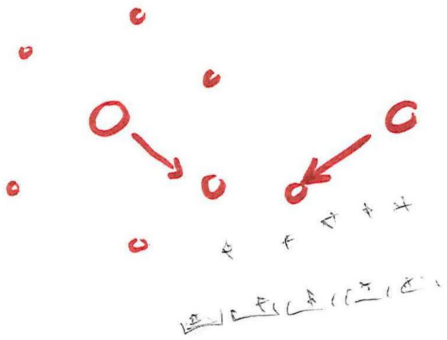
Zur Verfügung gestellter Code (Cederman, 1993)

```
unit action_mod;
{July 12, 1993: dividing the decide function for each culture type}
{see action_mod_old.pas file}

interface
 uses
  implement_mod, base_mod, system_mod, relations_mod;

 const
  d_trust = 1000;
  offset_trust = 1000;

 var
  harvest_m, harvest_s, min_res: res_t;
  bell_pred: integer;
  superior_ratio, victory_ratio, grad, mobil: integer;
  peace_div, victory_cost, defeat_cost, war_cost: res_t;
  min_threat: res_t;
  pr_oblig: integer;
  forgive, provoke: integer;
  soc_neg, soc_pos: trust_t;

 procedure perceptions;

 procedure decisions;

 procedure interactions;

 procedure update_resources;

 procedure update_all_resources;

 procedure collapse (j: id_t);

 procedure conquer (i, j: id_t);

 procedure structural_change;

 procedure diplomacy;



(*  ------------------------------------------------------------  *)

implementation

 type
  random_buffer_t = array[id_t] of id0_t;

 function random_act (p: real): act_t;
 begin
  if bern(p) then
   random_act := d
  else
   random_act := c;
 end;
```

```
procedure update_trust_OLD (i: id_t);
 var
  neigh: link_t;
begin
 neigh := system[i].neighs;
 repeat
  neigh := neigh^.next;
  if neigh^.pol then
   if neigh^.res = 0 then
    neigh^.trust := 0
   else
    neigh^.trust := -(neigh^.link^.res div neigh^.res);
 until last(neigh);
end;

procedure update_threat (i: id_t);
 var
  neigh: link_t;
  trust, min_trust: longint;
  threat: id0_t;
  pol_neighs: integer;
begin
 threat := 0;
 min_trust := 999999999;
 system[i].trust := 0;
 pol_neighs := 0;
 neigh := system[i].neighs;
 repeat
  neigh := neigh^.next;
  if neigh^.pol then
   begin
    pol_neighs := pol_neighs + 1;
    if alliances then
     if system[i].all <> nil then {need assistance?}
      if system[i].all^.threat = neigh^.id then
       if (neigh^.history[other] = d) then
        begin
         system[i].all^.obligation := true;
{writeln('CALL FOR HELP: ', i);}
        end;
    trust := neigh^.trust; {* res prop to res anyway}
    system[i].trust := system[i].trust + trust;
    if trust < min_trust then
     begin
      threat := neigh^.id;
      min_trust := trust;
     end;
   end;
 until last(neigh);
{update threat}
 system[i].pol_neighs := pol_neighs;
 if min_trust < min_threat then
  system[i].threat := threat
 else
  system[i].threat := 0;
```

```
    if (soc_neg <> 0) or (soc_pos <> 0) then
     if system[i].trust < soc_neg * pol_neighs then
      if system[i].culture = prey then
       begin
        system[i].culture := pred;
        pred_states := pred_states + 1;
       end
      else if system[i].trust > soc_pos * pol_neighs then
       if system[i].culture = pred then
        begin
         system[i].culture := prey;
         pred_states := pred_states - 1;
        end;
  end;

  procedure perceptions;
   var
    i: id_t;
  begin
   for i := 1 to n do
    if sov(i) then
     begin
{update_trust(i);}
 {use res_i and res_j instead}
      update_threat(i);
     end;
  end;

(* ---------------------------------------------------------------- *)

  function valid (i: integer): boolean;
  begin
   valid := (i > 0) and (i <= n);
  end;

  function select_agent (i, j: id_t): id0_t;
   var
    k: id0_t;
    buffer: random_buffer_t;
    agentp: link_t;
    result: id0_t;
  begin
   result := 0;
   if atom(i) then
    result := i
   else
    begin
     if adjacent(i, j) then
      begin
       k := 1;
       buffer[1] := i;
      end
     else
      k := 0;
     agentp := system[i].neighs;
     repeat
      agentp := agentp^.next;
```

```
{select agent adjacent to corp(j)}
      if agentp^.child then
       if adjacent(agentp^.id, j) then
        begin
         k := k + 1;
         buffer[k] := agentp^.id;
        end;
     until last(agentp);
     if k <> 0 then
      result := buffer[random_integer(k)];
   end;
  if not valid(result) then
   writeln('ERROR IN SELECT AGENT ', i : 4, j : 4, '=', result);
  select_agent := result;
end;
```



```
function select_target (agent, j: id_t): id0_t;
 var
  k: id0_t;
  buffer: random_buffer_t;
  targetp: link_t;
  result: id0_t;
begin
 if atom(j) then
  result := j
 else
  begin
   result := 0;
   k := 0;
   targetp := system[agent].neighs;
   repeat
    targetp := targetp^.next;
    if targetp^.terr then
     if compatriots(targetp^.id, j) then  {problem, compatriot called with illegal indices}
      begin
       k := k + 1;
       buffer[k] := targetp^.id;
      end;
   until last(targetp);
   if k <> 0 then
    result := buffer[random_integer(k)];
  end;
 select_target := result;
end;
```

```
function ext_res (i, threat: id_t): res_t;
begin
 if aligned(i) then
  if system[i].all^.threat = threat then
   ext_res := trunc((1.0 - all_contribution / 100.0) * system[i].res + all_contribution * system[i].all^.res / 1(
  else
   ext_res := system[i].res
 else
  ext_res := system[i].res;
end;
```

```
 function balance (i, j: id_t): real;
{returns true if superior, res_j is total res of j regardless of allocation}
{alliance resources should be taken into consideration}
  var
   own_res, other_res: res_t;
 begin
  if alliances then
   begin
    own_res := ext_res(i, j);
    other_res := ext_res(j, i);
   end
  else
   begin
    own_res := system[i].res;
    other_res := system[j].res;
   end;
  balance := own_res / other_res;
 end;

 function capita_res (i: id_t): res_t;
 begin
  capita_res := system[i].res div (system[i].children + 1);
 end;

 function binary_act (a: act_t): res_t;
 begin
  if a = d then
   binary_act := 1
  else
   binary_act := 0;
 end;

 function peace (neigh: link_t): boolean;
 begin
  peace := (neigh^.history[self] = c) and (neigh^.history[other] = c);
 end;

 function distance (i, j: id_t): real;
  var
   dist: pos_t;
 begin
  dist.x := (system[i].pos.x - system[j].pos.x);
  dist.y := (system[i].pos.y - system[j].pos.y);
  distance := sqrt(dist.x * dist.x + dist.y * dist.y);
 end;

 function battle_res (i, agent: id_t; res: res_t): res_t;
  var
   c: real;
 begin
  if (grad = 100) or atom(i) then
   c := 1
  else
   begin
    c := exp(distance(i, agent) * ln(grad / 100.0));
   end;
```

```
  battle_res := round(c * res);
end;

procedure decide_prey (i: id_t; var attacks: integer);
 var
  neigh: link_t;
begin
 attacks := 0;
 neigh := system[i].neighs;
 repeat
  neigh := neigh^.next;
  if neigh^.pol then
   begin
    neigh^.act := neigh^.history[other]; {TFT rule}
    if alliances then
     if system[i].all <> nil then
      if (system[i].all^.threat = neigh^.id) and system[i].all^.obligation then
       if bern(pr_oblig / 100.0) then
        begin
         neigh^.act := d; {assistance to alliance partners}
         if peace(neigh) then
          begin
           neigh^.agent := select_agent(i, neigh^.id);
           neigh^.target := select_target(neigh^.agent, neigh^.id);
          end;
{WRITELN('OBLIG FULFILLED', i, neigh^.agent, neigh^.target);}
         if (neigh^.history[self] = d) or (neigh^.history[other] = d) then
          attacks := attacks + 1;
        end;
   end;
  until last(neigh);
end;


procedure decide_pred (i: id_t; var attacks: integer);
 var
  neigh, weakest: link_t;
  rest: longint;
  max, ratio: real;
begin
 weakest := nil;
 max := -999999999;
 attacks := 0;
 neigh := system[i].neighs;
 repeat
  neigh := neigh^.next;
  if neigh^.pol then
   begin
    neigh^.act := neigh^.history[other]; {TFT rule}
    if (system[i].culture = pred) and (neigh^.history[self] = d) then
     neigh^.act := d; {unforgiving and exploitative}
    if alliances then
     if system[i].all <> nil then
      if (system[i].all^.threat = neigh^.id) and system[i].all^.obligation then
       begin
        neigh^.act := d; {assistance to alliance partners}
```

```
            if peace(neigh) then
              begin
                neigh^.agent := select_agent(i, neigh^.id);
                neigh^.target := select_target(neigh^.agent, neigh^.id);
              end;
{WRITELN('OBLIG FULFILLED', i, neigh^.agent, neigh^.target);}
          end;
        if (neigh^.history[self] = d) or (neigh^.history[other] = d) then
          attacks := attacks + 1;
        ratio := balance(i, neigh^.id);
{res := ext_res(neigh^.id, i) - battle_res(i, neigh^.id, system[i].res);}
        if ratio > max then
          begin
            weakest := neigh;
            max := ratio;
          end;
      end;
    until last(neigh);
{predators opportunistically make an unprovoked attack against weakest neigh}
    if (system[i].culture = pred) and (weakest <> nil) and (attacks = 0) then
      begin
        if bern(bell_pred / 100.0) and (max > superior_ratio / 100.0) then
          begin
            weakest^.agent := select_agent(i, weakest^.id);
            weakest^.target := select_target(weakest^.agent, weakest^.id);
            weakest^.act := d;
            attacks := 1;
          end;
      end;
  end;

  procedure allocate_res (i: id_t; attacks: integer);
    var
      neigh: link_t;
      share: real;
      available_res: res_t;
  begin
  {if attacks > 0}
    available_res := system[i].res - (1 + system[i].children) * min_res;
    neigh := system[i].neighs;
    repeat
      neigh := neigh^.next;
      if (neigh^.pol) then {and (neigh^.act = d)}
        begin
          if capita_res(i) >= min_res then
            begin
              neigh^.res := round(((1 + (mobil / 100.0) * binary_act(neigh^.act)) * available_res) / (system[i].pol_neig
(mobil / 100.0) * attacks));
            end
{share := (offset_trust + d_trust - neigh^.trust) / (system[i].pol_neighs * (offset_trust + d_trust) - system[i].tr
{neigh^.res := trunc(share * available_res)}
{NOTE: the factor 2 in the denominator #REAL}
          else
            neigh^.res := 0;
        end
      until last(neigh);
    end;
```

Choose both once ∀ and I have subsequent calculation

Problem:
- target/agent not chosen
- how calculate distance?

random choose weakest

battle_field

local balance:
victim := buffer[random_integer(k)]

victim

add pull out

pull out

A B C D E F G
H I J K L M N

x y A B

- res
- fronts ....
- combat_res ...
- old_combat_res

total # fronts

fronts:

```
procedure decide (i: id_t);
 var
  attacks: integer;
begin
 if system[i].culture = prey then
  decide_prey(i, attacks)
 else
  decide_pred(i, attacks);
 allocate_res(i, attacks);
end;

procedure decisions;
 var
  i: id_t;
begin
 if sov_states > 1 then
  for i := 1 to n do
   if sov(i) then
    decide(i);
end;
```

Laws of Interaction

```
function attack (neigh_i, neigh_j: link_t): boolean;
begin
 attack := (neigh_i^.act = d) and (neigh_j^.act = d) and (neigh_i^.history[self] = c) and (neigh_j^.history[self] =
end;

function win_battle (i, j: id_t; res_i, res_j: res_t): boolean;
begin
 win_battle := res_i > res_j * (victory_ratio / 100.0);
end;

function in_same_alliance (i, j: id_t): boolean;
 var
  a1, a2: id0_t;
begin
 a1 := counter_balance(i);
 a2 := counter_balance(j);
 in_same_alliance := (a1 = a2) and (a1 <> 0);
end;

procedure update_trust (var trust: trust_t; sensitivity, delta: real);
begin
 trust := trunc((1.0 - sensitivity) * trust + sensitivity * delta);
 if trust > d_trust then
  trust := d_trust
 else if trust < -d_trust then
  trust := -d_trust;
end;
```

```
 procedure update_threatened (i, j: id_t);
{update threat levels of all units j threatened by i, provided that i is a predator}
  var
   neigh: link_t;
   sensitivity: real;
   obligation: boolean;
 begin
  obligation := false;
  if counter_balance(i) = j then
   if system[i].all^.obligation then
    obligation := true;
  if not obligation then
   begin
    neigh := system[i].neighs;
    repeat
     neigh := neigh^.next;
     if neigh^.pol then
      begin
       if neigh^.id = j then
        sensitivity := 1.0
       else
        sensitivity := provoke / 100.0;
       update_trust(neigh^.link^.trust, sensitivity, -d_trust);
      end;
    until last(neigh);
   end;
 end;


{$PUSH}
{$R-}
 procedure interaction (i, j: id_t; neigh_i, neigh_j: link_t);
  var
   ires, jres: res_t;

  procedure discount_res;
  begin
   ires := battle_res(i, neigh_i^.agent, neigh_i^.res);
   jres := battle_res(j, neigh_j^.agent, neigh_j^.res);
  end;
{$POP}

 begin
{need to avoid simultaneous attacks because agent/target can only be set}
{by one party}
  if attack(neigh_i, neigh_j) then
   if bern(0.5) then
    neigh_i^.act := c
   else
    neigh_j^.act := c;
{update history}
  neigh_i^.history[self] := neigh_i^.act;
  neigh_i^.history[other] := neigh_j^.act;
  neigh_j^.history[self] := neigh_j^.act;
  neigh_j^.history[other] := neigh_i^.act;

  if (neigh_i^.act = c) and (neigh_j^.act = c) then
```

```
      begin
        neigh_i^.dres := peace_div;
        neigh_j^.dres := peace_div;
        update_trust(neigh_i^.trust, forgive / 100.0, +d_trust);
        update_trust(neigh_j^.trust, forgive / 100.0, +d_trust);
      end
    else if (neigh_i^.act = c) and (neigh_j^.act = d) then
      begin
        neigh_i^.target := neigh_j^.agent;
        neigh_i^.agent := neigh_j^.target;
        discount_res;
        neigh_i^.dres := -defeat_cost * jres div 100;
        neigh_j^.dres := -victory_cost * ires div 100;
        update_threatened(j, i);
        if in_same_alliance(j, i) then
          begin
            remove_member(j);
          end;
      end
    else if (neigh_i^.act = d) and (neigh_j^.act = c) then
      begin
        neigh_j^.target := neigh_i^.agent;
        neigh_j^.agent := neigh_i^.target;
        discount_res;
        neigh_i^.dres := -victory_cost * jres div 100;
        neigh_j^.dres := -defeat_cost * ires div 100;
        update_threatened(i, j);
        if in_same_alliance(i, j) then
          begin
            remove_member(i);
          end;
      end
    else {d,d}
      begin
        discount_res;
        neigh_i^.dres := -war_cost * jres div 100;
        neigh_j^.dres := -war_cost * ires div 100;
        update_trust(neigh_i^.trust, provoke / 100.0, -d_trust);
        update_trust(neigh_j^.trust, provoke / 100.0, -d_trust);
      end;

  {check claims}
    neigh_i^.claim := false;
    neigh_j^.claim := false;
    if (neigh_i^.act = d) and win_battle(i, j, ires, jres) then
      neigh_i^.claim := true;
    if (neigh_j^.act = d) and win_battle(j, i, jres, ires) then
      neigh_j^.claim := true;
    event := neigh_i^.claim or neigh_j^.claim;
  end;


  procedure interactions;
{only confronts states; no internal interaction; update history}
    var
      i, j: id_t;
```

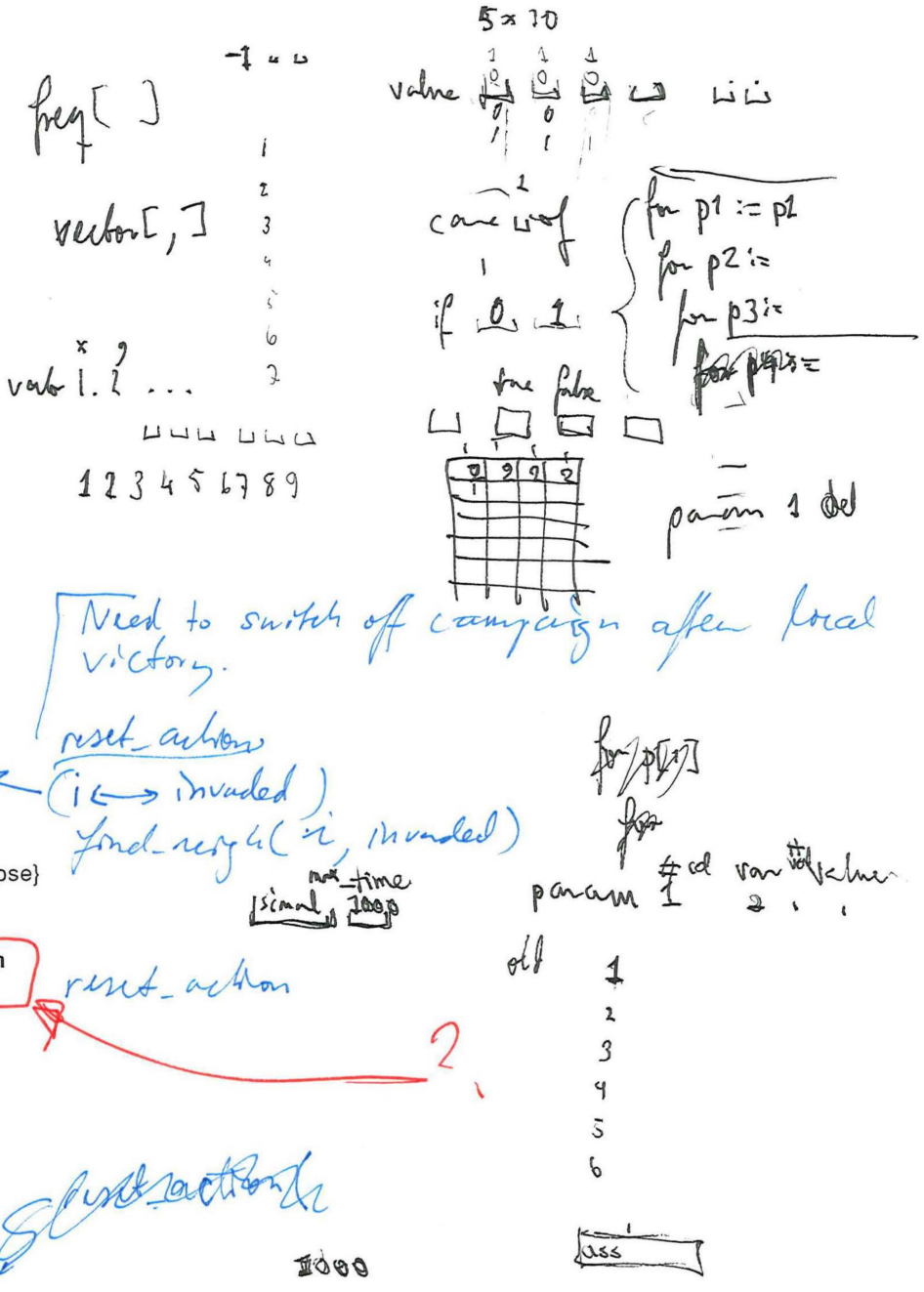
```
    neigh: link_t;
begin
 for i := 1 to n do
  if sov(i) then
   begin
    neigh := system[i].neighs;
    repeat
     neigh := neigh^.next;
     j := neigh^.id;
     if neigh^.pol then
      if i < j then
       interaction(i, j, neigh, neigh^.link);
    until last(neigh);
   end;
end;

function harvest (m, s: res_t): res_t;
begin
 if s = 0 then
  harvest := m
 else
  harvest := trunc(normal(m, s));
end;

procedure update_resources;
 var
  i: id_t;
  neigh: link_t;
  dres, tax: res_t;
begin
 for i := 1 to n do
  if sov(i) then
   begin
    dres := 0;
    tax := 0;
    neigh := system[i].neighs;
    repeat
     neigh := neigh^.next;
     if neigh^.pol then
      dres := dres + neigh^.dres
     else if neigh^.child then
      tax := tax + harvest(harvest_m, harvest_s);
    until last(neigh);
    system[i].res := system[i].res + harvest(harvest_m, harvest_s) + tax + dres;
    if system[i].res < 0 then
     system[i].res := 0;
   end;
end;

procedure update_all_resources;
 var
  all: all_t;
  member: link_t;
begin
 all := alliance_list;
 if not empty(alliance_list) then
  repeat
```

```
    all := all^.next;
    all^.res := 0;
    all^.obligation := false;
    member := all^.members;
    repeat
     member := member^.next;
     all^.res := all^.res + system[member^.id].res;
    until last(member);
  until last(all);
end;
```

Structural Change

```
procedure unlock;
 var
  i: id_t;
begin
 for i := 1 to n do
  locked[i] := false;
end;

procedure reset_reach (i: id_t);
 var
  child: link_t;
begin
 child := system[i].neighs;
 repeat
  child := child^.next;
  if child^.child then
   system[child^.id].reached := false;
 until last(child);
end;

procedure reach (i: id_t);
 var
  neigh: link_t;
begin
 neigh := system[i].neighs;
 repeat
  neigh := neigh^.next;
  if neigh^.terr and compatriots(i, neigh^.id) and not system[neigh^.id].reached then
   begin
    system[neigh^.id].reached := true;
    reach(neigh^.id);
   end;
 until last(neigh);
end;

procedure topology (i: id_t);
 var
  child: link_t;
begin
 reset_reach(i);
 reach(i);
 child := system[i].neighs;
 repeat
  child := child^.next;
  if child^.child and not system[child^.id].reached then
```

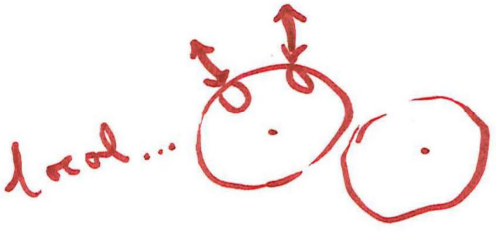
```
      remove_child(i, child^.id);
    until last(child);
  end;

  procedure collapse (j: id_t);
    var
      child_j: link_t;
  begin
    child_j := system[j].neighs;
    repeat
      child_j := child_j^.next;
      if child_j^.child then
        remove_child(j, child_j^.id);
    until last(child_j);
  end;

  procedure conquer (i, j: id_t);
    var
      invaded: id0_t;
  begin
{rewire hierarchy}
{assume target j is state}
    locked[i] := true;
    locked[j] := true;
    if subord(j) then
      begin {intrusion}
        invaded := system[j].parent;
        remove_child(invaded, j);
        topology(invaded);
      end
    else if not atom(j) then {collapse}
      collapse(j);
    add_child(i, j);
    if system[i].culture = pred then
      update_threatened(i, j);
  end;


  procedure structural_change;
    var
      i, state: id0_t;
      up: boolean;
      neigh: link_t;
  begin
{add spontaneous collapse}
    unlock;
    up := bern(0.5); {random direction of execution}
    for i := 1 to n do
      begin
        if up then
          state := i
        else
          state := n - i + 1;
        if sov(state) and not locked[state] then
          begin
            neigh := system[state].neighs;
            repeat
```



```
          neigh := neigh^.next;
          if neigh^.pol and neigh^.claim and not locked[neigh^.id] then
           begin
            neigh^.claim := false;
            if (neigh^.target >= 1) and (neigh^.target <= n) then {ERROR CHECK}
             conquer(state, neigh^.target)
            else
             writeln('ERROR CALLING CONQUER: ', state, neigh^.target);
           end;
         until last(neigh);
        end;
      end;
     for i := 1 to n do
      if sov(i) then
       if head(i) then
        if not locked[i] then
         if capita_res(i) < min_res then
          collapse(i);
end;

procedure diplomacy;
 var
  threat, threatee, counter: id0_t;
  neigh: link_t;
  all: all_t;
begin
 for threat := 1 to n do
  if sov(threat) then
   begin
    threatee := 0;
    neigh := system[threat].neighs;
    repeat
     neigh := neigh^.next;
     counter := counter_balance(neigh^.id);
     if neigh^.pol then
      if (system[neigh^.id].threat = threat) and not ((counter <> 0) and (counter <> threat)) then
{can be added if threatened by threat and not engaged in other alliance}
       threatee := threatee + 1;
    until last(neigh);
    all := system[threat].counter_all;
    if all = nil then {no alliance exists}
     if threatee < 2 then
{do nothing}
      else
       create_all(threat)
     else {alliance already exists}
      if threatee < 2 then
       delete_all(all)
      else
       adjust_all(threat, all);
   end;
end;


end.
```

spontaneous collapse

res transf



Literaturverzeichnis

[Cederman 1993] Cederman, Lars-Erik: *simul.proj.* Altcode. 1993. – Sourcecode des EPM, von Herrn Cederman zur Verfügung gestellt

[Cederman 1997] Cederman, Lars-Erik: *Emergent Actors in World Politics: How States and Nations Develop.* Princeton University Press, 1997

[Cederman 2001] Cederman, Lars-Erik: Agent-Based Modeling in Political Science. In: *The Political Methodologist* 10 (2001), Nr. 1, S. 16 – 22

[Drogoul u. a. 2003] Drogoul, Alexis ; Vanbergue, Diane ; Meurisse, Thomas: Multi-agent based simulation: where are the agents? In: Sichman, Jaime (Hrsg.) ; Bousquet, Francois (Hrsg.) ; Davidsson, Paul (Hrsg.): *Proceedings of MABS 2002 multi-agent-based simulation, vol 2581, Lecture notes in computer science.* Springer, 2003, S. 1–15

[Galán u. a. 2013] Galán, José M. ; Izquierdo, Luis R. ; Izquierdo, Segismundo S. ; Santos, José I. ; Olmo, Ricardo del ; López-Paredes, Adolfo: Checking Simulations: Detecting and Avoiding Errors and Artefacts. In: Edmonds, Bruce (Hrsg.) ; Meyer, Ruth (Hrsg.): *Simulating Social Complexity. A Handbook.* Springer, 2013, S. 95–116

[Gilbert 2008] Gilbert, Nigel: *Agent-Based Models.* Sage Publications, 2008

[Gilbert und Troitzsch 2005] Gilbert, Nigel ; Troitzsch, Klaus G.: *Simulation for the Social Scientist.* Open University Press, 2005

[Hales u. a.] Hales, David ; Rouchier, Juliette ; Edmonds, Bruce: Model-to-model analysis. In: *Journal of Artificial Societies and Social Simulation*

[Larman und Basili 2003] Larman, Craig ; Basili, Victor R.: Iterative and Incremental Development: A Brief History. In: *IEEE Computer* 36 (2003), Nr. 6

[Lazer 2001] Lazer, David: Review of „Emergent Actors in World Politics: How States and Nations Develop“. In: *The Journal of Artificial Societies and Social Simulation* 4 (2001), Nr. 2

[Sayama 2015] Sayama, Hiroki: *Introduction to the Modeling and Analysis of Complex Systems*. Open SUNY Textbooks, Milne Library, 2015

[Tange 2011] Tange, O.: GNU Parallel - The Command-Line Power Tool. In: *;login: The USENIX Magazine* 36 (2011), Feb, Nr. 1, S. 42–47. – URL http://www.gnu.org/s/parallel

[Weber 1946] Weber, Max: *Wirtschaft und Gesellschaft: Grundriss der verstehenden Soziologie*. Mohr Siebeck, 1946

[Wolfers 1962] Wolfers, Arnold: *Discord and Collaboration: Essays on International Politics*. John Hopkins University Press, 1962

[Wooldridge 2009] Wooldridge, Michael: *An Introduction to MultiAgent Systems*. John Wiley & Sons Ltd, 2009

[Wooldridge und Jennings 1995] Wooldridge, Michael J. ; Jennings, Nicholas R.: Agent theories, architectures, and languages: A survey. In: Wooldridge, Michael J. (Hrsg.) ; Jennings, Nicholas R. (Hrsg.): *Intelligent Agents*. Springer Verlag, 1995, S. 1–22

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 22. März 2018 Alena Störmer