CAS DHBW
Kurs TM40501 – Intelligente Agenten und Multiagentensysteme

Dokumentation – NEPO

Projekt in Gruppenarbeit

Vorgelegt am

31. März 2017

von

Sascha Weisenberger, Stefan Ghizelea, Michael Stahlberger, Patrick Schulz

1. Inhalt

2. Problemstellung

Bei der Abdeckung von ländlichen Regionen mit mobilem Netz stehen sich verschiedene Parteien mit unterschiedlichen Interessen gegenüber:

- Die **Netzanbieter** versuchen, mit möglichst wenigen Funksendemasten die größtmögliche Netzabdeckung zu erreichen. So lassen sich Kosten optimieren während die Erträge maximiert werden. Manche Plätze sind für den Anbieter besser geeignet als andere, da beispielweise die Errichtungskosten niedriger sind. Die optimalen Standorte sollen jedoch geheim gehalten werden, um Konkurrenten keinen Vorteil zu verschaffen.
- Die **Kommunen** agieren gemeinsam als Agent. Sie möchten für jede Kommune die bestmögliche Netzabdeckung. Allerdings sollen die Sendemasten einen Mindestabstand von bewohnten Gebieten einhalten, um Beeinträchtigungen und Gesundheitsrisiken zu vermeiden
- Hinzu kommt die zu Grunde liegende **Karte**. Auf dieser kann es Gebiete geben, die nicht zur Errichtung von Funksendemasten freigegeben sind. Ein Beispiel hierfür sind Naturschutzgebiete. Diese gesperrten Gebiete dürfen bei der Positionierung der Funksendemasten nicht eingeschlossen werden.

Mithilfe der *Negotiation-based evolutionary positioning optimisation* (NEPO) soll dieses Optimierungsproblem mit gegenläufigen Interessen für alle Parteien zufriedenstellend als Multiagentensystem gelöst werden.

3. Konzeption

Das Multiagentensystem wurde als tatsächliches Multiagentensystem konzipiert. Die verschiedenen Agenten werden also nicht nur simuliert, sondern sind tatsächlich unabhängige Programminstanzen, die mit dem als Webservice konzipierten Mediator kommunizieren. Somit wäre mit diesem Prototyp also auch eine verteilte Ausführung auf verschiedenen Computern möglich.

Außerdem soll das System flexibel gehalten werden. Mit verschiedenen Parametern kann es somit für viele verschiedenen Optimierungsprobleme angepasst und eingesetzt werden.

Im Folgenden wird die Konzeption genauer beleuchtet.

3.1. Gegebene Parameter

Vor Beginn der Verhandlung wird die zu Grunde liegende Umgebung und Rahmenparameter definiert. Folgende Objekte und Parameter werden hierbei betrachtet:

Karte:	• Größe in Pixel • Sperrgebiete mit Position (Koordinaten) und Größe
Städte:	• Position als Koordinaten • Größe (Einwohnerzahl)
Sendemast:	• Minimaler Radius: Radius, in dem sich keine Städte befinden sollten • Maximaler Radius: Maximaler Empfangsradius
Verhandlungsparameter:	• Vorschläge pro Verhandlungsrunde • Minimale Anzahl akzeptierter Vorschläge

3.2. Klassendiagramm

Folgendes Klassendiagramm liegt dem entworfenen Multiagentensystem zu Grunde:

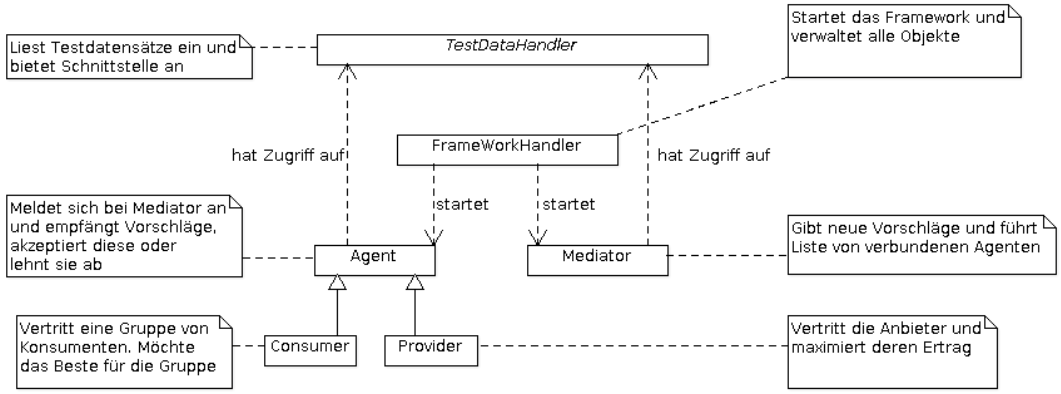

Abbildung 1: Klassendiagramm

Die Klasse *Agent* bildet das Rahmengerüst, um mit dem Mediator zu kommunizieren. Hierbei gilt es zu beachten, dass die verschiedenen Agenten jeweils Kindelemente der Klasse *Agent* darstellen. Die Ausprägungen der verschiedenen Agenten werden im folgenden Kapitel genauer definiert.

Der Mediator ist eine eigene Klasse, der als Webservice aufgebaut ist. Er liefert auf Anfrage der Agenten neue Vorschläge.

3.3. Verhandlungsprozess

Das Herzstück eines Multiagentensystems bildet der zentrale Verhandlungsprozess. Bei dem vorliegenden System generiert der Mediator jede Runde ein Tupel von neuen Verträgen. Die Agenten dürfen hiervon maximal eine definierte Anzahl ablehnen. Anhand der Überschneidungen der von beiden Agenten akzeptierten Lösungen bestimmt der Mediator die neue beste Lösung und generierte ein neues Tupel an Lösungsvorschlägen.

Eine Verhandlung ist beendet, wenn für eine vordefinierte Anzahl an Runden keine neue beste Lösung mehr gefunden wurde. Die Verhandlung stagniert an diesem Punkt, weil die Agenten jeweils unterschiedliche Lösungen ablehnen und deshalb keine Überschneidungen mehr vorhanden sind. Damit eine Verhandlung nicht unendlich andauern kann, wird außerdem eine maximale Rundenanzahl definiert.

Um eine Verhandlung ermöglichen zu können, müssen bei der Initialisierung die Rahmenbedingungen festgelegt werden. Folgende Parameter bedürfen der Definition:

- Anzahl Lösungsvariationen pro Verhandlungsrunde
- Anzahl an Lösungsvorschlägen, denen zugestimmt werden muss
- Maximale Anzahl an Verhandlungsrunden ohne Veränderung
- Maximale Anzahl an Verhandlungsrunden

3.4. Aufbau der Agenten

Beide Agenten verfolgen eigene Ziele, die in der Regel gegenläufig zueinander sind. Im Folgenden werden die Zielfunktionen der Agenten allgemein dargestellt. Die Parameter werden von den jeweiligen Parteien vor der Verhandlung konfiguriert.

Netzanbieter

Der Netzanbieter versucht, möglichst viele *Städte* abzudecken. Er geht pro *Einwohner* von einem pauschal definierten *Gewinn* aus.

Um seine Kosten zu optimieren, versucht der Anbieter, dies mit möglichst wenigen *Sendemasten* zu erreichen. Der Netzanbieter ist indifferent, in welchen Abstand der Sendemast zu den Städten platziert wird.

Je nach Standpunkt kann die Position für den Anbieter mit unterschiedlichen *Kosten* verbunden sein. Beispielweise ist es teurer, einen Sendemast auf einem Berg zu installieren als auf einer Wiese.

Die Zielfunktion des Netzanbieters lautet folglich:

$$Z_{Netzanbieter} = \sum_{Stadt} (Einwohner * Gewinn) - \sum_{Sendemast} (Kosten)$$

Kommunen

Im Gegensatz zum Netzanbieter ist es den Kommunen nicht nur wichtig, möglichst viele *Städte* mit Mobilfunk abzudecken, sondern ist auch auf die Gesundheit der Bürger bedacht. Es wird deshalb darauf geachtet, dass keine Stadt im inneren Radius *Min* der Funksendemasten liegt. In diesem Radius wird von einem zu hohen Krankheitsrisiko ausgegangen. Desto weiter die Stadt vom Sendemast entfernt liegt, desto niedriger ist das Krankheitsrisiko. Ziel ist es also, möglichst zwischen dem maximalen Radius *Max* und dem minimalen Radius *Min* zu liegen.

Somit ergibt sich pro Stadt folgende Zielfunktion *Z* in Abhängigkeit von der Entfernung *x*:

$$Z_{Stadt}(x) = -\frac{1}{\left(\frac{Max - Min}{2}\right)^2} * \left(x - \frac{Min + Max}{2}\right)^2 + 1 \; ; Z \in [-1; 1]$$

Um eine möglichst gute Lösung zu erhalten, wird die Zielfunktion in den ersten Verhandlungsrunden toleranter eingestellt und zum Ende der Verhandlung verfeinert. So erreichen die Kommunen, dass in den ersten Schritten möglichst viele Kommunen einbezogen werden. Anschließend wird die Feinjustierung vorgenommen. Dabei wird versucht, dass die Städte möglichst im mittleren, grünen Bereich der Sendemasten liegen:

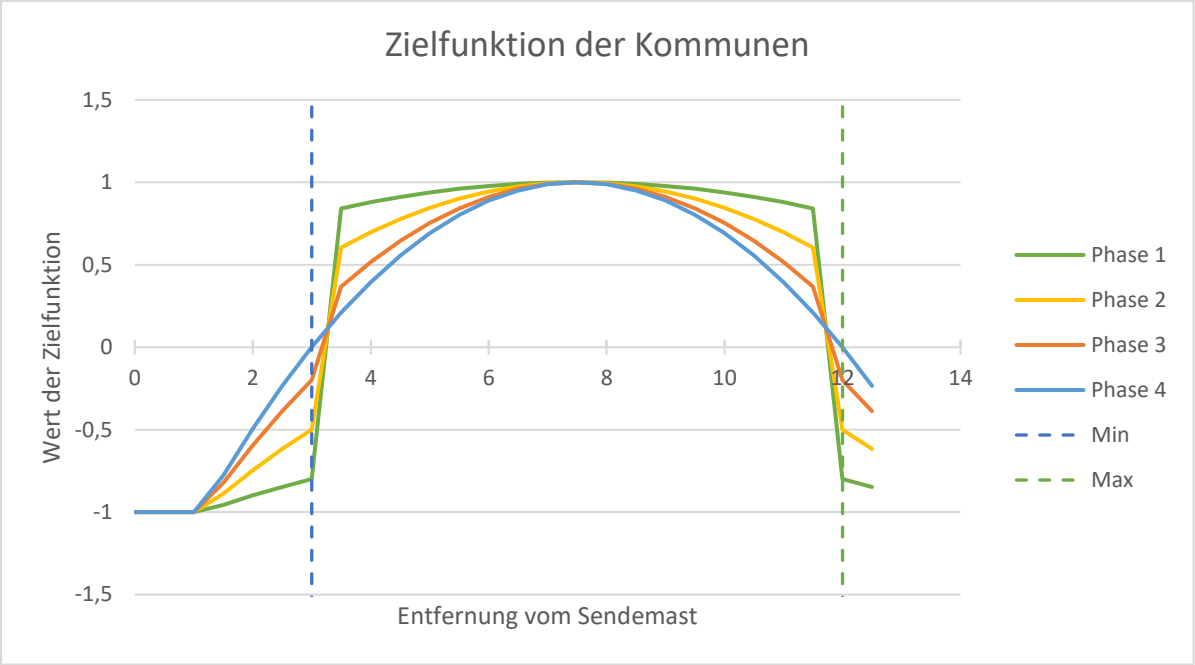

Abbildung 2: Zielfunktion der Kommunen mit minimalem und maximalem Radius

Die Belange größerer Städte werden hierbei anhand der Anzahl an Einwohnern stärker gewichtet als die Belange kleinerer Kommunen. Zusammengefasst stellt sich die Zielfunktion somit in folgender Form dar:

$$z_{Kommunen} = \frac{Einwohner}{\sum_{Stadt} Einwohner} * \sum_{Stadt} \left(-\frac{1}{\left(\frac{Max - Min}{2}\right)^2} * \left(x - \frac{Min + Max}{2}\right)^2 + 1 \right)$$

3.5. Programmablauf

Im Folgenden wird der Ablauf einer Verhandlung dargestellt.

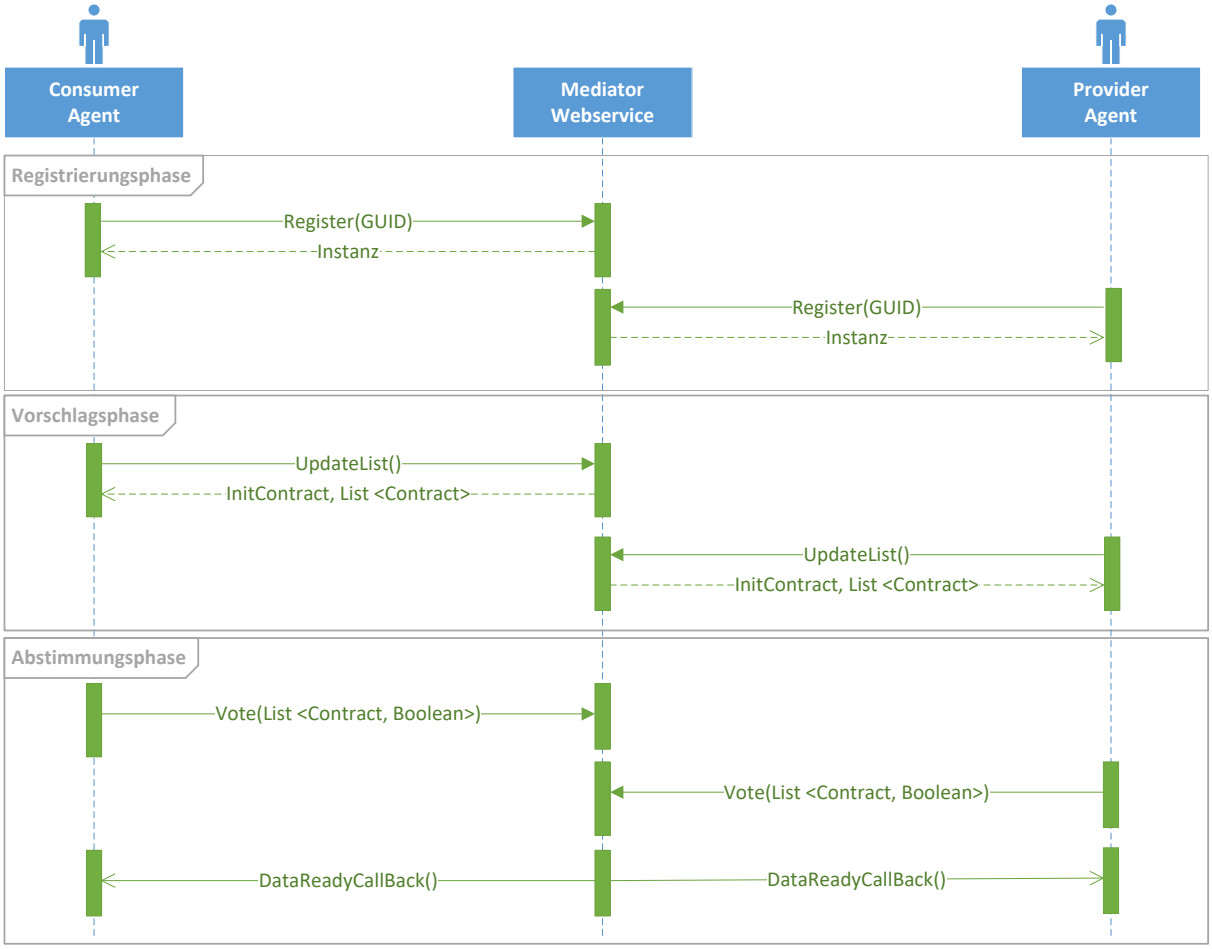

Abbildung 3: Programmablauf als Sequenzdiagramm

Registrierungsphase

1. Der Mediator wird gestartet, die Kartendaten hinterlegt (Karte, Städte, Sperrzonen) und die Verhandlungsregeln konfiguriert.
2. Register():
   Die Agenten registrieren sich beim Mediator (Webservice) und erhalten die Kartendaten. Bis zur eigentlichen Verhandlung können jetzt die individuelle Zielfunktion angepasst werden.

Verhandlung

3. UpdateList():
   Die Agenten fragen den Mediator an und erhalten eine Vergleichslösung (die aktuell beste Lösung) sowie eine Liste mit einer Anzahl mutierten Lösungen

4. Vote():
   Die Agenten berechnen den Zielwert für jede Lösung und senden die bewertete Liste an den Mediator. Hierbei muss eine vordefinierte Anzahl an Lösungen akzeptiert werden.
5. DataReadyCallBack():
   Nachdem alle Parteien ihre bewertete Lösung an den Mediator rückgemeldet haben, werden die Agenten per Callback informiert und eine neue beste Lösung sowie Alternativlösungen bestimmt.
6. UpdateList():
   Die Agenten fragen daraufhin erneut die Daten an und erhalten wiederum die beste Lösung sowie eine Liste mit Alternativlösungen. Eine neue Verhandlungsrunde beginnt.

4. Implementierung

Das Multiagentensystem wurde in der Programmiersprache C# im .NET-Framework von Microsoft umgesetzt. An dieser Stelle wird auf die verschiedenen Komponenten im Detail eingegangen.

Eine Stärke der Implementierung ist die Anpassbarkeit an verschiedene Szenarien. Alle relevanten Parameter lassen sich für jede Verhandlung einzeln konfigurieren. Jeder Agent kann selbst mit der Benutzeroberfläche die für ihn passenden Einstellungen evaluieren und definieren.

4.1. Architektur der verteilten Implementierung

Die verteile Kommunikation basiert auf Microsofts Windows Communication Foundation (WCF). Dafür werden serverseitig ein Serviceinterface und ein Callbackinterface für die Kommunikation definiert, wodurch später clientseitig Proxyklassen generiert werden können die den Remoteaufruf kapseln sodass er für Konsumenten der Proxyklassen wie ein lokaler Aufruf zu verwenden ist. Eine Übersicht über die Methoden des Interfaces bietet das Flussdiagram in Kapitel 4.2.

Serverseitig wird das Interface in einem Zwischenlayer implementiert das für die Registrierung von Clients und Callbackaufrufe an ebendiese zuständig ist. Dafür muss sich jeder Client zunächst mit einer GUID registrieren, welche im späteren Verlauf der Kommunikation und Programmlogik zur Identifizierung des Clients benutzt wird. Dieses Zwischenlayer ruft schließlich die eigentliche Programmlogik auf.

Clientseitig besteht über die generierte Proxyklasse die Möglichkeit die im Serviceinterface definierten Methoden an benötigter Stelle aufzurufen und deren Rückgabewerte zu verarbeiten. Zusätzlich wird das Callbackinterface implementiert und von einer kleinen Klasse gekapselt sodass der Callback als Event im Client nutzbar ist.

4.2. Detaillierter Programmablauf

Der in der Konzeption grob dargestellte Ablauf ist im Folgenden detailliert dargestellt. Da die Agenten (Netzanbieter und Kommunen) auf derselben Klasse basieren und dieselben Methoden ausführen, ist der Ablauf bei beiden Agenten gleich. Deshalb wird im Folgenden auf ein vereinfachtes Modell mit einem Agenten zurückgegriffen.

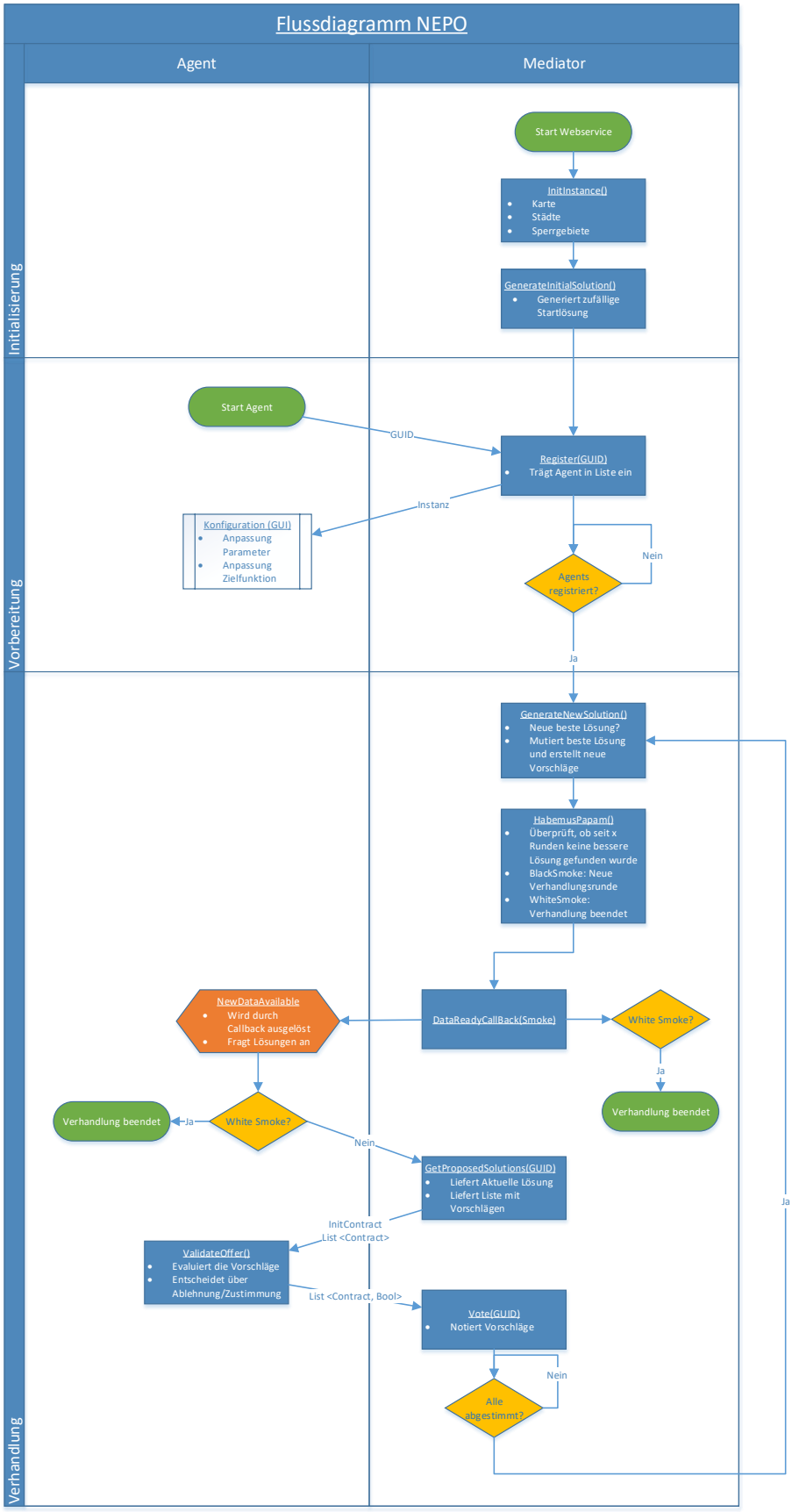

Abbildung 4: Programmablauf im Detail

Initialisierung

Einige Zeit vor der eigentlichen Verhandlung wird der Mediator als Webservice gestartet und initialisiert. Die Rahmenbedingungen wie die Karte, die Position und Größe der Städte sowie die ausgeschlossenen Sperrgebiete werden eingepflegt. Sie bilden die Instanz dieser Verhandlung. Anschließend wird eine erste Lösung als Startlösung generiert, die den Agenten als Vergleich dient.

Vorbereitung

Kurz vor der Verhandlung starten die Parteien Ihre Agenten und melden sich am Mediator an. Dieser registriert die Agenten und stellt Ihnen die Instanz bestehend aus Karte, den Städten und Sperrgebieten sowie die Initiallösung zur Verfügung.

Bis zum eigentlichen Start der Verhandlung kann jede Partei in ihrem Agenten über die bereitgestellte Benutzeroberfläche die eigene Zielfunktion anpassen und das lokale Optimum, als den maximalen Wert der eigenen Zielfunktion, errechnen.

Verhandlung

Wenn sich alle Agenten registriert haben kann die eigentliche Verhandlungsphase beginnen. Jede Verhandlungsrunde hat den gleichen Ablauf, der hier beschrieben wird.

Der Mediator mutiert die aktuell beste Lösung (am Anfang die Initiallösung) und generiert ein Tupel aus Vorschlägen. Dieses Tupel wird den Agenten zur Verfügung gestellt und diese berechnen die Güte jeder einzelnen Lösung. Die Art und Weise wie die Güte berechnet wird ist dabei jedem einzelnen Agenten überlassen und muss dem Mediator auch nicht mitgeteilt werden. Jeder Agent muss aber eine vordefinierte Anzahl ein Lösungen auswählen, denen er „zustimmt“. Die Mindestanzahl an Zustimmungen bekommt jeder Agent über die Konfiguration am Anfang mitgeteilt.

Über die Vote()-Funktion teilt er diese Lösungen dann dem Mediator mit. Der Mediator sucht nun aus allen Lösungen die für „gut“ befunden wurden diejenige aus, die die meisten Zustimmungen erhalten hat. Wenn es beispielsweise 2 Agenten gibt, sucht der Mediator die Lösung, für die beide Agenten gestimmt haben. Gibt es keine solche Lösung (das ist möglich wenn die Mindestzustimmung kleiner als Mutationszahl/2 ist), dann wird im Wechsel immer eine Lösung akzeptiert, die nur ein einziger Agent akzeptiert hat. Das Voranschreiten in irgendeine Richtung ist hier wichtiger als eine gemeinsame Richtung zu finden. Dadurch, dass dies immer im Wechsel geschieht, ist das Verfahren auch fair und keiner wird benachteiligt.

Nachdem der Mediator nun einen neue „beste“ Lösung gefunden hat, generiert er daraus wieder eine gewisse Anzahl an Mutationen und teilt die „aktuelle“ Lösung und ihre Kinder den registrierten Agenten mit. Somit beginnt eine neue Verhandlungsrunde.

Bei jeder Verhandlung ist vorher definiert, wie viele Verhandlungsrunden es gibt. Den Agenten wird dann mit jeder Lösung der aktuelle Fortschritt in Prozent mitgeteilt. Dies kann der Agent dazu benutzen, in seiner Zielfunktion einen sog. Cooldown einzubauen. Die Zielfunktion kann sich also im Laufe der Optimierung ändern, sodass am Anfang schnell eine annehmbare Lösung gefunden wird, und das Feintuning dann erst im späteren Verlauf stattfindet.

Die Optimierung ist beendet, sobald ein Fortschritt von 100% erreicht ist. Die vom Mediator zuletzt als aktuelle Lösung gewählt wurde ist dann der Gewinner bzw. das Ergebnis der Optimierung.

4.3. Benutzeroberfläche

Zur einfachen Konfiguration und Anpassung der eigenen Zielfunktion verfügt der Agent über eine grafische Benutzeroberfläche. Über diese kann jede Partei eigene Regeln angeben, die bei der Berechnung des Zielwerts berücksichtigt werden. Zudem bereitet die Benutzeroberfläche die Verhandlung grafisch auf und bietet die Möglichkeit einer lokalen Optimierung nach den eigenen Regeln.

Konfiguration

Bevor die Verhandlung startet, kann jeder Agent mit Hilfe der Benutzeroberfläche eigene Regeln definieren. Die Rahmenbedingungen wie die Anzahl und Position der Städte werden in einer Instanz vom Mediator bereitgestellt (vgl. 4.2: Detaillierter Programmablauf).

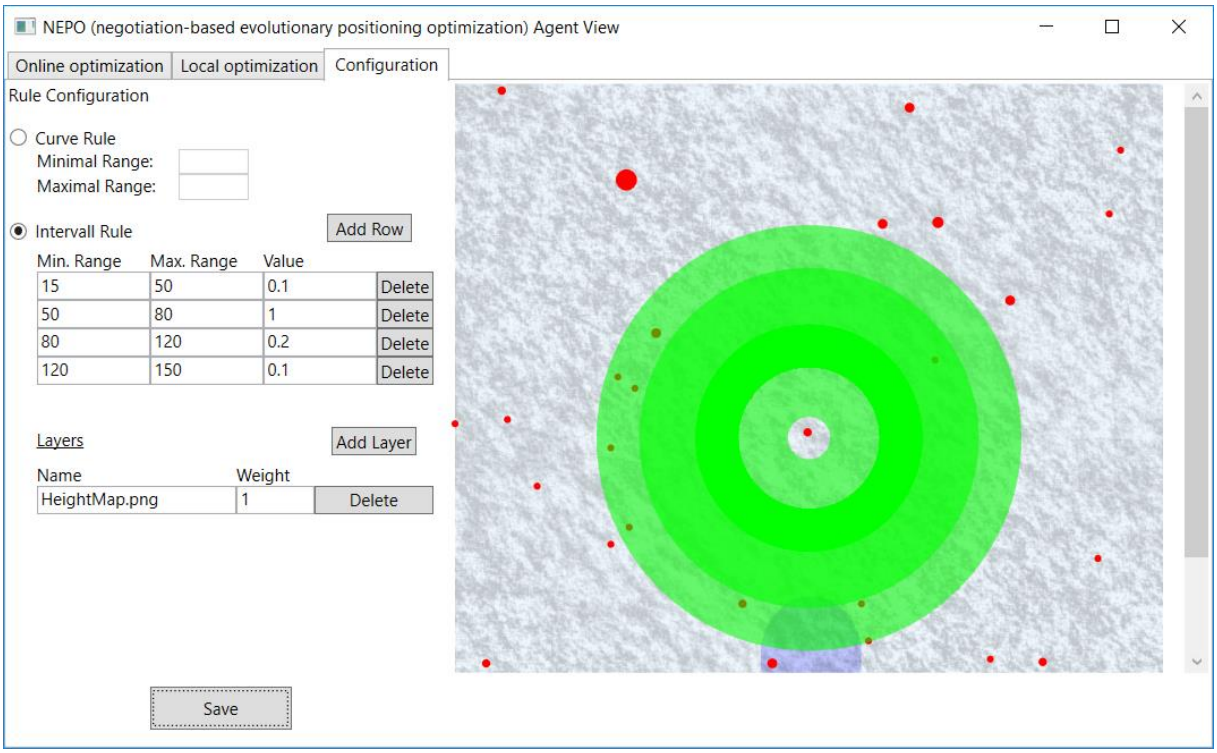

Abbildung 5: Benutzeroberfläche mit verschiedenen Intervallen und einem „Layer“

Hierfür bietet die Benutzeroberfläche die Möglichkeit, entweder eine Funktion mit einem Minimal- und einem Maximalwert anzugeben (Curve Rule) oder selbst Intervalle mit einer Wertung abzustecken. Die verschiedenen Regeln werden dabei grafisch auf der rechten Seite angezeigt.

Zusätzlich kann jeder Agent eigene Schichten zur Bewertung der Landschaft hinzufügen. Diese sogenannten „Layers“ werden als Bitmap-Datei hochgeladen und zeigen die Bewertung der jeweiligen Position als Helligkeitswerte an.

Optimierung

Vor der eigentlichen Verhandlung kann der Agent die Benutzeroberfläche nutzen, um eine lokale Optimierung durchzuführen. Hierbei wird das lokale Optimum wie bei einer Verhandlung mit nur einem Agenten gesucht. Der Anwender während der Optimierung verfolgen, wie sich die Sendemasten auf der Karte bewegen bis schließlich die optimale Lösung gefunden wird.

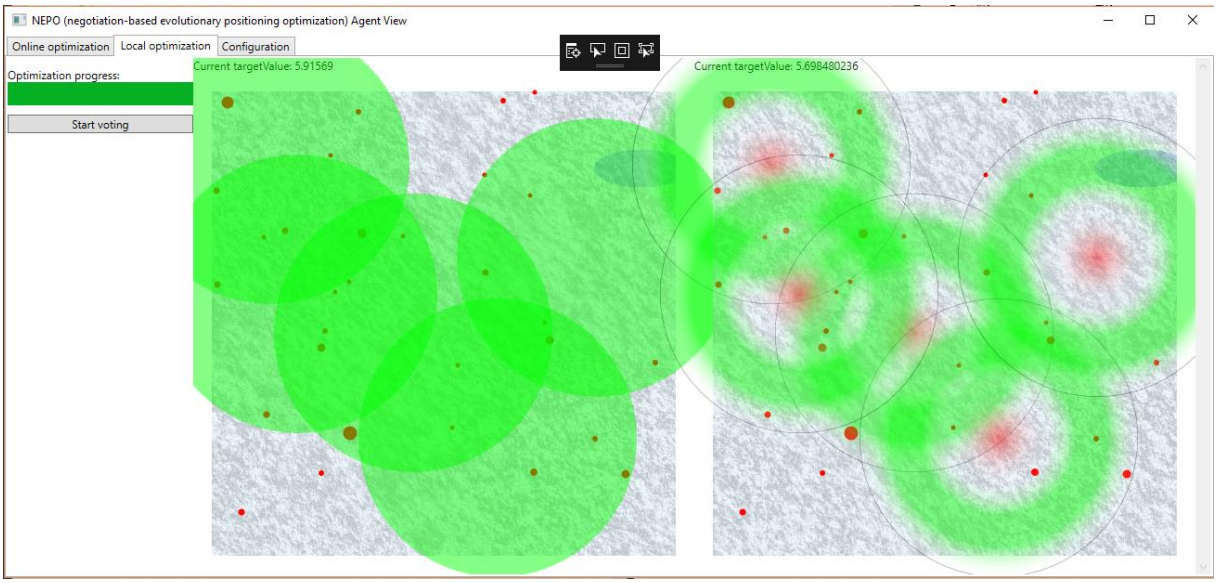

Abbildung 6: Lokale Simulation der Optimierung für beide Agenten

Auch während der Verhandlung kann der Agent mit Hilfe der Benutzeroberfläche die Optimierung beobachten. Ein grüner Balken auf der linken Seite zeigt hierbei den Fortschritt der Verhandlung in Abhängigkeit von der maximalen Rundenanzahl an (vgl. Abbildung 6).

4.4. Testdatengenerierung

Um das selbst entworfene Szenario testen zu können, wurden verschiedene Instanzen bestehend aus einer Karte mit Sperrgebieten und Städten mit einem Hilfsprogramm generiert.

Für relevante Testdaten werden vor der Generierung die Grenzen in einem XML-Dokument festgehalten. Anschließend erstellt das Programm eigenständig die gewünschte Anzahl an Instanzen mit jeweils unterschiedlichen Optimierungsproblemen und -situationen.

Für folgende Parameter kann jeweils ein Minimal-, sowie ein Maximalwert definiert werden:

- Größe der Karte in Pixel
- Anzahl an Sendemasten
- Anzahl der Städte
- Anzahl und Größe der Sperrzonen
- Anzahl der „Layers“ zur Positionsbewertung

Die so generierten Testinstanzen wurden für die Tests und Auswertung im folgenden Kapitel genutzt.

5. Test und Auswertung

Für den Test des entwickelten Prototyps wurden zehn eigens generierte Testinstanzen verwendet. Die Instanzen enthielten entweder sechs oder neun Städte sowie verschiedene „Sperrzonen“, um verschiedene mögliche Szenarien zu berücksichtigen und trotzdem noch eine gewisse Vergleichbarkeit zu ermöglichen.

Als Verhandlungsbedingungen wurden variable Rundenanzahlen zwischen zehn und 1000 Runden verwendet. Pro Runde wurde den Agenten jeweils 100 Vorschläge zur Abstimmung vorgelegt. Diese mussten jeweils mindestens einem bzw. zehn Vorschlägen zustimmen.

5.1. Evaluierungsmethodik

Um die Ergebnisse des Prototyps bewerten zu können, erzeugt das Programm ein Logfile, das die Rahmenbedingungen der Instanz sowie den Wert der Zielfunktion des Agenten enthält (vgl. Abbildung 7).

Type	AgentConfig	TestInstanzID	MaxRounds	VorschlaegeProRunde	ErzwungeneAkzeptanz	AnzahlPlanningObjects	ClientID	TargetValue
remote	c9ec95e6-665f-417f-a623-74a81f5d5cbb	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	49b9131d-eee8-4ce0-bdb9-aff7486f8995	30,46323
remote	c9ec95e6-665f-417f-a623-74a81f5d5cbb	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	042e885e-9572-4385-ae01-142a568f9efd	30,46323
remote	27d1c7b9-4fa3-4daf-ae8f-110cee7b0954	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	042e885e-9572-4385-ae01-142a568f9efd	6,40615
remote	c9ec95e6-665f-417f-a623-74a81f5d5cbb	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	49b9131d-eee8-4ce0-bdb9-aff7486f8995	30,48549
remote	27d1c7b9-4fa3-4daf-ae8f-110cee7b0954	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	042e885e-9572-4385-ae01-142a568f9efd	6,320852745
remote	c9ec95e6-665f-417f-a623-74a81f5d5cbb	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	49b9131d-eee8-4ce0-bdb9-aff7486f8995	30,14830373
remote	27d1c7b9-4fa3-4daf-ae8f-110cee7b0954	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	042e885e-9572-4385-ae01-142a568f9efd	6,53035
remote	c9ec95e6-665f-417f-a623-74a81f5d5cbb	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	49b9131d-eee8-4ce0-bdb9-aff7486f8995	30,56726
remote	27d1c7b9-4fa3-4daf-ae8f-110cee7b0954	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	042e885e-9572-4385-ae01-142a568f9efd	6,47643
remote	c9ec95e6-665f-417f-a623-74a81f5d5cbb	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	49b9131d-eee8-4ce0-bdb9-aff7486f8995	30,47643
local	27d1c7b9-4fa3-4daf-ae8f-110cee7b0954	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	042e885e-9572-4385-ae01-142a568f9efd	3,916254706
local	c9ec95e6-665f-417f-a623-74a81f5d5cbb	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	49b9131d-eee8-4ce0-bdb9-aff7486f8995	18,08171235
local	c9ec95e6-665f-417f-a623-74a81f5d5cbb	7311644a-a47e-4b00-abf3-6047ca61576d	10	100	1	6	49b9131d-eee8-4ce0-bdb9-aff7486f8995	13,95093745

Abbildung 7: Beispieldatensatz zu einer Testinstanz

Ein Ergebnisdatensatz enthält folgende Felder:

Feldname	Inhalt	Beschreibung
Type	local / remote	Beschreibt, ob TargetValue über eine alleinige, lokale Optimierung oder über die Optimierung mit dem Mediator errechnet wurde
AgentConfig	GUID	Jede AgentConfig enthält eine eigene eindeutige GUID. Eine AgentConfig enthält alle notwendigen Regeln für die Zielfunktion eines Agenten
TestInstanzID	GUID	Jede Testinstanz (Beispieldatensatz) enthält eine eindeutige GUID.
MaxRounds	[0-9]+	Beschreibt die Anzahl der Abstimmungsrunden
VorschlaegeProRunde	[0-9]+	Beschreibt die Anzahl der pro Runde unterbreiteten L

Feldname	Inhalt	Beschreibung
ErzwungeneAkzeptanz	[0-9]+	Beschreibt die Anzahl der pro Runde erzwungenen Akzeptanzlösungen
AnzahlPlanningObjects	[0-9]+	Beschreibt die Anzahl der auf der Testinstanz (TestInstanzID) befindlichen Sendemasten (Planning Objects)
ClientID	GUID	Beschreibt die temporäre ClientID einer Session. Ist nach jedem Programmstart anders. (Für Debugging hilfreich)
TargetValue	[0-9]+ **‘.‘** [0-9]+	Beschreibt den Wert der Ergebnisfunktion, bedingt durch die aktuelle AgentConfig

Zur Bewertung wurde zuerst der Wert der Zielfunktion bei einer lokalen Optimierung für beide Agenten ermittelt. Diese Werte wurden anschließend mit den Ergebnissen der tatsächlichen Verhandlung verglichen. Analysiert wurde hierbei die prozentuale Abweichung des Zielwerts von der lokalen Optimierung.

Um bessere Ergebnisse zu erhalten, wurde die Optimierung für jede Instanz mehrmals durchgeführt und die Ergebnisse anschließend gemittelt. So können etwaige zufällige „Ausreißer“ neutralisiert werden.

5.2. Ergebnisse der Tests

Der Übersicht halber wird nun exemplarisch auf zwei Beispiele eingegangen. Die kompletten Ergebnisse können im GitHub-Projektverzeichnis eingesehen werden.

Um die Fairness der Optimierung für beide Agenten vergleichen zu können, werden im Folgenden die Abweichungen für beide Parteien gegenübergestellt.

Für die erste hier gezeigte Testinstanz wurde die Optimierung sowohl mit zehn, 100, sowie 1000 Verhandlungsrunden durchgeführt (vgl. Diagramm 1). Das Diagramm zeigt für eine Testinstanz die mittleren Abweichungen beider Agenten in Prozent von deren lokalen Optimum für die unterschiedlichen Verhandlungslängen.

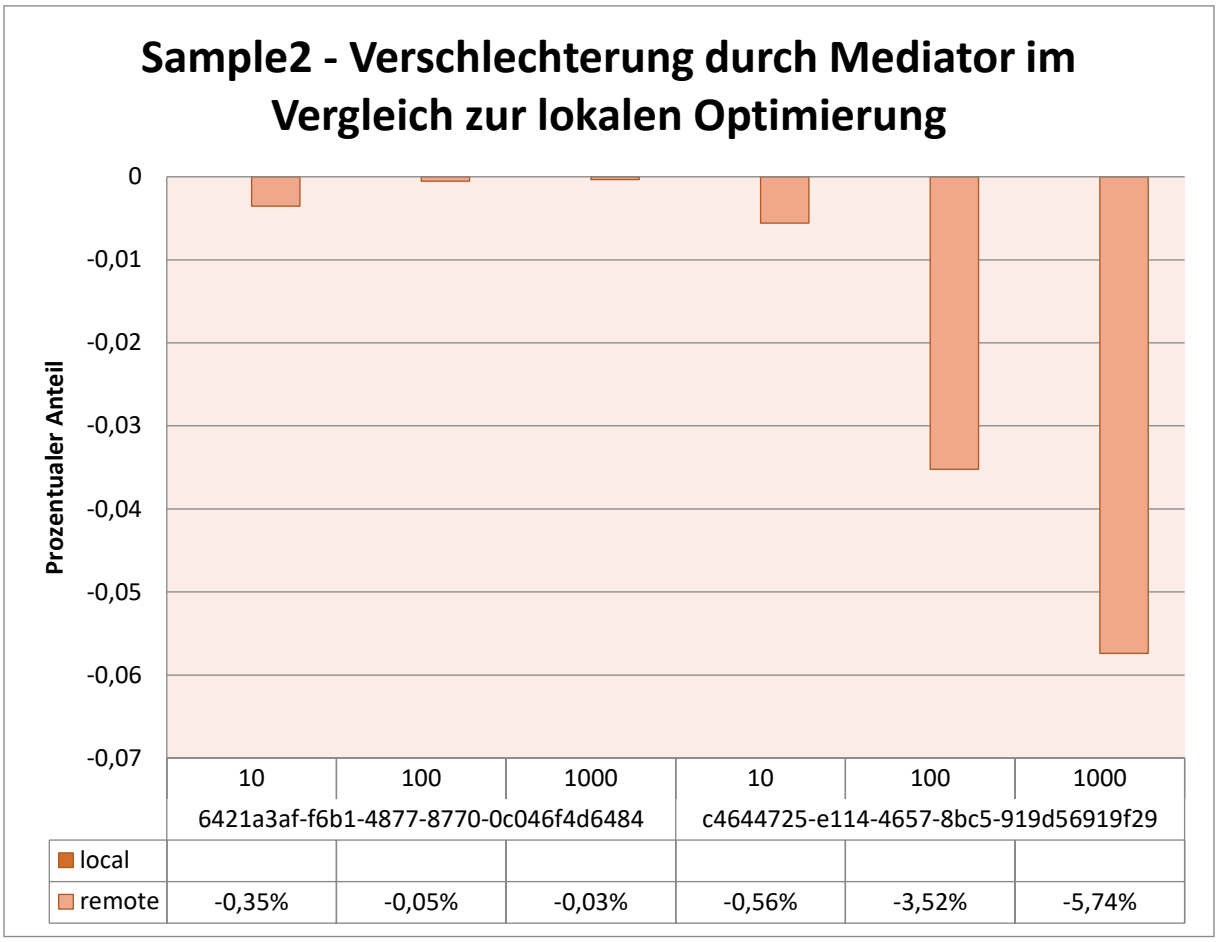

Diagramm 1: Vergleich der Agenten für eine Testinstanz bei variablen Verhandlungsrunden

Es zeigt sich, dass bei einer kurzen Verhandlung über zehn Runden beide Agenten ähnliche Abweichungen kleiner 1% von ihrem lokalen Optimum aufweisen. Bei längeren Verhandlungen gewinnt ein Agent die Oberhand und kann sich weiter verbessern, während der zweite Agent eine deutliche Verschlechterung hinnehmen muss.

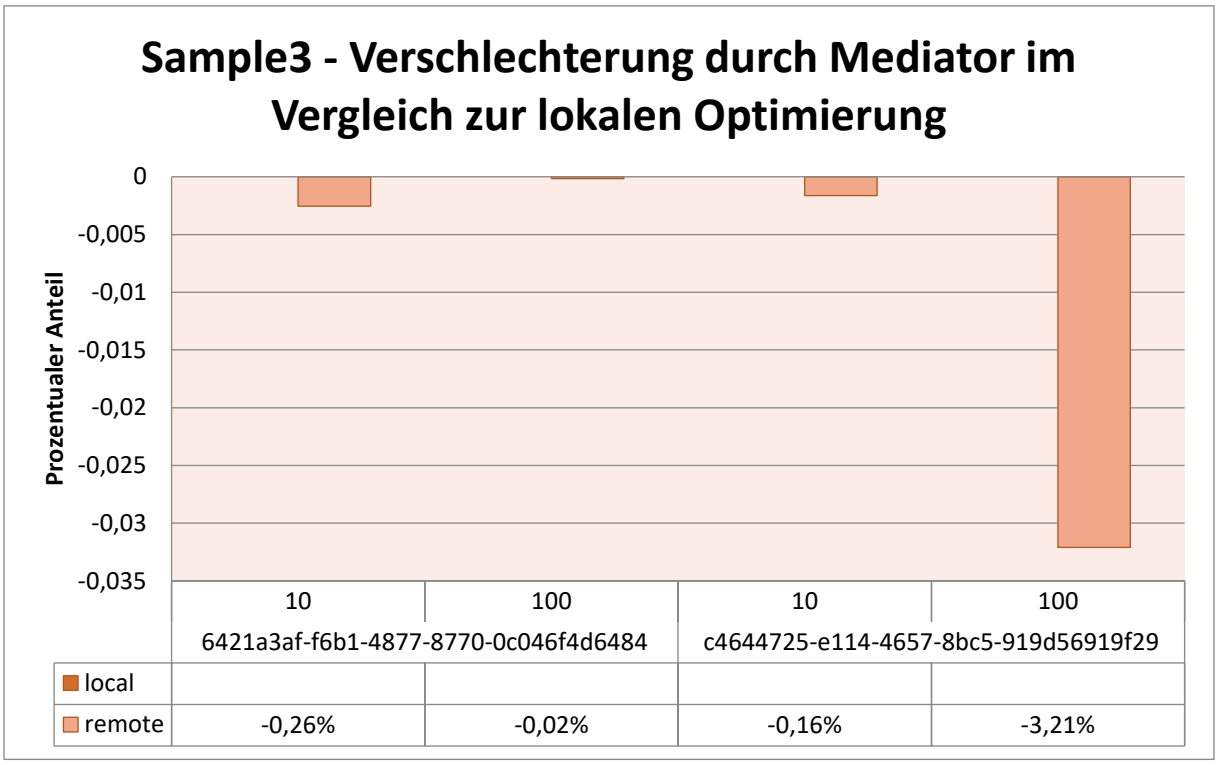

Diagramm 2: Vergleich der Agenten für eine Testinstanz bei variablen Verhandlungsrunden

Dasselbe Bild zeigt sich auch bei der zweiten betrachteten Testinstanz. Hier wurden nur zehn und 100 Verhandlungsrunden durchgeführt (vgl. Diagramm 2). Auch hier kann sich ein Agent bei der längeren Verhandlung durchsetzen und erzielt beinahe sein lokales Optimum. Der andere Agent muss hierfür von seinem lokalen Optimum stärker abweichen.

5.3. Bewertung der Ergebnisse

Es zeigt sich, dass bei längeren Verhandlungen ein Agent favorisiert wird und der Mediator sich mehr und mehr dessen lokalen Optimum annähert. Dies lässt sich eventuell damit erklären, dass der schlechtergestellte Agent viele kleine Verschlechterungen in Kauf nehmen muss. Die Abweichung ist mit -3,2% sowie -5,7% doch deutlich höher als bei der Opposition. Für einen produktiven Einsatz dieses Prototyps besteht an dieser Stelle noch Verbesserungspotential. Schließlich soll eine für beide Agenten faire Lösung gefunden werden.

Generell muss angemerkt werden, dass die durchgeführten Tests lediglich erste Anhaltspunkte bieten können. Um NEPO besser bewerten zu können, müssten weitere Tests durchgeführt werden. Sicherlich wäre es auch interessant, einen realen Fall nachzustellen und nicht nur mit zufallsgenerierten Daten zu experimentieren.

Durch die enorme Anpassbarkeit des Multiagentensystems an verschiedenste Szenarien und Optimierungsprobleme wäre ein weiteres mögliches Einsatzgebiet beispielweise auch die Positionierung von Windrädern. Diese allgemeine Herangehensweise birgt den Nachteil, dass der Prototyp auf kein explizites Szenario zugeschnitten wurde.

Die Tests des ersten Prototyps beweisen, dass NEPO durchaus in der Lage ist, das geschilderte Optimierungsproblem zufriedenstellend zu lösen. Hierbei sind alle Abweichungen vom lokalen Optimum für die Agenten nicht zu gravierend und könnten in der Praxis sicherlich akzeptiert werden. Es wird sich zeigen, ob in Zukunft für Problemstellungen dieser Art tatsächlich ein Multiagentensystem zu Rate gezogen werden wird. Da NEPO als ein Open Source-Projekt auf GitHub zur Verfügung steht, könnte in diesem Fall auf den Code und die ersten Erkenntnisse dieses Prototyps zurückgegriffen werden.

6. Projektverzeichnis und Aufgabenverteilung

6.1. Projektverzeichnis auf GitHub

Das Projekt NEPO steht also Open Source-Projekt auf GitHub allen interessierten Personen zur Verfügung. Neben dem Quellcode findet sich hier auch die komplette Dokumentation sowie die Konzeptions- und Testdokumente.

Hier der Link zum Verzeichnis:
https://github.com/tminf16/nepo

6.2. Aufgabenverteilung im Projektteam

Das Gruppenprojekt NEPO wurde von allen Teammitgliedern gemeinsam erdacht, konzipiert, umgesetzt und dokumentiert. Gewisse Schwerpunkte bei der Projektarbeit sind in der folgenden Tabelle festgehalten

	S. Ghizelea	S.Weisenberger	M. Stahlberger	P. Schulz
Konzeption	■	■	■	■
Framework	■			
Logik		■		
Dokumentation	■			■
Mediator			■	
GUI		■		■
Auswertung			■	■