**Dokumentation – NEPO**

# Inhalt

[1. Inhalt 1](#_Toc474316529)

[2. Problemstellung 2](#_Toc474316530)

[3. Konzeption 2](#_Toc474316531)

[3.1. Gegebene Parameter 2](#_Toc474316532)

[3.1. Klassendiagramm 3](#_Toc474316533)

[3.2. Aufbau der Agenten 3](#_Toc474316534)

[3.3. Programmablauf 5](#_Toc474316535)

[4. Implementierung 6](#_Toc474316536)

[4.1. Detaillierter Programmablauf 6](#_Toc474316537)

[4.2. Benutzeroberfläche 8](#_Toc474316538)

[4.3. Lokale Konfiguration 9](#_Toc474316539)

[4.4. Testdatengenerierung 9](#_Toc474316540)

[5. Test und Auswertung 9](#_Toc474316541)

[5.1. Evaluierungsmethodik 9](#_Toc474316542)

[5.2. Bewertung der Ergebnisse 9](#_Toc474316543)

[6. Aufgabenverteilung im Projektteam 9](#_Toc474316544)

# 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.

# Konzeption

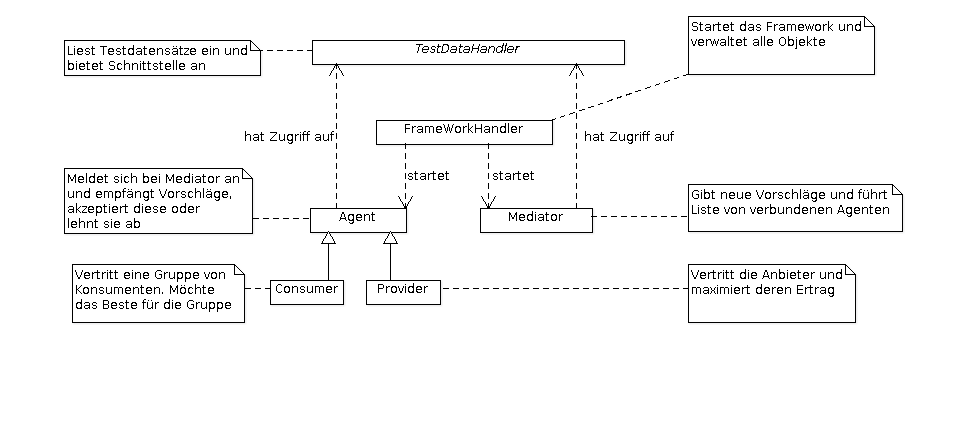
## Gegebene Parameter

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 |

## Klassendiagramm

Folgendes Klassendiagramm liegt dem entworfenen Multiagentensystem zu Grunde:



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.

## 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 ablehnbarer Lösungsvorschläge
* Maximale Anzahl an Verhandlungsrunden ohne Veränderung
* Maximale Anzahl an Verhandlungsrunden

## 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:

### 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 *Max* und dem minimalen Radius *Min* zu liegen.

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

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:

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:

## Programmablauf

Im Folgenden wird der Ablauf einer Verhandlung dargestellt.

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

1. UpdateList():  
   Die Agenten fragen den Mediator an und erhalten eine Vergleichslösung (die aktuell beste Lösung) sowie eine Liste mit einer Anzahl mutierten Lösungen
2. 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.
3. 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.
4. UpdateList():  
   Die Agenten fragen daraufhin erneut die Daten an und erhalten wiederum die beste Lösung sowie eine Liste mit Alternativlösungen. Eine neue Verhandlungsrunde beginnt.

# 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.

## 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.

## 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.



### 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.

## 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).

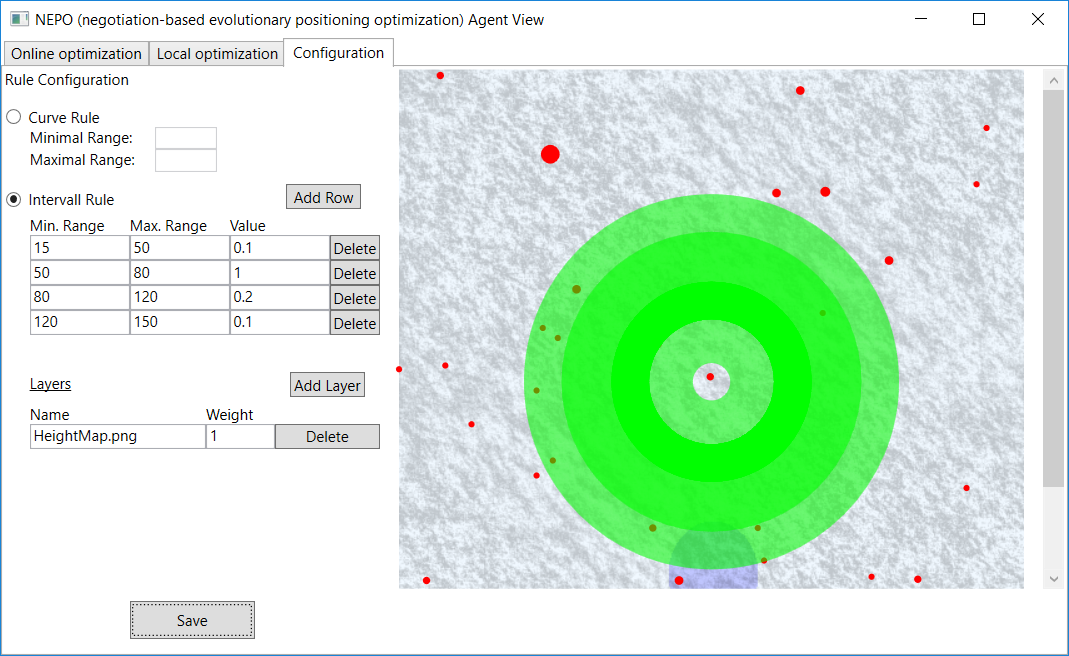


Abbildung : Grafische Darstellung verschiedener Intervalle mit einem Layer

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.

Auch während der Verhandlung kann der Agent mit Hilfe der Benutzeroberfläche die Optimierung beobachten.

## Testdatengenerierung

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

# Test und Auswertung

## Evaluierungsmethodik



**Beschreibung der Ergebnisdatei:**

Ein Ergebnisdatensatz enthält folgende Felder:

|  |  |  |
| --- | --- | --- |
| 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 Lösungen |
| ErzwungeneAkzeptanz | [0-9]+ | Beschreibt die Anzahl der pro Runde erzwungenen Akzeptanzlösungen |
| AnzahlPlanningObjects | [0-9]+ | Beschreibt die Anzahl der auf der Testinstanz (TestInstanzID) befindlichen |
| 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 |

## Bewertung der Ergebnisse

# Aufgabenverteilung im Projektteam

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