**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

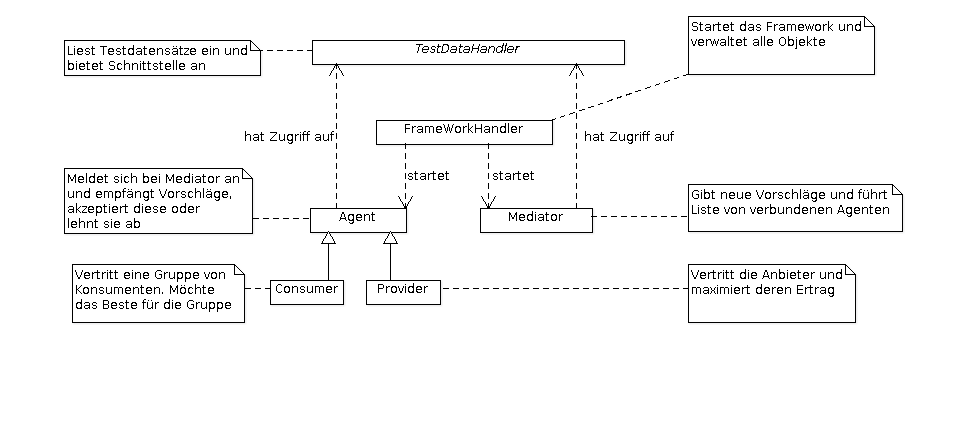
## Gegebene Parameter

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.

## 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:

## Verhandlungsprozess

## 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

→ Stefan

## 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. Anschließend wird überprüft, ob es seit einer vordefinierten Anzahl Runden keine neue beste Lösung gibt. Falls dies der Fall ist, ist die Verhandlung beendet. Ansonsten wird eine neue Runde gestartet. Die Entscheidung wird den Agents per Callback gemeldet.

Falls die Verhandlung nicht beendet ist, fragt der Agent beim Mediator die neuen Lösungsvorschläge an. Der Mediator sendet daraufhin als Antwort die aktuell beste Lösung sowie die generierten Lösungsvorschläge an die Agenten.

Jeder Agent überprüft die Vorschläge anhand der eigenen Zielfunktion und entscheidet über die Ablehnung oder Zustimmung jedes einzelnen Vorschlags. Jeder Agent muss einer vordefinierten Anzahl an Vorschlägen zustimmen. Nach der Validierung der Vorschläge wird die Entscheidung an den Mediator zurückgemeldet.

Nachdem alle Agenten Ihre Entscheidung an den Mediator übermittelt haben, überprüft dieser, ob es Kongruenzen bei den zugestimmten Lösungsvorschlägen gibt. Falls dies der Fall ist, wird die aktuell beste Lösung durch die neue beste Lösung ersetzt.

Nun beginnt eine neue Verhandlungsrunde mit der Mutation der aktuell besten Lösung und einer erneuten Abstimmung.

## 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.

## Lokale Konfiguration

## 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 |  |  |  |  |