Universität
Augsburg
University

Bachelorarbeit

# Nutzenbasierte Konfliktlösung für Batch Request und Single Request in SON


Autorin:
Name: Sabrina Einberger

Korrektur:
Prof. Dr. Bernhard Bauer
Prof. Dr. Bernhard Möller
Institut für Informatik
Universität Augsburg

Betreuer:
Christoph Frenzel


Abgabe:
24.09.2013

# Inhaltsverzeichnis

## Kurzzusammenfassung

***In Mobilfunknetzen werden zur Konfiguration, Optimierung und Fehlerbehebung verschiedene Parametern z.B. Antennenneigungswinkel, Downlink Sendeleistung in Self-Organising Networks (SON) Funktionen verwendet. Da mehrere Funktionen auf die gleichen Parameter zugreifen können und zum Teil gegensätzliche Ziele haben, stehen diese Aktionen im Konflikt miteinander. Mit Hilfe der SON Koordination werden diese Konflikte anhand der Nutzwerte, die jede Aktion besitzt aufgelöst. Die Batch Request (BR) Ansätze zur SON Konfliktlösung wählen aus jedem Konflikt die Aktion mit dem größten Nutzwert aus, und versuchen durch die Maximierung der einzelnen Konflikte den Gesamtnutzwert aller auszuführenden Aktionen zu optimieren. Durch diese isolierte Konfliktbetrachtung wird zwar der Nutzwert jedes einzelnen Konfliktes maximiert, aber nicht immer der Gesamtnutzwert aller auszuführenden Aktionen. Mit Hilfe der Constraint-Optimierung wird eine neue Methode zur SON Konfliktlösung ausgearbeitet, die den Gesamtnutzwert aller auszuführenden Aktionen optimiert. Ein anderer Ansatz zur Konfliktlösung ist die Single Request (SR) Methode, die ausführende SON Funktionen durch eine anfragende SON Funktionen unterbricht, wenn die anfragende Funktion mit den ausführenden Funktionen in Konflikt steht und einen höheren Nutzwert besitzt. Dabei wird nicht beachtet, dass der angegebene Nutzwert einer SON Funktion erst am Ende der Ausführungszeit erreicht wird. Die Konfliktlösung der SR Methode wird mit Hilfe der Barwertmethode verbessert, um einen höheren Gesamtnutzwert der ausgeführten SON Funktionen zu erzielen.***

# 1. Einleitung

## 1.1. Motivation

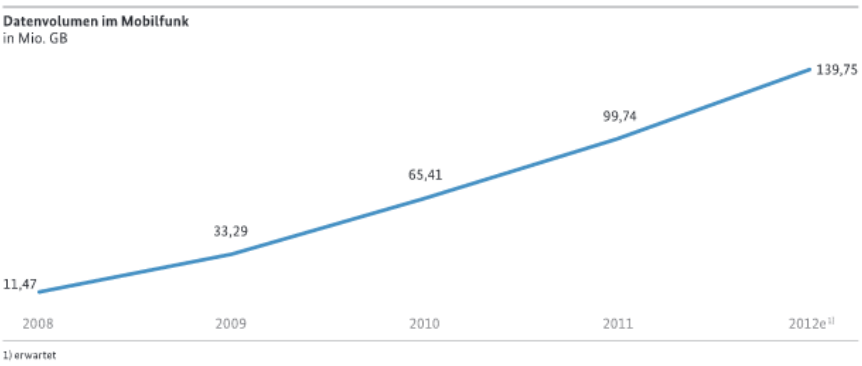


Abbildung 1.: Datenverkehr Deutschland [Bun13, S. 78]

Der Jahresbericht 2012 der Bundesnetzagentur berichtet über den Datenverkehr folgendes: „Der wachsende Datenverkehr im Mobilfunk ist für die Leistungsfähigkeit der Netze eine Herausforderung. 2012 wurden rund 140 Mio. GB übertragen. Würden die Nutzungsbedingungen von Fair-Flatrate-Tarifen Datenverkehr nicht eindämmen, wäre das Volumen noch viel größer. Immer mehr SIM-Karten werden in Endgeräten eingesetzt, um mobile Datenübertragungsdienste zu nutzen"[Bun13, S. 79].
Abbildung 1 der Bundesnetzagentur verdeutlicht, sowie die zunehmende Menge des Datenverkehrs in den letzten Jahren. Dieses hohe Datenaufkommen erfordert Mobilfunknetze, die immer komplexer werden und dadurch schwieriger zu verwalten sind. Self-Organising Networks (SON) ist ein Konzept, das durch die Einführung von SON-Funktionen in das Netzwerk, die Selbst-Konfiguration, Selbst-Optimierung und Selbstheilung aktiviert und dadurch die manuellen Einstellungen von Parametern z. B. dem Antennenneigungswinkel durch Operatoren reduziert [Net09, S. 2].
Verschiedene SON Funktionen stehen auf Grund der Nutzung gleicher Parameter in Konflikt miteinander. Jede Aktion besitzt einen Nutzwert $v$ der die Aktion gewichtet. Dieser Wert berechnet sich aus der Differenz zwischen dem positiven Nutzwert und den Kosten die eine Aktion verursacht. Die Batch Request (BR) und Rational Policy System (RPS) Konfliktlösungsansätze bestehen darin, bei einem Konflikt zwischen zwei Aktionen, die Aktion mit dem größten Nutzwert $v$ auszuführen und die andere Aktion zu verwerfen. Es wird versucht den Gesamtnutzwert aller Aktionen zu maximieren, indem die einzelnen Konflikte isoliert voneinander betrachtet werden und jeweils die Aktion mit dem

größten Nutzwert $v$ auswählt wird. Am Ende der Konfliktlösung werden alle Nutzwerte der ausgewählten Aktionen summiert und daraus ergibt sich der Gesamtnutzwert, den es gilt zu optimieren [Chr13], [Rap13].

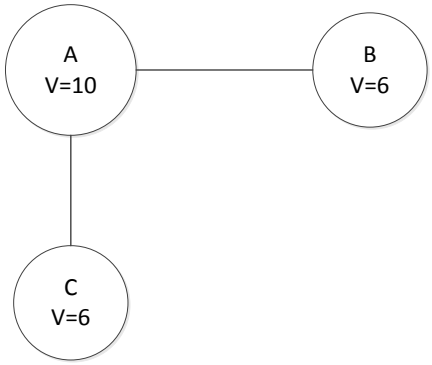


Abbildung 2.: ABC Konfliktbeispiel

Abbildung 2 skizziert einen Beispielkonflikt namens ABC Konfliktbeispiel zwischen den Aktionen A, B und C. Die Verbindungslinien stehen für eine Konfliktrelation zwischen den Aktionen. Somit steht die Aktion A mit dem Nutzwert $v(A) = 10$ mit den Aktionen B und C mit jeweils dem Nutzwert $v(B) = v(C) = 6$ in Konflikt. Die Konfliktlösungsansätzen betrachten den Konflikt zwischen der Aktion A und der Aktion B und die Aktion A mit dem größeren Nutzwert $v(A)$=10 wird ausgewählt. Danach wird der Konflikt zwischen der Aktion A und der Aktion C betrachtet und die Aktion mit dem größten Nutzwert nämlich A wird ausgewählt. Der Gesamtwert aller ausgewählten Aktionen ist somit $v_{\text{gesamt}} = v(A) = 10$ [Chr13].
Die Herausforderung besteht darin eine neue Konfliktlösungsmethode zu entwickeln, die den Gesamtnutzwert weiter maximiert, d. h. einen höheren Gesamtnutzwert erzielt als die BR und RPS Konfliktlösungsansätze.

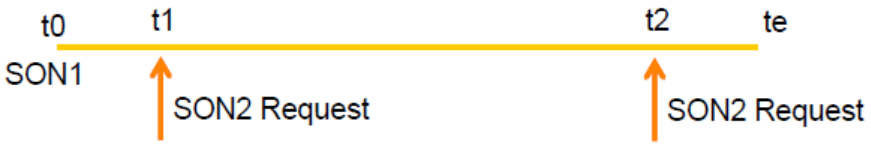


Abbildung 3.: Single Request Konfliktdarstellung [Fre13]

Die Single Request (SR) Methode behandelt Konflikte zwischen der Anfrage einer SON Funktion und einer oder mehrere nicht in Konflikt stehender ausführender SON Funktionen. Falls eine laufende Funktionen mit der anfragenden Funktion in Konflikt steht, wird der Nutzwert der laufenden Funktion mit dem Nutzwert der anfragenden Funktion verglichen. Falls die anfragende Funktion den größeren Nutzwert besitzt, wird diese ausgeführt und die laufende Funktion unterbrochen. Die laufende Funktion wird nicht

unterbrochen, wenn der Nutzwerte der anfragenden Funktion kleiner dem Nutzwert der laufenden Funktion ist. Zusätzlich wird die Anfrage der SON Funktion verworfen. Diese Methode der Konfliktlösung trifft die Auswahl der Funktion auf Grund der Nutzwerte, die erst am Ende der Laufzeit erzielt werden [Fre13].
In Abbildung 3 wird die SON1 Funktion ausgeführt und eine Anfrage der SON2 Funktion zum Ausführen wird zum Zeitpunkt $t_1$ und zum Zeitpunkt $t_2$ gestellt. Die SR Konfliktlösungsmethode vergleicht die Nutzwerte der SON1 und SON2 Funktionen miteinander. Falls die anfragend SON2 Funktion einen höheren Nutzwert als die laufende SON1 Funktion besitzt, wird unabhängig zu welchem Zeitpunkt $t_1$ oder $t_2$ die Anfrage gestellt wird, die SON1 Funktion abgebrochen. Auch kurz vor Ende der Laufzeit $t_2$ wird die SON1 Funktion abgebrochen [Fre13].
Der angegebene Nutzwert einer Funktion, wird erst am Ende der Ausführungszeit erzielt, d. h. der Nutzwert einer Funktion die gerade am Anfang ihrer Ausführungszeit steht, ist um einiges geringer als der Nutzwert, der am Ende der Laufzeit erzielt wird. Somit erhöht sich der Nutzwert einer Aktion während der Ausführungszeit [Fre13].

## 1.2. Ziel

Ziel ist es, eine neue Konfliktlösung der BR Methode zu entwickeln, die den Gesamtnutzwert der auszuführenden Aktionen optimieren. Dabei müssen die Konflikte zwischen den Aktionen berücksichtigt werden und die auszuführenden Aktionen so gewählt werden, dass es keine Konflikte mehr gibt. Die Summe der Nutzwerte der ausgewählten Aktionen ergibt den Gesamtnutzwert, der maximiert wird. Mit Hilfe eines Coinstraint-Lösers, wird die neue Konfliktlösung der BR Methode implementiert.
Ein weiteres Ziel ist es die Konfliktlösung der SR Methode zu verbessern, indem die Nutzwerte der laufenden Funktionen, mit Hilfe der Barwertberechnung zu einem bestimmten Zeitpunkt korrekt wiedergegeben werden können. Dadurch werden die Nutzwerte laufender und anfragende Funktionen besser verglichen und Funktionsabbrüche kurz vor dem Laufzeitende einer Funktion werden minimiert. Durch die Verringerung der Funktionsabbrüche soll ein höherer Gesamtnutzwert der ausgeführten Funktionen erzielt werden.

## 1.3. Herangehensweise

Zu Beginn, werden die aktuellen Konfliktlösungen der BR und RPS Methoden betrachtet, um herauszufinden worin eine Verbesserung des Gesamtnutzwertes erzielen werden kann. Im nächsten Schritt wird erkennt, dass es sich bei der Konfliktlösung für die BR Methode um ein Constraint-Optimierungsproblem handelt, weil die Konflikte der Aktionen sich mit Hilfe von Constraints formulieren lassen. Anschließend wird eine neue Methode zur Berechnung des Gesamtnutzwertes erstellt und die nötigen Bedingungen hierfür formuliert. Danach folgt die Implementierung der BR Konfliktlösung mit dem Constraint-Löser google-or-tools. Die Evaluation verdeutlicht die Unterschie-

de zwischen den BR Konfliktlösungsansätzen ohne Constraint-Optimierung und der BR Konfliktlösung mit Constraint-Optimierung.
Als nächstes wird die Konfliktlösung der SR Methode betrachtet, um eine neue Methode zur Berechnung des Nutzwertes einer Funktion zu verschiedenen Zeitpunkten der Ausführungszeit zu finden. Mit Hilfe der Barwertberechnung kann der "Zeitwert" des Nutzwertes einer Funktion berechnet werden. Dadurch können die Nutzwerte einer laufenden und einer anfragenden SON Funktion besser verglichen werden. Die Konfliktlösungen mit und ohne Barwertmethode werden implementiert und durch eine Evaluation die Ergebnisse verglichen.

# 2. SON Grundlagen

## 2.1. SON Definition

SON wird im Zusammenhang mit Mobilfunknetzen gebraucht, in denen die Konfiguration, Operation und Optimierung zum größten Teil automatisiert wird. Zwei wichtige Punkte, die für die Automatisierung von Mobilfunknetzen spricht, sind zum einen die Benutzerfreundlichkeit und zum anderen durch gesunkene Kosten die höhere Wirtschaftlichkeit. Die Inbetriebnahme und Installation von neuen Netzelemente (NE) soll schneller und kostengünstiger sein [Net09, S. 3, 4]. Es gibt drei wichtige Funktionen die SON auszeichnen [Net09, S. 5]:

- Selbst-Konfiguration: Selbst-Konfiguration umfasst alle nötigen Aufgabe um die Inbetriebnahme von neuen Netzen zu automatisieren und Parametern zu konfigurieren. Die NE arbeiten autonom, führen Setup-Routinen aus, authentifizieren und verbinden sich mit dem Operations Support System (OSS).
- Selbst-Optimierung: Selbst-Optimierung dient der Verbesserung der Netzqualität durch Abstimmung der Netzparameter. Eine der Hauptaufgaben ist die Datenlastverteilung zwischen benachbarten Zellen. Datenlastverteilung zwischen zwei Mobilfunkzellen bedeutet, dass eine überlastete Zelle einen Teil seiner Datenlast an eine Nachbarzelle übergeben kann.
  Zusätzlich leistet SON mit der Energiesparfunktion einen Beitrag zum Umweltschutz.
- Selbst-Heilung: Die Selbst-Heilung bietet bei einem Serviceausfall die Erkennung eines solchen, und eine Analyse der Ursachen und Mechanismen, die den Netzbetrieb wiederherstellen sollen.
  [Net09, S. 5]

Dank Smartphones, tragbarer Computer etc. ist es uns möglich zu jeder Zeit und überall das Internet zu nutzen. Abbildung 4 verdeutlicht wie stark der Datenaustausch von Paketen in den letzten Jahren gestiegen ist und in Zukunft noch weiter steigen wird. Die Menge der Sprachdaten ist seit Jahren gleich geblieben und im Vergleich zum Paketverkehr sehr gering. Durch den enormen Anstieg des Datenverkehrs, kann die Bandbreite stark schwanken. Dies hat zur Folge, dass die Datenverkehrslast schwierig vorhersehbar ist. Bei Spitzen in der Datenverkehrslast liegt die Herausforderungen in der gerechten Zuordnung der Bandbreite zu jedem Benutzer, der Optimierung der Lastverteilung unter den Zellen und die zuverlässige Übergabe von Parametern zu anderen Zellen. SON reagiert schnell auf diese Anforderungen und minimiert dadurch Ausfallzeiten [Net09, S. 6].

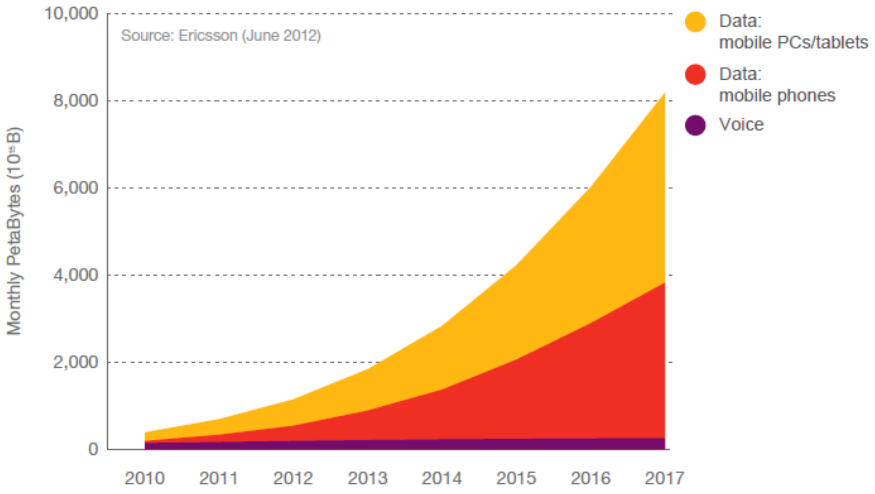


Abbildung 4.: Globaler Datenverkehr [Eri12]

## 2.2. SON Funktionen

Damit die Selbst-Optimierung automatisiert werden kann, werden viele verschiedene SON Funktionen benötigt, die auf verschiedene Parameter zugreifen und diese auslesen oder verändern können. Eine dieser Funktionen ist die Coverage and Capacity Optimisation (CCO) Funktion. Diese bestimmt die flächenmäßige Abdeckung und Kapazität des Netzes in einer bestimmten Mobilfunkzelle, die je nach Bedarf verändern werden kann. Einige Parameter dieser Funktion werden in der Abbildung 5 dargestellt. Es werden der Antennenneigungswinkel und die Downlink Transmission (Tx) Power (Sendeleistung) von der CCO Funktion gesetzt. Je größer der Antennenneigungswinkel gewählt wird, desto kleiner ist die Fläche, die durch diese Mobilfunkzelle abgedeckt wird und desto größer ist die Sendeleistung innerhalb dieser Fläche (siehe Abbildung 6). Dadurch, dass durch den Antennenneigungswinkel die Sendeleistung beeinflusst wird, sind die Parameter Downlink Tx Power und Antennenneigungswinkel eng miteinander verbunden und beeinflussen sich gegenseitig. Auf die Handover (HO) Parameter hat die CCO Funktion nur einen indirekten Einfluss. Als HO wird die Verbindungsübergabe bezeichnet, d. h. wenn ein mobiles Endgerät während einer „Datenverbindung ohne Unterbrechung dieser Verbindung von einer Funkzelle in eine andere wechselt“ [Wik13c]. Für diesen HO Vorgang werden sogenannten HO Parameter benötigt, die einen reibungslosen Vorgang gewährleisten. Die Funktion Mobility Robustness Optimisation (MRO) ist u. a. dafür zuständig neue HO Parameter einer Mobilfunkzelle zu berechnen. Die Energiesparfunktion versucht durch das An- und Ausschalten der Mobilfunkzellen oder durch das Heruntersetzen der Sendeleistung Energie zu sparen [Tob12, S. 329, 330, 332].

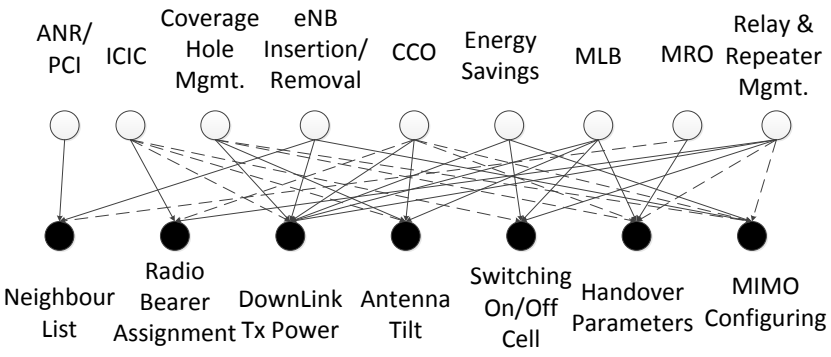


Abbildung 5.: SON Funktionen und Parameter [Tob12, S. 330]


Abbildung 6.: Neigunswinkel einer Mobilfunkantenne [Inf11]

## 2.3. SON Koordination

### 2.3.1. SON Koordination Definition

Auf Grund der vielen verschiedenen SON Funktionen und der gemeinsamen Parameter, auf die die verschiedenen Funktionen zugreifen, entstehen Konflikte. Es gibt zwei grundlegende Ansätze diese Konflikte zu erkennen und zu lösen. Der Co-Design Ansatz der SON Funktionen, erkennt und löst Konflikte während der ‘Design-Zeit‘ d. h. (wenn SON Funktionen entwickelt oder Policies erstellt werden) [Tob12, S. 334].
Die SON Funktionskoordination ist die Konfliktlösung zur Laufzeit und somit der logische nächste Schritt, wenn die Co-Design Konfliktlösung kein konfliktfreies SON System erreicht. Die Funktionskoordination ermöglicht das automatische Erkennen und Lösen von Konflikten und bietet dem Operator die Möglichkeit das Verhalten der SON Funktionen zu steuern [Tob12, S. 338].

### 2.3.2. SON Konflikte

Abbildung 5 verdeutlicht, dass viele verschiedene SON Funktionen auf dieselben Parameter entweder direkt oder indirekt zugreifen. Dadurch entstehen Konflikte zwischen den Funktionen. Somit entsteht beispielsweise ein Konflikt zwischen der Energiesparfunktion und der CCO Funktion, und der MRO und der CCO Funktion (Siehe Abbildung 7). Die Funktionen haben z. T. gegensätzliche Ziele z. B. fordert die Energiesparfunktion eine möglichst geringe Sendeleistung um dadurch weniger Energie zu verbrauchen, wohingegen die CCO Funktion die optimale flächenmäßige Abdeckung u. a. durch das Erhöhen der Sendeleistung als Ziel hat. Es entsteht ein direkter Konflikt zwischen der CCO Funktion und der Energiesparfunktion. Ein weiterer Konflikt entsteht zwischen der MRO Funktion und der CCO Funktion durch die HO Parameter. Die Veränderung des Neigungswinkels einer Antenne hat einen Einfluss auf die Größe der Funkzelle. SON Funktionen haben eine Verzögerungszeit bei der Sichtbarkeit von veränderten Parametern. Deshalb wird eine Veränderung des Antennenneigungswinkels durch die CCO Funktion erst nach einer Verzögerungszeit sichtbar. Wenn innerhalb dieser Verzögerungszeit die MRO Funktion getriggert wird, dann werden die HO Parameter auf Basis der nicht mehr aktuellen Messungen berechnet [Tob12, S. 332].

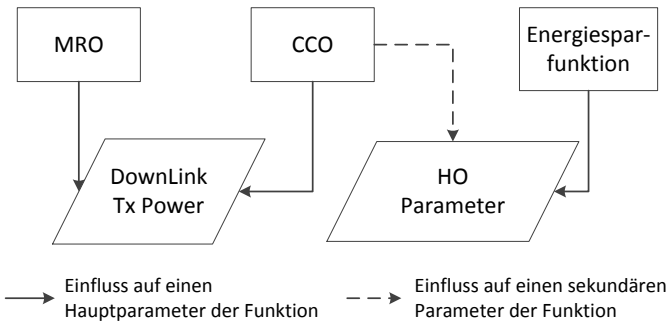


Abbildung 7.: Beispiel SON Funktionskonflikte

### 2.3.3. Rational Policy System

Das Managen von Mobilfunknetzen ist auf Grund seiner Komplexität eine ständige Herausforderung. Diese Komplexität entsteht unter anderem durch die hohe Anzahl an NE, die verteilt und organisiert werden müssen [Ric12, S. 39].
Für die Konfiguration dieser NE ist das Netzmanagement (NM) zuständig. Die Aufgaben des NM werden im FCAPS Modell der ISO für Netzmanagement definiert [Wik13b]. Aus folgenden Punkten setzt sich das FCAPS zusammen:

- Fault Management (FM): Das FM ist zuständig für die Erkennung und Behebung von Fehlern im Netz z. B. der Ausfall eines NE.
- Configuration Management (CM): Das CM ermöglicht die Vereinfachung bei der Konfiguration von NE.

- Accounting Management (AM): Das AM erstellt Statistiken für User über die Nutzung der Netzressourcen. Netzressourcen sind logische und physikalische Einheiten z. B. Bandbreite, die zusammen eine Netzverbindung ausmachen [Wik13e].
- Performance Management (PM): Das PM analysiert Leistungsdaten, z. B. die Sendeleistung in einer Mobilfunkzelle.
- Security Management (SM): Das SM steuert den Zugriff auf die Daten des Netzes [Wik13b].

Bei auftretenden Problemen wird das Netzmanagement System (NMS) mit Hilfe von Events informiert. Zum Überprüfen der NE werden Sonden eingesetzt, die z. B. Performance Daten auswerten und ein Event z. B. zu geringer Sendeleistung erzeugt, das das NMS über das Vorkommnis informiert. Wie auf diese Events reagiert wird, hängt von dem NMS ab [Chr13].
Das Policy-based Network Management (PBNM) stellt eine Policy, als ein Set aus Regeln dar. Diese Regeln werden von den Betreibern auf Grund von genauen Zielvorstellungen definiert und triggern Aktionen und bestimmen somit welche Aktion ausgeführt werden muss um das aufgetretene Problem nach den Zielvorstellungen des Betreibers zu lösen. Das PBNM verwendet die Event-Condition-Action (ECA) Policy Regeln, d. h. auf ein Event folgt eine Aktion. Z. B. wird das Problem der zu geringen Sendeleistung innerhalb einer Mobilfunkzelle durch das Event, zu geringe Sendeleistung, dem NMS mitgeteilt. Daraufhin wird mit Hilfe der Policy Regeln eine Aktion auf Basis der Zielvorstellungen z. B. hohe Netzqualität, ausgewählt. Es wird ein Operator benötigt, der zwischen den Events und den getriggerten Aktionen "übersetzt", d. h. festlegt, mit welcher Aktion die Zielvorstellungen erreicht werden kann. Das Technischen Wissen, d. h. welche Aktion auf welches Event folgt und die Zielvorstellungen des Operators bilden gemeinsam die Policy Regeln. Auf Grund dieser engen Verflechtung ist das Ändern der Zielvorstellungen eine kostspielige Angelegenheit, da alle Regeln überprüft und evtl. angepasst werden müssen [Chr13].
Ein anderes Policy System ist das Utility Function (UF) basierende Policy System. Dieses System wählt die Aktionen, die den Zielvorstellungen des Operators entsprechen, automatisch aus. Die Bildung der Policy ist sehr komplex, auf Grund der vielen verschiedenen Faktoren z. B. den Voraussetzungen und den Auswirkungen von Aktionen. Zusätzlich wird ein Nutzwert berechnet, der sich aus dem Systemkontext zusammensetzt. Auf Grund der vielen verschiedenen Faktoren, die für das Triggern von Aktionen verantwortlich sind, ist eine genaue Bestimmung der Faktoren eine grundlegende Voraussetzung, damit das gewünschte Ergebnis erzielt werden kann. Bei komplexen Systemen ist die Erstellung der Policy deshalb sehr aufwendig [Chr13].
Das RPS ist ein Policy System, das ECA Regeln mit UFs kombiniert. Das RPS erweitert das ECA-basierende Policy System mit UF-basierender Konfliktlösung, die die rationalste Aktion bei einem Konflikt auswählt. Im RPS geben die ECA Policy Regeln nur an, welche Aktion auf welches Event folgen soll. Diese Auswahl wird nicht auf Grund von Zielvorstellungen des Betreibers getroffen, sondern nur auf Grund des technischen Wissens über das Netzwerk. Z. B. erfolgt die Aktion, CCO Funktion ausführen, auf Grund

des Events der geringen Sendeleistung unabhängig von der Zielvorstellung einer hohen Netzqualität oder eines möglichst kostengünstigen Netzbetriebes. Die Zielvorstellungen des Netzbetreibers sind in den UFs enthalten. Der Nutzwert einer Aktion hängt von vier Faktoren ab [Chr13].

1 Aktionskosten: In den Aktionskosten fließt die Bevorzugung einer Aktion durch den Operator mit ein.

2 Effektivität: Die Effektivität einer Aktion bezieht sich auf die Wahrscheinlichkeit, dass die Aktion den gewünschten Effekt bezüglich eines Events erzielt.

3 Positive Nutzen: Der positive Nutzen eines Events ist ein Maß dafür, wie wichtig es dem Operator ist, auf ein Event zu reagieren, d. h. eine Aktion zu triggern. Die Zielvorstellung des Operators sind ebenfalls im positiven Nutzen des Events enthalten.

4 Wahrscheinlichkeit: Die Wahrscheinlichkeit eines Events gibt an, wie wahrscheinlich es ist, dass ein Event auftritt.

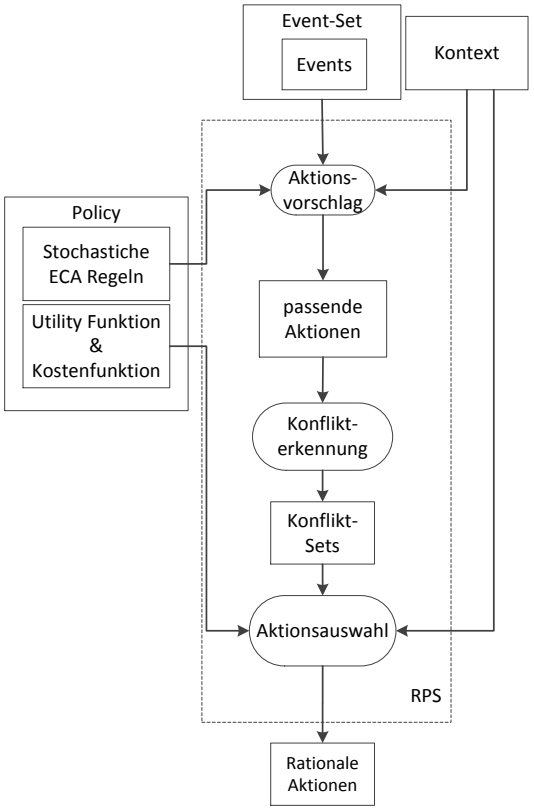


Abbildung 8.: Das Rational Policy System [Chr13]

In Abbildung 8 wird der Prozess des RPSs dargestellt. Am Anfang des Prozesses treffen Events, z. B. der Ausfall eines NE X ein, die in Event-Sets zusammengefasst werden. Danach folgen die drei Hauptkomponenten des RPS [Chr13]:

1. Aktionsvorschlag (AV): Auf die eingegangenen Events muss reagiert werden, um die aufgetretenen Fehler oder die nicht ausreichende Performance auszugleichen. Die Reaktion auf diese Events sind Aktionen, die getriggert werden. Somit wird auf jedes Event eine Aktion vorgeschlagen. Z. B. wird auf Grund des Events, Ausfall eines NE X, die Aktion, Datenlastverteilung vorgeschlagen. Um die Aktionsvorschläge zu bilden, werden sogenannten stochastischen ECA Regeln verwendet. Die ECA Regeln haben folgende Form:

   ON event IF condition THEN action WITH effectiveness (1)

   In dieser Policy ist das **event** das eingegangene Ereignis. **Condition**, ist ein logischer Ausdruck über den Netzkontext des events. **Action** ist die Aktion, die vorgeschlagen wird um das Problem zu lösen. **Effectiveness** drückt die Wahrscheinlichkeit aus, dass die vorgeschlagene Aktion das Problem löst. Das Ergebnis des AV sind die passenden Aktionen, zu den angegebenen Events.

2. Konflikterkennung (KE): Die Aktionen die auf die verschiedenen Events vorgeschlagen wurden, stehen mit großer Wahrscheinlichkeit im Konflikt miteinander. Ein Beispiel ist der Energiesparmodus für das NE Y und dieses NE Y soll gleichzeitig die Datenmenge eines anderen NE X übernehmen. Das Vorschlagen solcher gegensätzlicher Aktionen führt zu Konflikten. Aus den Aktionen, die in Konflikt miteinander stehen, werden Konflikt Sets gebildet. In einem Konflikt-Set steht jede Aktion mit jeder anderen Aktion des Sets im Konflikt. Z. B. entsteht ein Konflikt-Set aus der Aktion Energiesparmodus für NE Y und Datenlastverteilung durch NE Y.

3. Aktionsauswahl (AA): Die Konflikte, die in der vorherigen KE Komponente erkannt werden, müssen aufgelöst werden und die daraus resultierenden Aktionen werden danach ausgeführt. Für jedes Konflikt-Set entscheidet die AA, auf Grund der Kostenfunktion und der Nutzwertfunktion welche Aktion ausgewählt wird. Je niedriger die Kosten einer Aktion sind und je höher der Nutzwert eines Events ist, desto besser.
   Der erwartete Nutzen einer Aktion $U_{exp}$ ist die Summe der positiven Nutzwerte $U$, aller behandelten Events $E$, gewichtet mit der Event-Wahrscheinlichkeit $P$ und der Effektivität der Aktion $P_{eff}$ .

$$U_{exp} = \sum_{e \in E} P_e * P_{eff} * U_e \tag{2}$$

Die Funktion v(x) berechnet die Differenz zwischen dem erwarteten Nutzen $U_{exp}$ einer Aktion und den Kosten $C$ einer Aktion. Das Ergebnis liefert den Nutzwert $v(X)$ einer Aktion X.

$$v(X) = U_{exp} - C \tag{3}$$

Die Aktion mit dem höheren Nutzwert wird aus dem Konflikt-Set ausgewählt und anschließend ausgeführt. Die Aktion Energiesparmodus erhält z. B. den Nutzwert $v$(Energiesparmodus) = 6 und die Aktion Datenlastverteilung den Nutzwert $v$(Datenlastverteilung) = 8. Die Aktion mit dem größeren Nutzwert, nämlich Datenlastverteilung, wird aus dem Konflikt-Set ausgewählt.
[Chr13]

### 2.3.4. Batch Request Methode

#### 2.3.4.1. Unterschied zwischen Priorität und Nutzwert einer Aktion

Die BR Methode verwendet zur Bewertung der Aktionen Prioritäten. Das RPS bewertet die Aktionen mit Nutzwerten. Ein Vergleich der beiden Konfliktlösungen mit unterschiedlichen Bewertungen der Aktionen ist nicht möglich. Die Präferenzrelation beschreibt die Beziehung zwischen Elemente die mit Prioritäten bewertet werden und wird wie folgt definiert:

**Definition 1.** *Eine Präferenzrelation über ein set $\Omega$ ist eine transitive binäre Relation $\succeq$ über $\Omega$. Wenn für jedes Element $o$, $o' \in \Omega$, entweder $o \succeq o'$ oder $o' \succeq o$ gilt, so handelt es sich um eine totale Ordnung [Ron09, S. 59].*

Die BR Methode verfügt über Aktionen, die Prioritäten besitzen und über diese Prioritäten lassen sich die Aktionen in eine Rangordnung sortieren. Der höchste Rang ist der Rang 1 mit die höchsten Priorität $p$ =1. Ausgehend davon, dass der höchste Priorität zu bevorzugen ist, kann z. B. über die Beziehung zwischen einer Aktion X mit Priorität $p(X) = 1$ und einer Aktion Y mit der Priorität $p(Y) = 2$ nur die Aussage getroffen werden, dass die Aktion X einen höheren Rang als die Aktion Y hat und deshalb zu bevorzugen ist. Die Aussage, dass die Aktion Y nur halb so "wichtig" ist, wie die Aktion X, kann nicht getroffen werden, weil es sich bei der Präferenz Relation um eine total Ordnung handelt und auf Grund dessen kann keine Angabe über das genaue Maß der Differenz zwischen den beiden Aktionen gemacht werden [Wik13d], [Ron09, S. 59].
Das RPS verwendet zur Bewertung der Aktionen den Nutzwert $v$, der mit Hilfe der UF berechnet wird. Die UF wird bei Entscheidungsproblemen, die Hilfe einer Entscheidungsanalyse benötigen, eingesetzt. In Abschnitt 2.3.3 ist erläutert worden, dass der Nutzwert viele verschiedene Faktoren enthält, die eine Aktion bewerten. Somit spiegelt der Nutzwert nicht nur die Bevorzugung einer Aktion wieder, sondern gibt einen genauen Wert bezüglich verschiedener Entscheidungskriterien an, mit Hilfe derer ein Vergleich zwischen Aktionen getroffen werden kann und ein Maß der Differenz zwischen verschiedenen Aktionen angegeben werden kann. Z. B. besitzt die Aktion X den Nutzwert $v(X) = 10$ und die Aktion Y den Nutzwert $v(Y) = 5$. Der Vergleich der beiden Aktionen ergibt eine

Bevorzugung der Aktion X gegenüber der Aktion Y und ein genaues Maß der Differenz $v(X) = 2 * v(Y)$ lässt sich ebenfalls angeben. Je größer der Nutzwert einer Aktion ist, desto besser. Die Aktion X besitzt einen doppelt so großen Nutzwert, wie die Aktion Y [Wik13a], [Ron09, S. 74, 75].
Um den Gesamtnutzwert der verschiedenen Konfliktlösungen berechnen und vergleichen zu können ist es notwendig, die Aktionen mit Nutzwerten zu bewerten. Deshalb werden die Aktionen in der BR Methode mit Nutzwerten bewertet.

#### 2.3.4.2. Koordination der Batch Request Methode

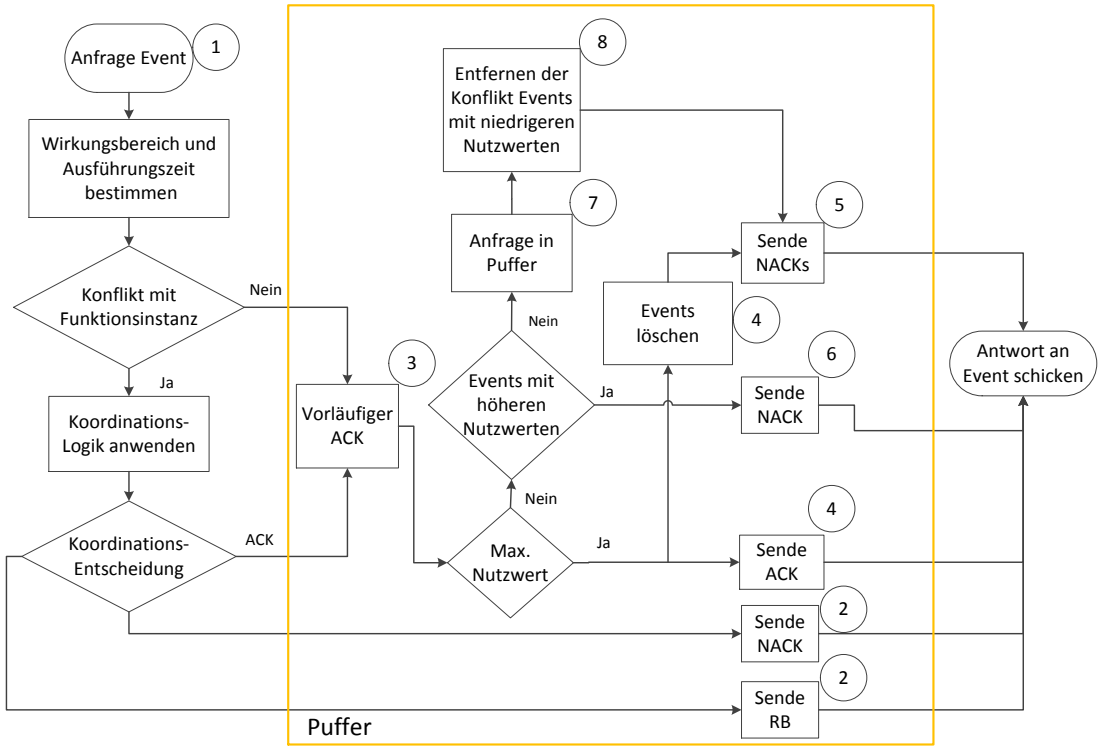


Abbildung 9.: Batch Request Koordination mit Puffer [Rap13]

Jede SON Funktion besitzt einen Funktionsbereich, der angibt, welche Netzressourcen z. B. Antennenneigungswinkel von dieser Funktion manipulieren werden können. Eine SON Funktions-Instanz ist ein Laufzeitprozess, der eine SON Funktion in z. B. einer Netzumgebung instanziiert. Diese SON Funktionsinstanzen besitzen die Fähigkeit Funktionen zu überwachen (diese sind aktiv, wenn die SON Funktion ausgeführt wird), Daten zu sammeln und zu analysieren und Aktionen auszuführen [Tob12, S. 323]. Im Gegensatz zu herkömmlichen Netzoperationen, ist die Ausführung von SON Funktions-Instanzen individuell und dynamisch. Aus diesem Grund ist es möglich, dass Konflikte zwischen SON Funktions-Instanzen während der Laufzeit auftreten [Rap13].

Eine Option der dynamischen Erkennung und Vermeidung von Konflikten ist die Vor-Aktions-Koordination. Diese koordiniert die Ausführung von SON Funktions-Instanzen. Hierfür müssen für jede SON Funktion folgende Informationen festgelegt werden [Rap13]:

- Koordinationslogik: Die Koordinationslogik drückt die Konfliktlösungsstrategie des Operators aus, z. B. wird ein Nutzwert angegeben, der ausdrückt welche Funktion bei einem Konflikt zu bevorzugen ist. Der ursprüngliche Begriff Priorität ist durch den Nutzwert ersetzt worden 2.3.4.1.

- Wirkungsbereich: Der Wirkungsbereich einer Funktion umfasst den Funktionsbereich und die Netzressourcen auf die die Funktion zusätzlich Einfluss hat. D. h. die Funktions-Instanzen ändern nur Konfigurationen, die in ihrem Funktionsbereich liegen, aber die Auswirkungen der Änderungen können auch Einfluss auf anderen Netzressourcen haben z. B. einer Nachbarzelle [Rap13]. Z. B. verändert die Instanz der CCO Funktion der Mobilfunkzelle X die Downlink Tx power und den Antennenneigungswinkel. Diese Veränderungen haben einen Einfluss auf die Größe der Zelle X. Eine MLB Funktion erlaubt einer überladenen Zelle einen Teil ihrer Datenmenge an eine Nachbarzelle zu übergeben. Das Ziel dieser Funktion liegt darin die Datenlast aufzuteilen und somit eine besseren Performance des Netzes zu erzielen. Hierfür sind HO Parameter nötig, um diesen Austausch zu ermöglichen. Die Größenveränderung der Zelle X hat einen Einfluss auf die HO Parameter der Nachbarzelle Y. Somit wird bei der Ausführung der CCO Funktion der Mobilfunkzelle X, die Nachbarzelle Y beeinflusst [Ric12, S. 72,73], [Tob12, S. 333,334].

- Ausführungszeit: Die Ausführungszeit gibt an, wie lange eine SON Funktion dauert [Rap13].

Zur Laufzeit werden die ausgezählten Informationen von der Koordinations-Logik evaluiert und auf der Koordinationsschicht instanziiert. Anschließend muss eine Entscheidung getroffen werden, wie eine Anfrage behandelt wird [Rap13]:

- Acknowledged (ACK): In diesem Fall wird das Event mit den Kontextinformationen weitergeleitet um ausgeführt zu werden.

- Reject (NACK): Die Anfrage wird gelöscht und das Event wird nicht weitergeleitet.

- Rollback (RB): Die Aktion die von der vorherigen Funktions-Instanz ausgeführt wird, wird rückgängig gemacht und die Kontextinformationen über diese Funktion werden gelöscht [Rap13].

Abbildung 9 zeigt ein Ablaufdiagramm der BR Koordination mit einem Puffer. Event Anfragen, die nicht mit Funktions-Instanzen im Konflikt stehen, werden im Puffer weiterverarbeitet, d. h. die Anfragen werden mit bereits im Puffer enthaltenen Events auf Konflikte überprüft [Rap13].
Ein Event z. B. CCO stellt eine Anfrage ausgeführt zu werden (1). Danach werden die Funktionsinformationen bestehend aus Wirkungsbereich und Ausführungszeit bestimmt.

Wenn ein Konflikt mit einer Funktions-Instanz besteht, wird von der Koordinations-Logik auf Grund des Nutzwertes einer Funktion bestimmt, ob die aktuelle Anfrage abgewiesen wird oder ein RB durchgeführt wird (2), oder mit einem ACK weitergeleitet wird (2). Falls kein Konflikt mit einer Funktions-Instanz besteht, wird ein vorläufiger ACK der Funktion zugewiesen (3). Jetzt beginnt die Konfliktlösung im Puffer. Wenn die Anfrage den maximalen Nutzwert besitzt, wird diese dem Puffer mit einem ACK hinzugefügt (4) und alle anderen Events, die mit der Anfrage mit maximalem Nutzwert in Konflikt stehen, werden aus dem Puffer entfernt (4) und mit einem NACK abgewiesen (5). Falls die Anfrage nicht den höchsten Nutzwerte besitzt, wird verglichen, ob sich im Puffer Events befinden, die einen höheren Nutzwert besitzen, als die Anfrage. Wenn es Events im Puffer gibt, die einen höheren Nutzwert besitzen, wird die Anfrage mit einem NACK verworfen (6). Falls die Anfrage einen höheren Nutzwert hat, als die anderen Events im Puffer, die in Konflikt mit der Anfrage stehen, wird die Anfrage dem Puffer hinzugefügt (7) und die Events, die mit der Anfrage in Konflikt stehen, mit niedrigerem Nutzwert werden aus dem Puffer entfernt (8) und mit einem NACK verworfen. Der Prozess im Puffer ist ein unabhängiger Prozess und überwacht das Systemverhalten. Somit wird das Ende der Input Batch erkannt und die Events, die sich im Puffer befinden, werden mit einem ACK bestätigt und ausgeführt [Rap13].

### 2.3.5. Single Request Methode

Die SR Methode kombiniert die Konflikterkennung und Konfliktlösung in den ECA Regeln. Ein **Event** ist eine Anfrage zum Ausführen einer SON Funktion. Die **Condition** ist eine oder mehrere SON Funktionen, die zum Zeitpunkt der Anfrage ausgeführt werden und mit der anfragenden SON Funktion in Konflikt stehen. **Action** ist die Reaktion auf eine SON Funktionsanfrage. Wenn ein ACK auf eine Anfrage folgt, wird die anfragende SON Funktion ausgeführt und die in Konflikt mit der anfragenden Funktion, stehende laufende SON Funktion wird abgebrochen. Falls ein NACK auf eine Anfrage folgt, wird die Anfrage verworfen und die laufenden Funktionen werden weiter ausgeführt. Alle laufenden SON Funktionen stehen nicht in Konflikt miteinander. Ein Konflikt einer laufenden Funktion tritt nur mit einer anfragenden Funktion auf. Der Begriff Konflikt-Funktion/en bezieht sich immer auf eine oder mehrere laufende Funktionen, die in Konflikt mit einer anfragenden Funktion stehen. Die angegebenen Nutzwerte der SON Funktionen werden immer erst am Ende der Laufzeit erreicht, d. h. wenn eine SON Funktion in der Mitte der Laufzeit abgebrochen wird, ist der erzielte Nutzwert $v = 0$. Es gibt keine Angaben über die restliche Ausführungszeit einer laufenden Funktion [Fre13].

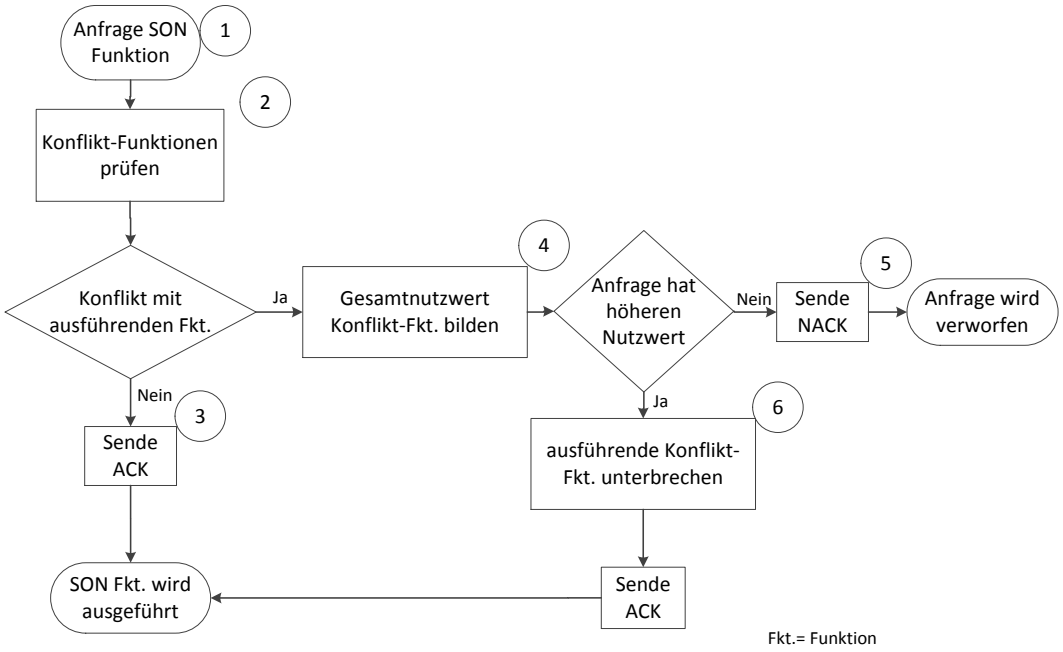


Abbildung 10.: Single Request Koordination [Fre13]

Der Ablauf einer Anfrage einer SON Funktion ist in Abbildung 10 dargestellt. Während der Laufzeit von einer oder mehrere SON Funktionen trifft eine Anfrage einer SON Funktion ein (1). Als nächstes wird geprüft, ob ein oder mehrere Konflikte zwischen der anfragenden Funktion und laufender Funktionen bestehen. Falls kein Konflikt besteht, wird ein ACK gesendet (3), und die anfragende SON Funktion wird der Liste laufender Funktionen hinzugefügt. Wenn mehrere Konflikt-Funktionen vorhanden sind, wird der Gesamtnutzwert über die Nutzwerte der Konflikt-Funktionen gebildet (4) und anschließend mit dem Nutzwert der anfragenden Funktion verglichen. Falls nur eine Konflikt-Funktion vorhanden ist, ist der Gesamtnutzwert gleich dem Nutzwert dieser einen Funktion. Wenn die anfragende Funktion einen niedrigeren oder gleichen Nutzwert besitzt, wird ein NACK gesendet (5), und die Anfrage wird verworfen. Auch bei Gleichheit der Nutzwerte wird die Anfrage verworfen, weil es aufwendiger ist eine Funktion abzubrechen, als diese weiter auszuführen. Wenn der Gesamtnutzwert der Konflikt-Funktionen kleiner dem Nutzwert der anfragenden Funktion ist, werden die laufenden Konflikt-Funktionen unterbrochen (6) und ein ACK wird gesendet und die anfragende SON Funktion wird ausgeführt [Fre13].

# 3. Optimierung der Batch Request Konfliktlösung durch Constraint-Optimierung

## 3.1. Beispiel Batch Request Konfliktlösungen

Um den Ablauf der RPS und BR Konfliktlösungen besser zu verstehen, wurde in den folgenden zwei Abschnitten 3.1.1, 3.1.2 die Konfliktlösungen anhand des ABC Konfliktbeispiels näher betrachtet.

### 3.1.1. Beispiel Rational Policy System Konfliktlösung

Der Ablauf des Konfliktlösungsansatzes des RPS ist in Abschnitt 2.3.3 beschrieben worden. In der folgenden Ausführung wurde die AA des RPS anhand des ABC Konfliktbeispiels gezeigt. In jedem Konflikt-Set (KS) stehen alle Aktionen in Konflikt miteinander. Somit gibt es ein $KS_1$ für den Konflikt zwischen A und B und ein $KS_2$ für den Konflikt zwischen A und C. Zusätzlich gibt es ein Aktions-Set, das zu Beginn der Konfliktlösung alle Aktionen A, B, C enthält. Aus diesem Aktions-Set werden die Aktionen gelöscht, die in einem KS den niedrigeren Nutzwert besitzen. Am Ende der Konfliktlösung stehen nur noch die auszuführenden Aktion im Aktions-Set [Chr13].
Aktions-Set zu Beginn der AA:

$$\text{Aktions-Set} = \{A, B, C\} \tag{4}$$

$KS_1$ zwischen A und B:

$$KS_1 = \{A, B\} \tag{5}$$

$KS_2$ zwischen A und C:

$$KS_2 = \{A, C\} \tag{6}$$

Der Nutzwerte der einzelnen Aktionen sind folgende:

$$v(A) = 10,\ v(B) = 6,\ v(C) = 6 \tag{7}$$

Es wird aus jedem KS die Aktion mit dem niedrigeren Nutzwert $v_{min}(KS)$ ausgewählt und aus dem Aktions-Set entfernt:

Aus $KS_1$ Aktion B mit niedrigerem Nutzwert auswählen und aus Aktions-Set entfernen:

$$\begin{aligned} &v(A) = 10 > v(B) = 6 \Rightarrow \text{Aktion B aus Aktions-Set entfernen} \\ &\text{Aktions-Set} = \{A, C\} \end{aligned} \tag{8}$$

Aus $KS_2$ Aktion C mit niedrigerem Nutzwert auswählen und aus Aktions-Set entfernen:

$$\begin{aligned} &v(A) = 10 > v(C) = 6 \Rightarrow \text{Aktion C aus Aktions-Set entfernen} \\ &\text{Aktions-Set} = \{A\} \end{aligned} \tag{9}$$

Der Gesamtnutzwert aller Aktionen ist die Summe über die Nutzwerte aller Aktionen aus dem Aktions-Set.

$$\begin{aligned} v_{\text{gesamtRPS}} &= \sum_{a \in \text{Aktions-Set}} v(a) \\ v_{\text{gesamtRPS}} &= v(A) = 10 \end{aligned} \tag{10}$$

[Chr13]

### 3.1.2. Beispiel Batch Request Konfliktlösung

Der Ablauf des BR Konfliktlösungsansatzes ist in Abschnitt 2.3.4 beschrieben worden. In diesem Kapitel wird die Konfliktlösung der BR Methode anhand des ABC Konfliktbeispiels durchgeführt. Zu Beginn des Prozesses ist der Puffer leer. Die Batch ist befüllt mit den Aktionen A, B und C. Mit jeder Anfrage einer Aktion aus der Batch wird überprüft, ob der Puffer ein Element enthält, welches mit der anfragenden Aktion in Konflikt steht. Anhand der Nutzwerte wird entschieden welche Aktion im Puffer abgespeichert wird und welche Anfrage verworfen wird [Rap13].
Puffer zu Beginn:

$$\text{Puffer} = \{\} \tag{11}$$

Batch zu Beginn:

$$\text{Batch} = \{B, A, C\} \tag{12}$$

Konfliktmenge K:

$$\text{K} = \{(A, B), (A, C)\} \tag{13}$$

Erste Anfrage der Aktion B aus der Batch:

$$\text{Request Element} = \{B\} \tag{14}$$

Überprüfung auf Konfliktelement im Puffer:

$$\text{(B, Element aus Puffer)} \notin K \tag{15}$$

Hinzufügen des angefragten Elements zum Puffer:

$$\text{Puffer} = \{B\} \tag{16}$$

Zweite Anfrage des Elements A aus der Batch:

$$\text{Request Element} = \{A\} \tag{17}$$

Überprüfung auf Konfliktelement im Puffer:

$$\text{(A, B)} \in K \tag{18}$$

Falls ein Konfliktelement im Puffer vorhanden ist, werden die Nutzwerte verglichen, und das Element mit dem kleineren Nutzwert wird aus dem Puffer entfernt und das Element mit dem größeren Nutzwert wird dem Puffer hinzugefügt:

$$\begin{aligned} &v(A) = 10 > v(B) = 6 \Rightarrow \text{B aus Puffer entfernen und A hinzufügen} \\ &\text{Puffer} \{\} \qquad \text{B aus Puffer entfernt} \\ &\text{Puffer} \{A\} \qquad \text{A dem Puffer hinzugefügt} \end{aligned} \tag{19}$$

Dritte Anfrage des Elements C aus der Batch:

$$\text{Request Element} = \{C\} \tag{20}$$

Überprüfung auf Konfliktelement im Puffer:

$$\text{(A, C)} \in K \tag{21}$$

Vergleich der Nutzwerte der Konfliktelemente und aktualisieren des Puffers:

$$v(A) = 10 > v(C) = 6 \Rightarrow \text{Puffer} = \{A\} \tag{22}$$

Der Gesamtnutzwert aller Aktionen ist die Summe über die Nutzwerte aller Aktionen aus dem Puffer.

$$v_{\text{gesamtBR}} = \sum_{a \in \text{Puffer}} v(a) \tag{23}$$

$$v_{\text{gesamtBR}} = v(A) = 10 \tag{24}$$

[Rap13]

## 3.2. Constraint Definition

In dem Abschnitt 1.3 wird als zweiter Punkt das Erkennen der BR Konflikte als Constraint Problem genannt. „Aus formaler Sicht sind Constraints spezielle prädikatenlogische Formeln, mit deren Hilfe man Eigenschaften von Problemen und deren Lösungen beschreibt. Das können z. B. Gleichungen oder Ungleichungen über Zahlen sein, aber auch andere Ausdrücke über Zahlen, Booleschen Werten oder beliebigen anderen Mengen wie Buchstaben oder Wörtern“ [PH07, S. 53]. Die wörtliche Übersetzung von „Constraint“ ist Zwangsbedingung, z. B. legen Konflikte Bedingungen für das Ausführen der verschiedenen Aktionen fest [PH07, S. 52]. Das ABC Konfliktbeispiel legt als erste Bedingung fest, wenn Aktion A ausgeführt wird, darf Aktion B nicht ausgeführt werden und umgekehrt. Die zweite Bedingung legt das Ausführen zwischen der Aktion A und der Aktion C fest, d. h. wenn Aktion A ausgeführt wird, darf Aktion C nicht ausgeführt werden und umgekehrt. Jede dieser Bedingungen stellt ein Constraint dar. Die Constraints können bezüglich der Reihenfolge beliebig angeordnet werden, d. h. es spielt keine Rolle, ob der Konflikt zwischen den Aktionen A und B Constraint 1 oder Constraint 2 bildet.
Mit Hilfe des Constraint-Lösers kann überprüft werden, ob die Constraints erfüllbar sind. Zusätzlich können die berechneten Ergebnisse, z. B. der Gesamtnutzwert, ausgegeben werden. „Ein Constraint-Löser ist dabei auf eine bestimmte Klasse von Constraints eingeschränkt, z. B. auf lineare arithmetische Constraints, Boolsche Constraints oder Finite-Domain-Constraints. Das liegt daran, dass die Algorithmen, mit denen man beispielsweise arithmetische Constraints untersuchen kann, im Allgemeinen ganz andere sind als jene, mit denen man Boolsche Constraints behandelt“ [PH07, S. 53].
Die Konflikte und die Optimierung der Nutzwerte ist ein lineares Optimierungsproblem, weil bei „der linearen Optimierung geht es meist um Probleme der Kapazitätsauslastung oder der Gewinnmaximierung, d. h. um Probleme mit einer Menge von möglichen Lösungen, wobei man von diesen hinsichtlich bestimmter Bedingungen optimale Lösungen sucht. Eine Zielfunktion legt fest, wann eine Lösung optimal ist“ [PH07, S. 99]. Bei der BR Konfliktlösung ist die „Gewinnmaximierung“ die Gesamtnutzwertmaximierung d. h. der größte Gesamtnutzwert ist hinsichtlich der Konflikte zu finden. Die Probleme stellen die Konflikte dar und die Menge von möglichen Lösungen sind die Aktionen, die an den Konflikten beteiligt sind. Die optimale Lösung, die hinsichtlich der Bedingungen gesucht wird, ist der größtmögliche Gesamtnutzwert. Die Bedingungen geben die Konflikte vor, d. h. es darf nur eine oder keine Aktion aus einem Konflikt ausgewählt werden und die Aktionen, die ausgewählt werden, dürfen nicht in Konflikt miteinander stehen.

## 3.3. Constraint-Optimierung

Bei den RPS und BR Konfliktlösungsansätzen, werden die Konflikte einzeln betrachtet und aus diesen die Aktion mit dem größten Nutzwert ausgewählt. Es wird versucht den Gesamtnutzwert zu maximieren, indem man den Nutzwert jedes einzelnen Konfliktes maximiert. Das ABC Konfliktbeispiel ist ein Beispiel dafür, dass die isolierte Konfliktbetrachtung nicht immer den maximalen Gesamtnutzwert ergibt. Angenommen die Op-

timierung des Gesamtnutzwertes $v_{\text{gesamtOptimiert}}$ des ABC Konfliktes erfolgt durch die Auswahl der Aktionen B und C, dann ergibt sich folgender Gesamtnutzwert:

$$v_{\text{gesamtOptimiert}} = v(B) + v(C) = 6 + 6 = 12 \tag{25}$$

Das Ergebnis lässt erkennen, dass die Vorgehensweise der BR Konfliktlösungsansätze nicht den maximalen Gesamtnutzwert aller Aktionen ergibt.

$$\begin{aligned} v_{\text{gesamtOptimiert}} &> v_{\text{gesamtBR}} \\ v_{\text{gesamtOptimiert}} &> v_{\text{gesamtRPS}} \end{aligned} \tag{26}$$

Um den Gesamtnutzwert zu optimieren, bedarf es einer neuen Konfliktlösungsmethode. Ein wichtiger Ansatz ist, dass die Konfliktlösung nicht isoliert auf einen einzelnen Konflikt betrachtet werden darf, sondern auf alle Konflikte. Hierfür ist die Constraint-Optimierung zu verwenden.
Für die Constraint-Optimierung wurde als erstes eine zu maximierende Zielfunktion $Z$ erstellt. Für das ABC Konfliktbeispiel wurde der Gesamtnutzwert aller auszuführenden Aktionen durch folgende zu maximierende Zielfunktion $Z$ angegeben:

$$Z = f(A) * v(A) + f(B) * v(B) + f(C) * v(C) \tag{27}$$

v(X): gibt den Nutzwert einer Aktion X an
f(X): liefert den Wert 1, wenn eine Aktion X ausgeführt wird und den Wert 0, wenn die Aktion X nicht ausgeführt wird
Die Zielfunktion der Konfliktlösung allgemein für eine Menge $A$ von Aktionen ist folgende:

$$Z = \text{maximiere} \sum_{X \in A} f(X) * v(X) \tag{28}$$

Als nächstes müssen die Constraints für die Zielfunktion angegeben werden. Die Konflikte geben die Constraints vor, deshalb wird die Menge aller Konflikte $K$ als Constraints definiert. Allgemein formuliert werden die Constraints zur Konfliktlösung folgendermaßen:

$$f(X) + f(Y) \leq 1 \text{ für alle Konflikte } k = (X, Y) \in K \tag{29}$$

Angewendet auf das ABC Konfliktbeispiel ergeben sich zwei Constraints. Constraint 1 bezieht sich auf den Konflikt zwischen der Aktion A und der Aktion B. Constraint 1 gibt folglich an, dass entweder die Aktion A oder die Aktion B oder keine der beiden Aktionen ausgeführt wird.

Constraint 2 beschreibt die Konfliktauflösung zwischen der Aktion A und der Aktion C.

$$\begin{aligned} &\textbf{Constraint 1} && f(A) + f(B) \leq 1 \\ &\textbf{Constraint 2} && f(A) + f(C) \leq 1 \end{aligned} \tag{30}$$

Falls die Aktion A ausgeführt werden soll, gilt folgende Belegung:

$$\begin{aligned} &f(A) = 1, f(B) = 0, f(C) = 0 \\ &f(A) + f(B) \leq 1 \Rightarrow 1 + 0 \leq 1 \\ &f(A) + f(C) \leq 1 \Rightarrow 1 + 0 \leq 1 \end{aligned} \tag{31}$$

Falls die Aktionen B und C ausgeführt werden sollen, gilt folgende Belegung:

$$\begin{aligned} &f(A) = 0, f(B) = 1, f(C) = 1 \\ &f(A) + f(B) \leq 1 \Rightarrow 0 + 1 \leq 1 \\ &f(A) + f(C) \leq 1 \Rightarrow 0 + 1 \leq 1 \end{aligned} \tag{32}$$

Beide Belegungen (31)(32) erfüllen die Constraints. Durch die Angabe der Maximierung des Gesamtnutzwerte wird die Belegung, die den größten Gesamtnutzwert erzielt gewählt.
Es werden die Aktionen B und C ausgewählt:

$$\begin{aligned} &f(A) = 0, f(B) = 1, f(C) = 1 \\ &v(A) = 10, v(B) = 6, v(C) = 6 \\ &v_{\text{gesamtOptimiert}} = 0 * 10 + 1 * 6 + 1 * 6 = 12 \\ &v_{\text{gesamtOptimiert}} > v_{\text{gesamtBR}} \\ &v_{\text{gesamtOptimiert}} > v_{\text{gesamtRPS}} \end{aligned} \tag{33}$$

Durch die Methode der Konfliktlösung mit Hilfe der Constraint-Optimierung wird der Gesamtnutzwert weiter maximiert.

## 3.4. Implementierung der Constraint-Optimierung

Die Implementierung der BR Konfliktlösung mit Constraint-Optimierung wurde mit dem Google operations research (or)-tools durchgeführt. or-tools ist von Google in C++ entwickelt worden und steht als open source zur Verfügung. Es werden verschiedene Werkzeuge angeboten z. B. Constraint-Löser, Rucksackproblem Algorithmen und Graphik Algorithmen. Der Code ist in C++ geschrieben und ist dank Simplified Wrapper and Interface Generator (SWIG) auch in anderen Programmiersprachen wie z. B. Java, Phyton oder .NET verfügbar [Nik13a].
Die Implementierung der Konfliktlösung erfolgte in Java. In den folgenden Absätzen wurden die wichtigsten Programmfragmente für die Umsetzung der Konfliktlösung durch den Constraint-Löser erläutert:

Die Methode `solveConflict` ist für die Konfliktlösung zuständig und gibt die auszuführenden Aktionen mit dem maximalen Gesamtnutzwert zurück. Die Konfliktpaare `conflictPairs` und Aktionen `actions` wurden an die Methode `solveConflicts` übergeben.

```
1 public Set<Action> solveConflict(Set<Pair<Action, Action>>
      conflictPairs, ArrayList<Action> actions);
```

Der Constraint Programming (CP) solver ist die Hauptkomponente um ein Constraint Problem zu lösen. Er enthält eine sehr große Application Programming Interface (API) und den größten Teil an Methoden, die zur Constraint-Lösung notwendig sind. An den Konstruktor des Solvers wurde der Name zur Identifikation übergeben [Nik13b].

```
1 Solver solver = new Solver("SolveConflict");
```

Der Constraint-Löser wird mit Constraints definiert, d. h. es müssen Bedingungen formuliert werden. Diese Bedingungen setzen sich aus mehreren Teilen zusammen. In Abschnitt 3.3 wurde erwähnt, dass zu jedem Konflikt ein Constraint definiert werden muss. Ein Teil der Constraint Erzeugung liefert die zweidimensionale Matrix `constraints`. Die Zeilen repräsentieren jeweils einen Konflikt und in den Spalten wurden die Konfliktelemente angegeben. Von einem Konflikt betroffene Elemente wurden mit 1 und nicht betroffene Elemente mit 0 markiert.

```
1 int[][] constraints = new int[conflictPairs.size()][utilities.
      size()];
```

Für das ABC Konfliktbeispiel wurden auf Grund von zwei Konflikten, zwei Zeilen benötigt. In der ersten Zeile wurde der Konflikt zwischen den Aktionen A und B dargestellt und in der zweiten Zeile steht der Konflikt zwischen den Aktionen A und C. Die Matrix `constraints` wurde für das ABC Konfliktbeispiel mit dem Inhalt folgender Tabelle gefüllt:

| Konflikte \ Elemente | A | B | C |
|---|---|---|---|
| A und B | 1 | 1 | 0 |
| A und C | 1 | 0 | 1 |

Mit der Matrix `constraints` wurde nur gekennzeichnet, ob ein Element betroffen ist von einem Konflikt oder nicht, folgende Programmzeile definiert die Möglichkeit ob die Aktion ausgeführt wird mit 1 oder nicht ausgeführt wird mit 0. Mit der Klasse `Solver` können Entscheidungsvariablen `x` definiert werden. Die Größe des Arrays ist mit `n` angegeben und ergibt sich aus der Größe des übergebenen Sets von Aktionen. Die Werte 0, 1 definieren die Domäne der Entscheidungsvariablen und der letzte Parameter `assignment` ist die Bezeichnung.

```
1 IntVar[] x = solver.makeIntVarArray(n, 0, 1, "assignment");
```

Als nächstes wurden die eigentlichen Constraints formuliert. Mit der Methode `makeScaleProdLessOrEqual()`, wird das Skalarprodukt zwischen den Entscheidungsparametern und den `constraints` gebildet. Der dritte Parameter dieser Funktion gibt an, dass das Skalarprodukt $\leq$ dem gesetzten Limit 1, sein muss. Die einzelnen Constraints wurden mit `addConstraint` dem `Solver` hinzugefügt. Dies entspricht der Formel (29).

```
1 for (int i = 0; i < conflictPairs.size(); i++) {
2         solver.addConstraint(solver.makeScalProdLessOrEqual
3         (x,constraints[i], limit));
4 }
```

Die Zielfunktion $Z$ (27) wurde mit der Methode `makeScalProd` angegeben. Diese Methode erzeugt das Skalarprodukt zwischen den bereits erzeugen Entscheidungsparameter `x` und den Nutzwerten, die in der `utilitiesList` enthalten sind. Das Ergebnis von `utility` ist die Summe über die Nutzwerte der Aktionen, die ausgeführt werden (28).

```
1 IntVar utility = solver.makeScalProd(x, utilitiesList).var();
```

Dem Constraint-Löser wird mit `makeMaximize` vorgegeben, dass die Zielfunktion `utility` maximiert werden soll und 1 gibt die Schrittgröße bei der Suche an. Die Variable `obj` enthält während der Suche die beste Lösung zu einem bestimmten Zeitpunkt.

```
1 OptimizeVar obj = solver.makeMaximize(utility, 1);
```

Der `DecisionBuilder` ist für die Erstellung des Suchbaumes zuständig. Der `Solver` stellt die Methode `makePhase()` zur Verfügung, die einen `DecisionBuilder` zurückliefert. Der erste Parameter ist die Entscheidungsvariable `x`, die mit 0 oder 1 angibt, ob eine Aktion ausgeführt wird, oder nicht. Der zweite Parameter entscheidet, wie in dem Suchbaum die nächste `IntVar` Variable ausgewählt wird. Mit `CHOOSE_PATH` wird die nächste Variable in dem jeweiligen Pfad der durchlaufen wird ausgewählt. Der dritte Parameter maximiert den Gesamtnutzwert mit `MAX_VALUE`. Anschließend kann die Suche gestartet werden.

```
1 DecisionBuilder db = solver.makePhase(x, Solver.CHOOSE_PATH,
2                                       Solver.ASSIGN_MAX_VALUE);
3 solver.newSearch(db, obj);
```

Der Constraint-Löser testet alle möglichen Kombinationen von Aktionen durch und nimmt die Kombinationen mit dem größten Gesamtnutzwert bei seinen Lösungen mit auf. Wenn eine Kombination mit einem größeren Gesamtnutzwert, als dem bereits enthaltenen ist, wird diese Kombination ebenfalls hinzugefügt. Die letzte hinzugefügte Lösung ergibt den größtmöglichen Gesamtnutzwert, der nicht weiter maximiert werden konnte. Die Lösungen werden mit `nextSolution()` ausgegeben [Nik13b].

```
1 while (solver.nextSolution())
```

Die ausgewählten Aktionen mit dem größtmöglichen Gesamtnutzwert werden an die aufrufende Funktion zurückgegeben.

## 3.5. Evaluation Batch Request Konfliktlösung

Um die Erhöhung des Gesamtnutzwertes der neuen Konfliktlösung durch Constraint-Optimierung gegenüber den Konfliktlösungsansätzen ohne Constraint-Optimierung zu veranschaulichen, wurden in der folgenden Ausführung mehrere Konflikte simuliert und die jeweiligen Gesamtnutzwerte in einem Diagramm dargestellt.
Der BR Simulator erstellt beliebig viele Aktionen und Konflikte und führt die implementierten Methoden `solveConflict` für die RPS Konfliktlösung, BR Konfliktlösung ohne Constraint-Optmierung und die Konfliktlösung mit Constraint-Optimierung aus. Für diese Auswertung wurden 15 Aktionen erstellt und aus diesen wurden die Konflikte gebildet. Das Diagramm aus Abbildung 11 zeigt die durchschnittlichen Gesamtnutzwerte der verschiedenen Konfliktlösungen abhängig von der Konfliktanzahl.

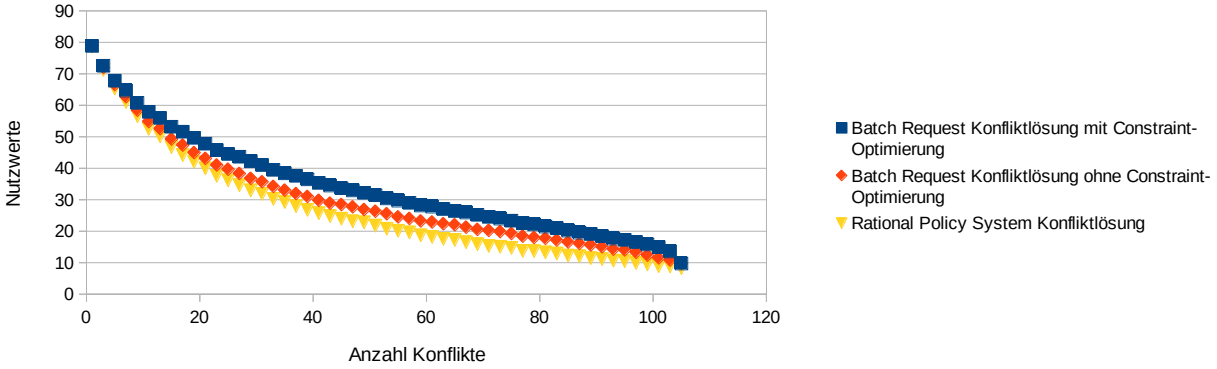


Abbildung 11.: Durchschnittlicher Gesamtnutzwert der BR Konfliktlösungen

Das Diagramm aus Abbildung 11 lässt erkennen, dass der Gesamtnutzwert der Konfliktlösung mit Hilfe der Constraint-Optimierung in den meisten Fällen größer und in Einzelfällen gleich groß ist, wie der Gesamtnutzwerten der Konfliktlösungen ohne Constraint-Optimierung. Bei geringer Konfliktanzahl liegen die verschiedenen Gesamtnutzwerte sehr nahe zusammen, weil bezügliche der Konflikte wenig selektiert werden muss. D. h. bei 15 Elementen und z. B. 3 Konflikten ist die Wahrscheinlichkeit, dass diese 3 Konflikte die gleichen Aktionen betreffen sehr gering. Somit ist die Differenz der Gesamtnutzwerte der Konfliktlösungen bei geringer Konfliktanzahl sehr gering. Je mehr Konflikte vorhanden sind, desto mehr Aktionen sind bei diesen Konflikten beteiligt und somit erzielen die Konfliktlösungen ohne Constraint-Optimierung, einen deutlich geringeren Gesamtnutzwert, als die Konfliktlösung mit Constraint-Optimierung. In der Mitte der Kurve, d. h. zwischen 20 und 80 Konflikten ist klar zu erkennen, dass der Gesamtnutzwert der Konfliktlösung des Constraint-Lösers größer ist, als der Gesamtnutzwert der Konfliktlösungen ohne Constraint-Löser. Dies liegt daran, dass der Constraint-Löser weder von der Reihenfolge von Konflikt-Sets, noch von der Reihenfolge der Aktionen in

einer Batch beeinflusst wird. Unter Berücksichtigung der Konflikte ermittelt die Konfliktlösung der Constraint-Optimierung immer die Kombination von Aktionen, die den größtmöglichen Gesamtnutzwert erzielen. Am Ende der Kurve, d. h. wenn die Konfliktanzahl sehr hoch ist, steht fast jede Aktion mit jeder anderen Aktion in Konflikt und damit ist die Auswahlmöglichkeit der auszuführenden Aktionen sehr gering. Mit dem Verringern der Auswahlmöglichkeit, der noch nicht in Konflikt stehenden Aktionen nähern sich die Gesamtnutzwerte aller drei Konfliktlösungen wieder an. Aus dem Diagramm aus Abbildung 11 ist der gewünschten Effekt der Konfliktlösung durch Contraint-Optimierung, nämlich eine weitere Erhöhung des Gesamtnutzwertes gegenüber den isolierten Konfliktlösungen signifikant erkennbar.
Weitere Auswertungen sind dem Anhang auf CD beigefügt.

### 3.5.1. Gesamtnutzwertvergleich zwischen den Konfliktlösungsansätzen ohne Constraint-Optimierung

Aus Abbildung 11 ist ebenfalls zu erkennen, dass der Gesamtnutzwert der Konfliktlösung der BR Methode ohne Constraint-Optimierung höher ist, als der Gesamtnutzwert der RPS Konfliktlösung.
Unterschiedliche Reihenfolgen der Konflikt-Sets, die vom RPS abgearbeitet werden, haben Einfluss auf den erzielten Gesamtnutzwert. Dies ist der Grund, weshalb der Gesamtnutzwert der RPS Konfliktlösung geringer ist, als der Gesamtnutzwert der BR Konfliktlösung ohne Constraint-Optimierung.
Es ist ebenfalls möglich, dass die RPS Konfliktlösung einen höheren Gesamtnutzwert erzielt als die BR Konfliktlösung ohne Constraint-Optimierung. Dies liegt daran, dass die BR Methode ohne Constraint-Löser, nacheinander die Aktionen aus der Batch abarbeitet und hierbei die Reihenfolge der Aktionen in der Batch einen Einfluss auf den Gesamtnutzwert haben. Das Vorkommnis, dass die RPS Konfliktlösung einen höheren Gesamtnutzwert erzielt als die BR Konfliktlösung ist aus dem Diagramm der Abbildung 11 nicht zu entnehmen, da es zu selten vorkommt.
In dem folgenden Beispiel wurden zwei verschiedene Konflikt-Set Reihenfolgen für die RPS Konfliktlösung und zwei verschiedene Aktionsreihenfolgen in der Batch, für die BR Konfliktlösung ohne Constraint-Löser angegeben. Anschließend wurden die Gesamtnutzwerte der verschiedenen Reihenfolgen ermittelt und gegenübergestellt. Beide Konfliktlösungsansätze wurden in verkürzter Version dargestellt. Der ausführliche Ablauf der Konfliktlösungen ist in den Abschnitten 2.3.4.2 und 2.3.3 nachzulesen.
Der Gesamtnutzwert der BR Konfliktlösung hängt von der Reihenfolge der Aktionen in der Batch ab.
Aktionsreihenfolge 1 in der Batch: [A, 10], [B, 6], [C, 5], [D, 5]
Aktionsreihenfolge 2 in der Batch: [B, 6], [C, 5], [A, 10], [D, 5]
Der Gesamtnutzwert der RPS Konfliktlösung hängt von der Reihenfolge der Konflikt-Sets ab.
Konfliktreihenfolge 1: [B, C], [A, B], [A, D]
Konfliktreihenfolge 2: [A, B], [B, C], [A, D]

**Konfliktlösung des RPSs nach Konfliktreihenfolge 1:**
Aktion C aus $KS_1$ wird aus Aktions-Set entfernt:

$$\begin{aligned} &v(B) = 6 > v(C) = 5 \Rightarrow \text{Aktion C aus Aktions-Set entfernen} \\ &\text{Aktions-Set} = \{A, B, D\} \end{aligned} \tag{34}$$

Aktion B aus $KS_2$ wird aus Aktions-Set entfernt:

$$\begin{aligned} &v(A) = 10 > v(B) = 6 \Rightarrow \text{Aktion B aus Aktions-Set entfernen} \\ &\text{Aktions-Set} = \{A, D\} \end{aligned} \tag{35}$$

Aktion D aus $KS_3$ wird aus Aktions-Set entfernt:

$$\begin{aligned} &v(A) = 10 > v(D) = 5 \Rightarrow \text{Aktion D aus Aktions-Set entfernen} \\ &\text{Aktions-Set} = \{A\} \end{aligned} \tag{36}$$

Der Gesamtnutzwert des RPSs nach Konfliktreihenfolge 1:

$$v_{\text{gesamtRPS Reihenfolge1}} = v(A) = 10 \tag{37}$$

**BR Konfliktlösung ohne Constraint-Optimierung nach Aktionsreihenfolge 1:**
Ersten Aktion A wird dem Puffer hinzugefügt:

$$\text{Puffer} = \{A\} \tag{38}$$

Zweite Aktion B wird verworfen:

$$v(A) = 10 > v(B) = 6 \Rightarrow \text{Puffer} = \{A\} \tag{39}$$

Dritte Aktion C wird dem Puffer hinzugefügt:

$$\text{Puffer} = \{A, C\} \tag{40}$$

Vierte Aktion D wird verworfen:

$$v(A) = 10 > v(D) = 5 \Rightarrow \text{Puffer} = \{A, C\} \tag{41}$$

Der Gesamtnutzwert der BR Konfliktlösung nach Aktionsreihenfolge 1:

$$v_{\text{gesamtBR Reihenfolge1}} = v(A) + v(C) = 15 \tag{42}$$

Gegenüberstellung Konfliktreihenfolge 1 und Aktionsreihenfolge 1:

$$v_{\text{gesamtBR Reihenfolge1}} = v(A) + v(C) = 15 > v_{\text{gesamtRPS Reihenfolge1}} = v(A) = 10 \tag{43}$$

**Konfliktlösung des RPSs nach Konfliktreihenfolge 2:**
Aktion B aus $KS_1$ wird aus Aktions-Set entfernt:

$$\begin{aligned} &v(A) = 10 > v(B) = 6 \Rightarrow \text{Aktion B aus Aktions-Set entfernen} \\ &\text{Aktions-Set} = \{A, C, D\} \end{aligned} \tag{44}$$

Aktion C aus $KS_2$ bleibt im Aktions-Set:

$$\begin{aligned} &B \notin \text{Aktions-Set} \Rightarrow \text{Aktions-Set unverändert} \\ &\text{Aktions-Set} = \{A, C, D\} \end{aligned} \tag{45}$$

Aktion D aus $KS_3$ wird aus Aktions-Set entfernt:

$$\begin{aligned} &v(A) = 10 > v(D) = 5 \Rightarrow \text{Aktion D aus Aktions-Set entfernen} \\ &\text{Aktions-Set} = \{A, C\} \end{aligned} \tag{46}$$

Der Gesamtnutzwert des RPSs nach Konfliktreihenfolge 2:

$$v_{\text{gesamtRPS Reihenfolge1}} = v(A) + v(C) = 15 \tag{47}$$

**BR Konfliktlösung ohne Constraint-Löser nach Aktionsreihenfolge 2:**
Erste Aktion B wird dem Puffer hinzufügen:

$$\text{Puffer} = \{B\} \tag{48}$$

Zweite Aktion C wird verworfen:

$$v(B) = 6 > v(C) = 5 \Rightarrow \text{Puffer} = \{B\} \tag{49}$$

Aktion B wird aus dem Puffer entfernt und dritte Aktion A wird dem Puffer hinzugefügt:

$$v(A) = 10 > v(B) = 6 \Rightarrow \text{Puffer} = \{A\} \tag{50}$$

Vierte Aktion D wird verworfen:

$$v(A) = 10 > v(D) = 5 \Rightarrow \text{Puffer} = \{A\} \tag{51}$$

Der Gesamtnutzwert der BR Konfliktlösung nach Aktionsreihenfolge 2:

$$v_{\text{gesamtBR Reihenfolge2}} = v(A) = 10 \tag{52}$$

Gegenüberstellung Konfliktreihenfolge 2 und Aktionsreihenfolge 2:

$$v_{\text{gesamtRPS Reihenfolge2}} = v(A) + v(C) = 15 > v_{\text{gesamtBR Reihenfolge2}} = v(A) = 10 \tag{53}$$

Aus diesem Beispiel ist zu erkennen, dass die Reihenfolge der Aktionen in der Batch und die Reihenfolge der Konflikt-Sets Einfluss auf die erzielten Gesamtnutzwerte haben. In der BR Konfliktlösung durch Constraint-Optimierung ist die Reihenfolge von angegebenen Aktionen oder Konflikte irrelevant. Der Constraint-Löser erstellt jede mögliche Kombination von Aktionen und wählt die Kombination von Aktionen mit dem größten Gesamtnutzwert aus.

# 4. Optimierung der Single Request Konfliktlösung

## 4.1. Beispiel Konfliktlösung Single Request

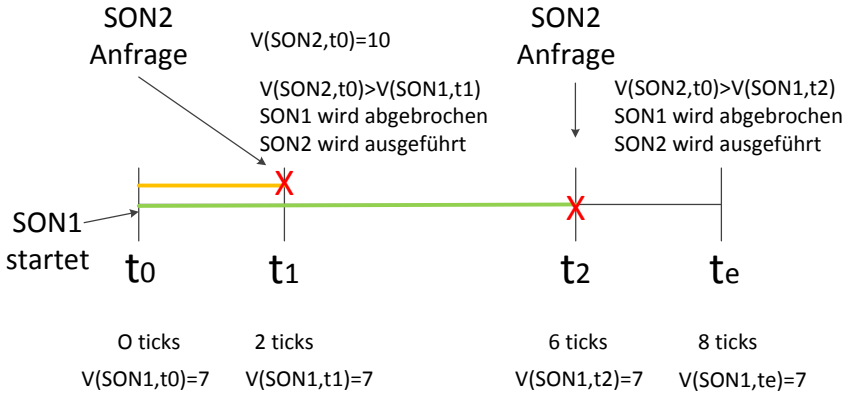


Abbildung 12.: SON1+SON2 Konfliktbeispiel

Der Ablauf der SR Methode wurde in Abschnitt 2.3.5 erklärt. In der folgenden Ausführung wurde näher auf die Konfliktlösung der SR Methode anhand eines Beispiels eingegangen.
Die Funktionen besitzen einen Nutzwert und eine Laufzeit. Die Laufzeit ist die gesamte Ausführungszeit einer SON Funktion, angegeben in Ticks. Das SON1+SON2 Konfliktbeispiel aus Abbildung 12 zeigt die SON1 Funktion mit dem Nutzwert $v(\text{SON1}) = 7$ und der Laufzeit $t_e(\text{SON1}) = 8$ Ticks. Der Nutzwert der SON1 Funktion besitzt zum Zeitpunkt $t_0 = 0$ Tick, $t_1 = 2$ Ticks, $t_2 = 6$ Ticks und $t_e(\text{SON1}) = 8$ Ticks immer denselben Nutzwert $v(\text{SON1}) = 7$. Die SON2 Funktion mit dem Nutzwert $v(\text{SON2}) = 10$ und einer Laufzeit von $t_e(\text{SON2}) = 12$ Ticks stellt eine Anfrage zum Zeitpunkt $t_1$ und zum Zeitpunkt $t_2$. Zum Zeitpunkt $t_1$ ist die SON1 Funktion noch nicht so lange ausgeführt worden, wie zum Zeitpunkt $t_2$. Die SON1 Funktion wird auf Grund des niedrigeren Nutzwertes unabhängig von der verstrichenen Laufzeit $t_1$ oder $t_2$ abgebrochen. Die bisherige Ausführungszeit der SON1 Funktion spielt keine Rolle. Hierbei wird nicht bedacht, dass es viel "rentabler" wäre die SON1 Funktion zum Zeitpunkt $t_2$ zu Ende laufen zu lassen, d. h. die bisherige Laufzeit der Funktion SON1 wäre nicht ohne erzielten Nutzwert ver-

strichen [Fre13].
Um die Nutzwerte der Funktionen besser vergleichen zu können und laufende Funktionen nicht kurz vor dem Laufzeitende abzubrechen, muss der Nutzwert zu verschiedenen Zeitpunkten berechnet werden und in den Berechnungen die Laufzeit einbezogen werden [Fre13].

## 4.2. Optimierung der Single Request Konfliktlösung

### 4.2.1. Barwertmethode

In der Wirtschaft wird zur Bewertung eines Investitionsobjektes die Barwertmethode angewandt. „Der Barwert rechnet Zahlungen, die zu unterschiedlichen Zeitpunkten anfallen, auf einen bestimmten Zeitpunkt (z.B. den 1. Januar eines Jahres oder das heutige Datum) um, indem er den Zeitwert des Geldes berücksichtigt (Barwertmethode)“ [Oli13b]. Weil der Wert der Investition auf den heutigen Wert, also die Gegenwart umgerechnet wird, wird der Barwert auch als Gegenwartswert bezeichnet. Bei der Barwertberechnung wird unterschieden, ob es sich um eine einmalige Zahlung handelt, oder um mehrere Zahlungen im Laufe der Zeit. Für die Barwertberechnung einer einmaligen Zahlung ist die Abzinsung ein passendes Beispiel [Oli13b].
Die Abzinsung ermittelt den Gegenwartswert einer Zahlung, die erst in der Zukunft getätigt wird. Die Zahlung, die in der Zukunft stattfindet wird mit Hilfe des Kalkulationszinssatz abgezinst und der Wert dieser Zahlung kann zu verschiedenen Zeitpunkten, die vor der Zahlung liegen, ermittelt werden [Oli13a].
Die Formel für die Abzinsung ist folgende [Oli13a]:

$$W_0 = \frac{W_n}{(1+i)^n} \tag{54}$$

$W_0$: Barwert zum Zeitpunkt 0
$W_n$: Wert nach n Perioden
$i$ : Kalkulationszinssatz
$1/(1+i)^n$: Abzinsungsfaktor

Beispiel für eine Abzinsung:
Ein Anleger hat die Möglichkeit eine Immobilie für 100.000 Euro am 01.01.2013 zu kaufen und der Immobilienmakler versichert ihm, die Möglichkeit zu haben diese Immobilie für 110.000 Euro am 31.12.2014 wieder verkaufen zu können. Anwendung der Abzinsungsformel bei einem Kalkulationszinssatz von 5% [Oli13a]:

$$W_0 = \frac{W_2}{(1+0.05)^2} = \frac{110.000 \text{ Euro}}{(1,05)^2} = 99.773,24 \text{ Euro} \tag{55}$$

$W_2$: Wert nach 2 Jahren

Das Ergebnis (55) zeigt, dass sich die Investition nicht gelohnt hätte, weil 110.000 Euro in 2 Jahren, umgerechnet auf den heutigen Barwert, weniger wert ist, als die investierten 100.000 Euro. Die Abzinsung bietet die Möglichkeit, den Wert einer zukünftigen Zahlung auf den aktuellen Zeitpunkt umzurechnen und damit vergleichbar zu machen [Oli13a].

### 4.2.2. Anwendung der Barwertmethode auf die Nutzwertberechnung

Bei der Barwertmethode wird der „Zeitwert des Geldes berücksichtigt“ [Oli13b], genauso muss der Zeitwert der Nutzwerte einer Funktion berücksichtigt werden. Die Möglichkeit, die die Abzinsung der Barwertmethode bei Investitionsentscheidungen liefert, kann auf die Berechnung der Nutzwerte angewandt werden. Der angegebene Nutzwert einer Funktion, der erst am Ende der Laufzeit erzielt wird, ist mit dem Verkaufswert, z. B. einer Immobilie nach zwei Jahren gleich zu setzen. Die angegebenen Perioden werden als Ticks interpretiert und der Kalkulationszinssatz ist der prozentuale Anstieg der Nutzwerte während der Laufzeit einer Funktion. Das Ergebnis ist statt dem aktuellen Geldwert der aktuelle Nutzwert einer Funktion. Aus diesen Angaben ergibt sich folgende Barwertmethode angewendet auf Nutzwerte:

$$V_{net}(\text{SON1}, t_1) = \frac{V(\text{SON1})}{(1+i)^{(t_e - t_1)}} \tag{56}$$

$V_{net}$: Nutzwert zum Zeitpunkt $t_1$ der SON1 Funktion
$V(\text{SON1})$: Nutzwert der SON1 Funktion am Ende der Laufzeit
$i$: Prozentuale Anstieg der Nutzwerte während der Laufzeit
$t_e$: Komplette Laufzeit der SON1 Funktion
$t_1$: Bisherige Laufzeit der SON1 Funktion

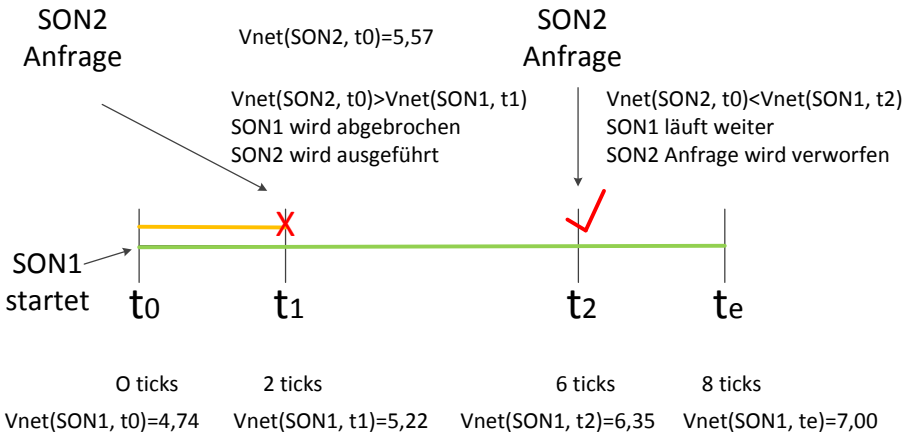


Abbildung 13.: SON1+SON2 Konfliktbeispiel mit Barwertmethode

Abbildung 13 zeigt den Ablauf der Konfliktlösung des SON1+SON2 Konfliktbeispiels mit der umgewandelten Barwertmethode. Der prozentuale Anstieg der Nutzwerte während der Laufzeit ist auf 5% festgelegt. Wenn die SON2 Funktion zum Zeitpunkt $t_1 = 2$ Ticks anfragt ausgeführt zu werden, werden die aktuellen Nutzwerte gemäß der Laufzeit auf zwei Nachkommastellen berechnet:

$$\begin{aligned} V_{net}(\text{SON2}, t_0) =& \frac{10}{(1+0.05)^{(12-0)}} = 5,57 \\ V_{net}(\text{SON1}, t_1) =& \frac{7}{(1+0.05)^{(8-2)}} = 5,22 \\ V_{net}(\text{SON2}, t_0) >& V_{net}(\text{SON1}, t_1) \end{aligned} \tag{57}$$

$\Rightarrow$ SON1 Funktion wird abgebrochen, SON2 Funktion wird ausgeführt

Es ist zu beachten, dass der Nutzwert der SON2 Funktion immer den Zeitpunkt $t_0$ berechnet wird, weil diese Funktion noch keinen Tick lang ausgeführt wurde. Falls die Anfrage der SON2 Funktion zu einem späteren Zeitpunkt $t_2 = 6$ Ticks stattfinden würde, wäre der Nutzwert der SON2 Funktion immer noch $V_{\text{net}}(\text{SON2}, t_0) = 5{,}57$ und der Nutzwert der SON1 Funktion wäre weiter angestiegen:

$$\begin{aligned} V_{net}(\text{SON1}, t_2) =& \frac{7}{(1+0.05)^{(8-6)}} = 6,35 \\ V_{net}(\text{SON2}, t_0) <& V_{net}(\text{SON1}, t_2) \end{aligned} \tag{58}$$

$\Rightarrow$ SON1 Funktion wird weiter ausgeführt, SON2 Funktionanfrage wird verworfen

Mit Hilfe der umgewandelten Barwertmethode werden die Nutzwert entsprechend dem jeweiligen Fortschritt der Funktionen berechnet. Somit können die Nutzwerte besser verglichen werden und Funktionsabbrüche kurz vor dem Laufzeitende einer SON Funktion vermieden werden.

## 4.3. Implementierung der Single Request Konfliktlösung

Die SR Konfliktlösung wird in Java implementiert und die folgenden Programmfragmente schaffen einen Überblick über die Methode `solveConflict`, die Konflikte zwischen laufenden und anfragenden Funktionen mit Hilfe der Barwertmethode löst.
Als Übergabeparameter sind die laufenden Aktionen `runningActions`, die Konflikte `conflicts`, die anfragende Aktion `requestedAction` und der prozentuale Anstieg des Nutzwertes während der Laufzeit `i` angegeben. Der Rückgabewert ist ein Set aus Aktionen, die abgebrochen werden sollen, d. h. wenn die laufend/en Aktion/en unterbrochen werden, dann sind diese in dem Set enthalten, wenn die laufend/en Aktion/en weiter laufen dürfen, ist das Set leer.

```
1 public Set<Action> solveConflict(Map<Action, Integer>
      runningActions, Set<Pair<Action, Action>> conflicts, Action
      requestedAction, double i);
```

Als erstes muss überprüft werden, mit welchen laufenden Aktionen die anfragende Aktion in Konflikt steht. Das Ergebnis ist ein Set von Konflikt-Aktionen.

```
1 Set<Action> runningConflicts = isInConflict(runningActions,
    requestedAction, conflicts);
```

Die Methode `calculateCurrentUtility` ermittelt mit Hilfe der gesamten Laufzeit `totalTime`, der verstrichenen Laufzeit `runningTime`, dem Nutzwert `utility` und der Prozentangabe `i` den aktuellen Nutzwert einer Aktion, gemäß der Formel (56).

```
1 public double calculateCurrentUtility(int totalTime, int
    runningTime, int utility, double i) {
2         double v = utility / Math.pow((1 + i), (totalTime -
            runningTime));
3         return v;
4 }
```

Die `getDifferenceRequestedRunningAction` berechnet in Zeile 9 die Summe der Nutzwerte der Konflikt-Aktionen. In Zeile 11 wurde der Nutzwert der anfragenden Aktion berechnet. Die Differenz zwischen dem Nutzwerte der anfragenden Aktion und der Summe der Nutzwerte der Konflikt-Aktionen wird in Zeile 12 zurückgegeben.

```
1  private double getDifferenceRequestedRunningAction(Set<Action>
     runningConflicts, Map<Action, Integer> runningActions, Action
      requestedAction, double i) {
2   double sumRunningUtilites = 0.0;
3   Iterator<Action> runningConflictElements = runningConflicts.
     iterator();
4   Action element;
5   while (runningConflictElements.hasNext()) {
6         element = runningConflictElements.next();
7         int tRunning = runningActions.get(element);
8         double utilityRunningAction = calculateCurrentUtility(
            element.getTickTotal(), tRunning, element.getUtility
            (), i);
9         sumRunningUtilites = sumRunningUtilites +
            utilityRunningAction;
10  }
11  double utilityRequestedAction = calculateCurrentUtility(
     requestedAction.getTickTotal(), 0, requestedAction.
     getUtility(), i);
12  return (utilityRequestedAction - sumRunningUtilites);
13 }
```

Wenn die Differenz größer 0 ist bedeutet das, dass die anfragende Aktion einen höheren Nutzwert hat, als die laufenden Konflikt-Aktionen. Somit werden die laufenden Aktionen, die mit der anfragenden Aktion in Konflikt stehen als `actionsToAbort` zum Abbrechen zurückgegeben. Falls die Differenz $\leq$ ist, die Liste leer, weil es keine Aktionen zum Abbrechen gibt, sondern nur die anfragende Aktion verworfen wird.

```
1 if (diff > 0) {
2         actionsToAbort = runningConflicts;
3 }
```

## 4.4. Evaluation Single Request Konfliktlösung

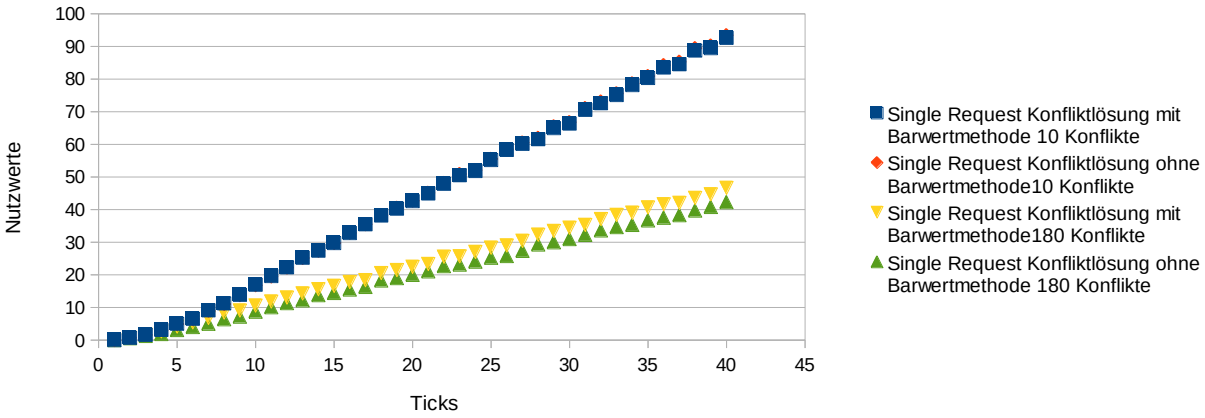


Abbildung 14.: Durchschnittlicher Gesamtnutzwert der SR Konfliktlösungen

Um die Gesamtnutzwerte der Konfliktlösungen mit Barwertmethode (mB) und ohne Barwertmethode (oB) gegenüberstellen zu können, wurden in der folgende Auswertung Konflikte und unterschiedliche Laufzeiten simuliert. Die Werte, aus denen sich die folgenden Diagramme ergeben und zusätzliche Auswertungen sind dem Anhang auf CD angefügt.
Der SR Simulator erstellt beliebig viele Aktionen und Konflikte und führt die Konfliktlösung für die SR Konfliktlösung mB und oB aus. Für diese Auswertung wurden 20 Aktionen erstellt und aus diesen wurden die Konflikte gebildet. Der prozentuale Anstieg der Nutzwerte ist mit 5 % angegeben. Im Durchschnitt wurde alle vier Ticks eine Anfrage einer Funktion gestellt. Das Diagramm aus 14 zeigt den durchschnittlichen Gesamtnutzwert der verschiedenen Konfliktlösungen, abhängig von der Konfliktanzahl und der Auswertungszeit. Die Auswertungszeit bezieht sich auf den Zeitrahmen für die Auswertung, angegeben in Ticks.
Das Diagramm aus Abbildung 14 zeigt den Gesamtnutzwert der SR Konfliktlösung mB und oB. Die Kurven der beiden Konfliktlösungen für 180 Konflikte zeigen deutlich, dass der Gesamtnutzwert der Konfliktlösung mB höher ist, als der Gesamtnutzwert der Konfliktlösung oB. Mit zunehmender Konfliktanzahl steigt der Unterschied des Gesamtnutzwertes mB, gegenüber dem Gesamtnutzwert oB. Der Kurve für 180 Konflikte aus dem Diagramm aus Abbildung 15 ist nur schwer zu entnehmen, dass die Anzahl der Aktionsabbrüche der Konfliktlösung mB immer niedriger ist, als die Anzahl der Aktionsabbrüche für die Konfliktlösung oB. Mit zunehmendem Anstieg der Auswertungszeit steigt die Anzahl der Aktionsabbrüche der Konfliktlösung oB stärker an, als die Aktionsabbrüche der Konfliktlösung mB. Die niedrigere Anzahl der Aktionsabbrüche ist der Grund für den

höheren Gesamtnutzwert der Konfliktlösung mB, gegenüber dem Gesamtnutzwert der Konfliktlösung oB.
Das Diagramm aus Abbildung 14 zeigt ebenfalls den Gesamtnutzwert der beiden Konfliktlösungen bei 10 Konflikten. Die Wahrscheinlichkeit, dass eine anfragende Aktion mit einer laufenden Aktion in Konflikt steht, ist bei 15 Aktionen und 10 Konflikten sehr gering. Bis zu einer Auswertungszeit von 17 Ticks erzielt die Konfliktlösung mB einen gering höheren Gesamtnutzwert, als die Konfliktlösung oB. Ab 19 Ticks ist der der Gesamtnutzwert der Konfliktlösung oB in geringem Maß höher, als der Gesamtnutzwert der Konfliktlösung mB. Obwohl das Diagramm aus Abbildung 15 deutlich zeigt, dass die Konfliktlösung mB weniger Aktionsabbrüche erzeugt als die Konfliktlösung oB, ist der Gesamtnutzwert der Konfliktlösung oB höher als der Gesamtnutzwert der Konfliktlösung mB. Der Grund für diesen Unterschied der Gesamtnutzwerte lässt sich anhand eines Konfliktbeispiels erklären.

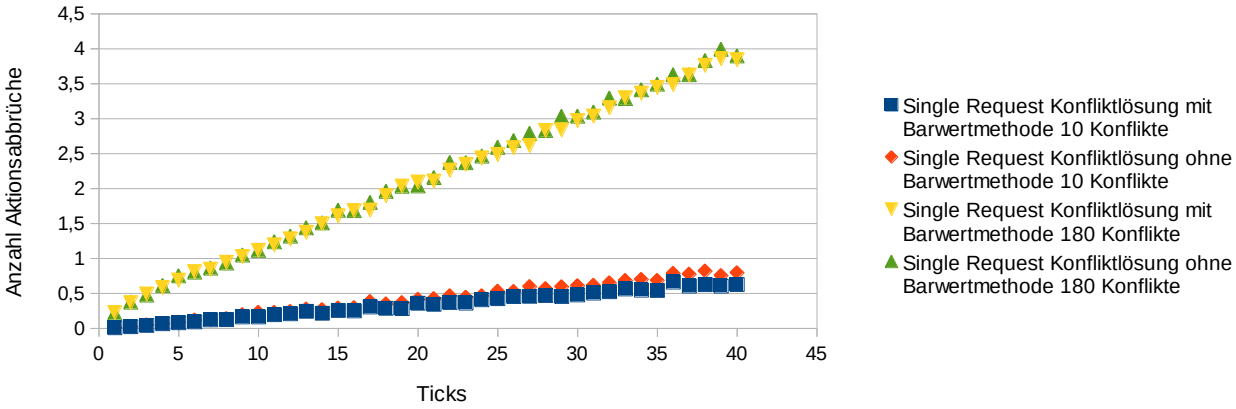


Abbildung 15.: Durchschnittliche Anzahl der Aktionsabbrüche der SR Konfliktlösungen

Z. B. wird eine SON1 Funktion mit einer Laufzeit von $t_{\text{eSON1}} = 10$ Ticks und einem Nutzwert von $v_{\text{SON1}} = 8$ ausgeführt. Eine SON2 Funktion mit einem Nutzwert von $v_{\text{SON2}} = 10$ und einer Laufzeit von $t_{\text{eSON2}} = 10$ Ticks fragt zum Zeitpunkt $t_5 = 5$ Ticks an ausgeführt zu werden. Die Auswertungszeit ist auf $t_{\text{end1}} = 10$ festgelegt. Abbildung 16 zeigt den erzielten Gesamtnutzwert $v_{\text{gesamtmB}}(t_{\text{end1}}) = 8$ der Konfliktlösung mB nach der Auswertungszeit an. Mit der Konfliktlösungsmethode oB im zweiten Teil der Abbildung 16 wird die SON1 Funktion abgebrochen und die SON2 Funktion gestartet. Die SON2 Funktion ist nach der Auswertungszeit $t_{\text{end1}} = 10$ noch nicht beendet und somit ist der erzielte Gesamtnutzwert der Konfliktlösung oB $v_{\text{gesamtoB}}(t_{\text{end1}}) = 0$. Daraus entsteht ein höherer Gesamtnutzwert der Konfliktlösungsmethode mB gegenüber dem Gesamtnutzwert der Konfliktlösung oB bei geringer Auswertungszeit. Wenn die Auswertungszeit auf $t_{\text{end2}} = 15$ erhöht wird, ist die SON2 Funktion am Ende der Auswertungszeit abgeschlossen und der Gesamtnutzwert $v_{\text{gesamtoB}}(t_{\text{end2}}) = 10$. Der Gesamtnutzwert der

Konfliktlösung mB bleibt gleich groß $v_{\text{gesamtmB}}(t_{\text{end1}}) = 8$. Bei höherer Auswertungszeit und geringer Konfliktanzahl erzielt die Konfliktlösung oB einen gering höheren Gesamtnutzwert gegenüber der Konfliktlösung mB. Dies ist der Fall, weil die Funktionen wegen der geringen Konfliktanzahl mit großer Wahrscheinlichkeit zu Ende laufen können und somit ist der Nutzwert, der am Ende der Laufzeit erzielt wird maßgebend.
Für $t_{\text{end1}} = 10$ gilt folgendes:

$$v_{\text{gesamtmB}}(t_{\text{end1}}) > v_{\text{gesamtoB}}(t_{\text{end1}}) \tag{59}$$

Für $t_{\text{end2}} = 15$ gilt folgendes:

$$v_{\text{gesamtmB}}(t_{\text{end2}}) < v_{\text{gesamtoB}}(t_{\text{end2}}) \tag{60}$$

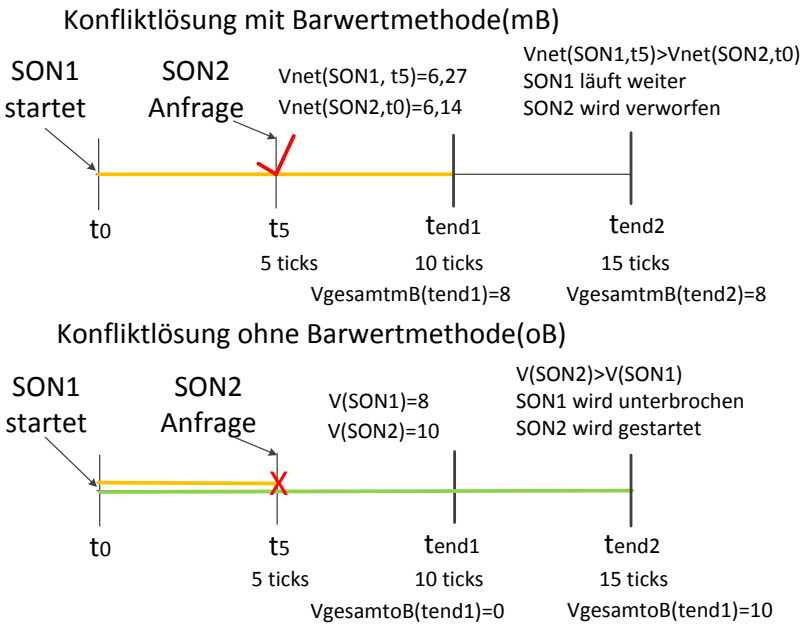


Abbildung 16.: SR Konfliktbeispiel bei geringer Konfliktanzahl

Das Verhältnis der Konfliktanzahl zur Aktionsanzahl ist ein wichtiges Maß, das den Gesamtnutzwert beeinflusst und entscheidet welche SR Konfliktlösung den höheren Gesamtnutzwert erzielt. Der SR Simulator ermöglicht es viele verschiedene Angaben über die Aktionen und Konflikte zu verarbeiten. Es ist möglich die Ober- und Untergrenzen der Nutzwerte und Ticks anzugeben. Die Anzahl der Aktionen und Konflikte können variiert werden. Der prozentuale Anstieg der Nutzwerte kann ebenfalls verändert werden. Die Häufigkeit einer Anfrage einer Funktion ist anzugeben und die Auswertungszeit wird ebenfalls bestimmt. Dem Anhang auf CD sind noch weitere Auswertungen der SR Konfliktlösung angefügt.

# 5. Schluss

## 5.1. Ergebnisse

### 5.1.1. Ergebnis Batch Request Konfliktlösung

Die Evaluation der verschiedenen BR Konfliktlösungen hat ergeben, dass die Konfliktlösung mit Constraint-Optimierung eine Maximierung des Gesamtnutzwertes erzielt hat. Dies liegt daran, dass der Constraint-Löser keine Reihenfolge von Aktionen oder Konflikten kennt, wohingegen die BR Konfliktlösung ohne Constraint-Optimierung abhängig von der Reihenfolge der Aktionen in der Batch ist und die RPS Konfliktlösung abhängig von der Reihenfolge der Konflikt-Sets ist. Die Evaluation 3.5 hat deutlich gezeigt, dass das Ziel der Maximierung der Gesamtnutzwerte mit Hilfe der Constraint-Optimierung erreicht wurde.

### 5.1.2. Ergebnis Single Request Konfliktlösung

Der erzielte Gesamtnutzwert der SR Konfliktlösung hängt von vielen verschiedenen Faktoren ab. Das Verhältnis von Aktionsanzahl zu Konfliktanzahl und die Auswertungszeit sind zwei wichtige Faktoren. Bei sehr geringer Konfliktanzahl und langer Auswertungszeit erzielt die SR Konfliktlösung oB einen gering höheren Gesamtnutzwert als die Konfliktlösung mB. Dies liegt daran, dass die Wahrscheinlichkeit eines Konfliktes gering ist und die Funktionen zu Ende ausgeführt werden können. Deshalb ist ein Vergleich der Nutzwerte, die erst am Ende der Ausführungszeit erzielt werden sinnvoll.
Bei hoher Konfliktanzahl erzielt die Konfliktlösung mB einen höheren Gesamtnutzwert als die Konfliktlösung oB. Die Berechnung des "Zeitwertes" der Nutzwerte verhindert in den meisten Fällen Funktionsabbrüche kurz vor dem Laufzeitende einer Funktion und deshalb ist der Gesamtnutzwert der Konfliktlösung mB höher als der Gesamtnutzwert der Konfliktlösung oB. Je höher die Auswertungszeit und Konfliktanzahl ist, desto größer wird die Differenz zwischen den Gesamtnutzwerten der Konfliktlösungen.

## 5.2. Ausblick

In der BR Konfliktlösung werden die Aktionen über einen bestimmten Zeitraum gesammelt und anschließend bestimmt der Constraint-Löser welche Aktionen den größten Gesamtnutzwert liefern. Man kann diese Methode als statische Konfliktlösung betrachten. Die SR Konfliktlösung löst Konflikte zwischen einer Menge laufender Funktionen und jeweils einer anfragenden Funktion. Die Konfliktlösung erfolgt während der Laufzeit.

Eine mögliche Weiterentwicklung der SON Konfliktlösung, wäre eine Kombination beider Konfliktlösungsmethoden. D. h. es könnten in festgelegten Zeitintervallen Events gesammelt werden und anschließend mit Hilfe der BR Konfliktlösung mit Constraint-Optimierung die Aktionen mit dem optimierten Gesamtnutzwert ermittelt werden. Anschließend würden diese Aktionen ausgeführt werden. Falls in dem Zeitraum der Ausführung der ausgewählten Aktionen eine Anfrage auftreten würde, würde die SR Konfliktlösung mB angewandt werden um mögliche Konflikt aufzulösen. Diese Kombination der BR und SR Konfliktlösung könnte eine mögliche Weiterentwicklung der SON Koordination sein.

# A. Anhang

## A.1. Abbildungsverzeichnis

## A.2. Abkürzungsverzeichnis

**SON** Self-Organising Networks
**CCO** Coverage and Capacity Optimisation
**MRO** Mobility Robustness Optimisation
**HO** Handover
**PBNM** Policy-based Network Management
**ECA** Event-Condition-Action
**RPS** Rational Policy System
**NMS** Netzmanagement System
**NM** Netzmanagement
**NE** Netzelemente
**UF** Utility Function
**AV** Aktionsvorschlag
**KE** Konflikterkennung
**AA** Aktionsauswahl
**Tx** Transmission
**ACK** Acknowledged
**NACK** Reject
**RB** Rollback
**or** operations research
**SWIG** Simplified Wrapper and Interface Generator
**CP** Constraint Programming
**API** Application Programming Interface
**KS** Konflikt-Set
**FM** Fault Management
**CM** Configuration Management
**AM** Accounting Management
**PM** Performance Management
**SM** Security Management
**OSS** Operations Support System
**BR** Batch Request
**SR** Single Request
**mB** mit Barwertmethode
**oB** ohne Barwertmethode

## A.3. Literaturverzeichnis


[Bun13] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen. Jahresbericht der Bundesnetzagentur 2012, 2013. http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Allgemeines/Bundesnetzagentur/Publikationen/Berichte/2013/Jahresbericht2012.pdf?__blob=publicationFile Abruf am 25.08.2013.

[Chr13] Christoph Frenzel, Henning Sanneck, Bernhard Bauer. Rational Policy System for Network Management, 2013. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6573076 Abruf am 01.07.2013.

[Eri12] Ericsson. Global mobile traffic: voice and data, 2010-2017, 2012. http://www.flickr.com/photos/ericsson_images/7341597996/in/set-72157630058081910 Abruf am 25.06.2013.

[Fre13] Christoph Frenzel. Utility-based SON Coordination, 2013. Interview via TelCo, geführt vom Sabrina Einberger. Augsburg, 15.August 2013.

[Inf11] Informationszentrum Mobilfunk e.V (IZMF). Hauptsenderichtung einer Mobilfunkantenne, 2011. http://www.izmf.de/de/content/hauptsenderichtung-einer-mobilfunkantenne Abruf am 20.07.2013.

[Net09] Nokia Siemens Networks. Self-Organizing Network (SON) Introducing the Nokia Siemens Networks SONSuite - an efficient, future-proof platform for SON, 2009. http://www.globaltelecomsbusiness.com/pdf/NSN_SON_WP_Revised_Oct2009.pdf Abruf am 25.07.2013.

[Nik13a] Nikolaj van Omme, Laurent Perron, Vincent Furnon. or-tools, 2013. https://code.google.com/p/or-tools/ Abruf am 25.06.2013.

[Nik13b] Nikolaj van Omme, Laurent Perron, Vincent Furnon. or-tools user's manual, 2013. http://or-tools.googlecode.com/svn/trunk/documentation/user_manual/index.html Abruf am 25.06.2013.

[Oli13a] Oliver Glück. Abzinsung, 2013. http://www.welt-der-bwl.de/Abzinsung Abruf am 02.09.2013.

[Oli13b] Oliver Glück. Barwertberechnung, 2013. http://www.welt-der-bwl.de/Barwert Abruf am 02.09.2013.

[PH07] Armin Wolf Petra Hofstedt. *Einführung in die Constraint-Programmierung.* Springer-Verlag Berlin Heidelberg, 2007.

[Rap13] Raphael Romeikat, Henning Sanneck, Tobias Bandh. Efficient, Dynamic Coordination of Request Batches in C-SON Systems, 2013. http://sanneck.net/research/publications/papers/IWSON%20batch%20request%20coordination.pdf Abruf am 25.06.2013.

[Ric12] Richard Waldhauser, Markus Staufer, Seppo Hämäläinen, Henning Sanneck, Haitao Tang, Christoph Schmelz, Jürgen George, Paul Stephens, Krzystof Kordybach, Clemens Suerbaum. Self-Organising Networks (SON). In Seppo Hämäläinen, Henning Sanneck, Cinzia Sartori, editor, *LTE Self-Organising Networks(SON) : Network Management Automation for Operational Efficiency*. England: John Wiley & Sons Ltd, 2012.

[Ron09] Ronen Brafman, Carmel Domshlak. Preference Handling - An Introductory Tutorial, 2009. https://www.aaai.org/ojs/index.php/aimagazine/article/view/2114/2065 Abruf am 12.09.2013.

[Tob12] Tobias Bandh, Haitao Tang, Henning Sanneck, Christoph Schmelz. Self-Organising Networks (SON). In Seppo Hämäläinen, Henning Sanneck, Cinzia Sartori, editor, *LTE Self-Organising Networks(SON) : Network Management Automation for Operational Efficiency*. England: John Wiley & Sons Ltd, 2012.

[Wik13a] Wikipedia, Die freie Enzyklopädie. Cardinal utility, 2013. http://en.wikipedia.org/wiki/Cardinal_utility Abruf am 03.09.2013.

[Wik13b] Wikipedia, Die freie Enzyklopädie. FCAPS, 2013. http://en.wikipedia.org/wiki/FCAPS Abruf am 02.09.2013.

[Wik13c] Wikipedia, Die freie Enzyklopädie. Handover, 2013. http://de.wikipedia.org/wiki/Handover Abruf am 13.09.2013.

[Wik13d] Wikipedia, Die freie Enzyklopädie. Level of measurement, 2013. http://en.wikipedia.org/wiki/Ordinal_scale#Ordinal_scale Abruf am 03.09.2013.

[Wik13e] Wikipedia, Die freie Enzyklopädie. Netzwerkressource, 2013. http://de.wikipedia.org/wiki/Netzwerkressource Abruf am 15.09.2013.

## A.4. Inhaltsverzeichnis CD



Hinweis: Die Daten der CD-ROM werden nicht veröffentlicht, können aber ggf. bei der Autorin angefordert werden.

Erklärung

Ich versichere, dass die Bachelorarbeit von mir selbständig verfasst wurde und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Zitate habe ich klar gekennzeichnet.

| Datum | Name, Vorname | Matrikelnummer | Unterschrift |
|---|---|---|---|