Universität des Saarlandes
Naturwissenschaftlich-Technische Fakultät I
Fachrichtung Informatik
Master-Studiengang Informatik

Masterarbeit

Quantenbasierte Koordination von Multiagentensystemen

vorgelegt von

Stefan Nesbigall

am 14.01.2008

angefertigt unter der Leitung und Betreuung von
Dr. rer. nat. Matthias Klusch
Deutsches **F**orschungszentrum für **K**ünstliche **I**ntelligenz
Saarbrücken, Deutschland

begutachtet von
1st: Prof. Dr. Jörg Siekmann
2nd: Prof. Dr. Michael Backes

Erklärung
Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und alle verwendeten Quellen angegeben habe.

Saarbrücken, den 14.01.2008

Stefan Nesbigall

Einverständniserklärung
Hiermit erkläre ich mich damit einverstanden, dass meine Arbeit in den Bestand der Bibliothek der Fachrichtung Informatik aufgenommen wird.

Saarbrücken, den 14.01.2008

Stefan Nesbigall

Danksagung

An dieser Stelle möchte ich die Gelegenheit nutzen und einigen Personen danken.

Professor Siekmann für die Erstkorrektur und seine ansteckende Begeisterung für die künstliche Intelligenz.

Professor Backes für die Zweitkorrektur.

Meinem wissenschaftlichen Betreuer Dr. Matthias Klusch für die Vergabe des Themas der Arbeit, seinen vielseitigen Rat, sowie konstruktive Kritik und lange Diskussionen.

Rene Schubotz für die Hilfe beim Einstieg in die Welt der Quanteninformatik.

Den üblichen Verdächtigen für die moralische Unterstützung.

Nicht zuletzt meinen Eltern, Alfons und Dorette Nesbigall, für das Ermöglichen meines Studiums, die finanzielle Unterstützung und den Rückhalt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Liste der Algorithmen

Kapitel 1

Einleitung

Es gab eine schnelle technologische Entwicklung im Bereich der Informatik in den letzten 50 Jahren, die es ermöglichte viele Rechenprobleme auf eine effiziente Art und Weise zu lösen. Trotzdem gibt es immer noch Probleme, für die eine effiziente Lösung noch nicht bekannt ist. Es stellte sich heraus, dass die Quantenmechanik zum Lösen von verschiedenen Problemen im Bereich der Logistik, Bioinformatik und Kryptographie erweitert werden kann. Feymann [Fey82] war der erste, der die Frage formulierte, welchen Effekt das Verhalten eines Quantensystems auf die Berechnung auslösen konnte. Er behauptete, die Simulation eines Quantensystems sei auf einem klassischen Computer nicht effizient durchführbar und bemerkte, dass hierfür ein "Quantencomputer" verwendet werden soll. Daraus entwickelte Deutsch [Deu84] ein quantenrechnendes Modell einer Turingmaschine (QTM) und stellte die Frage in den Raum, ob die QTM mehr Möglichkeiten aufweist, als eine klassische Turingmaschine. Viele Wissenschaftler nahmen sich der Frage an. Das bemerkenswerteste Ergebnis ist wohl der Algorithmus zum Faktorisieren von Peter Shor [Sho94].
Seit Anfang der 90er Jahre werden auch Erfolge bei der physikalischen Umsetzung von Quantencomputern verzeichnet, so dass für die Jahre 2020 bis 2030 von vielen einen regelmäßigen, unterstützenden Einsatz von Quantencomputern vorausgesagt wird.
Die Möglichkeit zur Verschränkung und Verteilung von Partikeln bietet eine neue Ressource im Bereich gruppenbasierter Entscheidungsfindung. In [Klu04] wird sich mit der Frage beschäftigt, ob intelligente Softwareagenten einen Vorteil durch das Potential der Quantenrechnung und Quantenkommunikation erhalten, wenn Quantencomputer real verfügbar sind. Darin werden Grundlagen und erste Antworten zu Quantenagenten (QCA) und Quantenmultiagentensysteme (QCMAS) gegeben. Aufbauend darauf wird in [Sch07] eine Quantenerweiterung zu der Agentenarchitektur InteRRap [MP93] beschrieben: QuantumInteRRap. Damit wird ein Suchagent basierend auf Quanten Pattern Matching umgesetzt.
Auf dieser Basis wird in dieser Arbeit die Möglichkeiten untersuchen, Probleme aus dem Bereich der Multiagentensysteme mit Quantenagenten zu lösen. Dabei wird der Schwerpunkt auf die Kommunikation und Koordination der Agenten gelegt. Ziel ist es erste Quantenversionen für die klassischen Protokollschritte zu erhalten und diese auf eine eventuelle Verbesserung zu prüfen.

Dazu wurden in dieser Arbeit zwei klassische Probleme (Verhandlung von Kernel-stabilen Koalitionen [KS96], [BK04] und Kontraktnetzsysteme [Smi80], [San93]) untersucht. Für das erste Problem wurden mit Hilfe von Quantenalgorithmen Verbesserungen der Komplexität

im Bereich der lokalen Berechnungen und Kommunikation erreicht. Bei den Verbesserungen handelt es sich um einen quadratischen Faktor, sowie zusätzliche Eigenschaften, wie echte Zufälligkeit bei der Wahl eines Koalitionsführers. Für das Kontraktnetzsystem wurde ein Quantenprotokoll, bestehend aus Quantenauktionen, umgesetzt. Dabei wurden keine nennenswerten Verbesserungen in der Komplexität erreicht, jedoch wird durch die Quantenauktionen die Vertraulichkeit der Gebote selbst gegenüber dem Auktionär bewahrt und durch Quantenbroadcasts die Sicherheit der Übertragungen gewährleistet.

In Kapitel 2 werden die Grundlagen beschrieben die zum Lesen und das Verständnis der Arbeit nötig sind. Zuerst wird in Abschnitt 2.1 die klassische Koordination von Multiagentensystemen betrachtet und die Grundlagen, der in der Arbeit untersuchten Protokolle, vorgestellt. Abschnitt 2.2 beschreibt die Quantenrechnung und Quantenkommunikation. Besonders Abschnitt 2.2.3 ist für das Verständnis von Kapitel 3.2 nötig. In Kapitel 3 wird der Algorithmus für Kernel-stabile Koalitionsbildung betrachtet, und in Bezug auf eine Quantenversion analysiert. Diese wird für die klassisch aufwendigsten Schritte in Abschnitt 3.2 gegeben, in 3.2.3 analysiert und in 3.3 theoretisch ausgewertet. Das Ergebnis wird in 3.4 diskutiert. Kapitel 4 setzt sich mit der verteilten Problemlösung, genauer mit dem TRACONET Protokoll auseinander, um auch hier eine Quantenversion zu finden. Schwerpunkt hier liegt auf der Kommunikation und Koordination der Agenten. In Abschnitt 4.2.1 werden Ansätze beschrieben und diese in Abschnitt 4.2.2 zu einem Quantenprotokoll umgesetzt. Das Ergebnis wird in Abschnitt 4.3 theoretisch evaluiert und in 4.4 diskutiert. In Kapitel 5 wird auf die Implementierung der Quantenversion von Kapitel 3 und Kapitel 4 eingegangen. In Kapitel 6 werden die Ergebnisse der Arbeit noch einmal zusammengefasst und Ansätze für weitere Forschungen gegeben.

Kapitel 2

Relevante Arbeiten

Dieses Kapitel legt die Grundlagen, die für das Verständnis der weiteren Arbeit wichtig sind. Es ist nicht unbedingt nötig es zu lesen. In 2.1 wird auf die Koordination in klassischen Multiagentensystemen eingegangen, in 2.2 die Grundlagen der Quantenrechnung gelegt. In Abschnitt 2.3 werden die praktischen Seiten der Quantenmechanik dargelegt und in 2.4 Ansätze einer Verbindung von Multiagentensystemen und Quantenrechnung gegeben.

2.1 Koordination

In diesem Abschnitt werden die Koordination von Multiagentensystemen, sowie die klassischen Grundlagen der Spieltheorie und der verteilten Problemlösung gegeben.

2.1.1 Koordination in Multiagentensystemen

Für die Zusammenarbeit von Agenten in einem Multiagentensystem (MAS) sind Koordination, Kommunikation und Verhandlung Voraussetzung. Um ein Problem zu lösen, müssen die Agenten untereinander Informationen über ihre Aktivitäten austauschen können. Dies wird über Kommunikation erreicht. Sie müssen auch ihre Handlungen aufeinander abstimmen und, falls Konflikte auftreten, diese über Verhandlungen lösen. Konflikte können zum Beispiel auftreten, wenn mehrere Agenten versuchen, auf die gleiche Ressource zuzugreifen. Koordination wird deshalb benötigt, um eine Organisationsstruktur für eine Menge von Agenten zu errichten.

Koordination ist essenziell für MAS, ohne sie gehen alle Vorteile der Interaktion verloren und die Gruppe von Agenten degeneriert zu einer Gruppe von Individuen mit einem chaotischen Verhalten.
In [HBWC99, 1.5] wird Koordination als ein Prozess beschrieben, der sicher stellt, dass eine Gemeinschaft individueller Agenten auf eine harmonische und koherente Weise interagieren.
Green [GHN+97, 4.3] gibt einige Gründe dafür, warum mehrere Agenten Koordination benötigen:

- Um Chaos zu vermeiden. Im allgemeinen besitzt kein Agent eine komplette Übersicht des Systems, da dies, wegen der Komplexität, einfach nicht durchführbar wäre. Also hat jeder Agent eine lokale Sicht von Zielen, Handlungen und Wissen, die die Handlungen anderer Agenten eher behindert als unterstützt.

- Um globale Bedingungen zu erfüllen. Agenten, die Netzwerkmanagement betreiben, haben für manche Fehler nur Sekunden zum Antworten, für andere Stunden. Das Koordinieren von Agenten ist essentiell, um solche globalen Bedingungen zu erfüllen.
- Agenten in MAS besitzen verschiedene Fähigkeiten und verschiedenes Fachwissen. Aus diesem Grund müssen Agenten koordiniert werden, damit verschiedene Spezialisten ihre Fähigkeiten aufeinander abstimmen und kombinieren können.
- Die Handlungen eines Agenten sind manchmal von denen eines anderen Agenten abhängig. So muss ein Agent zum Beispiel auf einen anderen warten, bevor er eine Aufgabe abschließen kann.

In der Spieltheorie mit mehreren Spielern und der verteilten Problemlösung hat die Koordination großen Einfluss.

2.1.2 Spieltheorie

Die Spieltheorie ist ein Teilgebiet der Mathematik, sie analysiert Situationen, in denen mehrere Spieler Entscheidungen treffen, um ihren Gewinn zu maximieren. Durch die Spieltheorie lassen sich Entscheidungsprobleme im Bereich der Soziologie, der Psychologie, den Wirtschaftswissenschaften, den Politikwissenschaften, der Informatik und anderen formal modellieren und analysieren. Im folgenden werden die Grundlagen beschrieben, die für diese Arbeit von Bedeutung sind. Diese Grundlagen sind aus [HI91, Kapitel 6] und [KR84, Kapitel 2].

Koalitionsspiele

Koalitionsspiele (manchmal auch als Kooperationsspiele bezeichnet) sind Spiele, in denen sich Spieler (Agenten) zu Gruppen zusammenschließen. Der "Wettkampf" findet somit nicht zwischen individuellen Spielern, sondern zwischen Gruppen von Spielern statt.
Ein Koalitionsspiel $(\mathbf{A}, v)$ besteht aus einer Menge $\mathbf{A}$ von Spielern (Agenten) $a_i, i \in \{1, \ldots, n\}$ und einer charakteristischen Funktion v.
Die Spieler bilden eine Koalitionsstruktur, die durch die Menge $\mathbf{S} = \{\mathbf{C_1}, \ldots, \mathbf{C_m}\}$ von m Koalitionen beschrieben werden kann. $\mathbf{S}$ besteht dabei aus Teilmengen von $\mathbf{A}$ und erfüllt drei Bedingungen:

$$\mathbf{C_j} \neq \emptyset, j = 1, \ldots, m \tag{2.1}$$

$$\mathbf{C_i} \cap \mathbf{C_j} = \emptyset, \forall i \neq j \tag{2.2}$$

$$\bigcup_{\mathbf{C_j} \in \mathbf{S}} \mathbf{C_j} = \mathbf{A} \tag{2.3}$$

Diese besagen, dass jeder Spieler in genau einer der m nicht-leeren Koalitionen der Struktur ist.

Charakteristische Funktion

In einem kooperativen n-Personen Spiel ist v eine reellwertige Funktion, die auf einer Teilmenge von $\mathbf{A}$ definiert ist, und charakteristische Funktion heißt. Sie teilt jeder Teilmenge $\mathbf{C} \subseteq \mathbf{A}$ einen reellen Wert zu. Dieser Wert wird als Gewinn gesehen, den die Spieler von $\mathbf{C}$

erreichen, wenn sie zusammenarbeiten. Die Funktion $v(\emptyset)$ ist immer 0. Die charakteristische Funktion hat folgende Eigenschaften:

- Der Wert $v(\mathbf{C})$ einer Koalition ist als Gewinn zu sehen, den es zu maximieren gilt.
- Eine Koalition $\mathbf{C}$ wird geformt, indem eine bindende Übereinkunft getroffen wird, die die Aufteilung von $v(\mathbf{C})$ auf die Mitglieder in $\mathbf{C}$ regelt.
- $v(\mathbf{C})$ ist unabhängig von den Handlungen der anderen Spieler $\mathbf{A} \setminus \mathbf{C}$.

Auszahlung

Jedes Spiel gelangt in eine Art Endzustand, in dem ein Resultat des Spiels hervorgeht. Selbst bei Spielen, die eine nicht vorbestimmte Anzahl von Runden gehen, kann ein solches Resultat analysiert werden, indem die Interaktion in diskrete Teilhandlungen aufgespaltet und diese einzel betrachtet werden. Resultate sind verbale Ausdrücke. Zur Betrachtung von Koalitionsspielen, muss dieser Begriff eine quantitative Repräsentation bekommen. Diese Repräsentation eines Resultats für einen Spieler wird Auszahlung genannt. Ein Spieler i erhält am Ende eines Spiels oder als Zwischenstand eine Auszahlung u_i. Die Auszahlung aller Spieler wird als Zeilenvektor $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ einer nummerierten Menge von Spielern ausgedrückt.

Auszahlungskonfiguration

Die Auszahlungskonfiguration (AK) eines kooperativen n-Personen Spiels ist das Paar $(\mathbf{S}, \mathbf{u})$, bestehend aus einer Koalitionsstruktur $\mathbf{S}$ und einem Auszahlungsvektor $\mathbf{u}$. Die Konfiguration beinhaltet somit alle wichtigen Informationen über den Ausgang eines Spiels. Für die AK gelten folgende Eigenschaften:

$$(\mathbf{S}, \mathbf{u}) = (\{\mathbf{C_1}, \ldots, \mathbf{C_m}\}, (u_1, u_2, \ldots, u_n)) \tag{2.4}$$

$$u(\mathbf{C_j}) \equiv \sum_{i \in \mathbf{C_j}} u_i = v(\mathbf{C_j}), \forall j = 1, \ldots, m \tag{2.5}$$

Die zweite Eigenschaft besagt, dass jede Koalition genau den Wert der charakteristischen Funktion auf ihre Mitglieder verteilt.

Eine stabile Lösung eines Kooperationsspiels ist eine Menge von AKs. Der Begriff Stabilität basiert dabei auf einer bestimmten Koalitionstheorie die angewendet wird. Bekannte Koalitionstheorien sind der Kern, stabile Mengen, Verhandlungsmengen und der Kernel [DM65].

Der Kernel

Der Kernel $\mathbf{K}$ ist eine Menge von AKs, in welchen die Koalitionsstruktur stabil ist, bezogen auf ein Gleichgewicht von allen Spielerpaaren, die in der gleichen Koalition sind.

$$\mathbf{K} := \{(\mathbf{S}, \mathbf{u}) \mid \forall a_k, a_l \in \mathbf{C} \in \mathbf{S} : (a_k, a_l) \text{ sind im Gleichgewicht}\} \tag{2.6}$$

Überschuss

Für ein Spiel $(\mathbf{A}, v)$ und eine AK $(\mathbf{S}, \mathbf{u})$ gibt es einen Vergleichswert Überschuss $e(\mathbf{R}, \mathbf{u})$, der herangezogen wird, falls eine Menge von Spielern ihre Koalitionen aus $\mathbf{S}$ aufbrechen und eine neue Koalition $\mathbf{R}$ bilden. Er ist wie folgt definiert:

$$e(\mathbf{R}, \mathbf{u}) \quad = \quad v(\mathbf{R}) - u(\mathbf{R}) \tag{2.7}$$

Der Überschuss bezeichnet somit den gesamten Grad an Gewinn oder Verlust, den die Mitglieder durch die Bildung von $\mathbf{R}$ erfahren.

Einwandspotential

Das Einwandspotential (Surplus) s_{kl} von einem Agenten a_k über einen Agenten a_l in der gleichen Koalition $(a_k, a_l \in \mathbf{C} \in \mathbf{S})$ mit Rücksicht auf eine Nutzenkonfiguration $(\mathbf{S}, \mathbf{u})$ ist das Maximum des Überschusses, den a_k in alternativen Koalitionen erzielen kann, die a_k aber nicht a_l enthalten.

$$s_{kl} := \max_{\mathbf{R} \notin \mathbf{S}, a_k \in \mathbf{R}, a_l \notin \mathbf{R}} e(\mathbf{R}, \mathbf{u}) \tag{2.8}$$

Gleichgewicht

Ein Paar von Agenten $a_k, a_l \in \mathbf{C} \in \mathbf{S}$ ist in Gleichgewicht, genau dann wenn

$$(s_{kl} = s_{lk} \vee (s_{kl} > s_{lk} \wedge u_l = v(\{a_l\})) \quad \vee \quad (s_{kl} < s_{lk} \wedge u_k = v(\{a_k\}))) \tag{2.9}$$

Angebote und Nachfragen

Im Rahmen dieser Arbeit basiert die charakteristische Funktion auf den Angeboten und Nachfragen der einzelnen Spieler (Agenten). Jeder Agent a besitzt eine Menge $\mathbf{T}_a$ von Nachfragen, dh. Aufgaben, die er abarbeiten muss und eine Menge $\mathbf{O}_a$ von Aufgaben, die er im Stande ist selbst abzuarbeiten. Beide Arten von Aufgaben besitzen einen Wert, den ein Nutzer des Agenten entweder bereit ist zu zahlen w_a (im Falle von einer Aufgabe $\in \mathbf{T}_a$), oder Kosten c_a, die bei der Ausführung der Aufgabe entstehen (im Falle von einer Aufgabe $\in \mathbf{O}_a$).

Lokaler Wert

Der lokale Wert $lw_a(\mathbf{C})$ eines Agenten a im Hinblick einer Koalition $\mathbf{C}$ ist die Differenz zwischen den in Rechnung gestellten Nachfragen, die der Agent entweder selbst oder mit Hilfe der anderen Agenten der Koalition ausgeführt hat, und den Kosten für das Abarbeiten der Angebote des Agenten für andere Agenten der Koalition.
Es sei $\mathbf{E}_a(\mathbf{C})$ die Menge der Aufgaben in $\mathbf{O}_a$, die von anderen Agenten der Koalition $\mathbf{C}$ benötigt werden und $\mathbf{R}_a$ die Menge der Aufgaben in $\mathbf{T}_a$, die von anderen Agenten der Koalition $\mathbf{C}$ gelöst werden können.

$$lw_a(\mathbf{C}) := \sum_{x \in \mathbf{R}_a(\mathbf{C})} w_a(x) - \sum_{x \in \mathbf{E}_a(\mathbf{C})} c_a(x) \tag{2.10}$$

Im folgenden ergibt sich der Wert für die charakteristische Funktion wie folgt:

$$v(\mathbf{C}) := \sum_{a \in \mathbf{C}} lw_a(\mathbf{C}) \tag{2.11}$$

2.1.3 Verteiltes Problemlösen

In Sycara [SDP+96, Kapitel 3] werden Gründe angegeben, die gegen den Einsatz eines einzelnen Agenten für bestimmte Probleme sprechen. In [BZW98, Kapitel 4.3] wird der Einsatz von Agenten zur verteilten Problemlösung motiviert. Viele Probleme sind wegen ihrer Form verteilter Natur und benötigen verteilte Lösungseinheiten. So sind zum Beispiel spezielle Informationen nicht zentral an einer Stelle zu beziehen. Selbst wenn ein Agent ein Problem alleine lösen kann, gibt es jedoch meist einen Engpass, was Geschwindigkeit, Verlässlichkeit und Modularität der Lösung angeht.
Wenn mehrere Agenten an einem Problem arbeiten, erfordert dies eine Methodik, die für den Lösungsprozess verwendet wird. Dies setzt die Fähigkeit der Kommunikation und Kooperation zwischen den einzelnen Agenten voraus. Eine solche Methodik kann in drei Schritte aufgeteilt werden: Aufteilung des Problems in Teilprobleme, Lösen der Teilprobleme und Kombination der Lösungen. Die Teilprobleme müssen auf geeignete Agenten verteilt werden, die auch die Fähigkeit besitzen diese zu lösen. Dabei können Schwierigkeiten in Form von Konflikten entstehen, wenn mehrere Agenten fähig sind ein Teilproblem zu lösen. In diesem Fall helfen Kooperationsprotokolle, wie das Kontraktnetzsystem, eine optimale Aufteilung zu finden.

Kooperationsprotokolle

Verschiedene Formen der Kooperation werden in [DFJN97] diskutiert. Dabei werden kooperative MAS mit Kommunikation in beratende und verhandelnde unterteilt. [BZW98, Kapitel 4.3.4] gibt als Beispiel für beratende Kommunikation Partial Global Planning (PGP) [DL91], als Beispiel für verhandelnde Kommunikation das Kontraktnetzsystem an.

Kontraktnetzsysteme

Ein Kontraktnetzsystem ist ein Konzept, das für die effiziente Koordination von einer Menge von interagierenden Agenten in einem MAS verwendet werden kann. Ein Kontraktnetz besteht aus einer Anzahl von Knoten, die durch die einzelnen Agenten dargestellt werden. In einer Methodik ähnlich zu einem Handelssystem, werden die Teilprobleme als Waren angeboten, für die Agenten Gebote abgeben können. Die Aufgabenverteilung ist ein Prozess an dem alle Agenten beteiligt sind, mit dem Ziel, die vorhandenen Ressourcen und Informationen der einzelnen Agenten so effizient wie möglich zu nutzen. Dies wird erreicht, indem die Teilprobleme an den Agent, mit dem zur Zeit besten Gebot, weitergegeben werden.
Die Basis der Kontraktnetzsysteme wird durch das Kontraktnetzprotokoll [Smi80] gelegt. Dieses bietet eine Lösung für das so genannte Verbindungsproblem (connection problem): Das Finden eines geeigneten Agenten für das Abarbeiten einer bestimmten Aufgabe. Ein Agent,

der eine Aufgabe gelöst haben möchte, heißt *Manager*. Ein Agent, der die Aufgabe lösen kann, ist ein potentieller *Auftragnehmer*. Agenten nehmen diese Rollen dynamisch für einzelne Aufträge an, dass heißt jeder Agent kann eine Aufgabe als Manager ausschreiben bzw. sich für das Lösen ausgeschriebener Aufgaben bewerben.
Die Verhandlungen besitzen vier Eigenschaften:

- Sie sind ein lokaler Prozess, der nicht von einer zentralen Stelle beeinflusst wird.
- Sie beinhalten einen Informationsaustausch in beide Richtungen.
- Jede Seite der Verhandlungen beurteilt die Informationen aus ihrer eigenen Sicht.
- Einigungen werden nur mit gemeinsamer Übereinkunft getroffen.

Kontraktnetzprotokoll

Um einen Kontrakt abzuschließen treten die einzelnen Agenten in Verhandlungen. Dabei tauschen sie eine Reihe von Informationen aus, die in einer Serie von Nachrichten verteilt werden. Diese bestehen aus den verschiedenen Schritten:

- Aufgabenausschreibung
- Evaluation der Ausschreibung
- Bieten
- Evaluation der Gebote
- Kontraktabarbeitung und Ergebnisverkündung

Aufgabenausschreibung

Ein Agent, der eine auszuführende Aufgabe besitzt, initialisiert Verhandlungen über diese Aufgabe, indem er anderen Agenten mit der Ausschreibungsnachricht die Existenz der Aufgabe mitteilt. Er nimmt für diese Aufgabe die Rolle des Managers ein. Die Nachricht enthält verschiedene Informationen über die Aufgabe, darunter den Sender der Aufgabe, eine Aufgaben Id, eine kurze Aufgabenbeschreibung, Voraussetzungen, die für das Bieten erfüllt sein müssen und eine Deadline für eingehende Gebote. Sie kann entweder an alle Agenten im Netz (allgemeiner Broadcast) oder an eine ausgewählte Teilmenge (begrenzter Broadcast) gesendet werden.

Evaluation der Ausschreibung

Jeder, der die Ausschreibung erhält, nimmt für diese Aufgabe die Rolle des Auftragnehmers ein. Ein Auftragnehmer untersucht in diesem Schritt, ob er für die Aufgabe geeignet ist. Falls dies der Fall ist sortiert er die Aufgabe in eine Liste von Aufgaben ein, für die er Bieten will. Die Einordnung wird anhand unterschiedlicher Faktoren und mit den Informationen aus der Ausschreibung getroffen.

Bieten

Wenn ein Agent eine Aufgabe abgearbeitet hat, beginnt er wieder mit dem Kontraktprozess. Dazu sucht er sich aus der Liste der eingegangenen und evaluierten Aufgaben eine aus. Er gibt ein Gebot ab, falls entweder eine neue Aufgabe ausgeschrieben wird oder kurz bevor die Deadline einer Aufgabe erreicht wurde. Ein Gebot beinhaltet unter anderem auch Informationen über die Fähigkeiten des Agenten, die für die Gebotsevaluation ausschlaggebend sind.

Evaluation der Gebote

Der Manager speichert alle eingegangen Gebote für eine Aufgabe. Wenn er ein Gebot erhält, ordnet er dieses in die Liste der eingegangenen Gebote ein. Falls das eingegangene Gebot zufriedenstellend ist kann er dieses auswählen. Falls die Deadline der Aufgabe erreicht, diese aber noch nicht vergeben wurde, bestehen verschiedene Möglichkeiten:

- Die Aufgabe wird an das beste Gebot vergeben (falls ein akzeptables Gebot existiert).
- Sie wird erneut ausgeschrieben (falls keine Gebote eingegangen sind).
- Die Deadline wird erweitert (falls kein akzeptables Gebot existiert).

Wird eine Aufgabe vergeben, informiert der Manager den Auftragnehmer über den bestehen Kontrakt. Die Nachricht enthält auch Informationen, die für die Ausführung der Aufgabe nötig sind.

Kontraktabarbeitung und Ergebnisverkündung

Der Auftragnehmer arbeitet die im Kontrakt stehende Aufgabe ab. Dies kann er entweder lokal bewerkstelligen, oder indem er die Aufgabe bzw. Teilaufgaben davon im Kontraktnetz ausschreibt. Es besteht weiterhin die Möglichkeit mit dem Manager zu kommunizieren. Das Resultat der Aufgabe wird an den Manager zurückgesendet.

2.2 Quantenrechnung und Quantenkommunikation

In diesem Abschnitt werden die Grundlagen der Quantenmechanik und Quanteninformation aus [NC00, 1.2, 1.3, 1.4, 2.2, 2.3, 4] beschrieben. In den Abschnitten 2.2.3 und 2.2.5 werden einige Quantenschaltkreise und Algorithmen beschrieben, die im weiteren Verlauf dieser Arbeit verwendet werden.

2.2.1 Einführung

Im folgenden wird hier eine kurze Einführung in die Quantenmechanik gegeben. Dazu werden Grundlagen erklärt, auf denen die Quantenmechanik basiert.

Quantenbit

Das Analogon zu einem Bit in klassischen Rechnern ist das Quantenbit (Qubit) in der Quantencomputation. Der Unterschied zwischen einem Qubit und einem klassischen Bit ist, dass

während ein klassisches Bit entweder den Zustand 1 oder den Zustand 0 hat, das Qubit auch andere Zustände annehmen kann. Es ist dem Qubit möglich eine Linearkombination der klassischen Zustände (Superposition)

$$|\psi\rangle \quad = \quad \alpha|0\rangle + \beta|1\rangle \tag{2.12}$$

anzunehmen. Die Werte α und $\beta \in \mathbb{C}$ werden Amplituden genannt und besitzen die Normalisierungseigenschaft:

$$|\alpha|^2 + |\beta|^2 \quad = \quad 1 \tag{2.13}$$

Also ist der Zustand eines Qubits ein Vektor im 2-dimensional komplexen Hilbertraum $H_2 = \mathbb{C}^2$. Die Zustände $|0\rangle$ und $|1\rangle$ heißen Eigenzustände und bilden eine Orthonormalbasis im Vektorraum.

Quantenregister

Um einen Zustand aus mehreren Qubits $|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_n\rangle$ zu beschreiben, wird ein neuer Zustandsraum gebildet, der sich aus dem Tensorprodukt der Einzelzustände $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$ ergibt. Solch ein Zustand wird auch als Quantenregister $|\psi\rangle$ mit 2^n Eigenzuständen bezeichnet. Jeder Multiqubitzustand ist so wieder eine Linearkombination

$$\begin{aligned} |\psi\rangle &= |\psi_1 \ldots \psi_n\rangle \\ &= \sum_{k=0}^{2^n-1} \alpha_k |k\rangle \end{aligned} \tag{2.14}$$

und erfüllt die Normalisierungseigenschaft:

$$\sum_{k=0}^{2^n-1} |\alpha_k|^2 \quad = \quad 1 \tag{2.15}$$

Messungen

Quantenmessungen werden durch eine Menge M_m von Messoperatoren beschrieben. Der Index m weist dabei auf ein mögliches Ergebnis hin. Die Wahrscheinlichkeit, dass ein Quantenzustand $|\psi\rangle$ zum Wert des Eigenzustandes $|m\rangle$ gemessen wird, ist gegeben durch:

$$p(m) \quad = \quad \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{2.16}$$

Der Zustand des Systems ist dann:

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}} \tag{2.17}$$

Die Messoperatoren müssen die Vollständigkeitsgleichung erfüllen. Diese drückt aus, das die Wahrscheinlichkeiten aller Messoperatoren zu 1 summieren.

$$\begin{aligned} 1 &= \sum_m p(m) \\ &= \sum_m \langle\psi|M_m^\dagger M_m|\psi\rangle \end{aligned} \tag{2.18}$$

Verschränkung

In Abschnitt 2.2.1 wurde gesagt, dass ein Multiqubitzustand aus mehreren Einzelzuständen mittels des Tensorproduktes erstellt werden kann. Es gibt aber auch Multiqubitzustände, die nicht durch ein Tensorprodukt aus Einzelzuständen erreicht werden können. Diese Zustände nennt man *verschränkt*.
Verschränkung ist eine Form miteinander verbundener Qubits. Dabei können die Quantenpartikel räumlich voneinander getrennt sein. Ihr Zustand ist trotzdem auf eine Art miteinander verbunden, so dass nach der Messung des einen Quantenpartikels auch auf den Zustand des anderen geschlossen werden kann. Um zwei Qubits miteinander zu verschränken, müssen sie beide miteinander interagieren. Danach können sie getrennt werden, behalten aber ihre Verschränkung bei. Zu den bekanntesten verschränkten Zuständen gehören die Bell oder EPR-Zustände:

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.19}$$

$$|\phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{2.20}$$

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.21}$$

$$|\psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{2.22}$$

Entanglement Swapping

In [BVK98, 2] und [SPG06, 3] wird ein Phänomen beschrieben, bei dem eine Verschränkung zwischen zwei Parteien A und C erzeugt werden kann, wenn Verschränkungen (A, B) und (B, C) existieren.

Beispiel 1 *Sei das System in dem Zustand:*

$$|\Psi\rangle = \frac{1}{2}(|0\rangle_A|1\rangle_{B_1} - |1\rangle_A|0\rangle_{B_1})(|0\rangle_{B_2}|1\rangle_C - |1\rangle_{B_2}|0\rangle_C) \tag{2.23}$$

Ausgedrückt in der Bell-Basis ist der Zustand:

$$\begin{aligned}|\Psi\rangle = \frac{1}{2}&(|\psi^+\rangle_B|\psi^+\rangle_{AC} - |\psi^-\rangle_B|\psi^-\rangle_{AC} \\ &-|\phi^+\rangle_B|\phi^+\rangle_{AC} + |\phi^-\rangle_B|\phi^-\rangle_{AC})\end{aligned} \tag{2.24}$$

Die Messung der Quantenbits von B nach $\{|\psi^+\rangle_B, |\psi^-\rangle_B, |\phi^+\rangle_B, |\phi^-\rangle_B\}$ *hinterlässt einen verschränkten Restzustand von A und C.*

Das Phänomen lässt sich auf den n-qubit Fall erweitern (vgl. [BVK98, 3]).

Evolution von Quantensystemen

Die Veränderung eines Quantenzustandes $|\psi\rangle$ zum Zeitpunkt t_1 in einem geschlossenen System zu einem Quantenzustand $|\psi'\rangle$ zum Zeitpunkt t_2 kann durch einen unitäre Operator U ausgedrückt werden.

$$|\psi'\rangle = U|\psi\rangle \tag{2.25}$$

U hängt dabei nur von t_1 und t_2, niemals von dem Quantenzustand $|\psi\rangle$ ab.

2.2.2 Quantenschaltkreise

Quantenschaltkreise sind unitäre Matrizen, die einen Quantenzustand in einen anderen überführen.

Einzelne Qubit Operatoren

Hierbei handelt es sich um einfache Operatoren, die den Zustand eines Qubits verändern. In Abschnitt 2.2.1 wurde gezeigt, dass ein Qubit aus einem Vektor (vgl. Gleichung (2.12)) besteht. Operationen auf einem Qubit müssen diese Norm erhalten. Dies wird durch unitäre 2×2 Matrizen erreicht.
Im folgenden werden die wichtigsten Operatoren aufgelistet.

Identität Der Identitätsoperator verändert das Qubit nicht.

$$\begin{aligned} I &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ &= \text{—}\boxed{I}\text{—} \end{aligned} \tag{2.26}$$

Pauli Gatter Pauli-Gatter sind Spiegelungen an der x, y und z-Achse der Bloch-Sphäre.

$$\begin{aligned} X &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \text{—}\boxed{X}\text{—} \end{aligned} \tag{2.27}$$

$$\begin{aligned} Y &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\ &= \text{—}\boxed{Y}\text{—} \end{aligned} \tag{2.28}$$

$$\begin{aligned} Z &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &= \text{—}\boxed{Z}\text{—} \end{aligned} \tag{2.29}$$

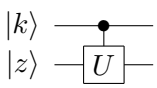
Hadamard Beschreibt eine Drehung um die y-Achse um 90° gefolgt von einer Drehung um die x-Achse um 180°.

$$\begin{aligned} H &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \end{aligned} \quad (2.30)$$

Kontrollierte Operatoren

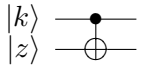
Sei U ein beliebiger Operator auf einem Qubit, dann ist ein kontrollierter Operator eine Operation auf zwei Qubits (Kontrollqubit und Zielqubit $|k, z\rangle$), die genau dann U auf $|z\rangle$ anwendet, wenn $|k\rangle$ auf *eins* gesetzt ist.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{11} & u_{12} \\ 0 & 0 & u_{21} & u_{22} \end{pmatrix}$$



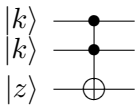
Einer der wichtigsten kontrollierten Operatoren ist das controlled-NOT (CNOT). Hier entspricht der Operator U einem X-Gatter.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

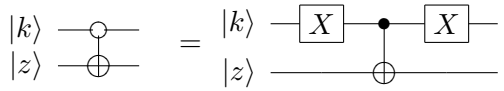


Dieser Operator kann auch mit 2 Kontrollqubits ausgeführt werden. Die Bezeichnung dafür lautet Toffoli-Gatter oder CCNOT.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



Jetzt ist es möglich kontrollierte Gatter auch für eine beliebige Anzahl von Kontrollqubits zu konstruieren (siehe Abb. 2.1).
Es besteht auch die Möglichkeit negative Kontrollbits zu verwenden. Hier werden kontrollierte Operationen genau dann ausgeführt, wenn das Kontrollqubit auf *null* gesetzt ist. Dies lässt sich folgendermaßen erreichen:



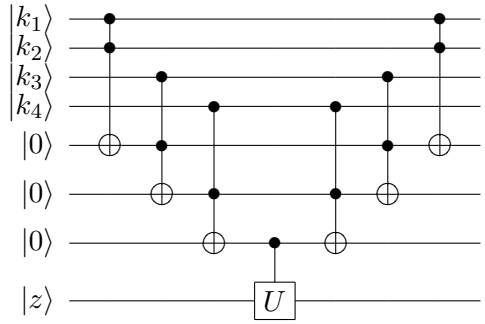


Abbildung 2.1: 4-qubit kontrollierter U Schaltkreis

Universelle Operatoren

Eine Menge von Operatoren heißt *universell* in Bezug auf Quantenberechnung, wenn sich jede unitäre Matrix durch einen Quantenschaltkreis darstellen lässt, der nur aus dieser Menge von Operatoren besteht. In [NC00, Seite 191 ff.] wird gezeigt, dass die Menge der 1-Qubit-Operatoren zusammen mit dem CNOT Operator eine universelle Operatorenmenge bildet.

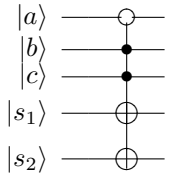
Nomenklatur

Für den weiteren Verlauf der Arbeit werden hier ein paar Funktionen definiert, die für die verschiedenen Arten von Kontrollgattern stehen. Diese werden später in Algorithmen verwendet.

Für alle kontrollierten X-Gatter (CNOT, Toffoli, negativ und positiv kontrollierte Gatter und Gatter mit mehreren Kontrollbits) wird der Name x_1*nx*x_2*p-CNOT(Kontrollbits → Zielbits)* verwendet. x_1 ist die Anzahl der negativen Kontrollbits, x_2 ist die Anzahl der positiven Kontrollbits, *Kontrollbits* ist die Aufzählung aller Kontrollbits, angefangen mit den negativen. *Zielbits* ist eine Aufzählung aller Qubits die durch die Kontrollbits invertiert werden.

Beispiel 2

1n2p-CNOT(a, b, c → s1, s2) $=$



(2.31)

Nach diesem Schema werden in den Algorithmen und Beispielen alle einfache Quantenoperationen angegeben.

Komplexität

Bei der Analyse von Quantenschaltkreisen betrachten wir die Anzahl einfacher Operatoren, die Anzahl der benötigten Quantenbits und, im Falle von Kommunikation, die Anzahl der versendeten Quantenbits. Genauer wäre es bei der Anzahl der Operatoren nur die Operatoren zu nehmen, die in der universellen Operatorenmenge (vgl. Abschnitt 2.2.2) enthalten sind, da wir jedoch in dieser Arbeit nur die asymptotische Laufzeit der Algorithmen betrachten, fallen

die Kosten für die Transformationen nicht ins Gewicht.
Da Quantencomputer und Simulatoren durch die Anzahl ihrer Qubits beschränkt sind, werden diese hier auch betrachtet. Beim Entwurf der Algorithmen wurde zwar keine Rücksicht auf die Anzahl der Quantenbits genommen, jedoch werden sie angegeben. Das Senden von Quantenbits wird direkt mit dem Versenden von klassischen Nachrichten verglichen. So wird angenommen, dass das Senden eines Qubits mindestens so aufwendig ist, wie das Senden eines klassischen Bits.

2.2.3 Logische und Arithmetische Quantenschaltkreise

Um Berechnungen durch Quantensuperpositionen zu parallelisieren, ist es nötig Quantenschaltkreise für logische und arithmetische Operationen einzuführen. Als Eingabe erhalten solche Quantenschaltkreise Quantenregister, auf denen eine Superposition von Werten gespeichert ist. Je nach Operation werden diese dann miteinander verglichen, addiert oder subtrahiert. Das Ergebnis wird dann auf ein Zielregister geschrieben und befindet sich ebenfalls in Superposition. Um die Quantenschaltkreise einführen zu können, muss zuerst die Wertekodierung in Bits definiert werden.

Wertekodierung in Bits

In dieser Arbeit müssen verschiedene Werte in Bits kodiert werden. Dazu werden, in Anlehnung an [KP95] und [MP00], Bitvektoren eingeführt, mit deren Hilfe sich Integer Zahlen im Binärsystem darstellen lassen.

Definition 1 *Sei $b \in \mathbb{B} = \{0,1\}$ ein Bit im klassischen Sinne. Ein Bitvektor $a \in \mathbb{B}^n$ der Länge $n \in \mathbb{N}$ ist ein Array $a[n-1:0] \in \mathbb{B}^n$.*

Wenn es im folgenden darum geht Werte durch Bitstrings auszudrücken, sei folgendes definiert:

Definition 2

$$\langle a[n-1:0]\rangle \quad = \quad \sum_{i=0}^{n-1} a[i] \cdot 2^i \tag{2.32}$$

Dadurch erhält man folgende Grenzen:

$$\langle a[n-1:0]\rangle \in \{0,1,\ldots,2^n-1\}$$

Um negative Zahlenwerte ausdrücken zu können, werden Two's Complement Zahlen verwendet.

Definition 3

$$[a[n-1:0]] \quad = \quad -2^{n-1} \cdot a[n-1] + \langle a[n-2:0]\rangle \tag{2.33}$$

Die Grenzen von Two's Complement Zahlen sind:

$$[a[n-1:0]] \in T_n := \{-2^{n-1}, -2^{n-1}+1, \ldots, 2^{n-1}-1\}$$

Es gelten folgende Eigenschaften:

$$
\begin{aligned}
[0, a] &= \langle a \rangle && (2.34)\\
[a] &\equiv \langle a[n-2:0] \rangle \mod 2^{n-1} && (2.35)\\
[a] &\equiv \langle a \rangle \mod 2^{n} && (2.36)\\
[a[n-1], a] &= [a] && (2.37)\\
-[a] &= [\neg a] + 1 && (2.38)
\end{aligned}
$$

den Beweis dieser Eigenschaften wird in [KP95, Kapitel 4.1.3] gegeben.
Für Floating Point können die in [MP00, Kapitel 7.1] beschriebenen Formate verwendet werden.

Logische Quantenschaltkreise

Im folgenden werden verschiedene Quantenschaltkreise beschrieben, die boolsche Tests und Vergleiche auf Quantenbits durchführen können.

n-qubit Oder-Schaltkreis

Dieser Schaltkreis (siehe Abb. 2.2) bekommt als Eingabe einen beliebigen n-qubit Quantenzustand und ein zusätzliches Quantenbit $|0\rangle$ und setzt dieses Quantenbit genau dann auf *eins*, wenn eines der n Quantenbits den Wert *eins* hat.

Beispiel 3 *Eine Oder-Schaltkreis Operation ist im Anhang A.1.1 zu finden.*

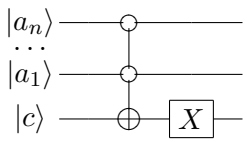


Abbildung 2.2: n-qubit Oder-Schaltkreis

n-qubit Gleichheit

Dieser Schaltkreis (siehe Abb. 2.3) vergleicht zwei n-qubit Quantenregister miteinander, ein zusätzliches Quantenbit $|0\rangle$ wird genau dann auf *eins* gesetzt, wenn die beiden Quantenregister den gleichen Wert haben.

Beispiel 4 *Eine Gleichheit Operation ist im Anhang A.1.2 zu finden.*

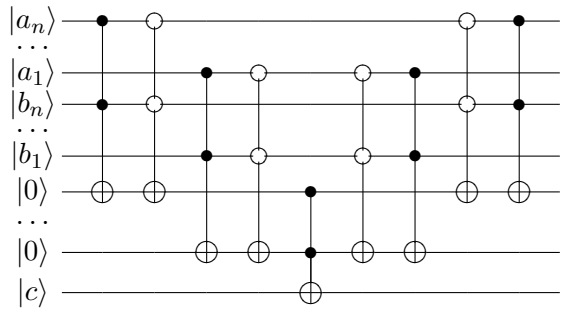


Abbildung 2.3: n-qubit Gleichheit

Größer/Kleiner Vergleich

Dieser Schaltkreis bekommt als Eingabe zwei n-qubit Quantenregister. Ein zusätzliches Quantenbit $|0\rangle$ wird genau dann auf *eins* gesetzt, wenn der Wert im ersten Quantenregister größer/kleiner als der Wert im zweiten Quantenregister ist. Der folgende Schaltkreis (siehe Abb. 2.4) ist ein n-qubit größer Vergleich.

Beispiel 5 *Eine größer Operation wird in Anhang A.1.3 gezeigt.*

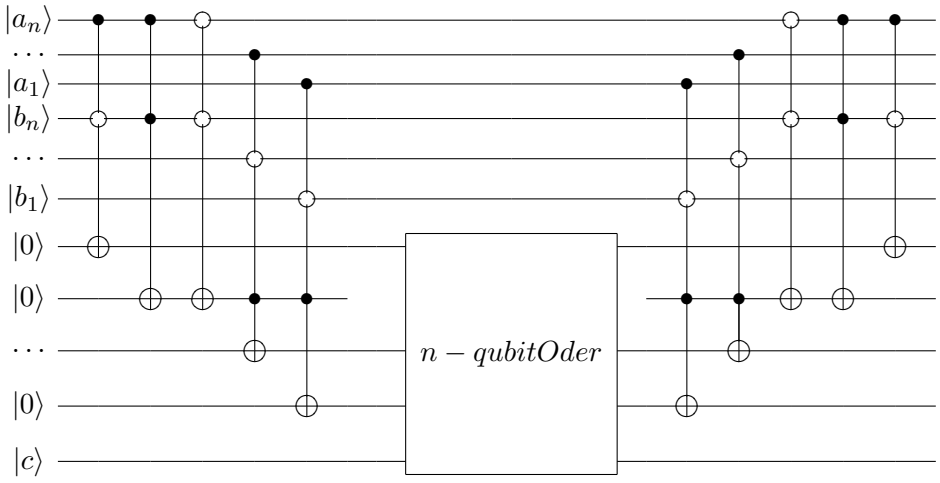


Abbildung 2.4: n-qubit größer Vergleich

Arithmetische Quantenschaltkreise

Im folgenden werden arithmetische Quantenschaltkreise beschrieben, die Quantenregister mit den Eingabewerten in Superposition erhalten und das Ergebnis auf ein leeres Quantenregister schreiben.

Addierer

Gegeben seien zwei Binärzahlen $a[n-1:0]$, $b[n-1:0]$ und ein Übertragseingangsbit c_{in}. Als Ausgabe werden zwei Bitvektoren $c[n-1:-1]$ und $s[n:0]$ wie folgt definiert.

$$\begin{aligned} c[-1] &= c_{in} && (2.39)\\ \langle c[i], s[i]\rangle &= c[i-1] + a[i] + b[i] \forall i \in \mathbb{N}_{<n} && (2.40)\\ s[n] &= c[n-1] && (2.41) \end{aligned}$$

Dadurch ergibt sich die Eigenschaft,

$$\begin{aligned} \langle a[n-1:0]\rangle + \langle b[n-1:0]\rangle + c_{in} &= \langle c[n-1], s[n-1:0]\rangle \\ &= \langle s[n:0]\rangle && (2.42) \end{aligned}$$

Der Beweis wird in [MP00, Kapitel 2.2.1] gezeigt.

3-*qubit Addierer*

Dieser Addierer (siehe Abb. 2.5) addiert die Werte von drei einzelnen Qubits (c_{in}, a, b) auf die beiden Ergebnisbits (s_0, s_1). Die Ergebnisbits müssen vor der Addition für ein sinnvolles Ergebnis im Zustand $|0\rangle$ sein.

Beispiel 6 *Eine 3-qubit Addier-Operation ist in Anhang A.1.4 zu finden.*

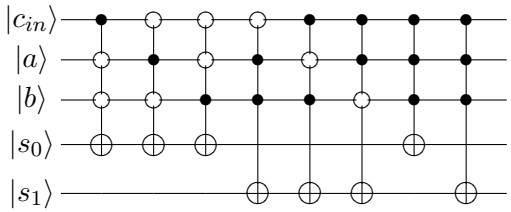


Abbildung 2.5: 3-qubit Addierer

n-qubit Addierer

Im n-bit Fall werden zwei n-qubit Quantenregister (a, b) auf ein Ergebnisregister addiert. Als Eingabe ist ein Quantenzustand $|a, b, s, c\rangle$ nötig, wobei a, b die beiden zu addierenden Quantenregister sind, s ist ein $n+1$-qubit Quantenregister, auf das das Ergebnis geschrieben wird und c ist das Carry-Register für einzelne Überträge. Die Quantenregister s und c werden mit $|0\ldots0\rangle$ initialisiert. Da ein Schaltkreis zu unübersichtlich wäre, wird hier ein Algorithmus (Algorithmus 1) für den n-qubit Addierer gegeben.

Dies ist die Umsetzung eines Carry-Chain Addierers [MP00, Kapitel 2.4.1]. Bei der Addition von Two's Complement Zahlen ist zu beachten, dass die Zahlen vorher mittels Sign extension

Algorithmen 1 n-qubit Addierer

Quantenvariablen: Quantenregister |a[n-1:0], b[n-1:0], s[n:0] = 0^{n+1}, c[n-1:0] = $0^n\rangle$
Bei Two's Complement Addition: $\otimes|a_{se}$, b_{se}, c_{se}, s[n+1] = $0^4\rangle$
Klassische Variablen: $n \in \mathbb{N}$ Anzahl der zu addierenden Quantenbits.

1: CNOT($a[n-1] \rightarrow a_{se}$) ▷ $(*)$ - Sign extension von a
2: CNOT($a[n-1] \rightarrow a_{se}$) ▷ $(*)$ - Sign extension von b
3: Interpretiere a_{se} als $a[n]$, b_{se} als $b[n]$ und c_{se} als $c[n]$ ▷ $(*)$

4: 2-qubit Addierer($a[0]$, $b[0] \rightarrow c[0]$, $s[0]$)
5: **für** $i = 1$ bis $n-1$ **tue** ▷ addiere Quantenregister a und b
▷ bei Two's Complement bis n
6: 3-qubit Addierer($a[i]$, $b[i]$, $c[i-1] \rightarrow c[i]$, $s[i]$)
7: **ende für**
8: CNOT($c[n-1] \rightarrow s[n]$) ▷ bei Two's Complement entfällt dieser Schritt

9: **für** $i = n-1$ bis 1 **tue** ▷ reinige Carry-Register c
▷ bei Two's Complement starte mit n
10: $1n2p$-CNOT($c[i-1]$, $a[i]$, $b[i] \rightarrow c[i]$)
11: $1n2p$-CNOT($a[i]$, $c[i-1]$, $b[i] \rightarrow c[i]$)
12: $1n2p$-CNOT($b[i]$, $a[i]$, $c[i-1] \rightarrow c[i]$)
13: $0n3p$-CNOT($c[i-1]$, $a[i]$, $b[i] \rightarrow c[i]$)
14: **ende für**
15: $0n2p$-CNOT($a[0]$, $b[0] \rightarrow c[0]$)

16: CNOT($a[n-1] \rightarrow a_{se}$) ▷ $(*)$ - inverse Sign extension von b
17: CNOT($a[n-1] \rightarrow a_{se}$) ▷ $(*)$ - inverse Sign extension von a

(vgl. Gleichung 2.37) um ein Bit erweitert werden. Danach wird die Addition durchgeführt und das letzte Bit des Ergebnisses wird nicht benötigt (siehe dazu die mit $(*)$ markierten Zeilen in Algorithmus 1). Der Beweis hierfür wird in [MP00, Kapitel 2.2.2] gegeben.

$$\begin{aligned} [a[n-1:0]] + [b[n-1:0]] + c_{in} &\rightarrow \langle a[n-1], a\rangle + \langle b[n-1], b\rangle + c_{in} \\ &\rightarrow \langle s[n+1:0]\rangle \\ &\rightarrow [s[n:0]] \end{aligned}$$

Der 2-qubit Addierer (siehe Abb. 2.6) ist analog zu dem 3-qubit Addierer aufgebaut. Er kann entweder durch einen 3-qubit Addierer simuliert werden, bei dem der Input c_{in} gleich $|0\rangle$ ist oder es kann ein extra Schaltkreis, bestehend aus den 3 Operatoren, gebaut werden, die bei dem 3-qubit Addierer c_{in} als negatives Kontrollbit besitzen.

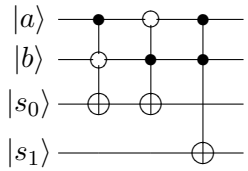


Abbildung 2.6: 2-qubit Addierer

Beispiel 7 *Eine Anwendung für einen n-qubit Addierer wird im Anhang A.1.5 durchgerechnet.*

Subtrahierer

Die Subtraktion läuft nach folgendem Algorithmus ab [MP00, Kapitel 2.2.2].

$$\langle a[n-1:0]\rangle - \langle b[n-1:0]\rangle \quad \equiv \quad \langle a[n-1:0]\rangle + \langle \neg b[n-1:0]\rangle + 1 \mod 2^n$$

Der Test, ob ein Overflow auftritt oder das Ergebnis negativ ist, kann wie folgt durchgeführt werden [MP00, Kapitel 2.4.5].

$$\begin{aligned} ovf \rightarrow [a] - [b] \notin T_n &= c_{n-1} \otimes c_{n-2} \\ neg \rightarrow [a] - [b] < 0 &= s_n \end{aligned}$$

Diese Eigenschaften werden benötigt, um die richtige Subtraktion von Two's Complement Zahlen zu ermöglichen. Der Algorithmus für den n-bit Fall subtrahiert zwei n-qubit Quantenregister (a, b) und schreibt das Ergebnis auf ein $(n+1)$-qubit Ergebnisregister s. Zusätzlich wird noch ein n-qubit Carry-Register c benötigt, sowie 2 zusätzliche Kontrollqubits. Das Quantenregister befindet sich anfangs in dem Zustand $|a, b, s, c, ctrl_1, ctrl_2\rangle$. Da ein Schaltkreis zu unübersichtlich wäre, wird hier ein Algorithmus (Algorithmus 2) für den n-qubit Subtrahierer gegeben.

Algorithmen 2 n-qubit Subtrahierer

Quantenvariablen: Quantenregister |a[n-1:0], b[n-1:0], s[n:0] = 0^{n+1}, c[n-1:0] = $0^n, 1, 0\rangle$
Klassische Variablen: $n \in \mathbb{N}$ Anzahl der zu addierenden Quantenbits.

1: **für** $i = 0$ bis $n-1$ **tue** ▷ invertiere Quantenregister b
2: Sigma-X($b[i]$)
3: **ende für**

4: 3-qubit Addierer($a[0]$, $b[0]$, $ctrl_1 \rightarrow c[0]$, $s[0]$) ▷ addiere 1
5: Sigma-X($ctrl_1$) ▷ setze $ctrl_1 = |0\rangle$
6: **für** $i = 1$ to $n-1$ **tue** ▷ addiere Quantenregister a und b
7: 3-qubit Addierer($a[i]$, $b[i]$, $c[i-1] \rightarrow c[i]$, $s[i]$)
8: **ende für**
9: CNOT($c[n-1] \rightarrow s[n]$)

10: $1n1p$-CNOT($c[n-1]$, $c[n-2] \rightarrow ctrl_1$) ▷ berechne ovf auf $ctrl_1$
11: $1n1p$-CNOT($c[n-2]$, $c[n-1] \rightarrow ctrl_1$)

12: $2n1p$-CNOT($ctrl_1$, $s[n-1]$, $s[n] \rightarrow ctrl_2$) ▷ falls kein ovf → Sign extension
13: $2n1p$-CNOT($ctrl_1$, $s[n]$, $s[n-1] \rightarrow ctrl_2$)
14: CNOT($ctrl_2 \rightarrow s[n]$)

15: $2n1p$-CNOT($ctrl_1$, $s[n-1]$, $c[n-1] \rightarrow ctrl_2$) ▷ reinigen von $ctrl_2$
16: $2n1p$-CNOT($ctrl_1$, $c[n-1]$, $s[n-1] \rightarrow ctrl_2$)

17: $1n1p$-CNOT($c[n-1]$, $c[n-2] \rightarrow ctrl_1$) ▷ reinigen von $ctrl_1$
18: $1n1p$-CNOT($c[n-2]$, $c[n-1] \rightarrow ctrl_1$)

19: **für** $i = n-1$ bis 1 **tue** ▷ reinige Carry-Register c
20: $1n2p$-CNOT($c[i-1]$, $a[i]$, $b[i] \rightarrow c[i]$)
21: $1n2p$-CNOT($a[i]$, $c[i-1]$, $b[i] \rightarrow c[i]$)
22: $1n2p$-CNOT($b[i]$, $a[i]$, $c[i-1] \rightarrow c[i]$)
23: $0n3p$-CNOT($c[i-1]$, $a[i]$, $b[i] \rightarrow c[i]$)
24: **ende für**
25: $0n2p$-CNOT($a[0]$, $b[0] \rightarrow c[0]$)
26: $1n1p$-CNOT($a[0]$, $b[0] \rightarrow c[0]$)
27: $1n1p$-CNOT($b[0]$, $a[0] \rightarrow c[0]$)

28: **für** $i = 0$ bis $n-1$ **tue** ▷ invertiere Quantenregister b
29: Sigma-X($b[i]$)
30: **ende für**

Nach dem Durchlaufen des Algorithmus ist das Carry-Register und die beiden Kontrollqubits wieder auf $|0\rangle$.

Beispiel 8 *Eine Anwendung für einen n-qubit Subtrahierer wird im Anhang A.1.6 durchgerechnet.*

2.2.4 Quantenkommunikation

In diesem Abschnitt werden die Grundlagen der Quantenkommunikation (Teleportation, Superdense Coding), sowie Kommunikationsmodelle beschrieben.

Teleportation

Bei der Quantenteleportation (siehe Abb. 2.7) können n Qubits mit $2n$ Bits klassischer Information übertragen werden. Um ein Quantenbit ψ von einem Sender A zu einem Empfänger B zu übertragen, müssen sich beide ein verschränktes Quantenpaar (zum Beispiel ein EPR Paar) $(\phi_1\phi_2)$ teilen. Wie in [BBC$^+$93] beschrieben wird folgendes Protokoll durchgeführt.

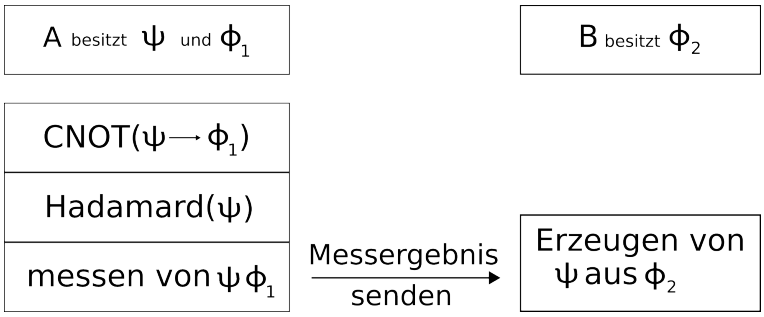


Abbildung 2.7: Quantenteleportation eines unbekannten Quantenzustandes ψ

Für die einzelnen Messergebnisse werden folgende Pauli Operatoren (siehe Abschnitt 2.2.1 Einzelne Qubit Operatoren) angegeben.

$$\begin{array}{lcl} 00 & \rightarrow & \mathbf{I} \\ 01 & \rightarrow & \mathbf{X} \\ 10 & \rightarrow & \mathbf{Z} \\ 11 & \rightarrow & \mathbf{XZ} \end{array}$$

In [KB98, 3] wird die Teleportation eines unbekannten Qubitzustandes $|\Psi_A\rangle = a|0\rangle_1 + b|1\rangle_1$ mit Hilfe einer 3-qubit Verschränkung $|\Psi_{GHZ}\rangle = \frac{1}{\sqrt{2}}(|0\rangle_2|0\rangle_3|0\rangle_4 + |1\rangle_2|1\rangle_3|1\rangle_4)$ zu einem von zwei Empfängern (E_1 und E_2) gezeigt. Nach einer Dekomposition in die Bell Zustände kann das System wie folgt beschrieben werden:

$$
\begin{aligned}
|\Psi_A\rangle \otimes |\Psi_{GHZ}\rangle &= \frac{1}{2}[|\phi_{12}^+\rangle \otimes (a|0\rangle_3|0\rangle_4 + b|1\rangle_3|1\rangle_4) \\
&\quad +|\phi_{12}^-\rangle \otimes (a|0\rangle_3|0\rangle_4 - b|1\rangle_3|1\rangle_4) \\
&\quad +|\psi_{12}^+\rangle \otimes (a|0\rangle_3|1\rangle_4 + b|1\rangle_3|0\rangle_4) \\
&\quad +|\psi_{12}^-\rangle \otimes (a|0\rangle_3|1\rangle_4 - b|1\rangle_3|0\rangle_4)] \qquad (2.43)
\end{aligned}
$$

Nach einer Messung von Qubit 1 und 2 in der Bellbasis ist das System ohne Beschränkung der Allgemeinheit in Zustand $a|0\rangle_3|0\rangle_4 + b|1\rangle_3|1\rangle_4$ (die anderen Fälle werden ähnlich behandelt). Das Messergebnis wird an beide Empfänger geschickt. Soll nun E_2 das Qubit bekommen, führt E_1 eine Messung in einer neuen Basis ($|x_1\rangle$, $|x_2\rangle$) durch.

$$
\begin{aligned}
|0\rangle_3 &= \sin\theta|x_1\rangle_3 + \cos\theta|x_2\rangle_3 \\
|1\rangle_3 &= \cos\theta|x_1\rangle_3 - \sin\theta|x_2\rangle_3 \qquad (2.44)
\end{aligned}
$$

Das System ist vor der Messung in Zustand:

$$
\begin{aligned}
|\Psi_{34}\rangle &= (a\sin\theta|0\rangle_4 + b\cos\theta|1\rangle_4)|x_1\rangle_3 \\
&\quad +a\cos\theta|0\rangle_4 - b\sin\theta|1\rangle_4)|x_2\rangle_3) \qquad (2.45)
\end{aligned}
$$

Ist das Messergebnis x_1 und wurde $\theta = \frac{\pi}{4}$ gewählt, ist Qubit 4 in Zustand $a|0\rangle_4 + b|1\rangle_4$. Ist das Messergebnis x_2 kann E_2 den gleichen Zustand erhalten, indem er den $|1\rangle_4$ Zustand um den Faktor π flipt. Dazu muss E_1 eine 1-bit Nachricht an E_2 schicken, um ihm das Ergebnis mitzuteilen.
Im Falle von m-qubit Zuständen bietet [YCH04, 2] einen effizienteren Ansatz. Dort wird dieser auch für den Fall mehrerer kontrollierender Agenten verallgemeinert [YCH04, 3].

Superdense Coding

Mit Quanten Dense Coding (siehe Abb. 2.8) kann ein n-bit String mittels $\frac{n}{2}$ Qubits übertragen werden. Der Sender A kodiert jedes Paar von klassischen Bits b_1b_2, wie in [BW92] beschrieben. Er wendet einen der Pauli Operatoren (siehe Abschnitt 2.2.1 Einzelne Qubit Operatoren) auf sein Partikel eines bekannten verteilten Paares ($\phi_1\phi_2$) an.

Eine vorherige Abstimmung von A und B über die Kodierung mittels der Pauli Operatoren ist nötig. Zum Beispiel kann diese wie bei der Teleportation getroffen werden.

Kommunikationsmodelle

In [Klu04, 3.3] werden aus diesen beiden Kommunikationsmethoden verschiedene Quantenkommunikationsmodelle erstellt.

QCOMM-1 A und B teilen sich verschränkte Qubits und verwenden einen klassischen Kanal zur Kommunikation.

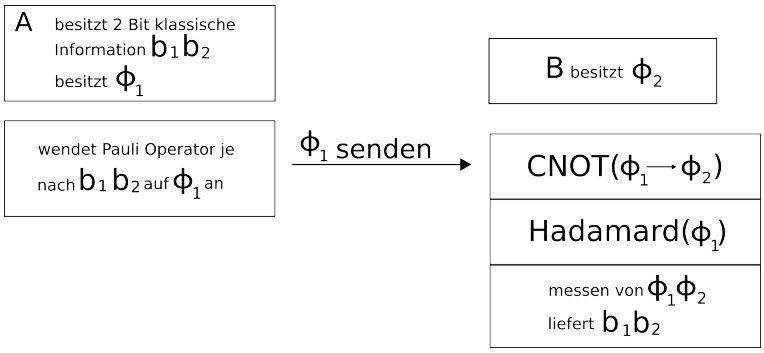


Abbildung 2.8: Superdense Coding

QCOMM-2 A und B teilen sich verschränkte Qubits und verwenden einen Quantenkanal zur Kommunikation.

QCOMM-3 A und B teilen sich keine verschränkten Qubits und verwenden einen Quantenkanal zur Kommunikation.

In QCOMM-1 kann somit Quantenteleportation eingesetzt werden, in QCOMM-2 Dense Coding.

2.2.5 Quantenalgorithmen

In diesem Abschnitt werden einige Quantenalgorithmen beschrieben, die in der Arbeit verwendet werden.

Grovers Suchalgorithmus

Grovers Suchalgorithmus [Gro96] erlaubt eine Suche in einer unsortierten Datenbank von N Einträgen innerhalb von $O(\sqrt{N})$ Schritten. Hier wird der Algorithmus (Algorithmus 3) in aller Kürze vorgestellt.

Maximum Suchalgorithmus

In [AK99] wird ein Quantenalgorithmus zum Auffinden des Maximums in einer unsortierten Datenbank $T[0, \ldots, N-1]$ vorgestellt. Der Algorithmus verwendet intern Grovers Suchalgorithmus. Wobei dieser als Orakel die Funktion

$$f_i(j) \quad = \quad \begin{cases} 1 & \text{falls } T[j] > T[i] \\ 0 & \text{sonst} \end{cases} \tag{2.46}$$

verwendet. Die Laufzeit ist bei N Einträgen $O(\sqrt{N})$, wobei diese durch die Wechselwirkung zwischen der Anzahl der durchlaufenden Schleifen (Zeile 2) und der Anzahl an Groveriterationen (Zeile 4) zustande kommt. Dabei spielt die Anzahl der vom Orakel in jedem Durchgang markierte Einträge eine wichtige Rolle.

Algorithmen 3 Grover Suchalgorithmus

Quantenvariablen: $|s\rangle \in H_2^{\otimes n}$ Indexregister,
$|0\rangle$ Orakelbit.
Klassische Variablen: N = Größe des Suchraums,
k = Anzahl der richtigen Lösungen,
$U_\omega = I - 2|\omega\rangle\langle\omega|$.

1: Initialisiere das System in dem Startzustand $|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$
2: $r(N) = \frac{\pi\sqrt{N/k}}{4}$
3: **für** $i = 0$ bis $r(N)$ **tue**
4: Wende $U_\omega = \begin{cases} -|x\rangle & \text{falls} |x\rangle = |\omega\rangle \\ |x\rangle & \text{sonst} \end{cases}$ an.
5: Hadamard$(s_1, \ldots, s_N)$ ▷ Hadamard auf das Indexregister.
6: Sigma-X$(s_1, \ldots, s_N)$ ▷ Sigma-X auf das Indexregister.
7: Hadamard(s_N) ▷ Hadamard auf das letzte Qubit des Indexregisters.
8: $0n(N-1)p$-CNOT$(s_1, \ldots, s_{N-1} \to s_N)$
9: Hadamard(s_N) ▷ Hadamard auf das letzte Qubit des Indexregisters.
10: Sigma-X$(s_1, \ldots, s_N)$ ▷ Sigma-X auf das Indexregister.
11: Hadamard$(s_1, \ldots, s_N)$ ▷ Hadamard auf das Indexregister.
12: **ende für**
13: Führe eine Messung durch, um den Index der gesuchten Zahl zu erhalten.

Algorithmen 4 Maximum Suchalgorithmus

Quantenvariablen: $|\psi\rangle = \sum_i \frac{1}{\sqrt{N}} |i\rangle|y\rangle$
Klassische Variablen: N = Größe des Suchraums, y = Index mit bisher größtem gefundenen Inhalt,
x = Messergebnis.

1: Wähle zufällig einen Index $y \in \{0, \ldots, N-1\}$. ▷ Rate initiales Maximum.
2: **für** 0 bis $O(\sqrt{N})$ **tue**
3: Initialisiere Zustand $|\psi\rangle = \sum_i \frac{1}{\sqrt{N}} |i\rangle|y\rangle$.
4: Wende Grovers Suchalgorithmus an, um einen markierten Zustand zu finden.
5: x = Messergebnis des ersten Quantenregisters, $y = x$
6: **ende für**
7: y = Index des maximalen Elements der Datenbank.

Quantenauktionen

Bei der Quantenauktion [HHC07] handelt es sich um eine Auktion im klassischen Sinne, wobei Superpositionen die Gebote repräsentieren und durch eine verteilte Suche der/die Gewinner der Auktion festgestellt wird. Durch Messen des Endzustandes wird das Ergebnis festgestellt und die Superposition zerstört, so dass Gebote von Nicht-Gewinnern niemals eingesehen werden. Das Auktionsprotokoll enthält folgende Schritte:

- Der Auktionär (Manager) definiert in der Ausschreibung die Aufgabe, die Kosten usw.

- Der Auktionär gibt die Einzelheiten der Quantenauktion und einen initialen Quantenzustand mit p Qubits für jeden Bieter bekannt.
- Jeder Bieter wählt einen Operator U_j für seine p Qubits. Sie halten die Wahl des Operators geheim.
- Der Auktionär erzeugt den initialen Quantenzustand $|\Psi_{init}\rangle$.
- Der Auktionär und die Bieter führen eine verteilte Suche durch.

Im folgenden wird die Implementierung des Quantenzustandes und die verteilte Suche einer Auktion für einen Gegenstand beschrieben. Dabei werden folgende Variablen verwendet:

Anzahl der Bieter	n
Anzahl der Qubits pro Bieter	p
Zustand der Qubits für Bieter j	ψ_j
Zustand aller Qubits	$\Psi = \psi_1 \otimes \cdots \otimes \psi_n$
Suchoperator für Bieter j	U_j
Gesamter Suchoperator	$U = U_1 \otimes \cdots \otimes U_n$

Implementierung des Startzustandes

Jeder Bieter bekommt p Qubits und kann nur auf diesen Qubits operieren, so dass jeder Bieter 2^p mögliche Bietwerte hat. Bieter j wählt einen Operator U_j von p Qubits und wendet diesen auf seine Qubits des initialen Zustandes $|\psi_{init}\rangle$ an, der vom Auktionär spezifiziert wurde. Der resultierende Zustand $|\psi_j\rangle = U_j|\psi_{init}\rangle$ ist somit eine Superposition von Geboten.

Verteilte Suche

Die verteilte Suche wird in Algorithmus 5 beschrieben. Dabei wird vom Auktionär zuerst der initiale Startzustand $|\psi_{init}\rangle$ erstellt. In S Iterationsrunden wird der Zustand $|\psi\rangle$ so verändert, dass nach der letzten Iterationsrunde eine Messung den Bieter mit dem höchsten Gebot liefert. Eine Iterationsrunde ist in Abbildung 2.9 veranschaulicht.
Die Diagonalmatrix $P(f)$ paßt die Phasen der Amplituden an die Kosten in jeder Zuordnung an. Die Kosten $c(x)$ für eine Zuordnung $|x\rangle$ ist die negative Evaluationsfunktion $-F(x)$, die jeder Zuordnung einen Ertrag für den Auktionär zuweist. Dabei ist $f = s/S$ und exp bezeichnet das Matrixexponential $e^X = \sum_{k=o}^{\infty} \frac{X^k}{k!}$

$$P_{xx}(f) \quad = \quad \exp(-ifc(x)\Delta) \tag{2.47}$$

In Anlehnung daran werden die Phasen der Amplituden von Diagonalmatrix $D(f)$ durch die Funktion $d(x)$ angepasst. Wobei $d(x)$ der geringsten Zuordnung $|x\rangle$ den Wert 0 zuweist.

$$D_{xx}(f) \quad = \quad \exp(-i(1-f)d(x)\Delta) \tag{2.48}$$

Beispiel 9 *Quantenauktionen und den Gebrauch der Funktionen $c(x)$ und $d(x)$ beinhaltet die Anwendung in Anhang A.5.2.*

Algorithmen 5 Quantenauktion

Quantenvariablen: $|\Psi_{init}\rangle = |0 \dots 0\rangle$ Quantenregister der Länge $n \cdot p$.
Klassische Variablen: S = Anzahl der Iterationsrunden,
$s = 0$, aktuelle Iterationsrunde,
Δ = Konstante für $D(f)$ und $P(f)$.

▷ **Auktionärsseitige Quantenauktion**
für alle $i \in n$ **tue** ▷ Verteile Zustand, um $|\Psi_0\rangle = U|\Psi_{init}\rangle$ zu erhalten.
  sende($|\psi^i_{init}\rangle$, Bieter$_i$) ▷ Sende jedem Bieter seine p Qubits.
  empfange($|\psi^i_0\rangle$, Bieter$_i$) ▷ Empfange die veränderten Qubits aller Bieter.
ende für
für $s = 0$ bis S **tue** ▷ Start der Iterationsrunde, Ziel: $|\Psi_{s+1}\rangle = UD(f)U^\dagger P(f)|\Psi_s\rangle$
  $|\Psi_{s'}\rangle = P(f)|\Psi_s\rangle$.
  für alle $i \in n$ **tue** ▷ Verteile Zustand, um $|\Psi_{s^+}\rangle = U^\dagger|\Psi_{s'}\rangle$ zu erhalten.
    sende($|\psi^i_{s'}\rangle$, Bieter$_i$) ▷ Sende jedem Bieter seine p Qubits.
    empfange($|\psi^i_{s^+}\rangle$, Bieter$_i$) ▷ Empfange die veränderten Qubits aller Bieter.
  ende für
  $|\Psi_{s^*}\rangle = D(f)|\Psi_{s^+}\rangle$.
  für alle $i \in n$ **tue** ▷ Verteile Zustand, um $|\Psi_{s+1}\rangle = U|\Psi_{s^*}\rangle$ zu erhalten.
    sende($|\psi^i_{s^*}\rangle$, Bieter$_i$) ▷ Sende jedem Bieter seine p Qubits.
    empfange($|\psi^i_{s+1}\rangle$, Bieter$_i$) ▷ Empfange die veränderten Qubits aller Bieter.
  ende für
ende für
Messe $|\Psi_{S+1}\rangle$ und evaluiere das Messergebnis.

▷ **Bieterseitige Quantenauktion** für Bieter$_i$
empfange($|\psi^i_{init}\rangle$, Auktionär) ▷ Empfange die eigenen p Qubits.
$|\psi^i_0\rangle = U_i|\psi^i_{init}\rangle$
sende($|\psi^i_0\rangle$, Auktionär) ▷ Sende die veränderten Qubits zurück.

für $s = 0$ bis S **tue** ▷ Start der Iterationsrunde.
  empfange($|\psi^i_{s'}\rangle$, Auktionär) ▷ Empfange die eigenen p Qubits.
  $|\psi^i_{s^+}\rangle = U_i^\dagger|\psi^i_{s'}\rangle$
  sende($|\psi^i_{s^+}\rangle$, Auktionär) ▷ Sende die veränderten Qubits zurück.

  empfange($|\psi^i_{s^*}\rangle$, Auktionär) ▷ Empfange die eigenen p Qubits.
  $|\psi^i_{s+1}\rangle = U_i|\psi^i_{s^*}\rangle$
  sende($|\psi^i_{s+1}\rangle$, Auktionär) ▷ Sende die veränderten Qubits zurück.
ende für

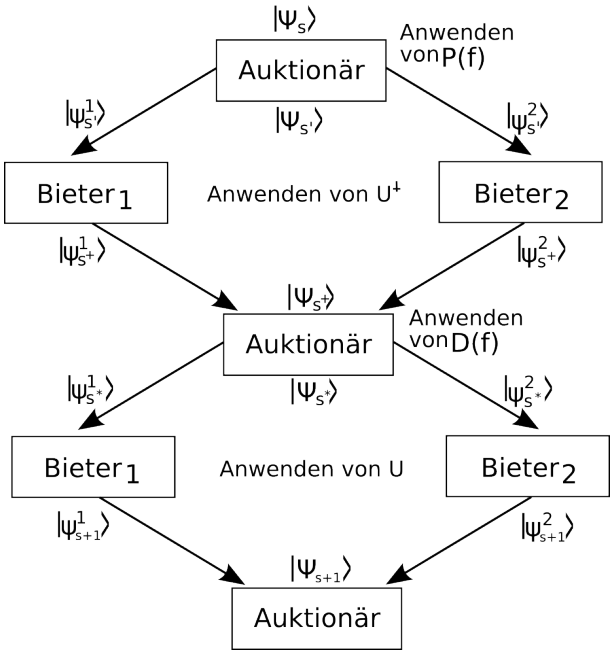


Abbildung 2.9: Verteilte Iterationsrunde der Quantenauktion

2.3 Quantencomputer und Simulatoren

In Abschnitt 2.3.1 werden Quantencomputer in aller Kürze beschrieben. In 2.3.2 werden Quantensimulatoren angesprochen und die Wahl für die Implementierung in Kapitel 5 getroffen.

2.3.1 Quantencomputer

Auch auf der physikalischen Ebene machen Quantencomputer Fortschritte. 1995 betrachtete das National Institute of Standards and Technology und das California Institute of Technology das Problem ein Quantensystem von äußeren Einflüssen abzuschirmen und machten Versuche mit magnetischen Feldern, um Ionen als Quantenzustände zu fangen. Auf diese Weise konnten jedoch nur Quantensysteme aus wenigen Bits erreicht werden [Rin98]. 1996 wurde eine ähnliche Technik von einem Team aus der Universität von Kalifornien in Berkeley, MIT, der Havard University und IBM vorgestellt, die kernmagnetische Resonanz (NMR) verwendete [GC97]. Damit konnte ein 2-qubit Quantencomputer realisiert werden. 1998 wurde an der Universität von Innsbruck eine Teleportation eines Quantenbits praktisch durchgeführt [BPM$^+$97]. 2005 wurde an der gleichen Universität erstmals ein QuByte erzeugt, verschränkt und gemessen [HHR$^+$05]. Am 14. Februar 2007 wurde *Orion*, ein 16-qubit Quantencomputer von D-Wave Systems, vorgestellt [Ric07] und eine 512-qubit Version bis 2008 angekündigt.
Die Autoren von [SAC$^+$06] sehen zukünftige Quantencomputer als ein hybrides Modell, bestehend aus einem klassischen und einem quantenrechnenden Teil. Die klassische Maschine steuert und kontrolliert die Operationen auf der Quantenmaschine. In [Klu04, 4] und [Sch07, 3.1, 3.2] wird dieser Ansatz aufgegriffen und weiter ausgeführt.

Hybrides Modell

Beim hybriden Modell findet eine Unterteilung der Architektur in einen klassischen und einen quantenrechnenden Teil statt. Der klassische Teil (CM classical machine) kontrolliert und steuert die Operationen der quantenrechnenden Maschine (QM quantum machine). Die QM besteht aus Quantenspeichern, der Quantenrecheneinheit (QPU quantum processing unit) mit Fehlerkorrektur und einem Quanteneinheitkontroller (QDC quantum device controller), der ein Interface für die klassische Maschine bereitstellt.
Die klassische Maschine besteht aus einer CPU zur Kontrolle und Planung der QM und Speicher, der sowohl klassisch, als auch durch ein Quantenaddress-System [Klu04, 4] angesteuert werden kann. Es gibt einige Quantenprogrammiersprachen ([BCS03], [Oem00], [Sel04]) die von der CM verwendet werden können. Dabei beinhalten diese high-level Primitive für Quantenoperationen verbunden mit klassischen Arbeitsfluss Anweisungen.
Die QPL wird von der CPU kompiliert und in low-level Anweisungen übersetzt, die an den QDC übergeben werden. Der QDC übersetzt diese in physikalische Quantenoperationen, die von der QPU ausgeführt werden. Diese führt Messungen und Operationen einer universellen Menge von 1- und 2-qubit Operatoren (vgl. Abschnitt 2.2.2) aus. Die Fehlerkorrektur wird eingesetzt, um Dekohärenz zu minimieren. Die QPU gibt nur das Ergebnis der Messung an die CPU zurück.

2.3.2 Quantensimulatoren

Da es Quantencomputer derzeit noch nicht außerhalb von Laboratorien gibt, müssen Forschungen an Quantenalgorithmen mittels Simulatoren durchgeführt werden. Der Vorteil dieser Simulatoren ist es, Einblicke in die Quantenzustände zu erhalten, während der Algorithmus läuft. Bei richtigen Quantencomputern kann derzeit nur das gemessene Ergebnis ausgegeben werden. Auch für die Analyse von Auswirkungen von Fehlern auf ein Quantensystem können Simulatoren verwendet werden. Um ein Quantensystem simulieren zu können muss ein klassischer Computer exponentiell viele Berechnungen ausführen. Dadurch erfährt das System auf einem klassischen Computer eine Verlangsamung, die mit jedem zusätzlichen Qubit exponentiell verstärkt wird [Ste97, 7.1].

[Wal99, 2, 3] setzt sich detailliert mit den Vor und Nachteilen von Quantensimulatoren auseinander. Im Rahmen dieser Arbeit wurden in Anlehnung an [Sch07, 3.3] QCL, QuIDDPro und libquantum betrachtet. Aufgrund der dort beschriebenen Ergebnisse, Tests mit einer hohen Anzahl von Quantenbits und der Kompatibilität wird in dieser Arbeit *libquantum* zur Simulation verwendet. *libquantum* kann unter *www.libquantum.de* kostenlos heruntergeladen werden (GNU General Public License (GPL), Version 3).

2.4 Quantenmultiagentensysteme

In diesem Kapitel werden Quantenagenten und ein quantenrechnendes MAS in aller Kürze beschrieben. Dies geschieht in Anlehnung an [Klu04, 5] und [Sch07, 4], wo auf das Thema dieses Abschnitts im Detail eingegangen wird. 2.4.1 beschreibt Quantenagenten, in 2.4.2 werden Systeme bestehend aus diesen Agenten vorgestellt.

2.4.1 Quantenagenten

Ein Quantenagent (QC Agent) ist eine Erweiterung zu einem intelligenten Softwareagenten. Er ist in einer Quantenprogrammiersprache implementiert und kann sowohl klassische, als auch Quantenberechnungen und Kommunikation ausführen, um seine Ziele alleine oder in Interaktion mit anderen Agenten zu erreichen. Das Programm des Agenten wird auf der CM und der QM ausgeführt. Er verwendet das Interface des hybriden Quantencomputers (vgl. Abschnitt 2.3.1). QC Agenten sollen mit Hilfe der Quantenrechnung die Komplexität der auszuführenden Probleme wenn möglich reduzieren.

2.4.2 Quantenmultiagentensysteme

Ein quantenrechnendes Multiagentensystem (QCMAS) ist ein Multiagentensystem, das aus klassischen und quantenrechnenden Agenten besteht. Die Agenten können miteinander interagieren, um ihre Ziele zu erreichen. Ein reines QCMAS besteht nur aus QC Agenten.
Ein QCMAS, dessen Agenten nicht mittels Quantenkommunikation interagieren können, heißt Typ-I QCMAS, ansonsten nennt man es Typ-II QCMAS.

Typ-I QCMAS

Die Agenteninteraktion in einem Typ-I QCMAS basiert nur auf klassischen Kommunikationskanälen. Die Agenten (Typ-I QC Agenten) teilen sich keine EPR Paare, so dass keines der Quantenkommunikationsmodelle von Abschnitt 2.2.4 anwendbar ist. In einem Typ-I QCMAS können Quantenberechnungen also nur lokal durchgeführt werden.

Typ-II QCMAS

In einem Typ-II QCMAS können Quantenkommunikationsmodelle genutzt werden. Dabei wird unterschieden, ob Agenten in einem solchen System nur über direkte Quantenkanäle Partikel austauschen (Typ-IIa QC Agenten), oder sich zusätzlich eine ausreichende Menge an EPR Paaren teilen (Typ-IIb QC Agenten). Der erste Fall kann in den zweiten überführt werden, da verschränkte Zustände lokal erzeugt und über Quantenkanäle verteilt werden können. Das Verteilen von verschränkten Paaren erhöht jedoch den Kommunikationsaufwand. In dieser Arbeit gehen wir im Allgemeinen von einem Typ-IIb QCMAS aus, bei dem Verschränkung nicht vorher erzeugt werden muss. Die Folgen aus dieser Annahme und die Verwendung von Typ-IIa QCMAS wird in Abschnitt 4.4 diskutiert, da in der Praxis, im Fall von zukünftigen Quantencomputern, wohl nicht jederzeit eine unbegrenzte Menge von Verschränkungen vorhanden ist. Das Wissen über die Quantenkodierungsoperationen von Abschnitt 2.2.4 ist in solchen Systemen vorhanden, oder kann, falls nötig, über klassische Kanäle ausgetauscht werden.

Kapitel 3

Quantenbasierte Verhandlung von Koalitionen

In diesem Kapitel wird ein klassischer Algorithmus zur Verhandlung von Kernel-stabilen Koalitionen (Abschnitt 3.1) vorgestellt, der als Betrachtung für eine Quantenumsetzung dient. In Abschnitt 3.2 werden dann einzelne Schritte des klassischen Algorithmus durch Quantenversionen ersetzt und das Ergebnis in Abschnitt 3.3 verglichen und ausgewertet. In Abschnitt 3.4 wird das Ergebnis diskutiert und einige offene Fragen gestellt.

3.1 KCA

Eine Menge **A** von Agenten in einem Netzwerk müssen verschieden Aufgaben erfüllen. Jeder Agent besitzt eine Menge **T** von Aufgaben, die er abarbeiten muss und eine Menge **O** von Aufgaben, die er selbst im Stande ist abzuarbeiten. Da nicht alle Aufgaben von **T** eines Agenten in seiner Menge **O** ist, muss er, um alle abzuarbeiten, andere Agenten finden, die diese Aufgaben lösen können. Im Gegenzug kann er selbst seine Fähigkeiten **O** einsetzen, um anderen Agenten bei ihren Aufgaben zu helfen.
Da die Aufgaben unterschiedlich schwer zu erledigen sind, bzw. eine variierende Anzahl von Ressourcen benötigen, versucht jeder Agent eine optimale Menge anderer Agenten zu finden, von denen er sich am meisten Gewinn verspricht. Dies erreicht er über Verhandlungen. Der KCA Algorithmus ([KS96, 5.1], [BK04, 2.2]) gewährleistet die Bildung von Kernel-stabilen Koalitionen, die allen Agenten anhand ihrer Aufgaben und Fähigkeiten eine optimale Lösung vorgibt, bei der kein Agent bevorzugt wird (vgl. Abschnitt 2.1.2).

Der Algorithmus besteht aus vier Schritten, wobei die letzten drei so lange ausgeführt werden bis der Algorithmus terminiert.

1. Die initialen Informationen werden ausgetauscht.

2. Angebote zu Koalitionzusammenschlüsse werden erstellt.

3. Die Angebote werden evaluiert.

4. Es wird eine Entscheidung über die neue Konfiguration getroffen.

3.1.1 Algorithmus

Im Folgenden wird in Algorithmus 6 das Protokoll für die Verhandlung von Kernel-stabilen Koalitionen angegeben. Darin werden folgende Funktionen verwendet:

sende(d, e) bezeichnet das Senden von Daten d an einen Empfänger e.

empfange(d, s) bezeichnet das Empfangen von Daten d von einem Sender s.

berechne bezeichnet alle lokalen Berechnungen, die ein Agent anhand der ihm vorliegenden Informationen durchführen kann.

wähle bezeichnet das Auswählen von Angeboten anhand verschiedener Kriterien.

bilateral(a) ist ein Angebot $a = (\mathbf{S}^*, u^*)$ einer Koalition $\mathbf{C}^*$ für eine Koalition $\mathbf{C}^+$, falls ein Angebot von $\mathbf{C}^+$ für $\mathbf{C}^*$ existiert.

Beispiel 10 *Ein Beispiel für den Algorithmus wird in Anhang A.2 gegeben.*

Algorithmen 6 KCA

Klassische Variablen: $\mathbf{A}$ die Menge der Agenten, $n \in \mathbb{N}$ die Anzahl der Agenten,
AListe nach Leistungsstärke sortierte Liste der Agenten,
$\mathbf{konf} = (\mathbf{S}, u)$ aktuelle Konfiguration, Führer($\mathbf{C}$) der Führer von Koalition $\mathbf{C}$,
$\mathbf{T}_a$ Menge der zu erfüllenden Aufgaben von Agent a,
$\mathbf{O}_a$ Menge der von Agent a erfüllbaren Aufgaben

1: **für** jeden Agenten $a' \in \mathbf{A} \wedge a' \in \mathbf{C}'$ **tue**
▷ **Initiale Kommunikation**
2: konf $= (\mathbf{S}, u) := (\{a_1\}, \ldots, \{a_n\}, (v(\{a_1\}), \ldots, v(\{a_n\})))$
3: Führer$(\{a'\}) := a'$
4: Aliste := sortiere($\mathbf{A}$)
5: sende$(\mathbf{T}_{a'}, a_i)$, $i = 1 \ldots n, a_i \neq a'$
6: empfange$(\mathbf{T}_{a_i}, a_i)$, $i = 1 \ldots n, a_i \neq a'$
7: berechne$(\mathbf{O}_{a'})$
8: sende$(\mathbf{O}_{a'}, a_i)$, $i = 1 \ldots n, a_i \neq a'$
9: empfange$(\mathbf{O}_{a_i}, a_i)$, $i = 1 \ldots n, a_i \neq a'$
10: **für** jede Koalition $C \subseteq A$ **tue**
11: berechne$lw'_a(C)$
12: sende$(lw'_a(C), a_i)$, $i = 1 \ldots n, a_i \neq a'$
13: **ende für**
14: empfange$(lw_{a_i}(C), a_i)$, $i = 1 \ldots n, a_i \neq a'$

▷ **Erstellung der Angebote**

15: **falls** $a' \neq$ Führer($\mathbf{C}'$) **dann**
16: springe nach Zeile 41
17: **ende falls**
18: **für** jede andere Koalition $\mathbf{C}^* \in \mathbf{S}$, $a \notin \mathbf{C}^*$ **tue**
19: berechne($k - stabile - konf = (\mathbf{S}^*, u^*)$), $\mathbf{S}^* = (\mathbf{S} \setminus \mathbf{C}', \mathbf{C}^*) \cup \{\mathbf{C}' \cup \mathbf{C}^*\}$
20: **falls** u^* dominiert u strikt **dann**
21: sende($(\mathbf{S}^*, u^*)$, Führer($\mathbf{C}^*$))
22: **ende falls**
23: **ende für**

▷ **Auswertung der Angebote**

24: empfange($(\mathbf{S}^*, u^*)$, Führer($\mathbf{C}$)) =: Prop$'$
25: **falls** $u^+ \in$ Prop$'$ dominiert u strikt $\wedge\neg\exists u^* \in$ Prop$'$: u^* dominiert u^+ strikt **dann**
26: wähle($\mathbf{S}^+, u^+$)
27: **ende falls**
28: sende($(\mathbf{S}^+, u^+)$, Führer($\mathbf{C}$)), $\forall \mathbf{C} \in \mathbf{S}$

▷ **Entscheidung über neue Konfiguration**

29: empfange($(\mathbf{S}^+, u^+)$, Führer($\mathbf{C}$)) =: Prop$''$
30: **falls** Prop$'' = \emptyset$ **dann**
31: STOP
32: **ende falls**
33: **falls** bilateral($(\mathbf{S}^+, u^+) \in$ Prop$''$) **dann**
34: wähle($(\mathbf{S}^+, u^+)$)
35: **sonst falls** $\sum_{a^* \in \mathbf{A}} u^*(a^*) > \sum_{a^* \in \mathbf{A}} u^+(a^*)$, $\forall u^+ \neq u^* \wedge u^+, u^* \in$ Prop$''$ **dann**
36: wähle($\mathbf{S}^*, u^*$)
37: **sonst**
38: wähle($\mathbf{S}^*, u^*$) von dem leistungsstärksten Agenten
39: **ende falls**
40: sende($(\mathbf{S}^+, u^+)$, Mitglieder($\mathbf{C}$)), konf := $(\mathbf{S}^+, u^+)$
41: **falls** $a' \neq$ Führer($\mathbf{C}$) **dann**
42: empfange($\mathbf{S}^+, u^+$), konf := $(\mathbf{S}^+, u^+)$
43: **ende falls**
44: **falls** $a' \in \mathbf{C}^{neu}$ **dann**
45: Führer($\mathbf{C}^{neu}$) := leistungsstärkster Agent
46: **ende falls**
47: **falls** $a' =$ Führer($\mathbf{C}'$) **dann**
48: **falls** $a' \in \mathbf{C}^{neu}$ **dann**
49: sende(Führer($\mathbf{C}^{neu}$), Führer($\mathbf{C}$)), $\forall \mathbf{C} \in \mathbf{S}$
50: **sonst**
51: empfange(Führer($\mathbf{C}^{neu}$), Führer($\mathbf{C}^{neu}$))
52: **ende falls**
53: **ende falls**
54: **falls** die große Koalition wurde gebildet $\vee$ nach einer definierten Zeit **dann**
55: STOP
56: **ende falls**
57: springe nach Zeile 15
58: **ende für**

3.1.2 Komplexität

Die Komplexität des Algorithmus unterteilt sich auf die Kosten für lokale Berechnungen und auf die Kommunikation. Bei der Kommunikation wird die Anzahl der Nachrichten gezählt,

nicht die Länge des Inhalts, da diese durch verschiedene Kompressionsalgorithmen reduziert werden kann. Für die initialen, lokalen Berechnungen stellt sich die worst-case Komplexität für jeden Agenten wie folgt zusammen.

$$\begin{aligned} \text{lokale Berechnung} &= \mathbf{O}(\text{sort} + \text{find} + 2^n \cdot \text{lw} \\ &= \mathbf{O}(2^n) \end{aligned} \tag{3.1}$$

Dabei ist *sort* die Komplexität für einen Sortieralgorithmus, der die Liste der Agenten nach ihrer Rechenleistung sortieren soll (Zeile 4). Da die Komplexität weit unter 2^n (n = Anzahl der Agenten) liegt, kann er vernachlässigt werden. Das gleiche gilt für *find*, was die Kosten für das Aufsuchen passender Aufträge, anhand der Fähigkeiten eines Agenten (Zeile 7), angibt. Die exponentielle Komplexität kommt durch die *lw*-Werte zustande, die für jede Koalition $C \subseteq A$ berechnet werden (Zeile 11).

Die Kommunikation der initialen Runde zwischen den Agenten wird wie folgt berechnet.

$$\begin{aligned} \text{Kommunikation} &= \mathbf{O}(n - 1 + n - 1 + n - 1) \\ &= \mathbf{O}(n) \end{aligned} \tag{3.2}$$

Dies ist das Senden der Aufgaben (Zeile 5), der erfüllbaren Aufgaben (Zeile 8) und der *lw*-Werte an alle anderen Agenten. In der Praxis wird das Senden der *lw*-Werte (Zeile 12) außerhalb der Schleife durchgeführt. Dazu werden erst alle 2^n Werte berechnet und dann in einer Nachricht zusammen gesendet werden. Aus diesem Grund entstehen hier nicht die 2^n Nachrichten für jeden einzelnen Wert.

Für eine Verhandlungsrunde belaufen sich die Kosten der lokalen Berechnungen auf:

$$\begin{aligned} \text{lokale Berechnung} &= \mathbf{O}((n - 1 \cdot \text{k-stabile konf}) + \text{Angebot auswählen} \\ &\quad + \text{Akzeptiertes Angebot auswählen}) \\ &= \mathbf{O}(n \cdot 2^n + n + n) \\ &= \mathbf{O}(2^n) \end{aligned} \tag{3.3}$$

k-stabile konf bezieht sich auf das Berechnen einer neuen Kernel-stabilen Konfiguration (Zeile 19). Dabei müssen die Surplus Werte berechnet werden und dies führt zu der exponentiellen Laufzeit (vgl. Abschnitt 2.1.2). Das Auswählen der Angebote (Zeile 25 und 33 bis 38) kann aus diesem Grund vernachlässigt werden.

Die Kommunikation zwischen den Agenten wird wie folgt berechnet.

$$\begin{aligned} \text{Kommunikation} &= \mathbf{O}((n - 1 \cdot 1) + n - 1 + n + n) \\ &= \mathbf{O}(n) \end{aligned} \tag{3.4}$$

Dies bezieht sich auf das Zusenden der Angebote bzw. das Informieren über veränderte Zustände (Zeile 20, 28, 40 und 49).

Über maximal n Verhandlungsrunden gesehen hat der Algorithmus also eine Komplexität von $\mathbf{O}(2^n)$ für die Berechnungen und $\mathbf{O}(n^2)$ für die Kommunikation eines Agenten, was für das Gesamtsystem eine Kommunikation von $\mathbf{O}(n^3)$ bedeutet.

3.2 KCA-Q

In diesem Abschnitt wird der klassische Algorithmus zur Bildung von Kernel-stabilen Koalitionen für ein QCMAS umgesetzt. Dabei wird die Frage untersucht, ob sich Verbesserungen erzielen lassen, wenn QC Agenten anstelle von normalen Softwareagenten zum Einsatz kommen.

3.2.1 Änderungen zu KCA

In dieser Arbeit wurde nicht ein neuer Algorithmus entworfen, sondern der klassische Algorithmus analysiert und Schritt für Schritt auf eine Modifikation mit verschiedenen Quantentechniken hingearbeitet. Bei der Bildung von Kernel-stabilen Koalitionen stehen dabei die exponentiellen Schritte des klassischen Algorithmus im Vordergrund, da eine Verbesserung hier die Gesamtkomplexität des Algorithmus reduzieren kann. Dabei stellt sich die Frage, ob durch mehr lokale Berechnungen, die durch Quantenparallelität optimiert wurden, die Kommunikation und Koordination der Agenten untereinander verbessert werden kann. Dazu wird versucht die lokalen aufwendigen Berechnungen (Zeile 11 und 19 im klassischen Algorithmus) zu parallelisieren. Auch die Führerwahl in Zeile 45 kann durch den Ansatz aus [HP06, 4] mittels richtiger Zufälligkeit effizient umgesetzt werden. Dabei verliert das Protokoll jedoch die Eigenschaft, dass immer der leistungsstärkste Agent einer Koalition der Führer ist.

3.2.2 Quantenumsetzung

Im folgenden werden einige Definitionen gegeben, wie klassische Werte in Quantenregistern kodiert werden. Danach folgen die Quantenalgorithmen zur Berechnung der lokalen Werte der Agenten ($lw_a(C)$), zum Berechnen des Surpluswertes eines Agentenpaares und für die Wahl des Koalitionsführers.

Kodierung

Der Einfachheit und Übersicht wegen werden einige Beschränkungen vorgenommen.

- l ist eine Konstante, die die maximale Anzahl von Aufträgen angibt, die ein Agent abarbeiten muss, bzw. abarbeiten kann. D.h. die Mengen $\mathbf{T}_a$ und $\mathbf{O}_a$ (vgl. Abschnitt 2.1.2) eines Agenten a besitzen jeweils maximal l Elemente.
- Die Zahlwerte in den Quantenberechnungen sind ganzzahlig.
- Falls mehrere Agenten einen Auftrag abarbeiten können, wird der Agent mit den geringeren Kosten ausgewählt.

Um die Vorteile der Quantenparallelität zu erhalten, müssen alle klassischen Werte in Quantenregistern gespeichert werden. Um dies zu erreichen werden folgende Definitionen eingeführt.

Definition 4 *Ein Agent in einem System bestehend aus n Agenten wird durch ein n-qubit Quantenregister dargestellt, das für Agent a_i an der i-ten Stelle den Wert $|1\rangle$ hat und ansonsten den Wert $|0\rangle$.*

Beispiel 11 *Agent a_2 in einem System mit 3 Agenten* $\rightarrow |010\rangle$

Definition 5 *Eine Koalition in einem System mit n Agenten ist ein n-qubit Quantenregister, das an der Stelle i genau dann den Wert $|1\rangle$ hat, wenn Agent a_i in der Koalition ist, ansonsten $|0\rangle$.*

Beispiel 12 *Koalition $\{a_1, a_3\}$ in einem System mit 3 Agenten* $\rightarrow |101\rangle$

Definition 6 *Ein Agentenpaar (a_i, a_j) in einem System von n Agenten wird als n-qubit Quantenregister dargestellt, das an der i-ten und j-ten Stelle den Wert $|1\rangle$ hat und ansonsten den Wert $|0\rangle$.*

Beispiel 13 *Agentenpaar a_2, a_3 in einem System mit 3 Agenten* $\rightarrow |011\rangle$

Definition 7 *Die Id einer Aufgabe wird in einem Quantenregister mit fester Länge binär kodiert. Die Quantenregisterlänge der Ids sei l_{id}.*

Beispiel 14 *Die Aufgabe mit Id 3 in einem System mit 3 Qubit für die Id Kodierung* $\rightarrow |011\rangle$

Definition 8 *Die Kosten einer Aufgabe wird, wie in 2.2.3, in einem Quantenregister als Two's Complement Zahl kodiert. Die maximale Anzahl der Qubits für diese Kodierung sei l_c.*

Beispiel 15 *Die Aufgabe mit Kosten 7 in einem System mit 4 Qubit für die Kosten Kodierung* $\rightarrow |0111\rangle$

Definition 9 *Eine Aufgabe wird als ein Tripel aus dem Agenten, einer Aufgaben-Id und dem Kosten Wert w bzw. c dargestellt. Die Länge des Quantenregisters sei lx.*
$|x\rangle = |agent; id; cost\rangle$

Beispiel 16 *Die Aufgabe von a_1 mit Id 3 und Kosten 7* $\rightarrow |100; 011; 0111\rangle$

Definition 10 *Die Liste der Nachfragen eines Agenten a_i besteht aus l Aufgaben, die nebeneinander auf einem Quantenregister stehen.*
$\mathbf{T}_{a_i} : |x_1, x_2, \ldots, x_l\rangle$

Definition 11 *Die Liste der Angebote eines Agenten a_i besteht aus l Aufgaben, die nebeneinander auf einem Quantenregister stehen.*
$\mathbf{O}_{a_i} : |x_1, x_2 \ldots, x_l\rangle$

Definition 12 *Die Liste aller Angebote ist eine Aneinanderreihung der $\mathbf{O}_{a_i}$ aller Agenten $a_i \in A$.*
$|All\mathbf{O}(\mathbf{C})\rangle = |\mathbf{O}_{a_1}; \mathbf{O}_{a_2}; \ldots; \mathbf{O}_{a_n}; \rangle$

Definition 13 *Die Liste der optimalen Angebote ist eine Aneinanderreihung aller Angebote aus $\mathbf{O}_a$ eines Agenten a, für die es kein günstigeres Angebot $\mathbf{O}_{a_i}$ eines anderen Agenten $a_i \neq a$ gibt. Die Bezeichnung des Quantenregisters ist $|\mathbf{optO}\rangle$.*

Durch die Einschränkung auf l Aufgaben für jeden Agenten, kann l auf den Wert:

$$l = \max_{a' \in \mathbf{A}} \{\max\{\#\mathbf{T}_{a'}, \#\mathbf{O}_{a'}\}\} \tag{3.5}$$

gesetzt werden.

Quantenberechnung der lokalen Werte ($lw_a(C)$)

Hier wird eine Quantenberechnung der lokalen Werte angegeben. Der Vorgang wird in der Übersicht in Unterschritte zerlegt, gegliedert und diese dann einzeln beschrieben.

Übersicht
In Abschnitt 2.1.2 wurde die Definition des lokalen Wertes (vgl. Gleichung 2.10) angegeben.

$$lw_a(\mathbf{C}) = \sum_{x \in \mathbf{R}_a(\mathbf{C})} w_a(x) - \sum_{ws \in \mathbf{E}_a(\mathbf{C})} c_a(x)$$

Die erste Summe sind die Kosten der Nachfragen von Agent a, die andere Agenten aus der Koalition ausführen können. Die zweite Summe sind die Kosten der Angebote von Agent a, die der Agent für andere Agenten in der Koalition ausführt.
Hier wird mittels Quantenparallelität über alle Koalitionen $\mathbf{C}$ versucht, den Rechenaufwand zu verringern. Dazu wird die Gleichung in Teilschritte zerlegt und die Quantenberechnung der Schritte dann beschrieben.

1. Berechne die Kosten der Nachfragen von a parallel für jede Koalition $\mathbf{C}$.
   (a) Berechne eine Liste mit allen Angeboten der Agenten von $\mathbf{C}$.
   (b) Für jede Nachfrage von a suche ein Angebot aus der Liste.
   (c) Summiere die Kosten der Nachfragen auf, für die ein Angebot existiert.
2. Berechne die Kosten der Angebote von a parallel für jede Koalition $\mathbf{C}$.
   (a) Berechne eine Liste mit allen Nachfragen der Agenten in $\mathbf{C}$.
   (b) Berechne eine Liste der optimalen Angebote von a.
      i. Berechne eine Liste mit den Angeboten aller Agenten in $\mathbf{C}$.
      ii. Für jedes Angebot von Agent a berechne eine Liste von gleichen Angeboten von a und der Liste aller Angebote aus $\mathbf{C}$.
      iii. Suche das Angebot mit den minimalen Kosten aus dieser Liste.
      iv. Falls dieses Angebot von a stammt, füge es in die Liste der optimalen Angebote von a hinzu.
   (c) Für jedes optimale Angebot von a suche eine Nachfrage aus der Liste.
   (d) Summiere die Kosten der Angebote auf, für die eine Nachfrage existiert.
3. Subtrahiere die Werte, um $lw_a(\mathbf{C})$ für jede Koalition $\mathbf{C}$ zu bekommen.

Schritt 1: Berechne die Kosten der Nachfragen von a

In diesem Schritt sollen die Kosten berechnet werden, die durch das Abarbeiten der Aufträge von Agent a durch andere Agenten der Koalition entstehen.

Schritt 1a: Berechne eine Liste mit allen Angeboten der Agenten von $\mathbf{C}$.

Dieser Schritt baut ein Quantenregister in Superposition über alle Koalitionen auf, das als Index die Koalition beinhaltet und alle Angebote der Agenten in dieser Koalition $|\text{All}\mathbf{O}(\mathbf{C})\rangle$ auflistet. Dazu wird zuerst eine Quantenkodierung aller Angebote eines Agenten $\mathbf{O}_{a_i}$ für alle a_i in $\mathbf{A}$ benötigt. Diese Quantenregister befinden sich nicht in Superposition und können so durch eine Reihe von Sigma-X Operationen aus den klassischen Werten auf ein mit $|0,\ldots,0\rangle$ initialisiertes Quantenregister geschrieben werden. Um die Superposition aller Koalitionen zu erreichen wird ein leeres (mit $|0\rangle$ an allen Stellen) n-qubit Quantenregister genommen und auf jedes Qubit eine Hadamard Operation ausgeführt. Das Ergebnis entspricht genau einem Quantenregister auf dem alle möglichen Koalitionen parallel gespeichert sind. Der Zustand $|0,\ldots,0\rangle$ bleibt hier weiterhin bestehen, obwohl diese Koalition nicht zulässig ist. Alle weiteren Quantenberechnungen werden mittels kontrollierten Operationen (vgl. Abschnitt 2.2.2) durchgeführt, wodurch dieser Zustand nicht weiter verändert wird. Nun kann Algorithmus 7 ausgeführt werden, um den Zustand mit den richtigen Werten in den einzelnen Superpositionen zu füllen.

Algorithmen 7 Berechne Angebotsliste

Quantenvariablen:

$|\mathbf{O}_a\rangle = |\mathbf{O}_{a_1},\ldots\mathbf{O}_{a_n}\rangle$,
$|\text{All}\mathbf{O}(\mathbf{C})\rangle = (n\cdot l\cdot lx)$-qubit Quantenregister das mit $|0,\ldots,0\rangle$ initialisiert ist,
$|\mathbf{C}\rangle$ ein n-qubit Quantenregister mit allen Koalitionen.

1: **für** jeden Agenten $a_i \in A$ **tue**
2: **für** jedes Qubits $j \in \mathbf{O}_{a_i}$ **tue** ▷ Falls $a_i \in \mathbf{C}$
3: $0n2p\text{-CNOT}(\mathbf{C}_i, \mathbf{O}^j_{a_i} \rightarrow \text{All}\mathbf{O}(\mathbf{C})^j_i)$ ▷ kopiere $\mathbf{O}_{a_i}$ auf das leere Quantenregister
4: **ende für**
5: **ende für**

Beim Kopieren in Zeile 3 gibt $|\text{All}\mathbf{O}(\mathbf{C})^j_i\rangle$ die Stelle des leeren Quantenregisters an, an der für Agent a_i das j-te Qubit von $|\mathbf{O}_{a_i}\rangle$ stehen soll. Somit ist $|\text{All}\mathbf{O}(\mathbf{C})\rangle$ genauso lang wie alle $|\mathbf{O}_{a_i}\rangle, i = 1,\ldots,n$ zusammen. Und der Algorithmus schreibt nun in Abhängigkeit der Koalitionen aus $|\mathbf{C}\rangle$, $\mathbf{O}_{a_i}$ auf das Quantenregister oder es wird leer gelassen.

Beispiel 17 *Siehe Anhang A.3.1.*

Schritt 1b: Für jede Nachfrage von a suche ein Angebot aus der Liste.

In diesem Schritt sollen nun Quantenregister $|val\rangle$ für jede Nachfrage von a genau dann mit dem Wert w_a gesetzt werden, falls es mindestens ein Angebot in $|\text{All}\mathbf{O}(\mathbf{C})\rangle$ aus Algorithmus 7 gibt, das die gleiche Id wie die Nachfrage hat. Dh. es gibt einen Agenten, der die Nachfrage von a ausführen kann. Da im ersten Schritt nur Angebote aus gleichen Koalitionen in $|\text{All}\mathbf{O}(\mathbf{C})\rangle$ geschrieben wurden, können hier auch nur Agenten aus der gleichen Koalition herangezogen

werden.
Dazu werden Quantenregister $|val\rangle$ für jede Nachfrage von a benötigt, auf die der Wert w_a geschrieben werden soll. Für die Tests werden für jede Nachfrage Kontrollregister $|tmp\rangle$ benötigt, auf die das Ergebnis der Gleichheitstest geschrieben wird. Ein zusätzliches Quantenbit c wird für eine Nachfrage gesetzt, falls eines der tmp-Register für diese Nachfrage den Wert $|1\rangle$ beinhaltet.

Algorithmen 8 Suche Angebote für Nachfragen

Quantenvariablen:

$|\text{All}\mathbf{O}(\mathbf{C})\rangle$ = Ergebnis aus Algorithmus 7,
$|\mathbf{T}_a\rangle$,
$l \cdot |tmp\rangle = |0, \ldots, 0\rangle$ mit der Länge $n \cdot l$,
$l \cdot |val\rangle = |0, \ldots, 0\rangle$ mit der Länge l_c,
$|c\rangle = |0, \ldots, 0\rangle$ ein l-qubit Kontrollregister.

1: **für** jede Nachfrage $i \in \mathbf{T}_a$ **tue**
2: **für** jedes j-te Angebot $o^j_{a'} \in \text{All}\mathbf{O}(\mathbf{C})$ von Agent a' **tue**
3: Gleich($i.id$, $o.id \rightarrow tmp^j_{a'}(i)$) ▷ Teste auf gleiche Id
4: **ende für**
5: Oder($tmp(i) \forall a' \rightarrow c$) ▷ Falls eine der Ids gleich war, setze c
6: **für** jedes Qubit k von $i.cost$ **tue**
7: $0n2p$-CNOT($c, i.cost \rightarrow val$) ▷ Falls ein Angebot existiert, kopiere w_a auf val
8: **ende für**
9: **ende für**
10: **für** jede Nachfrage $i \in \mathbf{T}_a$ **tue**
11: Oder($tmp(i) \forall a' \rightarrow c$) ▷ Reinige $|c\rangle$
12: **für** jedes j-te Angebot $o^j_{a'} \in \psi \in \phi$ von Agent a' **tue**
13: Gleich($i.id$, $o.id \rightarrow tmp^j_{a'}(i)$) ▷ Reinige $|tmp\rangle$
14: **ende für**
15: **ende für**

Algorithmus 8 liefert somit eine Reihe von l Quantenregister, in denen die Werte der Nachfragen von a stehen, wenn es in der Koalition einen Agenten gibt, der diese Aufgabe lösen kann.

Beispiel 18 *Siehe Anhang A.3.2.*

Schritt 1c: Summiere die Kosten der Nachfragen auf, für die ein Angebot existiert.

In diesem Schritt sollen nun die Werte aller $|val\rangle$ Quantenregister aufsummiert werden. Dazu werden n-qubit Adder (vgl. Algorithmus 1) verwendet, die immer zwei Werte auf ein temporäres Quantenregister $|tmp^j_i\rangle$ addieren. Nach $\log_2 l$ Schritten können so alle Werte aufsummiert werden. In jedem Schritt wird das Ergebnisregister ein Qubit länger. Der Einfachheit halber sei hier ohne Beschränkung der Allgemeinheit angenommen, dass $l = 2^y, y \in \mathbb{N}$. Das Quantenregister $|tmp^j_i\rangle$ ist somit das Ergebnisregister in Schritt i des Wertepaars j. Die Variablen (a) bzw. (b) bezeichnen den ersten bzw. zweiten Summanden im Wertepaar. In diesem Algorithmus werden nur die Quantenregister $|val\rangle$ benötigt. Der Gesamtzustand des Systems beinhaltet jedoch weiterhin die Quantenregister aus dem vorherigen Algorithmus.

Algorithmen 9 Summiere die Kosten der Nachfragen

Quantenvariablen:
$l \cdot |val\rangle$ aus Algorithmus 8,
$(\sum_{i=1}^{\log_2 l} 2^{i-1}) \cdot |tmp_i^j\rangle$ Ergebnisregister.

1: **für** jede Stufe $i = \log_2 l$ bis 1 **tue**
2: **für** jedes Wertepaar j **tue** ▷ Immer 2 Quantenregister sollen miteinander addiert werden
3: Adder($tmp_i^j(a)$, $tmp_i^j(b) \rightarrow tmp_{i-1}^{\lceil \frac{j}{2} \rceil}$)
4: **ende für**
5: **ende für**
6: **für** jede Stufe $i = 2$ bis $\log_2 l$ **tue** ▷ Reinige alle $|tmp\rangle$, außer das Endergebnis
7: **für** jedes Wertepaar j **tue**
8: Adder(tmp_i^j, $tmp_i^j \rightarrow tmp_{i-1}^{\lceil \frac{j}{2} \rceil}$)
9: **ende für**
10: **ende für**

Die $|val\rangle$ Quantenregister werden in $|tmp_{\log_2 l}\rangle$ umbenannt. Abbildung 3.1 zeigt eine schematische Darstellung, wie die Quantenregister miteinander addiert werden.

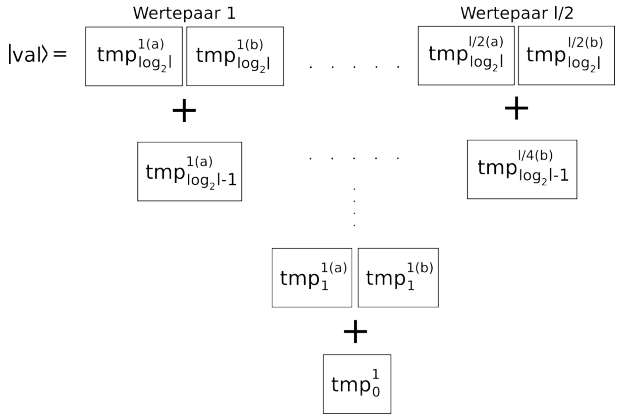


Abbildung 3.1: Aufaddieren der $|val\rangle$ Quantenregister.

Algorithmus 9 liefert als Ergebnis ein Quantenregister, das den Wert von $\sum_{x \in \mathbf{R}_a(\mathbf{C})} w_a(x)$ beinhaltet.

Beispiel 19 *Siehe Anhang A.3.3.*

Reinigen der Quantenregister

Um die nicht mehr benötigten Quantenregister mit den Zwischenergebnissen der einzelnen Schritte zu entfernen, müssen diese *gereinigt* werden. Reinigen bezeichnet dabei den Vorgang, die nicht mehr benötigten Quantenregister in allen Superpositionszuständen auf den gleichen Wert (meist der Ausgangswert) zu bringen. So sind diese Quantenregister nicht mehr mit den noch benötigten Quantenregistern verschränkt und können gemessen werden, ohne die Superposition der anderen Quantenregister zu zerstören oder zu reduzieren.

Durch die Evolution von Quantensystemen (vgl. Abschnitt 2.2.1) geht hervor, dass durch einen Matrixoperator U^{-1} ein Quantenzustand wieder in seinen Ausgangszustand überführt werden kann. Somit ist es möglich, durch das erneute Anwenden der Quantenschaltkreise, die Quantenregister wieder in ihren Ausgangszustand zu bringen. So wird durch erneutes Ausführen der Algorithmen in umgekehrter Reihenfolge, die Quantenregister auf den Wert $|0\rangle$ gesetzt und können danach gemessen werden.

1. Algorithmus 8 ausführen, um alle $|val\rangle$ Quantenregister zu reinigen.

2. Algorithmus 7 ausführen, um $|\text{All}\mathbf{O}(\mathbf{C})\rangle$ zu reinigen.

3. Die Quantenregister $|\mathbf{O}_{a_i}\rangle$ können direkt gemessen werden, da diese nie verändert wurden und so in allen Superpositionen zu dem gleichen Wert Messergebnis führen.

Schritt 2: Berechne die Kosten der Angebote von a

In diesem Schritt soll die Summe der Kosten, die durch das Ausführen von Aufgaben, von Agent a für andere Agenten in der Koalition, entstehen berechnet werden. Dabei gilt zu beachten, dass a nur dann eine Aufgabe zur Bearbeitung bekommt, wenn seine Kosten die niedrigsten für diese Aufgabe in der gesamten Koalition sind. Aus diesem Grund muss hier eine Liste erstellt werden, die alle optimalen Angebote von a enthält, dh. alle Angebote von a für die kein anderer Agent in der Koalition ein günstigeres Angebot hat.

Schritt: 2a: Berechne eine Liste mit allen Nachfragen der Agenten in $\mathbf{C}$.

Wegen der gleichen Kodierung von Angeboten und Nachfragen ist dieser Schritt analog zu Algorithmus 7. Es werden nur nicht die Quantenregister mit $\mathbf{O}_{a_i}$ verwendet, sondern mit $\mathbf{T}_{a_i}$.

Schritt 2b: Berechne eine Liste der optimalen Angebote von a.

In diesem Schritt soll eine Liste mit optimalen Angeboten erstellt werden. Diese beinhaltet alle Angebote von a, für die kein anderer Agent in der Koalition ein günstigeres Angebot hat.

Schritt 2(b)i: Berechne eine Liste mit den Angeboten aller Agenten in $\mathbf{C}$

Dieser Schritt ist in Algorithmus 7 beschrieben.

Schritt 2(b)ii: Berechne eine Liste von gleichen Angeboten

In diesem Schritt haben wir einen Quantenzustand, der die Koalitionen $\mathbf{C}$ und eine Liste aller Angebote $\text{All}\mathbf{O}(\mathbf{C})$ der Agenten $a_i \in \mathbf{C}$ in Superposition enthält. Ein weiteres leeres Quantenregister wird hinzugefügt, das genau dann mit einem Angebot beschrieben wird, wenn dieses die gleiche Id zu einem Angebot von a hat.

Beispiel 20 *Siehe Anhang A.3.4.*

Algorithmen 10 Berechne Liste von gleichen Angeboten

Quantenvariablen:

$|\mathbf{C}\rangle$ = Koalitionen in Superposition aus Algorithmus 7,
$|\text{All}\mathbf{O}(\mathbf{C})\rangle$ = Liste aller Angebote aus Algorithmus 7,
$|\mathbf{O}_a\rangle$ = Angebote von Agent a,
$|\mathbf{gO}\rangle$ = leeres Quantenregister gleicher Länge zu $|\psi\rangle$ aus Alg. 7,
$|c\rangle = |0\rangle$ ein Kontrollbit.

1: **für** jedes Angebot $o_a^i \in \mathbf{O}_a$ **tue**
2: **für** jedes j-Angebot $o_{a'}^j \forall a' \in \mathbf{C}$ **tue**
3: Gleich($o_a^i.id$, $o_{a'}^j.id \rightarrow c$) ▷ Vergleiche Angebote von a mit den anderen Angeboten aus $\mathbf{C}$.
4: **für** jedes Qubit $k \in o_{a'}^j$ **tue**
5: $0n2p$-CNOT(c, $o_{a'}^j(k) \rightarrow \mathbf{gO}_i^j(k)$ ▷ Kopiere das Angebot auf die richtige Stelle von $\mathbf{gO}$.
6: **ende für**
7: Gleich($o_a^i.id$, $o_{a'}^j.id \rightarrow c$) ▷ Reinigen von c.
8: **ende für**
9: **ende für**

Schritt 2(b)iii: Suche das Angebot mit den minimalen Kosten aus dieser Liste.

Aus dem vorherigen Schritt haben wir eine Liste mit gleichen Angeboten von Agenten aus $\mathbf{C}$ zu jedem Angebot von a berechnet. Da auch a in der Koalition $\mathbf{C}$ ist, beinhaltet diese Liste auch die Angebote von a selbst. Im folgenden betrachten wir die Liste als mehrere kleine Listen l_i von gleichen Angeboten i. Algorithmus 11 sucht nun das Angebot mit den kleinsten Kosten aus l_i und schreibt dieses auf ein neues Quantenregister min_i. Da "leere" Angebote die Kosten 0 haben, müssen diese abgefangen werden. Der Algorithmus vertauscht das erste Angebot der Liste mit einem anderen, falls dieses nicht-leer und geringere Kosten als das erste hat. Nach einem Listendurchgang steht somit das Angebot mit den geringsten Kosten an der ersten Stelle und es wird auf das Quantenregister min_i kopiert.

Beispiel 21 *Siehe Anhang A.3.5.*

Schritt 2(b)iv: Berechne die Liste der optimalen Angebote von a.

Als nächstes werden alle Angebote mit minimalen Kosten min_i geprüft, ob diese von Agent a stammen. Falls ja besitzt a ein optimales Angebot. Aus Algorithmus 11 stammen die Angebote min_i. Ein leeres Quantenregister $|\mathbf{optO}\rangle$ wird benötigt, auf das eventuelle optimale Angebote von a kopiert werden können. Algorithmus 12 berechnet somit eine Liste mit optimalen Angeboten von a.

Beispiel 22 *Siehe A.3.6.*

Algorithmen 11 Suche Angebot mit minimalen Kosten

Quantenvariablen:

$|\mathbf{l}_i\rangle \forall i \in \mathbf{O}_a$ = Ergebnis aus Algorithmus 10,
$|min_i\rangle \forall i \in \mathbf{O}_a = |0, \ldots, 0\rangle$ Ergebnisregister,
$|tmp\rangle$ = temporäres Quantenregister,
$|nzb\rangle, |bsa\rangle, |za\rangle, |c1\rangle, |c\rangle = |0\rangle$ Kontrollbits.

1: **für** jedes Angebot $i \in \mathbf{O}_a$ **tue**
2: Sei o_1 das erste Angebot der Liste $\mathbf{l}_i$.
3: **für** jedes Angebot $j \in \mathbf{l}_i$ ab dem 2. Angebot in der Liste **tue**
4: $xn0p$-CNOT($o_1.cost_0, \ldots, o_1.cost_{x-1} \rightarrow za$) ▷ Teste auf leeres erstes Angebot.
5: Oder($j.cost_0, \ldots, j.cost_{x-1} \rightarrow nzb$) ▷ Teste auf nicht-leeres Angebot.
6: Kleiner($j.cost, o_1.cost \rightarrow bsa$) ▷ Teste auf kleinere Kosten.
7: Oder($za, bsa \rightarrow c1$) ▷ falls das erste Angebot = 0 oder > j-te Angebot ist...
8: $0n2p$-CNOT($c1, nzb \rightarrow c$) ▷ ... und falls das j-te Angebot nicht null ist...
9: **für** jedes Qubit k in einem Angebot **tue** ▷ ... vertausche j mit o_1.
10: $0n2p$-CNOT($c, o_1^k \rightarrow tmp_k$) ▷ Kopiere o_1 nach tmp.
11: $0n2p$-CNOT($c, tmp_k \rightarrow o_1^k$) ▷ Reinige o_1.
12: $0n2p$-CNOT($c, j_k \rightarrow o_1^k$) ▷ Kopiere j nach o_1.
13: $0n2p$-CNOT($c, o_1^k \rightarrow j_k$) ▷ Reinige j.
14: $0n2p$-CNOT($c, tmp_k \rightarrow j_k$) ▷ Kopiere tmp nach j.
15: $0n2p$-CNOT($c, j_k \rightarrow tmp_k$) ▷ Reinige tmp.
16: **ende für**
17: $0n2p$-CNOT($c1, nzb \rightarrow c$) ▷ Reinige c.
18: Oder($za, bsa \rightarrow c1$) ▷ Reinige $c1$.
19: Kleiner($o_1.cost, j.cost \rightarrow bsa$) ▷ Reinige bsa.
20: Oder($0_1.cost_0, \ldots, o_1.cost_{x-1} \rightarrow nzb$) ▷ Reinige nzb.
21: $xn0p$-CNOT($j.cost_0, \ldots, j.cost_{x-1} \rightarrow za$) ▷ Reinige za.
22: **ende für**
23: **für** jedes Qubit k in einem Angebot **tue**
24: $0n1p$-CNOT($o_1^k \rightarrow min_i^k$) ▷ Kopiere das erste Angebot nach min.
25: **ende für**
▷ **Umkehrung der Vertauschungen.**
26: **für** jedes Angebot $j \in \mathbf{l}_i$ in umgekehrter Reihenfolge zu Zeile 3 **tue**
27: $xn0p$-CNOT($o_1.cost_0, \ldots, o_1.cost_{x-1} \rightarrow za$)
28: Oder($j.cost_0, \ldots, j.cost_{x-1} \rightarrow nzb$)
29: Kleiner($j.cost, o_1.cost \rightarrow bsa$)
30: Oder($za, bsa \rightarrow c1$)
31: $0n2p$-CNOT($c1, nzb \rightarrow c$)
32: **für** jedes Qubit k in einem Angebot **tue** ▷ Vertausche j mit o_1.
33: $1n2p$-CNOT($c, o_1^k \rightarrow tmp_k$) ▷ Kopiere o_1 nach tmp.
34: $1n2p$-CNOT($c, tmp_k \rightarrow o_1^k$) ▷ Reinige o_1.
35: $1n2p$-CNOT($c, j_k \rightarrow o_1^k$) ▷ Kopiere j nach o_1.
36: $1n2p$-CNOT($c, o_1^k \rightarrow j_k$) ▷ Reinige j.
37: $1n2p$-CNOT($c, tmp_k \rightarrow j_k$) ▷ Kopiere tmp nach j.
38: $1n2p$-CNOT($c, j_k \rightarrow tmp_k$) ▷ Reinige tmp.
39: **ende für**
40: $0n2p$-CNOT($c1, nzb \rightarrow c$)
41: Oder($za, bsa \rightarrow c1$)
42: Kleiner($o_1.cost, j.cost \rightarrow bsa$)
43: Oder($0_1.cost_0, \ldots, o_1.cost_{x-1} \rightarrow nzb$)
44: $xn0p$-CNOT($j.cost_0, \ldots, j.cost_{x-1} \rightarrow za$)
45: **ende für**
46: **ende für**

Algorithmen 12 Berechne Liste von optimalen Angeboten

Quantenvariablen:

$|min_i\rangle$ = Ergebnis aus Algorithmus 11,
$|\mathbf{optO}\rangle$ = leeres Ergebnisregister,
$|a\rangle$ = Quantenregister mit Agent a,
$|c\rangle = |0\rangle$ ein Kontrollbit.

1: **für** jedes Angebot min_i **tue**
2: Gleich(a, $min_i.agent \rightarrow c$) ▷ Teste ob min_i von a ist.
3: **für** jedes Qubit k in einer Aufgabe **tue**
4: $0n2p$-CNOT(c, $min_i^k \rightarrow \mathbf{optO}_i^k$) ▷ Kopiere das Angebot.
5: **ende für**
6: Gleich(a, $min_i.agent \rightarrow c$) ▷ Reinige c.
7: **ende für**

Schritt 2c: Für jedes optimale Angebot von a suche die Nachfragen aus der Liste.

Bisher wurde eine Liste mit allen Nachfragen $\mathbf{T}_{a_i}$ von allen Agenten a_i in der Koalition erstellt und eine Liste mit optimalen Angeboten $|\mathbf{optO}\rangle$ von Agent a berechnet. In Algorithmus 13 sollen Quantenregister $|val\rangle$ genau dann mit den Kosten eines optimalen Angebots beschrieben werden, falls dieses die gleiche Id wie eine Nachfrage aus $|\mathbf{T}_{a_i}\rangle$ hat. Der Unterschied zu Algorithmus 8 besteht darin, dass hier im Falle von mehreren gleichen Ids in $|\mathbf{T}_{a_i}\rangle$ die Kosten auch mehrfach berechnet werden müssen. Aus diesem Grund gibt es $l \cdot n$ $|val\rangle$ Quantenregister, da für jedes, der maximal l Angebote in $|\mathbf{optO}\rangle$, jeder, der maximal n Agenten in der Koalition, Nachfragen in $|\mathbf{T}_{a_i}\rangle$ hat.

Beispiel 23 *Wegen der großen Ähnlichkeit zu Algorithmus 12 siehe Anhang A.3.6.*

Algorithmen 13 Suche Nachfragen für optimale Angebote

Quantenvariablen:

$|\mathbf{T}_{a_i}\rangle$ = Ergebnis aus vorherigem Schritt,
$|\mathbf{optO}_a\rangle$ Ergebnis aus Algorithmus 12,
$|tmp\rangle = |0\rangle$ ein Kontrollbit,
$l \cdot l \cdot n|val\rangle = |0, \ldots, 0\rangle$ mit der Länge l_c.

1: **für** jedes Angebot $i \in \mathbf{optO}_a$ **tue**
2: **für** jede j-te Nachfrage $t_{a'}^j \in \mathbf{T}_{a'}$ von Agent a' **tue**
3: Gleich($i.id$, $t_{a'}^j.id \rightarrow tmp$) ▷ Teste auf gleiche Id.
4: **für** jedes Qubit k von $i.cost$ **tue**
5: $0n2p$-CNOT(tmp, $i.cost_k \rightarrow val_j^i(k)$) ▷ Kopiere Kosten auf entsprechendes val Quantenregister.
6: **ende für**
7: Gleich($i.id$, $t_{a'}^j.id \rightarrow tmp$) ▷ Reinige tmp.
8: **ende für**
9: **ende für**

Schritt 2d: Summiere die Kosten der Angebote auf, für die Nachfragen existieren.

Dieser Schritt ist analog zu Algorithmus 9 mit dem Unterschied, dass diesmal nicht l sondern $l \cdot l \cdot n$ $|val\rangle$ Quantenregister aufaddiert werden. Entsprechend steigt die Anzahl der benötigten Stufen. Als Ergebnis steht im letzten $|tmp\rangle$ Quantenregister der gewünschte Wert $\sum_{ws \in \mathbf{E}_a(\mathbf{C})} c_a(ws)$. Wie bei der vorherigen Summe können, durch erneutes Ausführen der Algorithmen, alle nicht weiter benötigten Quantenregister wieder gereinigt und danach gemessen werden.
Der resultierende Zustand ist ein Quantenregister mit den Koalitionen in Superposition und den Werten der Summe $\sum_{x \in \mathbf{R}_a(\mathbf{C})} w_a(x)$ und $\sum_{ws \in \mathbf{E}_a(\mathbf{C})} c_a(ws)$.

Beispiel 24 *Siehe Anhang A.3.3.*

Schritt 3: Subtrahiere die Werte, um $lw_a(\mathbf{C})$ für jede Koalition $\mathbf{C}$ zu bekommen.

Jetzt müssen die beiden Summenwerte nur noch subtrahiert werden. Dazu wird ein leeres Ergebnisregister $|lw\rangle$ benötigt, das durch einen n-qubit Subtrahierer beschrieben wird. Durch ein inverses Ausführen der Algorithmen zur Quantenberechnung der beiden Summen lassen sich diese Quantenregister wieder reinigen und können danach einfach gemessen werden. Der Endzustand sieht wie folgt aus.

$$|\chi\rangle \quad = \quad |\mathbf{C}; lw_a(\mathbf{C})\rangle \tag{3.6}$$

Beispiel 25 *Siehe Anhang A.3.7.*

Surplus-Quantenberechnung eines Agentenpaares (Zeile 19)

In diesem Abschnitt soll die Berechnung eines Surpluswertes für ein Agentenpaar quantenberechnet werden. Zuerst wird der Vorgang zerlegt und strukturiert, anschliessend dann Schritt für Schritt beschrieben.

Übersicht

Zeile 19 ist die zweite Stelle, an dem der klassische Algorithmus große lokale Berechnungen durchführen muss. Bei der Berechnung von Kernel-stabilen Konfigurationen, muss auf ein Gleichgewicht der Agentenpaare innerhalb der Koalition getestet werden (vgl. Kapitel 2.1.2). Die kritische Stelle dabei sind die Surplus-Werte, die für ein Agentenpaar (a_k, a_l) und alle möglichen Koalitionen (mit a_k und ohne a_l) berechnet werden müssen. Um den maximalen Excesswert zu erhalten, wird der Quantenalgorithmus zum Finden des Maximums (vgl. Abschnitt 2.2.5) verwendet. Ausgehend von den Koalitionen in Superposition liegt die eigentliche Arbeit in der Orakelfunktion. Dort werden die Excesswerte berechnet, die zum Markieren des richtigen Zustandes verwendet werden.

1. Erstellen eines Quantenzustandes in Superposition, der die Koalitionen $R \notin \mathbf{C}, a_k \in R, a_l \notin R$ enthält.
2. Paralleles Berechnen der Excesswerte für das Paar (a_k, a_l) und (a_l, a_k).

(a) Berechnen eines Quantenzustandes für die charakteristische Funktion $v(\mathbf{C})$.

(b) Berechnen eines Quantenzustandes für die Auszahlung $u(\mathbf{C})$.

(c) Berechnen eines Quantenzustandes für Excesswerte $e(\mathbf{C}, \mathbf{u})$.

3. Zwei mal Maximum Find Algorithmus von 2.2.5, um den Surplus s_{kl} und s_{lk} für das Agentenpaar zu erhalten.

Schritt 1: Erstellen eines Quantenzustandes in Superposition, der die Koalitionen entsprechend dem Agentenpaar enthält.

Als erstes muss ein Quantenzustand in Superposition erzeugt werden, der alle Koalitionen beinhaltet, die aus einem Agentenpaar (a_k, a_l), nach Definition von Surplus (vgl. Kapitel 2.1.2 $R \notin \mathbf{C}, a_k \in R, a_l \notin R$), gebildet werden können. Ausgabe sind zwei Quantenregister, eines für (a_k, a_l), das andere für (a_l, a_k). Diese beiden Quantenregister werden als Eingabe für den Quantenalgorithmus für das Finden des Maximums (Schritt 3) verwendet und müssen in jeder Iteration des Suchalgorithmus' neu berechnet werden.

Algorithmen 14 Erstellen der Koalitionen

Quantenvariablen:

$|p\rangle = n$-qubit Quantenregister für das Agentenpaar (a_k, a_l)

$|\mathbf{C1}; \mathbf{C2}\rangle = |0, \ldots, 0\rangle$ $2 \cdot n$-qubit Quantenregister für die Koalitionen.

Klassische Variablen:

n die Anzahl der Agenten

1: **für** i = 0 bis n-1 **tue**
2: $0n1p$-CNOT$(p_i \rightarrow \mathbf{C1}_i)$ ▷ Kopieren des Agentenpaares auf Quantenregister **C1**.
3: $1n0p$-Hadamard$(p_i \rightarrow \mathbf{C1})$ ▷ Erzeuge Superposition.
4: $0n1p$-CNOT$(\mathbf{C1}_i \rightarrow \mathbf{C2}_i)$ ▷ Kopieren von Quantenregister **C1** auf **C2**.
5: **ende für**
6: Sigma-X$(\mathbf{C1}_l)$ ▷ a_l aus Koalitionen **C1** entfernen.
7: Sigma-X$(\mathbf{C2}_k)$ ▷ a_k aus Koalitionen **C2** entfernen.

Algorithmus 14 erhält als Eingabe ein Quantenregister $|p\rangle$, das das Agentenpaar (a_k, a_l) darstellt. Ausgabe sind zwei Quantenregister mit Koalitionen, eines für (a_k, a_l) das andere für (a_l, a_k).

Beispiel 26 *Siehe Anhang A.3.8.*

Schritt 2: Paralleles Berechnen der Excesswerte für das Paar (a_k, a_l) und (a_l, a_k).

Für das Berechnen der Excesswerte müssen die Werte für die charakteristische Funktion und die Auszahlung der Koalitionen **C1** und **C2** berechnet werden. Im folgenden wird nur noch **C** geschrieben, da die Algorithmen einfach auf beide Koalitionsregister ausgeführt werden.

Schritt 2a: Berechnen eines Quantenzustandes für die charakteristische Funktion $v(\mathbf{C})$.

Als erstes soll die charakteristische Funktion (vgl. Kapitel 2.1.2) für die Koalitionen in **C** berechnet werden. Dazu werden die *lw*-Werte benötigt, die wie beschrieben anhand des Quantenregisterabschnittes für die Koalitionen berechnet werden können. Die *lw*-Werte werden

auf temporäre Quantenregister $|tmp\rangle$ kopiert, genau dann, wenn der entsprechende Agent in der Koalition ist. Die temporären Quantenregister können danach aufaddiert werden, wie in Algorithmus 9 beschrieben.

Algorithmen 15 Berechnen der charakteristischen Funktion

Quantenvariablen:

$|\mathbf{C}\rangle = n$-qubit Quantenregister für die Koalitionen,
$|index_{a_i}; lw_{a_i}\rangle = |0, \ldots, 0\rangle$ die n Index- und Werteregister für alle Agenten,
$|tmp_{a_i}\rangle = |0, \ldots, 0\rangle$ die n temporären Quantenregister für alle Agenten,
$|res\rangle = |0, \ldots, 0\rangle$ Ergebnisregister für $v(\mathbf{C})$.

Klassische Variablen:

n die Anzahl der Agenten

1: **für** $i = 0$ bis $n - 1$ **tue**
2: **für** $j = 0$ bis $n - 1$ **tue**
3: $0n1p$-CNOT($\mathbf{C}_j \rightarrow index^j_{a_i}$) ▷ Kopieren des Quantenregisters $\mathbf{C}$ auf das *index* Quantenregister.
4: **ende für**
5: Berechne die lw-Werte ▷ Wie oben in 3.2.2 beschrieben.
6: **für** jedes Qubit k von lw_{a_i} **tue**
7: $0n2p$-CNOT($\mathbf{C}_i$, $lw^k_{a_i} \rightarrow tmp^k_{a_i}$) ▷ Falls a_i in der Koalition, kopiere den Wert auf tmp_{a_i}.
8: **ende für**
9: **ende für**
10: Angepasster Algorithmus 9 ▷ Aufaddieren der *tmp* Quantenregister.
11: **für** $i = n - 1$ bis 0 **tue**
12: **für** jedes Qubit k von lw_{a_i} **tue**
13: $0n1p$-CNOT($\mathbf{C}_i$, $lw^k_{a_i} \rightarrow tmp^k_{a_i}$) ▷ Reinigen der *tmp* Quantenregister.
14: **ende für**
15: Inverse Berechnung der lw-Werte
16: **für** $j = 0$ bis $n - 1$ **tue**
17: $0n1p$-CNOT($\mathbf{C}_j \rightarrow index^j_{a_i}$) ▷ Reinigen der *index* Quantenregister.
18: **ende für**
19: **ende für**

Algorithmus 15 liefert so zu den Koalitionen in $|\mathbf{C}\rangle$ ein Quantenregister, das die Werte der charakteristischen Funktion enthält.

Beispiel 27 *Siehe Anhang A.3.9.*

Schritt 2b: Berechnen eines Quantenzustandes für die Auszahlung $u(\mathbf{C})$.

Hier soll die Auszahlung $u(\mathbf{C})$ berechnet werden. Dazu wird eine Kodierung des klassisch berechneten Auszahlungsvektors $\mathbf{u}$ benötigt. Die Kodierung besteht aus den Werten u_{a_i} aller Agenten und wird in die Quantenregister $|u_{a_i}\rangle$ geschrieben. Jetzt sollen die Werte in $|u_{a_i}\rangle$ genau dann aufsummiert werden, wenn a_i in der Koalition ist. Dies geschieht analog wie in Algorithmus 15 unter Verwendung des angepassten Algorithmus 9. Benötigt werden deshalb wieder temporäre Quantenregister $|tmp\rangle$ in Anzahl der Agenten.

Algorithmen 16 Berechnen der Auszahlung $u(C)$

Quantenvariablen:

$|\mathbf{C}\rangle = n$-qubit Quantenregister für die Koalitionen,
$|u_{a_i}\rangle = |0,\ldots,0\rangle$ die n Werteregister mit den Auszahlungen für alle Agenten,
$|tmp_{a_i}\rangle = |0,\ldots,0\rangle$ die n temporären Quantenregister für alle Agenten.

Klassische Variablen:

n die Anzahl der Agenten

1: **für** $i = 0$ bis $n-1$ **tue**
2: **für** jedes Qubit k von u_{a_i} **tue**
3: $0n2p$-CNOT($\mathbf{C}_i$, $u^k_{a_i} \rightarrow tmp^k_{a_i}$) ▷ Falls a_i in der Koalition, kopiere den Wert auf tmp_{a_i}.
4: **ende für**
5: **ende für**
6: Angepasster Algorithmus 9 ▷ Aufaddieren der *tmp* Quantenregister.
7: **für** $i = n-1$ bis 0 **tue**
8: **für** jedes Qubit k von u_{a_i} **tue**
9: $0n2p$-CNOT($\mathbf{C}_i$, $u_{a_i} \rightarrow tmp^k_{a_i}$) ▷ Reinigen der *tmp* Quantenregister.
10: **ende für**
11: **ende für**

Algorithmus 16 berechnet, in Abhängigkeit der Koalitionen, die Auszahlungen $u(\mathbf{C})$.
Der Algorithmus ist grundlegend analog zu Algorithmus 15, mit dem Unterschied, dass die Werte nicht durch einen anderen Algorithmus berechnet, sondern vorher aus klassischen Variablen auf das Quantenregister geschrieben werden.

Beispiel 28 *Siehe Anhang A.3.9.*

Schritt 2c: Berechnen eines Quantenzustandes für Excesswerte $e(\mathbf{C}, \mathbf{u})$.

Zur endgültigen Berechnung der Excesswerte (vgl. Kapitel 2.1.2) müssen die Werte $v(\mathbf{C})$ und $u(\mathbf{C})$ voneinander subtrahiert werden. Dazu wird ein leeres Ergebnisregister $|e\rangle$ benötigt, das durch einen n-qubit Subtrahierer beschrieben wird. Durch ein inverses Ausführen der Algorithmen zur Quantenberechnung von $v(\mathbf{C})$ und $u(\mathbf{C})$ lassen sich diese Quantenregister wieder reinigen und können danach einfach gemessen werden.

Beispiel 29 *Siehe Anhang A.3.10.*

Der Endzustand sieht wie folgt aus.

$$|\chi 1\rangle = |\mathbf{C1}; e(\mathbf{C}, \mathbf{u})\rangle \tag{3.7}$$
$$|\chi 2\rangle = |\mathbf{C2}; e(\mathbf{C}, \mathbf{u})\rangle \tag{3.8}$$

Jeweils für das Paar (a_k, a_l) und (a_l, a_k).

Schritt 3: Maximum Find Algorithmus, um den Surplus s_{kl} für das Agentenpaar zu erhalten.

Um das Maximum den Surpluswert zu berechnen wird ein Quantenalgorithmus 4 auf die Koalitionen von Quantenregister $|\mathbf{C1}\rangle$ und $|\mathbf{C2}\rangle$ angewendet, der genau die Koalition liefert die den maximalen Excesswert besitzt. Berechnet man diesen Surpluswert für alle Paare in einer Wunschkoalition, kann auf klassische Weise auf ein Gleichgewicht getestet werden.

Bei der Quantenberechnung des Maximums ist zu beachten, dass bei jedem Schleifendurchlauf der Ausgangszustand benötigt wird. Dh. er muss entweder geklont oder neu berechnet werden. Durch das No-Cloning-Theorem [WZ82] ist es nicht möglich genaue Kopien von unbekannten Quantenzuständen zu erhalten. Universal optimale Quanten Klon-Maschinen (UOQCM) können Approximationen eines Quantenzustandes herstellen [SIG$^+$05]. Diese erhöhen jedoch die Wahrscheinlichkeit auf ein falsches Endergebnis, weshalb hier die Werte jedes mal neu berechnet werden sollen. Dies geschieht mit Algorithmus 14 wie in Schritt 1 beschrieben.
Des Weiteren wird als y nicht der Index, sondern der Wert des aktuellen Maximums gespeichert. Dieser Wert und der Excesswert werden dann durch das Orakel erzeugt bzw. berechnet, verglichen und der Zustand eventuell markiert. Als Vergleichstest dient der Subtraktionsalgorithmus, wobei das letzte Ergebnis-Qubit genau dann den Wert $|1\rangle$ annimmt, wenn das Ergebnis der Subtraktion negativ ist. Wenn also alle Zustände markiert werden sollen, bei denen der aktuelle Excesswert größer als y ist, muss die Subtraktion von $y -$ Excesswert negativ sein. Als initialer y Wert wird Algorithmus 14 ausgeführt und die Excesswerte, wie in Schritt 2, berechnet. Der Zustand wird gemessen, um einen zufälligen Excesswert zu erhalten. y wird auf diesen Wert gesetzt. Eine schematische Arbeitsweise des Orakels ist in Abbildung 3.2 dargestellt.

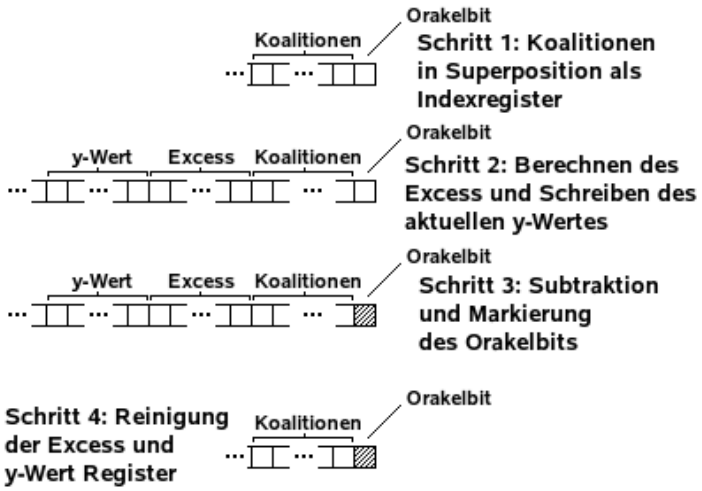


Abbildung 3.2: Orakel für Surplusberechnung

Vor dem Messen des Quantenregisters nach jeder Groversuche in Algorithmus 4 werden noch einmal mit Schritt 2 die Excesswerte auf das Quantenregister geschrieben, damit diese nach der Messung direkt abgelesen werden können. Sie dienen dann als neuer y Wert für den nächsten Durchgang und nach dem letzten Durchgang ist das aktuelle y gleich dem maximalen Excess und dieser ist gleich dem Surpluswert.

Beispiel 30 *Siehe Anhang A.3.11.*

Wahl des Koalitionsführers

Im klassischen Fall des Algorithmus wird der Koalitionsführer anhand einer Liste bestimmt, die nach der Rechenleistung der Agenten sortiert und in der initialen Phase erstellt wird.

In der Quantenspieltheorie gibt es einige Ansätze, um eine faire (rein zufällige) Wahl eines Führers zu vollziehen [PSK03], [HP06].
In [HP06] wird gezeigt, dass für eine korrekte, anonyme Quantenführerwahl (quantum leader election QLE) zwischen n Agenten, bei denen jeder Agent ein Qubit hält, ein verschränkter W_n Zustand ($|W_n\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |2^i\rangle$) eine notwendige und hinreichende Bedingung ist. Liegt ein solcher Zustand vor, kann jeder Agent, wie in Algorithmus 17 beschrieben, seinen Status (Führer oder Anhänger) herausfinden.

Algorithmen 17 Wahl eines Koalitionsführers

Quantenvariablen:
$p = i$-te Qubit von $W_{\mathbf{C}}$.
Klassische Variablen:
$b = 0$,
$result$.

1: b := measure(q)
2: **falls** $b = 1$ **dann**
3: $result$ = Führer
4: **sonst**
5: $result$ = Anhänger
6: **ende falls**

In einem Typ IIb QCMAS kann davon ausgegangen werden, dass ein solcher Zustand vorhanden ist. Falls dies jedoch nicht der Fall ist (es existieren nur ausreichend EPR Paare, aber keine verschränkten Zustände zwischen mehreren Agenten) stellt sich die Frage wie ein solcher W_n Zustand, bei dem jeder der n Agenten, einer neu gebildeten Koalition, ein Qubit besitzt, erzeugt bzw. verteilt wird. In einem ersten Ansatz wurde folgendes betrachtet:

- Zu Beginn einer Verhandlungsrunde verteilt jeder Führer einer Koalition mit c Mitgliedern einen c-qubit GHZ Zustand $|GHZ_c\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes c} + |1\rangle^{\otimes c})$. Diesen kann er entweder selbst erstellen und verteilen oder durch Entanglement Swapping (vgl. Abschnitt 2.2.1) aus vorhandenen EPR Paaren erzeugen.
- Im Falle einer Koalitionsfusion können die beiden betroffenen Führer, mit Hilfe von Entanglement Swapping, beide GHZ Zustände zu einem großen Zustand für die neue Koalitionen verbinden.
- Dieser neue GHZ Zustand kann zu einem W Zustand approximiert werden [WRZ05].
- Der approximierte W Zustand dient Algorithmus 17 als Eingabe [HP06].

In [DVC00] wird gezeigt, dass es zwei unterschiedliche Klassen von 3-qubit Verschränkungen gibt, wobei verschränkte Zustände der einen Klasse mit Hilfe von lokalen Operationen und klassischer Kommunikation (LOCC) nicht in die der anderen Klasse überführt werden können. So können GHZ_3 Zustände nicht in W_3 Zustände überführt werden. In [WRZ05] wird eine Approximation für diese Überführung gezeigt. Dabei besteht das Problem jedoch in genau dieser Approximation des W Zustandes. Je besser die gewünschte Approximation von W ist, desto

größer ist die Chance, dass bei der Approximation ein Fehler auftritt, da ein probabilistisches Protokoll zu Grunde liegt. Somit ist es für die Verwendung im KCA nicht wirklich geeignet, da bei guten verwendbaren Approximationen falsche Messergebnisse vorliegen können, bzw. bei weniger guten Approximationen diese Fehler in Algorithmus 17 zum Tragen kommen.

In einem primitiveren Ansatz würde der W Zustand erst nach der Koalitionsfusion verteilt werden. Ein solcher Zustand kann mit einem Trugenberger Speicher [Tru01] erstellt werden. Dieser würde dann auf die Mitglieder der Koalition verteilt werden, um danach Algorithmus 17 auszuführen.

Eine andere Möglichkeit ist, einfach die Führerrolle der neuen Koalition einem der beiden bisherigen Führer zu übertragen. Dabei verwenden diese einen $|\Psi^+\rangle$ Zustand (vgl. Gleichung 2.21) und führen Algorithmus 17 für einen 2 Agenten Fall durch.

3.2.3 Komplexität

In diesem Abschnitt wollen wir die Komplexität der in Kapitel 3.2.2 beschriebenen Algorithmen betrachten. Dabei wird ähnlich wie im vorherigen Kapitel vorgegangen. Zuerst werden die Quantenberechnungen der lw- und Excesswerte einzeln betrachtet und mit ihren klassischen Counterparts verglichen. Danach wird die Berechnung des Surpluswertes, bei der diese Werte eingesetzt werden, analysiert. Bevor am Ende die Komplexität für den gesamten KCA-Q vorliegt, muss noch der Schritt zur Wahl des Koalitionsführers analysiert werden.

Quantenberechnung der lw-Werte eines Agenten in allen Koalitionen

Wie in Abschnitt 3.2.2 erklärt, wird hier die Quantenberechnung der lw-Werte in die Quantenberechnung der beiden Summen (*Schritt 1* und *Schritt 2*) aufgeteilt, die am Ende miteinander subtrahiert werden. Durch die Quantenparallelität spielt es keine Rolle, für wie viele Koalitionen die Werte gleichzeitig berechnet werden. Die Komplexität zur Berechnung der lw-Werte beläuft sich damit auf:

$$\begin{aligned} \text{Berechnung} \quad : \quad & \text{Komplexität von Summe 1} + \text{Komplexität von Summe 2} + \\ & \text{Subtraktion} \end{aligned} \tag{3.9}$$

Die Quantenberechnung der ersten Summe (*Schritt 1*) wurde in folgende Teilschritte aufgeteilt.

- Berechne eine Liste mit allen Angeboten der Agenten von $\mathbf{C}$ (Alg. 7).
- Für jede Nachfrage von a suche ein Angebot aus der Liste (Alg. 8).
- Summiere die Kosten der Nachfragen auf, für die ein Angebot existiert (Alg. 9).

Algorithmus 7

Dieser Algorithmus berechnet eine Liste mit den Angeboten aller n Agenten aus der Koalition. Dazu werden die Angebote aller Agenten auf ein Quantenregister geschrieben und mit Hilfe von kontrollierten Operationen genau dann auf ein leeres Quantenregister geschrieben, falls

der Agent in der Koalition ist. Jeder der n Agenten hat maximal l Angebote der Länge lx (siehe dazu Abschnitt 3.2.2 Kodierung).

$$\begin{aligned} \text{Berechnung} &: n \cdot l \cdot lx \cdot CNOT \\ &= n \cdot l \cdot (n + l_{id} + l_c) \cdot CNOT \\ &= O(n^2) \end{aligned} \tag{3.10}$$

$$\begin{aligned} \text{Anzahl Qubits} &: n + n \cdot (l \cdot lx) + n \cdot (l \cdot lx) \\ &= n + 2nl \cdot (n + l_{id} + l_c) \\ &= O(n^2) \end{aligned} \tag{3.11}$$

Ein klassischer Algorithmus, der eine Liste aus Angeboten für Agenten einer Koalition erstellt könnte dies in $O(n)$ bewerkstelligen, indem er für jeden der maximal n Agenten in der Koalition die Angebote zu der Liste hinzufügt.

Algorithmus 8

In diesem Schritt werden jetzt die $n \cdot l$ Angebote aus der vorher erstellten Liste mit den l Nachfragen des Agenten verglichen und, im Falle von mindestens einem Treffer, die Kosten auf ein leeres Quantenregister kopiert.

$$\begin{aligned} \text{Berechnung} &: l \cdot (nll_{id} - \text{qubit}Gleich + nl - \text{qubit}Oder + l_c CNOT) + \\ &\quad l \cdot (nl - \text{qubit}Oder + nll_{id} - \text{qubit}Gleich) \\ &= O(n) \end{aligned} \tag{3.12}$$

$$\begin{aligned} \text{Anzahl Qubits} &: n \cdot l \cdot lx + l \cdot lx + l \cdot ln + l \cdot l_c + l \\ &= nl(n + l_{id} + l_c) + l(n + l_{id} + l_c) + lln + ll_c + l \\ &= O(n^2) \end{aligned} \tag{3.13}$$

Im klassischen Fall wären die Berechnungskosten für eine Koalition ebenfalls linear.

Algorithmus 9

Im letzten Schritt zur Quantenberechnung der ersten Summe werden die Quantenregister mit den Kosten der Nachfragen, für die es ein Angebot innerhalb der Koalition gibt, aufsummiert. Da dies maximal l Nachfragen sein können und l als Konstante angegeben wurde, hat dieser Algorithmus eine asymptotische Laufzeit von $O(1)$.

$$\begin{aligned} \text{Berechnung} &: 2 \cdot \sum_{i=1}^{\log_2 l} \frac{i^2}{2} \cdot (l_c + i) - \text{qubit}Adder \\ &= O(1) \text{da } l \text{ konstant} \end{aligned} \tag{3.14}$$

$$\begin{aligned} \text{Anzahl Qubits} &: l \cdot l_c + \sum_{i=1}^{\log_2 l} 2^{i-1} \cdot (l_c + i) \\ &= O(1) \end{aligned} \tag{3.15}$$

Auch der klassische Fall addiert hier nur eine konstante Anzahl von Werten für eine Koalition und hat deshalb eine konstante Laufzeit.

Zusammen hat die Quantenberechnung von Schritt 1 also folgende Laufzeit:

$$\begin{aligned} \text{Berechnung} &: \text{Schritt 1} \\ &= \text{Alg. 7} + \text{Alg. 8} + \text{Alg. 9} \\ &= O(n^2) + O(n) + O(1) \\ &= O(n^2) \end{aligned} \tag{3.16}$$

Auch die zweite Summe (*Schritt 2*) wurde in Teilschritte aufgeteilt:

- Berechne eine Liste mit allen Nachfragen der Agenten in **C** (Alg. 7).
- Berechne eine Liste der optimalen Angebote von a.
  - Berechne eine Liste mit den Angeboten aller Agenten in **C** (Alg. 7).
  - Für jedes Angebot von Agent a berechne eine Liste von gleichen Angeboten von a und der Liste aller Angebote aus **C** (Alg. 10).
  - Suche das Angebot mit den minimalen Kosten aus dieser Liste (Alg. 11).
  - Falls dieses Angebot von a stammt, füge es in die Liste der optimalen Angebote von a hinzu (Alg. 12).
- Für jedes optimale Angebot von a suche eine Nachfrage aus der Liste (Alg. 13).
- Summiere die Kosten der Angebote auf, für die eine Nachfrage existiert (Alg. 9).

Die noch nicht im ersten Schritt betrachteten Algorithmen werden auch hier zuerst einzeln analysiert.

Algorithmus 10

In diesem Algorithmus werden zu den l Angeboten von Agent a gleiche Angebote, aus der im vorherigen Schritt berechneten Liste aller Angebote, gesucht. Falls es solche gleichen Angebote gibt, werden diese zu einer Liste zusammengefasst.

$$\begin{aligned} \text{Berechnung} &: l \cdot ln(l_{id} - \text{qubit}Gleich + lxCNOT + l_{id} - \text{qubit}Gleich) \\ &= lln(2 \cdot l_{id} - \text{qubit}Gleich + (n + l_{id} + l_c)CNOT) \\ &= O(n^2) \end{aligned} \tag{3.17}$$

$$\begin{aligned} \text{Anzahl Qubits} &: n \cdot l \cdot lx + l \cdot lx + n \cdot l \cdot lx + 1 \\ &= 2nl(n + l_{id} + l_c) + l(n + l_{id} + l_c) + 1 \\ &= O(n^2) \end{aligned} \tag{3.18}$$

Ein vergleichbarer klassischer Algorithmus, der zu einem Agenten in einer Koalition die Angebote, die der Agent mit anderen aus seiner Koalition gemeinsam hat und diese einer Liste hinzufügt, hätte eine Laufzeit von $O(n)$.

Algorithmus 11

Hier wird aus den l Listen mit gleichen Angeboten das Angebot mit den minimalen Kosten auf ein leeres Quantenregister kopiert. Dazu wird für jedes Angebot getestet, ob die Kosten kleiner als die der bisherigen Angebote sind. Falls ja wird das Angebot mit dem veralteten Minimum ausgetauscht.

$$\begin{aligned} \text{Berechnung} &: l \cdot (2(n-1)(4 + (lx \cdot 6)CNOT + 2l_c - \text{qubit}Oder + \\ &\quad 22 - \text{qubit}Oder + 2l_c - \text{qubit}Kleiner) + lxCNOT) \\ &= l \cdot (2(n-1)(4 + ((n + l_{id} + l_c) \cdot 6)CNOT + 2l_c - \text{qubit}Oder + \\ &\quad 22 - \text{qubit}Oder + 2l_c - \text{qubit}Kleiner) + lxCNOT) \\ &= O(n^2) \end{aligned} \tag{3.19}$$

$$\begin{aligned} \text{Anzahl Qubits} &: l \cdot n \cdot lx + l \cdot lx + lx + 5 \\ &= ln(n + l_{id} + l_c) + l(n + l_{id} + l_c) + n + l_{id} + l_c + 5 \\ &= O(n^2) \end{aligned} \tag{3.20}$$

Ein vergleichbarer klassischer Algorithmus, der ein Minimum aus einer Liste sucht hat eine Laufzeit von $O(n)$.

Algorithmus 12

In diesem Schritt wird eine Liste erstellt, die nur Angebote der im vorherigen Schritt berechneten Minima beinhaltet, die von Agent a stammen. Auf diese Weise wird sicher gestellt, dass das Angebot das günstigste in dieser Koalition ist und dass es von Agent a kommt. Die resultierende Liste ist also eine Liste der Angebote von a, für die es keine günstigeren Angebote in der Koalition gibt.

$$\begin{aligned} \text{Berechnung} &: l \cdot (2n - \text{qubit}Gleich + lxCNOT) \\ &= l \cdot (2n - \text{qubit}Gleich + (n + l_{id} + l_c)CNOT) \\ &= O(n) \end{aligned} \tag{3.21}$$

$$\begin{aligned} \text{Anzahl Qubits} &: l \cdot lx + l \cdot lx + n + 1 \\ &= l \cdot (n + l_{id} + l_c) + l \cdot (n + l_{id} + l_c) + n + 1 \\ &= O(n) \end{aligned} \tag{3.22}$$

Um die Gleichheit von konstant vielen Angeboten zu einem Agenten zu testen, ist die Laufzeit bei einem klassischen Algorithmus $O(1)$.

Algorithmus 13

In diesem Algorithmus werden nun die Nachfragen aus der Koalition mit der Liste der optimalen Angebote von a abgeglichen. Für jede Nachfrage, für die a ein optimales Angebot besitzt, werden die Kosten dieses Angebots auf ein leeres Quantenregister kopiert.

$$\begin{aligned} \text{Berechnung} &: l \cdot l \cdot n(2l_{id} - \text{qubit}Gleich + l_c CNOT) \\ &= O(n) \end{aligned} \tag{3.23}$$

$$\begin{aligned} \text{Anzahl Qubits} &: n \cdot l \cdot lx + l \cdot lx + 1 + l \cdot l \cdot n \cdot l_c \\ &= n \cdot l \cdot (n + l_{id} + l_c) + l \cdot (n + l_{id} + l_c) + 1 + l \cdot l \cdot n \cdot l_c \\ &= O(n^2) \end{aligned} \tag{3.24}$$

Ein klassischer Algorithmus könnte dies in der gleichen asymptotischen Laufzeit bewerkstelligen.

Zusammen hat die Quantenberechnung von Schritt 2 also folgende Laufzeit:

$$\begin{aligned} \text{Berechnung} &: \text{Schritt 2} \\ &= \text{Alg. 7} + \text{Alg. 7} + \text{Alg. 10} + \\ &\quad \text{Alg. 11} + \text{Alg. 12} + \text{Alg. 13} + \\ &\quad \text{Alg. 9 (mit lln val-Quantenregistern)} \\ &= O(n^2) + O(n^2) + O(n^2) + O(n^2) + O(n) + O(n) + O(n) \\ &= O(n^2) \end{aligned} \tag{3.25}$$

Zum Quantenberechnen der lw-Werte eines Agenten für alle Koalitionen ist folgende Gesamtkomplexität nötig:

$$\begin{aligned} \text{Berechnung} &: \text{Schritt 1} + \text{Schritt 2} + \text{Schritt 3} \\ &= O(n^2) + O(n^2) + l_c - \text{qubit}Subtrahierer + n\mathbf{K} \\ &= O(n^2) \end{aligned} \tag{3.26}$$

$$\text{Anzahl Qubits} : O(n^2) \tag{3.27}$$

Wobei die Laufzeit verdoppelt wird, wegen der inversen Operationen zum Reinigen der Quantenregister, dies ändert jedoch nichts an der asymptotischen Laufzeit. **K** beschreibt eine Konstante, die für die Operationen stehen, die zum Erstellen der initialen Zustände benötigt werden.
Die Anzahl der Quantenbits, die für die gesamte Quantenberechnung der lw-Werte benötigt werden, ist das Maximum der Einzelschritte, da zwischen den Schritten nicht weiter benötigte Quantenregister gereinigt und gemessen werden. Auf diese Weise können diese Quantenregister entfernt werden.
Jetzt lässt sich die Komplexität des Gesamtschrittes auch mit der des klassischen Schrittes besser vergleichen. Der große Vorteil besteht darin, dass durch die Quantenparallelität und die Kodierung mit den verschiedenen Koalitionen als Superpositionszuständen, nur ein Durchlauf nötig ist, um die Daten für jede Koalition zu berechnen. Im klassischen Fall hingegen muss noch der exponentielle Faktor für das Berechnen der lw-Werte, für jede der 2^n möglichen Koalitionen, berücksichtigt werden. Am Ende des Quantenalgorithmus ist es zwar nicht möglich alle lw-Werte auszugeben, da sie nur in Superposition vorliegen. Bei der Surplusberechnung werden diese Werte jedoch direkt als Superposition benötigt, wodurch sich dieser Nachteil auflöst. Dadurch entsteht im Vergleich zur klassischen asymptotischen Laufzeit von $O(2^n)$ für diesen Schritt eine exponentielle Verbesserung.

Quantenberechnung der parallelen Excesswerte eines Agentenpaares

Die Quantenberechnung der parallelen Excesswerte eines Agentenpaares wird in folgenden Schritten, wie in Abschnitt 3.2.2 beschrieben, durchgeführt.

- Berechnen eines Quantenzustandes für die charakteristische Funktion $v(\mathbf{C})$ (Alg. 15).
- Berechnen eines Quantenzustandes für die Auszahlung $u(\mathbf{C})$ (Alg. 16).
- Berechnen eines Quantenzustandes für Excesswerte $e(\mathbf{C}, \mathbf{u})$.

Die Komplexität zum Berechnen der parallelen Excesswerte beläuft dadurch auf:

$$\text{Berechnung} \quad : \quad \text{Alg. } 15 + \text{Alg. } 16 + l_c - \text{qubit}\mathit{Subtrahierer} + \mathbf{K} \tag{3.28}$$

Wie schon bei den lw-Werten wird hier zuerst die Komplexität der einzelnen Algorithmen betrachtet und danach die des Gesamtschrittes.

Algorithmus 15

Hier werden die Werte der charakteristischen Funktion parallel für die gegebenen Koalitionen berechnet. Dabei werden die lw-Werte für alle Agenten in Superposition benötigt und aufaddiert, falls der Agent in der Koalition ist.

$$\begin{aligned} \text{Berechnung} \quad : \quad & n \cdot (nCNOT + O(n^2)(lw - \mathit{Berechnung}) + l_c CNOT) + \\ & O(n)(\text{Alg. } 9) + n(l_c CNOT + O(n^2) + nCNOT) \\ = \quad & O(n^3) \end{aligned} \tag{3.29}$$

$$\begin{aligned} \text{Anzahl Qubits} &: n + n \cdot (n + l_c) + n \cdot l_c + l_c + O(n^2) \\ &= O(n^2) \end{aligned} \tag{3.30}$$

Liegen im klassischen die lw-Werte bereits vor, kann dieser Schritt für eine Koalition in $O(n)$ durchgeführt werden.

Algorithmus 16

In diesem Schritt werden alle Auszahlungen der Agenten in einer Koalition aufaddiert. Dies geschieht parallel für jede Koalition. Die Auszahlungen liegen als klassische Variablen vor und können in linearer Laufzeit auf ein leeres Quantenregister geschrieben werden.

$$\begin{aligned} \text{Berechnung} &: n \cdot l_c CNOT + O(n)(\text{Alg. } 9) + n \cdot l_c CNOT \\ &= O(n) \end{aligned} \tag{3.31}$$

$$\begin{aligned} \text{Anzahl Qubits} &: n + n \cdot l_c + n \cdot l_c + l_c + O(n) \\ &= O(n) \end{aligned} \tag{3.32}$$

Im Klassischen hat dieser Schritt ebenfalls eine lineare Laufzeit.

Zum Quantenberechnen der Excesswerte ist die Gesamtkomplexität:

$$\begin{aligned} \text{Berechnung} &: \text{Alg. } 15 + \text{Alg. } 16 + l_c - \text{qubit}Subtrahierer + \mathbf{K} \\ &= O(n^3) + O(n) + l_c - \text{qubit}Subtrahierer + \mathbf{K} \\ &= O(n^3) \end{aligned} \tag{3.33}$$

Wobei die Laufzeit verdoppelt wird, wegen der inversen Operationen zum Reinigen der Quantenregister, dies ändert jedoch nichts an der asymptotischen Laufzeit. **K** beschreibt eine Konstante, die für die Operationen stehen, die zum Erstellen der initialen Zustände benötigt werden.
Auch hier ergibt sich die Anzahl der benötigten Quantenbits durch das Maximum der Einzelschritte.

$$\text{Anzahl Qubits} : O(n^2) \tag{3.34}$$

Im Vergleich zum Klassischen hat die Quantenberechnung der Excesswerte ebenfalls den Vorteil, dass die Werte parallel für jede Koalition berechnet werden, während ein klassischer Algorithmus diese für jede Koalition einzeln berechnen muss, was wieder zu einer exponentiellen Laufzeit, wegen der bis zu 2^n Koalitionen, führt. Da auch hier das Ergebnis in Superposition vorliegt, kann es so noch nicht wie im Klassischen verwendet werden.

Quantenberechnung der Surpluswerte eines Agentenpaares

Die Quantenberechnung der parallelen Excesswerte kann also durch einen polynomiellen Quantenalgorithmus erreicht werden. Um nun den Surpluswert eines Agentenpaares zu erhalten, was letztendlich das ist, was in dem klassischen KCA durch einen Quantenalgorithmus ersetzt wird, werden folgende Schritte ausgeführt.

- Erstellen eines Quantenzustandes in Superposition, der die Koalitionen entsprechend dem Agentenpaar enthält (Alg. 14).
- Durchführen von zwei Maximum Find Quantenalgorithmen, um die Koalition mit dem maximalen Excesswert zu finden (Alg. 4).

Algorithmus 14

Dieser Algorithmus erhält als Eingabe ein Agentenpaar auf einem Quantenregister und beschreibt zwei weitere Quantenregister mit allen möglichen Koalitionen, die den einen Agenten enthalten und den anderen nicht bzw. umgekehrt.

$$\begin{aligned} \text{Berechnung} &: 2 \cdot nCNOT + nHadamard + 2Sigma - X \\ &= O(n) \end{aligned} \tag{3.35}$$

$$\begin{aligned} \text{Anzahl Qubits} &: n + 2n \\ &= O(n) \end{aligned} \tag{3.36}$$

Zu beachten ist, dass hier direkt die Koalitionen sowohl für das Agentenpaar (a_k, a_l), als auch für (a_l, a_k) erstellt werden. Jedes der Koalitionsregister dient als Eingabe für einen der Quantenalgorithmen zum Finden eines Maximums.

Algorithmus 4

Der Algorithmus zum Finden des Maximums hat eine Laufzeit von $O(\sqrt{N})$ (vgl. [AK99, 4]), wobei N die Länge der Datenbank ist. Als Orakel werden die Excesswerte der Koalitionen mit dem derzeitigen Maximalwert verglichen und das Orakelbit entsprechend gesetzt. In unserem Fall ist die Länge $2^{(n-2)}$. Die Gesamtkomplexität ergibt sich durch:

$$\begin{aligned} \text{Berechnung} &: \text{Alg. } 14 + 2 \cdot \sqrt{2^{(n-2)}} \cdot \text{Orakelaufrufe} \\ &= O(n) + 2\sqrt{2^{(n-2)}} \cdot (2 \cdot O(n^3) + l_c - \text{qubit Größer}) \\ &= O(2^{\frac{n-2}{2}}) \end{aligned} \tag{3.37}$$

Im klassischen benötigt der gesamte Schritt zur Berechnung des Surpluswertes eines Agentenpaares $O(2^n)$ Schritte.

Führerwahl

Bei der Wahl eines Führers ist der einzige Rechenaufwand das Verteilen des verschränkten Zustandes. Algorithmus 17 selbst läuft in $O(1)$ ab, da jeder Agent nur sein Quantenbit messen muss. Das Erstellen eines W Zustandes erfordert $O(n)$ Quantenberechnungsschritte mit einem Trugenberger Speicher. Für das Verteilen eines Quantenzustandes auf n Parteien müssen n Quantenbits verschickt werden ($n-1$ falls einer der beteiligten Parteien den Zustand erstellt und verschickt).

$$\begin{aligned} \text{Berechnung} &: \text{Erstellen des W-Zustandes} + \text{Alg. 17} \\ &= O(n) + O(1) \\ &= O(n) \qquad (3.38) \\ \text{Kommunikation} &: \text{Verteilen des W-Zustandes} \\ &= O(n) \qquad (3.39) \end{aligned}$$

Ein vergleichbarer klassischer Algorithmus zur fairen Führerwahl hat mindestens eine Kommunikationskomplexität $O(n \log n)$ (vgl. [AG91]).

Gesamtkomplexität

Über n Verhandlungsrunden hat KCA-Q eine Komplexität von $O(n2^{\frac{n-2}{2}})$ für jeden Agenten. Die n Verhandlungsrunden des klassischen KCA haben eine Komplexität von $O(n2^n)$. Bei der Kommunikation entfällt nur das Versenden der lw-Werte aus der initialen Runde, was die Gesamtkomplexität für die Kommunikation jedoch nicht verändert. Sie beträgt, wie im klassischen Fall, $O(n^2)$ für jeden Agenten und $O(n^3)$ für das Gesamtsystem.

3.2.4 Beispiel

Da die Quantenversion KCA-Q nur einige Schritte im klassischen KCA ersetzt, bleibt die Grundstruktur (Angebote erstellen, Angebote evaluieren und eine Entscheidung über die neue Konfiguration treffen) in beiden Algorithmen erhalten. Abbildung 3.3 zeigt die Ausgangssituation eines Beispiels mit 3 Agenten. In Abbildung 3.4 sind die einzelnen Verhandlungsdurchläufe zu sehen, wie sich die Agenten zu Koalitionen zusammenschließen. Eine detaillierte Berechnung des Beispiels ist in Anhang A.2 zu finden. Die einzelnen Quantenalgorithmen zu diesem Beispiel werden in Anhang A.3 kurz und auf der CD-Rom (*examples.pdf*) zur Arbeit ausführlich durchgerechnet.

3.3 Evaluation

Eine richtige Evaluation der beiden Algorithmen KCA und KCA-Q ist nicht sinnvoll, da noch kein Quantencomputer existiert, auf dem KCA-Q getestet werden könnte. Hinzu kommt, dass alle Quantensimulatoren auf klassischen Computern eine exponentielle Verlangsamung erfahren (vgl. Abschnitt 2.3.2). Aus diesem Grund wird hier ein theoretischer Vergleich durchgeführt.
Die vorgestellte Quantenversion des KCA Algorithmus ersetzt die Schritte 11 und 19 des

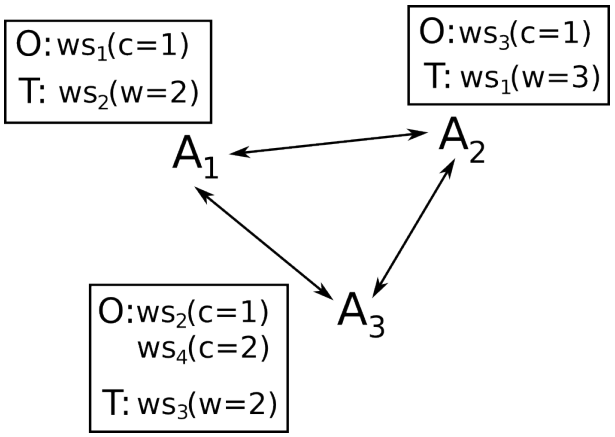


Abbildung 3.3: Ausgangssituation des Beispiels.

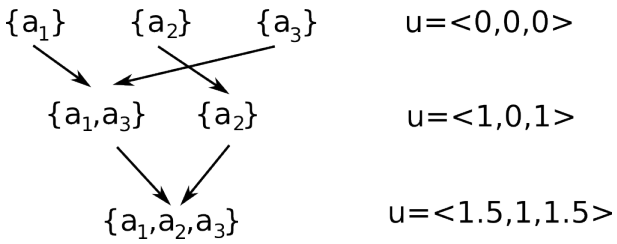


Abbildung 3.4: Zusammenschlüsse und Auszahlungen jeder Protokollrunde.

klassischen Protokolls. Durch die lokale Quantenberechnung der lw-Werte wird die gesamte Kommunikation gespart, die das Versenden der lw-Werte aller Agenten an alle Agenten verursacht. Es werden also für jeden der n Agenten n Nachrichten gespart. Auch die lokale Berechnung der Surpluswerte wurde verbessert. Die 2^n Rechnungsschritte für jeden Surpluswert kann durch die Quantenumsetzung und Groversuche um einen Wurzelfaktor reduziert werden. Asymptotisch gesehen fallen diese Verbesserungen jedoch nicht ins Gewicht. Da jeder Agent zu Beginn seine Angebote und Nachfragen an alle anderen Agenten versenden muss, fällt an dieser Stelle auch $O(n)$ Kommunikationsaufwand für alle Agenten an. Auch die lokalen Berechnungen, die in der Quantenversion in $O(2^{\frac{n}{2}})$ ablaufen, liegen noch in $O(2^n)$.
Die Führerwahl reduziert in keinster Weise den Kommunikationsaufwand, da im klassischen Fall der Führer einfach aus einer sortierten Liste ausgelesen und nicht durch Koordination mit anderen Agenten gewählt wird. Durch die Quantenversion wird jedoch das Erstellen und Sortieren der Liste in der initialen Phase gespart. Betrachtet man eine klassische, faire Führerwahl, ist diese mit einem Kommunikationsaufwand von mindestens $O(n \log n)$ Nachrichten verbunden ([AG91]). Verglichen damit kann auch hier eine Verbesserung erreicht werden, selbst wenn ein verschränkter Quantenzustand zuvor verteilt werden muss.
Die Implementierung von KCA-Q macht den zusätzlichen Rechenaufwand, der durch das Simulieren des Quantensystems anfällt deutlich. Schon ein 3-Agentensystem benötigt mit KCA-Q ein vielfaches mehr an Berechnungszeit, als das gleiche Beispiel mit KCA.

3.4 Diskussion

Durch die Quantenparallelität konnten zwar Verbesserungen erzielt werden, die gesamt gesehen jedoch die Komplexität des Protokolls nicht senken konnten. Durch die Aussagen in [Zal99, 2, 3] ist das hier vorgestellte Ergebnis, der Suche nach dem Maximum, optimal. Gäbe es einen Quantenalgorithmus, der aus der Superposition der Excesswerte das Maximum schneller findet, könnte er auch für andere Groversuchen verwendet werden, was der Aussage in [Zal99] widerspricht.
Bei der Führerwahl wurde, neben den Einsparungen in der initialen Phase, die richtige Zufälligkeit des Führers erreicht. Diese kann jedoch unterschiedlich gesehen werden. Auf der einen Seite besteht so für jeden Agenten die Chance Führer zu werden, auf der anderen Seite hat es Vorteile, wenn die Protokollschritte von den leistungsstärkeren Agenten ausgeführt werden. Auch zeichnet sich bei der beschriebenen Führerwahl das Problem ab, wie sehr ein solches System auf der Annahme beruht, dass ausreichend viele Verschränkungen vorhanden sind.
Es bleiben einige Fragen offen:

Wie lässt sich auf dieses Ergebnis aufbauen?

Eventuell können dennoch weitere Verbesserungen gefunden werden, indem die Superposition nach der Excess Quantenberechnung nicht mit Grover reduziert und durch die Messung aufgehoben wird, sondern indem auch noch weitere Schritte des klassischen Algorithmus quantenkodiert werden und den berechneten Zustand mit den Excesswerten weiter verwenden. Bei den Forschungen im Rahmen dieser Arbeit wurde dazu ein paralleler Gleichgewichtstest betrachtet. Der sich jedoch als nicht erfolgversprechend herausstellte.

Ergeben sich aus diesem Ansatz andere Möglichkeiten?

Neben dem Entwickeln neuer Quantenalgorithmen steht das Vereinfachen von bereits bestehenden Algorithmen. Bei Grovers Suchalgorithmus, macht vor allem das automatische Erstellen eines Orakels für kompliziertere Markierungen Probleme. Eine Fortführung der hier erstellten und verwendeten Schaltkreise für logische und arithmetische Operationen bis hin zu einer Art Quanten-ALU (arithmetisch-logische Einheit), die mit Hilfe eines Quanten-RISC Befehlssatzes angesprochen werden kann. Mit einer solchen Struktur und den formalen Werkzeugen, kann eine Automatisierung von Orakelfunktionen erreicht werden. Ähnlich wie sie in [Yam05, A] als C_f benötigt wird.

Zusammenfassend lässt sich sagen:

- Durch KCA-Q konnten Verbesserungen erreicht werden, die jedoch asymptotisch keinen Unterschied machen.
  - Bei der Berechnung ist KCA-Q wegen des Quantensuchalgorithmus schneller ($O(2^{\frac{n-2}{2}})$), als dies im Klassischen ($O(2^n)$) möglich ist.
  - Im Bereich der Kommunikation konnten, durch die Quantenführerwahl und die lokalen Berechnungen, quadratisch viele Nachrichten gespart werden.

- Der Charakter des Protokolls wurde nur durch die Quantenführerwahl verändert, ansonsten verhält es sich wie im Klassischen.
  - Der Führer einer Koalition wird rein zufällig und nicht aus einer sortierten Liste gewählt.

Kapitel 4

Quantenbasierte verteilte Problemlösung

In diesem Kapitel wird die verteilte Problemlösung untersucht und auf Hinblick einer Quantenumsetzung geprüft. Im Unterschied zu Kapitel 3 steht dabei die Koordination und Kommunikation im Vordergrund. Die lokalen Berechnungen, die in diesem Protokoll anfallen, werden weitgehend außer Acht gelassen. In Abschnitt 4.1 wird das klassische TRACONET Protokoll beschrieben. In Abschnitt 4.2 werden Ansätze für eine Quantenumsetzung vorgestellt und umgesetzt. Die Ergebnisse werden dann in Abschnitt 4.3 aufgezeigt und evaluiert. In Abschnitt 4.4 wird das Ergebnis diskutiert und einige offene Fragen gestellt.

4.1 TRACONET

In diesem Abschnitt wird das Kontraktnetzprotokoll TRACONET (TRansportation COoperation NET) [San93] genauer untersucht. Hier interagieren Agenten miteinander, um Aufgaben im Netzwerk zu verteilen und somit im Ganzen mehr Effizienz zu gewinnen. Dabei basiert das Protokoll auf grenzwertiger Kostenberechnung mit Hilfe von lokalen Kriterien der Agenten. Eine Gruppierung von Aufgaben ist möglich, um eine große Zahl von Ausschreibungen und Gebote zu bearbeiten und auch neue Ausschreibungen und Gebote abzugeben, obwohl das Resultat von vorherigen noch nicht bekannt ist. Das Protokoll ist asynchron, verteilt und unterstützt die Selbständigkeit der Agenten. Agenten berechnen die Kosten, um eine Menge von Aufgaben auszuführen. Auf Basis dieser Kosten werden alle Entscheidungen der Agenten getroffen.

Bei TRACONET versuchen geographisch verteilte Versandzentren von verschiedenen Firmen automatisch bei der Planung ihrer Fahrzeugrouten zu kooperieren. Dabei ist jedes Zentrum für die Auslieferungen von einer Reihe von Firmen zuständig und besitzt eine gewisse Anzahl von Fahrzeugen, um diese Lieferungen zu tätigen. Die Hauptoperationsgebiete der Zentren überlappen sich zu großen Teilen. Dies führt zu der Möglichkeit, dass mehrere Zentren eine bestimmte Lieferung effizient ausführen könnten. Jede Lieferung muss in die Route eines Fahrzeugs integriert werden. Das lokale Problem eines Agenten ist die kostengünstige Planung einer Auslieferungsroute mit mehreren Fahrzeugen und verschiedenen Depots. Die Fahrzeuge haben dabei Attribute wie Kosten pro Kilometer, maximale Reichweite, maximale Routendauer, maximale Transportgewicht, maximales Transportvolumen.

Um das Problem zu lösen, wird jedes Versandzentrum von einem intelligenten Agenten repräsentiert. Der Agent löst zuerst sein lokales Planungsproblem und betrachtet danach die Möglichkeit Aufgaben, für eine dynamisch berechnete Gebühr, eines anderen Zentrums zu übernehmen, bzw. eigene Aufgaben abzugeben. Bei den Verhandlungen tauschen Agenten Mengen von Aufgaben, wenn dies profitabel ist, dh. wenn ein Auftragnehmer die Aufgabe günstiger abarbeiten kann, als der Auftraggeber.

Jeder Agent besitzt zwei Protokollteile, das Verhandlungssystem und den lokalen Optimierer. Das Verhandlungssystem ist für die Kommunikation und Koordination zuständig und stellt somit das eigentliche Kontraktnetzprotokoll dar. Es besteht aus der lokalen Kontrolle die Nachrichten von anderen Agenten empfängt und diese in einer Kontrollschleife bearbeitet. Die Schleife ist in folgende Schritte unterteilt:

Ausschreibung: Ein Agent versucht für die Ausführung einer Menge von Aufträgen den Transportservice eines anderen Zentrums zu kaufen.

Gebot: Ein Agent versucht den eigenen Transportservice für eine Menge von Lieferungen zu verkaufen.

Zuschlag: Ein Agent kauft den Transportservice eines anderen Zentrums für eine Menge von Aufträgen.

Erhalten des Zuschlags: Ein Agent verkauft den eigenen Transportservice für eine Menge von Lieferungen.

Der lokale Optimierer berechnet die lokalen Lösungen eines Agenten, fügt neue Aufgaben zu diesen Lösungen hinzu oder entfernt sie und berechnet die Kosten für eine Menge von Aufgaben. Der lokale Optimierer ist somit für die Evaluation von Ausschreibungen und Geboten zuständig. Jeder Agent besitzt einen eigenen lokalen Optimierer, der auf die Bedürfnisse des Zentrums angepasst ist.

In dieser Arbeit wird vor allem das Verhandlungsprotokoll betrachtet, also die Interaktion der Agenten untereinander und nicht die Kostenberechnung unter Berücksichtigung der Interessen einzelner Agenten. Im folgenden (Abschnitt 4.1.1) wird deshalb nur das Protokoll für die Verhandlungen dargestellt.

Beispiel 31 *In Anhang A.4 wird ein Beispiel mit einem einfachen lokalen Optimierer gegeben.*

4.1.1 Protokoll

Im folgenden wird die Hauptschleife des Verhandlungsprotokolls von TRACONET in allgemeiner Form dargestellt und danach die in der Hauptschleife verwendeten Prozeduren einzeln erklärt.

Lokale Kontrolle *main()*

Jeder Agent im System durchläuft in der *main()* Prozedur (siehe Algorithmus 18.1) Verhandlungsrunden. Ein Listener empfängt alle Informationen (Ausschreibungen, Gebote, Zuschläge, Ablehnungen) die an den Agenten gesendet werden. Zuerst berechnet der Agent mit dem lokalen Optimierer eine Lösung für sein derzeitiges Planungssystem. Anhand dieser initialen Lösung kann er entscheiden, wie er sich in den Verhandlungen verhält, indem er die Kosten seiner lokale Lösung zum Vergleich nimmt. Bei erfolgreicher Verhandlung wird die lokale Lösung des Agenten aktualisiert. Als nächstes bearbeitet der Agent die bis dahin empfangenen Informationen durch die Prozeduren *bieten()*, *Zuschlag_geben()* und *Zuschlag_bekommen()*. Danach kann er neue Ausschreibungen mit der Prozedur *ausschreiben()* machen.
Zusätzliche Nachrichten, die während dem Ausführen von *bieten()*, *Zuschlag_geben()* und *Zuschlag_bekommen()* empfangen werden, werden in der nächsten Verhandlungsrunde behandelt. Dies verhindert, dass das System bei einer großen Menge von Nachrichten in einem der Prozeduren stecken bleibt.

Algorithmen 18 Verhandlungsprotokoll TRACONET

Klassische Variablen:
n die Anzahl der Agenten

```
1:  Prozedur main()                    ▷ Ablauf der Verhandlungsschleifen (lokale Kontrolle).
2:      starte Listener                ▷ Empfängt alle Verhandlungsdaten
3:      wiederhole
4:          Berechne lokale Lösung.    ▷ Durch lokalen Optimierer.
5:          bieten()
6:          Zuschlag_geben()
7:          Zuschlag_bekommen()
8:          ausschreiben()
9:      bis Netzwerk verlassen wurde
10: ende Prozedur
```

Ausschreibungen machen *ausschreiben()*

Die Ausschreibungen werden in Prozedur *ausschreiben()* (siehe Algorithmus 18.11) erstellt. Dabei sucht sich der Agent anhand der aktuellsten Version seiner Lösung eine Menge von Aufgaben, die er an andere Agenten abgeben möchte. Damit das Zustandekommen eines Vertrages wahrscheinlicher wird, wird beim TRACONET die Menge der Aufgaben nach dem Operationsgebiet der anderen Agenten gewählt. In einer Verhandlungsrunde, kann jeder Agent nur eine Ausschreibung machen. Beim Ausschreiben bestehen unterschiedliche Möglichkeiten zur Auswahl der auszuschreibenden Aufgaben und ob eine Aufgabe mehrmals ausgeschrieben werden darf. In Algorithmus 18.11 ist die Auswahl neben dem Operationsgebiet zufällig, Wiederholungen sind erlaubt und die Aufgabenmenge, die ausgeschrieben werden soll, ist auf eine Aufgabe pro Ausschreibung beschränkt. Jede Ausschreibung enthält Kosten, die der Agent bereit ist für die Abgabe der Aufgabe zu zahlen.

11: **Prozedur** *ausschreiben()* ▷ Schreibt Mengen von Aufgaben für andere Agenten aus.
12: wähle $T = \{$ausgewählte Aufgabe$\}$, die in dem Operationsgebiet anderer Agenten liegt.
13: erstelle Ausschreibung t
14: $c_{max} = c'_{rem}(T)$ ▷ Berechnet durch lokalen Optimierer.
15: **für** alle anderen Agenten a' **tue**
16: **falls** Aufgabenziel liegt in Operationsgebiet von a' **dann**
17: sende(t, a')
18: **ende falls**
19: **ende für**
20: **ende Prozedur**

Gebote abgeben *bieten()*

Die Prozedur *bieten()* (siehe Algorithmus 18.21) liest alle eingegangenen Ausschreibungen. Falls die Kosten für das Abarbeiten der Aufgabe geringer sind, als das, was der ausschreibende Agent bereit ist zu zahlen, wird ein Gebot abgegeben. Es ist für einen Agenten möglich mehrere Gebote für unterschiedliche Ausschreibungen abzugeben. So kommt es zu einer Situation, in der ein Agent Entscheidungen treffen muss, obwohl er nicht genau weiß, wie viele seiner aktuell abgegebenen Gebote den Zuschlag bekommen. In Algorithmus 18.21 gibt der Agent nur dann ein Gebot ab, wenn dieses für ihn, in dem Fall, dass andere offene Gebote nicht angenommen werden, profitabel ist.

21: **Prozedur** *bieten()* ▷ Liest die Ausschreibungen, die von anderen Agenten gesendet wurden.
22: $E = \{$empfangenen Ausschreibungen$\}$
23: **für** jede Ausschreibung $t \in E$ **tue**
24: **falls** $f'(T_{cur} \cup T_t \cup T^t_{pos}) < \infty$ **dann** ▷ Machbarkeits-Check; Berechnet vom lokalen Optimierer.
25: $c^t_{bid} = c'_{add}(T_t)$ ▷ Berechnet vom lokalen Optimierer.
26: **falls** $c^t_{bid} < c^t_{max}$ **dann**
27: erstelle Gebot b
28: sende(b,Ausschreiber(t))
29: **ende falls**
30: **ende falls**
31: **ende für**
32: **ende Prozedur**

Gebote evaluieren *Zuschlag_geben()*

Die Prozedur *Zuschlag_geben()* (siehe Algorithmus 18.33) bearbeitet alle eingegangenen Gebote nach Ablauf eines Timestamps. Der Agent mit dem besten Gebot bekommt einen Zuschlag geschickt, alle anderen eine Ablehnung. Falls kein Angebot eingegangen ist, kann der Agent einen zweiten Timestamp abwarten und danach Ablehnungen an alle schicken, wenn weitere Gebote ausblieben. Im Falle eines Zuschlags wird die Lösung des Agenten ohne die Aufgaben aus der Ausschreibung aktualisiert. Auch hier ist das beste Gebot abhängig von den noch offenen Geboten des auszuwertenden Agenten.

33: **Prozedur** *Zuschlag_geben()* ▷ Liest und evaluiert die Gebote.
34: **falls** Gebote nach *Timestamp1* empfangen wurden **dann**
35: **für** jedes Gebot b aus allen empfangenen Geboten B **tue**
36: **falls** $c_b = \min_{b' \in B}\{c_{b'}\}$ **dann**
37: sende(Zuschlag,Sender(b))
38: **ende falls**
39: **ende für**
40: **für** jeden anderen Agenten a, an den die Ausschreibung t gesendet wurde. **tue**
41: sende(Ablehnung,a)
42: **ende für**
43: $T_{cur} = T_{cur} \setminus T_t$
44: **sonst**
45: **falls** Zuschlag_geben() zum ersten mal für t ausgeführt wurde **dann**
46: Warte(*Timestamp2*)
47: Zuschlag_geben()
48: **sonst**
49: sende(Ablehnung, alle Agenten a die t gesendet bekamen)
50: **ende falls**
51: **ende falls**
52: **ende Prozedur**

Zuschlag_bekommen()

Für jeden Zuschlag den ein Agent bekommt (siehe Algorithmus 18.53) aktualisiert er seine Lösung, indem er die Menge der Aufgaben, die er neu erhalten hat, hinzufügt. Für den Fall, dass im Verlauf der Verhandlungen, durch andere parallel laufende Verhandlungen, die gerade erhaltene Aufgabenmenge für den Agent nicht mehr profitabel ist, muss er sie dennoch annehmen. Er kann sie aber danach ganz oder teilweise wieder ausschreiben.

53: **Prozedur** *Zuschlag_bekommen()* ▷ Liest und evaluiert die Zuschläge.
54: $T_{cur} = T_{cur} \cup T_t$
55: **ende Prozedur**

Netzwerk_betreten()

Das Betreten eines bereits bestehenden Verhandlungsnetzwerks ist recht einfach, der Agent löscht alle Daten von früheren Verhandlungen und beginnt mit den Verhandlungsrunde, indem er die lokale Kontrolle *main()* startet.

56: **Prozedur** *Netzwerk_betreten()* ▷ Neuer Agent betritt das Netzwerk.
57: Lösche Daten aus vorherigen Verhandlungen.
58: main()
59: **ende Prozedur**

Netzwerk_verlassen()

Beim Verlassen des Netzwerkes ist zu beachten, dass kein anderer Agent auf Nachrichten wartet. Aus diesem Grund wartet der Agent vor dem Verlassen auf Zuschläge oder Ablehnun-

gen für alle seine offenen Gebote und sendet Ablehnungen an alle Empfänger seiner offenen Ausschreibungen.

60: **Prozedur** *Netzwerk_verlassen()* ▷ Agent verlässt das Netzwerk.
61: **für** alle abgegebenen Gebote **tue**
62: Warte auf Ablehnung oder Zuschlag.
63: **ende für**
64: **für** alle offenen Ausschreibungen t **tue**
65: sende(Ablehnung,Empfänger(t))
66: **ende für**
67: **ende Prozedur**

Variablenerklärung:

- $Z = \{z_1, \ldots, z_n\}$**:** Die Menge der Agenten im System.
- T^z_{cur}**:** Die Menge der Aufgaben von z.
- $A^z = \{a^z_1, \ldots, a^z_n\}$**:** Die Menge der offenen Ausschreibungen von z.
- T_a**:** Die Menge der Aufgaben in der Ausschreibung a.
- c^a_{max}**:** Die maximalen Kosten, die der Sender von a zu zahlen bereit ist.
- $c_{rem}(T)$**:** Kosten die gespart werden, wenn die Menge der Aufgaben T von der Planung des Agenten gelöscht wird.
- $c'_{rem}(T)$**:** Approximation von $c_{rem}(T)$.
- c^a_{bid}**:** Kosten die der Agent bereit ist, die Aufgaben in a zu übernehmen.
- B_{uns}**:** Menge von offenen Geboten, die gesendet wurden.
- B_{pos}**:** Menge von offenen Geboten, für die der Agent ein Zuschlag bekommen kann, wenn er einen Zuschlag für das Gebot aktuelle b erhält. $B_{pos} = \{x | x \in B_{uns}, T_x \cap T_b = \emptyset\}$
- T_{pos}**:** Menge der Aufgaben, die in den Geboten B_{pos} enthalten sind.
- $f(T)$**:** Die Funktion berechnet die totalen Kosten der lokal optimalen Lösung.
- $c_{add}(T)$**:** Die grenzwertigen Kosten, die durch das hinzufügen der Menge T zu der lokalen Lösung entstehen.
- $c'_{add}(T_b)$**:** Die inkrementell berechnete heuristische Lösung unter der Annahme, dass der Agent den Zuschlag von keinem der offenen Gebote erhalten hat.
- **sende**(d, e)**:** Das Senden von Daten d an einen Empfängere.

4.1.2 Komplexität

Bei Kommunikationskomplexität eines einzelnen Agenten für die Verhandlung einer Ausschreibung muss zwischen der Rolle des Managers und des Auftragnehmers unterschieden werden. Der Manager hat folgenden worst case Kommunikationsaufwand:

$$\begin{aligned} \text{Kommunikation (Manager)} &= O(\text{ausschreiben}()) + O(\text{Zuschlag\_geben}()) \\ &= O(n+n) \qquad (4.1) \\ \text{Kommunikation (Auftragnehmer)} &= O(\text{bieten}()) + O(\text{Zuschlag\_bekommen}())) \\ &= O(1+0) \qquad (4.2) \end{aligned}$$

Dabei sind die beiden n auf der Managerseite, das Senden einer Ausschreibung an jeden Agenten und das Senden von Ablehnungen bzw. eines Zuschlages an diese Menge. Bei den Auftragnehmer wird nur ein Gebot gesendet.

Die lokalen Berechnungskosten belaufen sich auf $O(2^n)$, sowohl für den Manager, als auch für die Auftraggeber (siehe [San93]).

4.2 TRACONET-Q

Im Gegensatz zu Abschnitt 3.2 stehen bei der Untersuchung der verteilten Problemlösung die Interaktion zwischen den Agenten im Vordergrund. Die lokalen Berechnungen lassen sich, ähnlich wie in dem vorherigen Kapitel, beschränkt verbessern, werden hier aber nicht näher betrachtet. Zum einem werden vorhandene Ergebnisse in der Quantenberechnung und Kommunikation auf das TRACONET Protokoll angewendet, zum anderen werden die Kommunikationsstrukturen in TRACONET untersucht, um eine bessere Quantenumsetzung dafür zu finden.

4.2.1 Änderungen zu TRACONET

In diesem Abschnitt werden die Ansätze vorgestellt, die für eine Quantenumsetzung des TRACONET Protokolls vielversprechend aussehen. Die Ähnlichkeit der Struktur der Verhandlungen in TRACONET zu Auktionen führte direkt zu einer Umsetzung der in [HHC07, 2, 3] und [GHF$^+$07, 2] beschriebenen Quantenauktionen. Des Weiteren wird Verschränkung als Hilfsmittel zur Koordination und die Reduzierung der Nachrichtenlänge betrachtet. Um die Multicasts des Managers zu verbessern, können verschiedene Ansätze verwendet werden, die die Sicherheit zu erhöhen.

Verschränkung als Hilfsmittel zur Koordination

Wie schon bei der Führerwahl in Abschnitt 3.2.2 können auch im TRACONET verschränkte Zustände eingesetzt werden, um die Koordination zwischen den Agenten zu gewährleisten. Zum Beispiel kann die in Abschnitt 2.2.4 beschriebene kontrollierte Teleportation helfen die Reihenfolge verschiedener Teilergebnisse, die von unterschiedlichen Agenten ausgeführt wurden, zu koordinieren. Dazu werden Aufgaben quantenkodiert und der Manager vergibt einen verschränkten Zustand an alle beteiligte Agenten. Wenn der Agent mit dem ersten Teilergebnis fertig ist, kann er seinen Teil des verschränkten Zustandes messen und die Information

an den zweiten Agenten schicken. Damit ist dieser in der Lage, seine Aufgabe zu erhalten und erst jetzt auszuführen. Somit könnte der gleichzeitige Zugriff einer gemeinsam genutzten, sicherheitskritischen Ressource innerhalb der Aufgabe koordiniert werden.

Dense Coding zur Reduzierung der Nachrichtenlänge

Verschränkung kann auch zur Reduzierung der Nachrichtenlängen verwendet werden. Durch Superdense Coding (vgl. Abschnitt 2.2.4) kann eine im klassischen optimal komprimierte Nachricht von n Bits nach vorheriger Absprache der Kodierungsoperatoren mit $\frac{n}{2}$ Qubits übertragen werden, wenn genügend verschränkte Paare zur Verfügung stehen. So können alle Nachrichten im Protokoll (Ausschreibungen, Gebote, Zuschläge usw.) noch weiter komprimiert werden.

Quantenauktionen

Vielversprechend für eine Quantenumsetzung sind die in Abschnitt 2.2.5 beschriebenen Quantenauktionen. Der Auktionär ist dabei der Agent, der versucht eine Aufgabe an andere Agenten abzutreten. Die Bieter sind alle Agenten, an die der Manager im klassischen Fall eine Ausschreibung senden würde. Der Gewinner der Auktion bekommt diese Aufgabe dann übertragen. Dabei sind einige Modifikationen nötig. Der Auktionär legt einen Höchstpreis für die Menge von Aufgaben fest, die er ausschreiben möchte. Die Bieter unterbieten sich gegenseitig mit dem Preis, den sie bekommen, wenn sie die Aufgaben übernehmen.

Multicasting

In unserem Protokoll wird die meiste Kommunikation von der Seite des Managers ausgeführt. In Zeile 17 macht er eine Ausschreibung an eine Menge von Agenten. In der Quanteninformation ist es problematisch einen Multi- oder Broadcast auszuführen, da ein Quantenzustand nicht kopiert werden kann [WZ82]. Die Quantenkommunikation stellt trotzdem einer der größten Forschungsbereiche der Quantenmechanik dar. In [WZT07] werden 3 Möglichkeiten vorgestellt, um einen Multicast von klassischer Information mit Hilfe von Quantenkommunikation durchzuführen. Dabei ist die Resultierende Verbesserung nicht eingesparte Kommunikation, sondern der Benefit an Sicherheit, die erreicht wird.

- Quantenbroadcast-Kommunikation mit Entanglement Swapping
- Quantenbroadcast-Kommunikation mit Dense Coding
- Quantenbroadcast-Kommunikation mit Quantenverschlüsselung

Eines dieser Modelle kann auch direkt für TRACONET verwendet werden, um die Agenten sicher miteinander kommunizieren zu lassen. Dabei ist es nötig einen gemeinsamen Schlüssel sicher auszutauschen. Auch dies wurde in der Quantenkommunikation intensiv untersucht [ACL99], [BB84, 3]. Eine Veränderung des klassischen TRACONET wäre, hier die gesamte Multicast-Kommunikation durch Quantenmulticast-Kommunikation zu ersetzen, um die Vorteile der Sicherheit zu erhalten.

4.2.2 Quantenumsetzung

Der Kern der Quantenumsetzung für TRACONET bilden die Quantenauktionen. Dabei werden zu den Ausschreibungen Quantenzustände verteilt, die für die Auktionen verwendet werden. Das Protokoll für eine Verhandlung sieht wie folgt aus:

1. Ausschreibung mit Hilfe von Quantenbroadcast-Kommunikation.
2. Ausführung einer Quantenauktion unter den Interessenten.

Lokale Kontrolle

Die lokale Kontrolle (siehe Algorithmus 19 *main()*) ist stark an den klassischen Fall angelehnt. Die Agenten treffen alle Entscheidungen mit Hilfe des lokalen Optimierers (die Variablenbelegung für Kosten usw. sind wie im klassischen Fall). Der Q-Listener (Zeile 2) führt alle Kommunikationsprotokolle durch, die beim Empfangen von Nachrichten anfallen (vgl. Schritt 1 und Abschnitt 2.2.4). *q-bieten()* und *q-verteilen()* sind die bieter- bzw. auktionärsseitigen Prozeduren der Quantenauktion (vgl. Schritt 2). *q-Zuschlag_geben()* ist das Versenden der Ablehnungen und der Gewinnernachricht und das Löschen der Aufgabe aus der lokalen Lösung des Managers, *q-Zuschlag_bekommen()* bezeichnet das Hinzufügen der Aufgabe in die lokale Lösung des Agenten. *q-ausschreiben()* beinhaltet das Ausschreiben einer Aufgabe an eine Menge von Agenten (vgl. Schritt 1).

Algorithmen 19 Verhandlungsprotokoll TRACONET-Q

Klassische Variablen:
n die Anzahl der Agenten
$K_{ij} \forall i, j \in \{1, \ldots, n\}$ ein Schlüssel für der Agentenpaar (a_i, a_j)

```
1:  Prozedur q-main()                ▷ Ablauf der Verhandlungsschleifen (lokale Kontrolle).
2:      starte Q-Listener            ▷ Empfängt alle Verhandlungsdaten
3:      wiederhole
4:          Berechne lokale Lösung   ▷ Durch lokalen Optimierer.
5:          q-verteilen()
6:          q-bieten()
7:          q-Zuschlag_geben()
8:          q-Zuschlag_bekommen()
9:          q-ausschreiben()
10:     bis Netzwerk verlassen wurde
11: ende Prozedur
```

Schritt 1: Ausschreibung

Bei der Ausschreibung muss der Manager einer Aufgabe allen Agenten im Zielgebiet die für die Verhandlungen wichtigen Informationen der Aufgabe mitteilen. Bei anderen Anwendungen bei denen es nicht um den Transport von Lieferungen geht, kann der Manager auf andere Weise eine Teilmenge von Agenten auswählen, die sich für diese Aufgabe eignen. Dies kann durch Erfahrungen aus vorherigen Verhandlungen oder durch Informationen, die der Manager über andere Agenten besitzt, erreicht werden. Im Zweifelsfall ist auch eine Broadcast

Ausschreibung an alle Agenten möglich. Insbesondere enthält die Ausschreibung einen Timestamp, der den Agenten Zeit gibt, um sich zu einer Teilnahme bei der Quantenauktion zu entscheiden. Das Protokoll für die Ausschreibungen läuft ähnlich wie im klassischen Fall (vgl. Abschnitt 4.1.1). Das Versenden der klassischen Informationen kann mit Hilfe der in Abschnitt 4.2.1 angesprochenen Quantenbroadcast-Kommunikation durchgeführt werden. In der klassischen Prozedur *ausschreiben()* wird somit die Schleife von Zeile 15 durch das folgende Protokoll ersetzt. Zusätzlich zur klassischen Information versendet der Manager den initialen Quantenzustand der Auktion.

Quantenbroadcast-Kommunikation (QBC)

Das verwendete Protokoll basiert auf Superdense Coding. Es werden Schlüssel zwischen den Agentenpaaren benötigt, die beim Betreten eines Agenten in das Netz erzeugt werden müssen. Existieren gemeinsame Schlüssel K_i zwischen dem Manager M und $A_i, i \in \{1, \ldots, r\}$, kann M eine Nachricht an die Agenten $A_1, \ldots, A_r$, wie in Abbildung 4.1 gezeigt, versenden.

Beispiel 32 *Ein Beispiel wird in Anhang A.5.1 gegeben.*

12: **Prozedur** *q-ausschreiben()* ▷ Bietet eine Mengen von Aufgaben an.
13: wähle $T = \{$ausgewählte Aufgabe$\}$, die in dem Operationsgebiet anderer Agenten liegt.
14: erstelle Ausschreibung t
15: $c_{max} = c'_{rem}(T)$ ▷ Berechnet durch lokalen Optimierer.
16: $M =$ Menge aller Agenten, bei denen das Aufgabenziel im Operationsgebiet liegt.
17: QBC(t, M)
18: $|\Psi_{init}\rangle = |\psi^1_{init}\rangle \otimes \cdots \otimes |\psi^m_{init}\rangle = |0, \ldots, 0\rangle$ ▷ Erstelle initialen Quantenzustand.
19: **für** jeden Agenten a_i aus M **tue** ▷ Verteile $|\Psi_{init}\rangle$
20: sende-q($|\psi^i_{init}\rangle$,a_i)
21: **ende für**
22: **ende Prozedur**

Schritt 2: Quantenauktion

Alle Agenten die eine Ausschreibung erhalten, prüfen mit ihrem lokalen Optimierer, ob sie an der ausgeschriebenen Aufgabe interessiert sind. Falls ja erstellen sie einen Bietoperator für die Quantenauktion und wenden diesen auf den erhaltenen Quantenzustand an. Das Ergebnis senden sie an den Manager zurück. Bei Ablauf des Timestamp beginnt der Manager mit der ersten Iteration der Quantenauktion. Die Menge der an der Auktion teilnehmenden Agenten ergibt sich aus den empfangenen Quantenzuständen der interessierten Agenten. Falls keine Quantenzustände empfangen wurden wartet der Agent einen zweiten Timestamp. Falls danach immer noch keine Quantenzustände empfangen wurden versendet er Ablehnungen an alle Agenten, die die Ausschreibung erhalten haben. Zu spät eintreffende Quantenzustände zur Teilnahme an einer bereits laufenden Auktion werden ignoriert. Der Manager erstellt die Diagonalmatrizen $D(f)$ und $P(f)$ anhand der Anzahl der Teilnehmer. Der Quantenzustand zu diesem Zeitpunkt ist $U \cdot |\Psi_{init}\rangle$. Danach befindet sich die Auktion in der Iterationsschleife, die aus den Prozeduren *q-bieten()* und *q-verteilen()* besteht. Es entsteht ein Wechselspiel, das aus dem Anwenden der Bietoperatoren auf Seiten der Bieter und dem Anwenden der

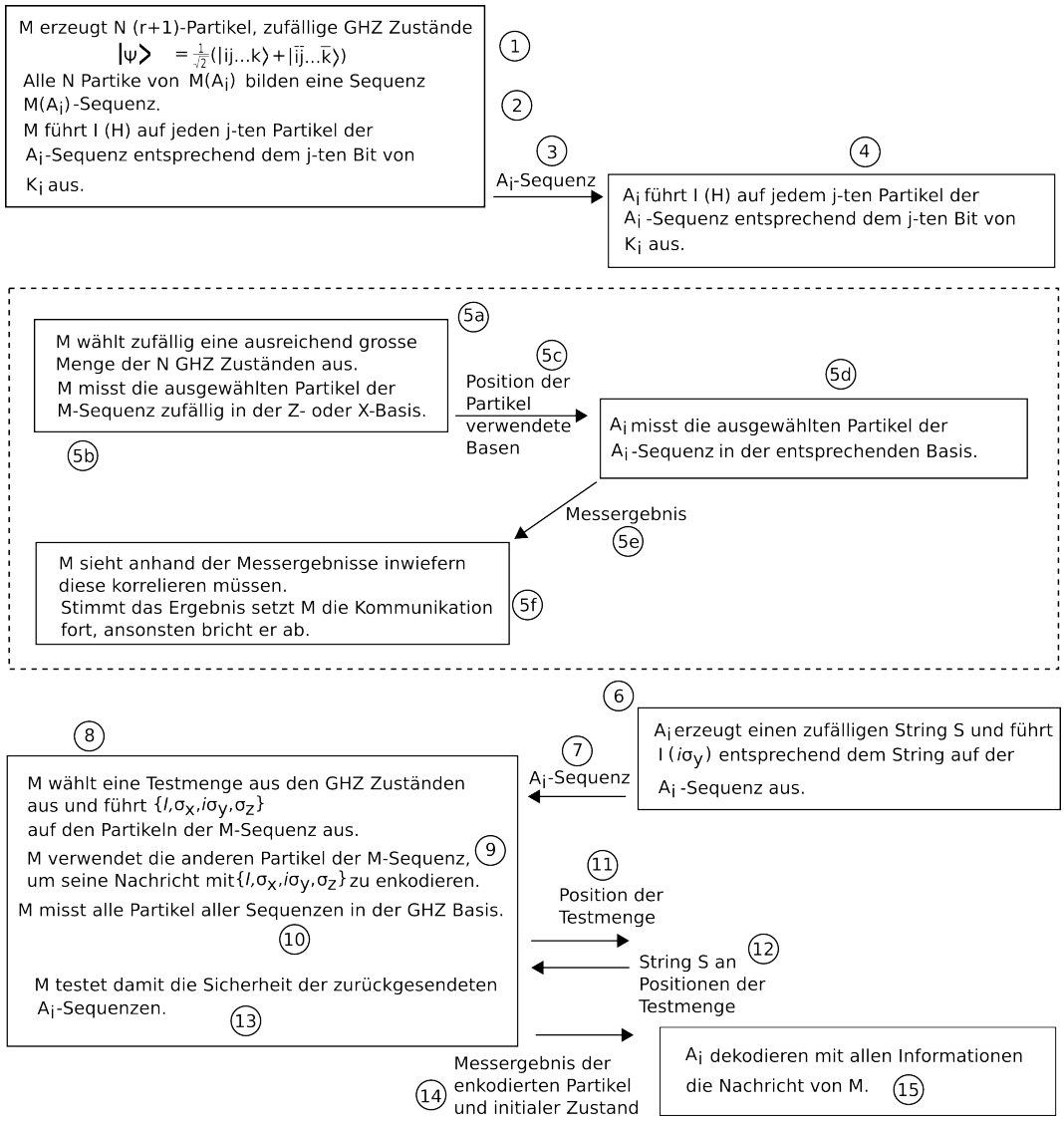


Abbildung 4.1: Quantenbroadcast-Kommunikation (QBC): Darstellung der verteilten Ausführungsschritte vom Manager (links) und den Agenten (rechts).

Diagonalmatrizen des Managers besteht. In einer Verhandlungsschleife wird so jede laufende Auktion um eine Iteration voran gebracht. Die Quantenauktion selbst läuft wie in Abschnitt 2.2.5 ab. Da sich in TRACONET die Bieter jedoch gegenseitig unterbieten müssen, wird die Quantenauktion hier wie folgt interpretiert:
Der Manager legt in der Ausschreibung wie im klassischen Fall die maximalen Kosten fest, die er bereit ist, für das Lösen der Aufgabe zu zahlen. Die Bieter berechnen die Kosten, für die sie bereit sind, die Aufgabe zu übernehmen. Der Bietoperator ergibt sich aus der Differenz zwischen den beiden Werten. Auf diese Weise gewinnt der Bieter mit der höchsten Differenz am wahrscheinlichsten, dh. der Bieter mit den niedrigsten Kosten für die er die Aufgabe übernimmt. Die Anzahl der Iterationen, die für eine Quantenauktion nötig ist, kann durch frühere Erfahrungswerte bei ähnlichen Auktionen oder durch lokale Berechnung einer Probeauktion ermittelt werden. Die Funktionen *sende-q* und *empfange-q* bezeichnen hier das Senden und Empfangen eines Quantenzustandes durch einen Quantenkanal.

23: **Prozedur** *q-bieten()* ▷ Bieter-seitige Auktionsiteration (Bieter$_i$)
24: **für** laufende Quantenauktion mit Ausschreibung t **tue**
25: empfange-q($|\psi_s\rangle$, Auktionär)
26: **falls** t eine neue Ausschreibung ist **dann** ▷ dh. $|\psi_s\rangle = |\psi_{init}\rangle$
27: **falls** $f'(T_{cur} \cup T_t \cup T^t_{pos}) < \infty$ **dann** ▷ Machbarkeits-Check; Berechnet vom lokalen Optimierer.
28: $c^t_{bid} = c'_{add}(T_t)$ ▷ Berechnet vom lokalen Optimierer.
29: **falls** $c^t_{bid} < c^t_{max}$ **dann**
30: $teilnahme(t) = true$
31: erstelle Quantenoperator U_i ▷ Operator der das Gebot für die Quantenauktion beinhaltet.
32: $|\psi_0\rangle = U_i \cdot |\psi_{init}\rangle$ ▷ Wende Operator U_i aus *q-anmelden()* an.
33: sende-q($|\psi_0\rangle$, Auktionär)
34: **ende falls**
35: **ende falls**
36: **sonst**
37: **falls** $teilnahme(t)$ **dann**
38: $|\psi_{s^+}\rangle = U_i^\dagger \cdot |\psi_{s'}\rangle$ ▷ Wende Operator $U_i^\dagger$ an.
39: sende-q($|\psi_{s^+}\rangle$, Auktionär)
40: empfange-q($|\psi_{s^*}\rangle$, Auktionär)
41: $|\psi_{s+1}\rangle = U_i \cdot |\psi_{s^*}\rangle$ ▷ Wende Operator U_i an.
42: sende-q($|\psi_{s+1}\rangle$, Auktionär)
43: **ende falls**
44: **ende falls**
45: **ende für**
46: **ende Prozedur**

47: **Prozedur** *q-verteilen()* ▷ Manager-seitige Auktionsiteration
48: **für** jede, als Manager beteiligte, offene Ausschreibung t **tue**
49: **falls** t eine neue Ausschreibung ist **dann**
50: empfange-q($|\psi_0\rangle$, a_i)
51: **falls** Quantenzustände nach *Timestamp1* empfangen wurden **dann**
52: $E = \{$Agenten die einen Quantenzustand $|\psi_0^i\rangle$ zurückgesendet haben$\}$
53: $|\Psi_0\rangle = |\psi_0^1\rangle \otimes \cdots \otimes |\psi_0^e\rangle$
54: **sonst**
55: **falls** erster Timestamp für t ausgeführt wurde **dann**
56: Warte(*Timestamp2*)
57: q-verteilen()
58: **sonst**
59: QBC(Ablehnung, alle Agenten a die t gesendet bekamen)
60: **ende falls**
61: **ende falls**
62: **ende falls**
63: $|\Psi_{s'}\rangle = P(f) \cdot |\Psi_s\rangle$ ▷ Beginn einer normalen Iteration.
64: **für** jeden Agenten a_i aus M **tue** ▷ $U^\dagger$ wird auf $|\Psi_{s'}\rangle$ angewendet
65: sende-q($|\psi_{s'}^i\rangle$,a_i)
66: empfange-q($|\psi_{s^+}\rangle$, a_i)
67: $|\Psi_{s^+}\rangle = |\psi_{s^+}^1\rangle \otimes \cdots \otimes |\psi_{s^+}^m\rangle$
68: **ende für**
69: $|\Psi_{s^*}\rangle = D(f) \cdot |\Psi_{s^+}\rangle$
70: **für** jeden Agenten a_i aus M **tue** ▷ U wird auf $|\Psi_{s^*}\rangle$ angewendet
71: sende-q($|\psi_{s^*}^i\rangle$,a_i)
72: empfange-q($|\psi_{s+1}\rangle$, a_i)
73: $|\Psi_{s+1}\rangle = |\psi_{s+1}^1\rangle \otimes \cdots \otimes |\psi_{s+1}^m\rangle$
74: **ende für**
75: **ende für**
76: **ende Prozedur**

Auswertung der Auktion

Alle abgelaufenen Auktionen (Auktionen, bei denen die letzte Iteration durchgeführt wurde) werden vom Manager der Aufgabe ausgewertet. Dieser versendet Ablehnungsnachrichten an jeden Agenten (außer dem Gewinner), an den er eine Ausschreibung gesendet hat. Dies kann wieder mit der Quantenbroadcast-Kommunikation oder auf klassischem Weg erreicht werden. Der Gewinner bekommt seine Nachricht, bzw. alle zur Ausführung der Aufgabe nötigen Informationen mit Hilfe von Superdense Coding (SDC, vgl. Abschnitt 2.2.4) zugesandt. Auf der Seite des Managers wird an dieser Stelle (*Zuschlag_geben()*) die Aufgabe aus der lokalen Lösung gelöscht und auf der Seite des Gewinners (*q-Zuschlag_bekommen()*) wird diese hinzugefügt. Die Prozedur *q-Zuschlag_bekommen()* ist analog zum klassischen Fall und aus diesem Grund hier nicht mehr aufgeführt.

77: **Prozedur** *q-Zuschlag_geben()* ▷ Misst und evaluiert das Ergebnis der Quantenauktion.
78: **für** jede eigene abgelaufene Quantenauktion **tue**
79: $|\Psi_{S+1}\rangle$ = Endzustand der Auktion
80: Messe $|\Psi_{S+1}\rangle$ und interpretiere das Ergebnis
81: SDC(Zuschlag, Auktionsgewinner)
82: E = Menge der Agenten, an die die Ausschreibung t gesendet wurde
83: QBC(Ablehnung,E {Auktionsgewinner})
84: $T_{cur} = T_{cur} \setminus T_t$
85: **ende für**
86: **ende Prozedur**

Verlassen oder Betreten des Systems

Möchte ein Agent das Netzwerk verlassen, muss er wie im klassischen Fall darauf achten, dass kein anderer Agent auf seine Nachrichten wartet. Aus diesem Grund führt er alle laufenden Auktionen, an denen er als Bieter teilnimmt, bis zum Ende (dh. bis er eine Ablehnung oder einen Zuschlag geschickt bekommt) aus. Auktionen, die er selbst leitet, kann er jederzeit durch das Versenden von Ablehnungen an alle teilnehmenden Agenten beenden.
Betritt ein Agent das Netzwerk benötigt er alle, für die Quantenalgorithmen nötigen, Voraussetzungen. Er benötigt Zugang zu dem Pool, der verschränkte Paare unter den Agenten verteilt, für die Broadcast-Kommunikation werden Paarschlüssel benötigt und für Superdense Coding initiales Wissen über die verwendeten Operatoren. Ansonsten kann der Agent direkt an den Verhandlungen teilnehmen.

Ergebnisübergabe

Im klassischen TRACONET wird die Übergabe des Ergebnisses einer Aufgabe zurück an den Manager wegen dem Anwendungsgebiet nicht betrachtet. Beim Ausführen von Lieferungen ist eine solche Ergebnisübergabe nicht nötig. An dieser Stelle wird dennoch für das TRACONET-Q betrachtet, wie eine Ergebnisübergabe, vor allem wenn die Aufgabe erneut ausgeschrieben bzw. in Unteraufgaben aufgeteilt und ausgeschrieben wurde, aussieht. An dieser Stelle lassen sich verschränkte Zustände gut einsetzen, um den gesamten Rückweg nicht noch einmal durchlaufen zu müssen. Übergibt jeder Manager in *Zuschlag_geben()* ausreichend verschränkte Paare an den Gewinner der Auktion, entsteht ein Baum aus Verschränkungen wie in Abb. 4.2 dargestellt, der von der Vergabe einer Aufgabenmenge als Wurzel bis zu den Agenten für das Ausführen kleiner Teilaufgaben an den Blättern reicht. Durch Entanglement Swapping (vgl. Abschnitt 2.2.1) lassen sich somit Verschränkungen zwischen jedem Agenten der an der Lösung der Aufgabenmenge arbeitet aufbauen. Ein solches Netzwerk kann zum Beispiel für die Übertragung der Ergebnisse der Aufgaben zurück zur Wurzel verwendet werden.

4.2.3 Komplexität

Alle Untersuchungspunkte haben gemein, dass die Komplexität nicht wesentlich reduziert wird. Die Quantenbroadcast-Kommunikation (vgl. Abb. 4.1) hat folgende Kommunikationskomplexität für n Empfänger:

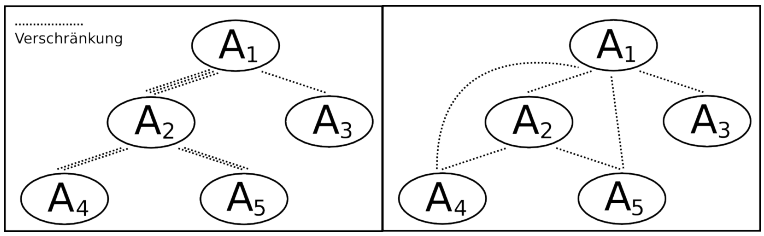


Abbildung 4.2: Verschränkung bei Aufgabenverteilung (links), Verschränkung zur Ergebnisübergabe an Wurzel nach Entanglement Swapping (rechts)

$$\begin{aligned}
\text{Kommunikation (Sender)} &= \text{Schritt 3} + \text{Schritt 5c} + \text{Schritt 11} + \text{Schritt 14} \\
&= O(4 \cdot n) \\
&= O(n) \qquad (4.3) \\
\text{Kommunikation (Empfänger)} &= \text{Schritt 5e} + \text{Schritt 7} + \text{Schritt 12} \\
&= O(1) \qquad (4.4)
\end{aligned}$$

Somit ergibt sich die Komplexität für eine Verhandlungsrunde durch:

$$\begin{aligned}
\text{Kommunikation (Manager)} &= \text{q-ausschreiben()} + \text{q-verteilen()} \\
&\quad + \text{q-Zuschlag_geben()} \\
&= QBC(n) + n + 2 \cdot n + QBC(n-1) + 1 \\
&= O(n) \qquad (4.5) \\
\text{Kommunikation (Auftragnehmer)} &= \text{q-bieten()} + \text{q-Zuschlag_bekommen()} \\
&= 3 + 0 + C \\
&= O(1) \qquad (4.6)
\end{aligned}$$

Die Konstante C ist dabei die Kommunikation, die bei einem Quantenbroadcast auf der Empfängerseite anfällt. Zu beachten ist, dass der Abschluss einer Auktion über mehrere Runden, bestehend aus *q-verteilen()* und *q-bieten()*, andauert. Die Wahrscheinlichkeit, dass der Bieter mit dem niedrigsten Angebot den Zuschlag erhält, geht gegen 1, wenn die Anzahl der Iterationen der verteilten adiabetischen Suche gegen unendlich geht. Um eine hinreichend gute Wahrscheinlichkeit zu erreichen, sind hier Erfahrungswerte aus vorherigen, ähnlichen Suchen nötig (vgl. [HHC07, 2]).

4.2.4 Beispiel

Auch hier wird ein kleines Beispiel zur Veranschaulichung der vorgestellten Algorithmen gegeben. Dieses beschränkt sich auf die einfache Abgabe einiger Aufgaben an andere Agenten, die zur Lösung der Aufgabe einen geringeren Kostenaufwand haben. Da die Berechnung der Kosten in dieser Arbeit nicht weiter betrachtet wird, ist hier nur ein einfaches 1-dimensionales Szenario gegeben, das die Positionen der Agenten und Aufgaben verwaltet und die Kosten

anhand der Abstände misst. Abbildung 4.3 zeigt die Ausgangssituation des 3-Agenten Systems. Abbildung 4.4 veranschaulicht die Verhandlung einer Ausschreibung, durch eine direkte Nebeneinanderstellung im klassischen und dem Quantenprotokoll.

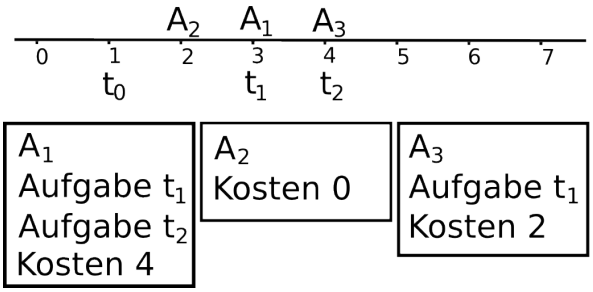


Abbildung 4.3: Ausgangssituation des Beispiels.

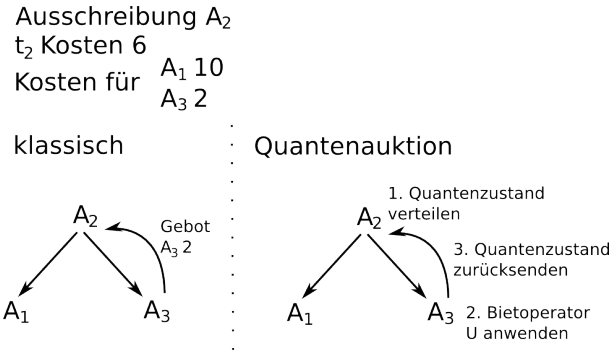


Abbildung 4.4: Verhandlung einer Ausschreibung von Agent A_2

4.3 Evaluation

Eine richtige Evaluation ist wegen dem Fehlen ausreichender Quantencomputern und der exponentiellen Verlangsamung beim Simulieren auf klassischen Computern nicht sinnvoll (vgl. 3.3). Deshalb wird an dieser Stelle nur ein theoretischer Vergleich des klassischen und des Quantenprotokolls geführt.
Wegen der gleichen Komplexität einer Verhandlungsrunde scheint keines der beiden Protokolle signifikant schneller zu sein. Es ist jedoch zu beachten, dass hier die Vergabe einer Aufgabe, wegen der Iterationen bei der Quantenauktion, mehr Kontrollschleifen in Anspruch nimmt, als im klassischen Fall. Somit zeichnen sich bei TRACONET-Q Nachteile im Bereich der Komplexität ab. Da die Anzahl der Iterationen bei einer adiabetischen Suche analytisch nicht fassbar ist, lässt sich auch nur schwer sagen, wie sehr dieser Unterschied ins Gewicht fällt.
Die Implementierung von TRACONET-Q zeigt die Schwierigkeit eines Mittelwegs aus der Zuverlässigkeit des Messergebnisses und einer vertretbaren Menge an Iterationen. Bei zu wenigen Iterationen kann es zu einem falschen Ergebnis beim Messen kommen, aber zu viele Iterationen führen auch zu unnötig mehr Kommunikationsaufwand (vgl. Kapitel 5).

4.4 Diskussion

Wie in Abschnitt 4.3 gesehen, weist TRACONET-Q keine Verbesserung im Bereich der Kommunikation auf. Bei den Quantenauktionen muss die Anzahl der Iterationen groß genug sein, um eine ausreichend hohe Wahrscheinlichkeit auf das Richtige Endergebnis, zu erhalten. Dazu muss der Quantenzustand mehrfach verteilt werden. Hinzu kommt, dass für die Multicastübertragung das Erstellen der Schlüssel bei einer großen Anzahl von Agenten aufwendig wird. Da für jedes Agentenpaar ein Schlüssel benötigt wird, muss ein Agent beim Betreten des Netzwerkes mit allen anderen einen solchen Schlüssel austauschen.
Bei den meisten der hier verwendeten Änderungen und Ansätzen fällt die Abhängigkeit der Protokolle von verschränkten Zuständen auf. Deshalb stellt sich an dieser Stelle die Frage:

Welche Folgen hat die Annahme eines Typ-IIb QCMAS?

Bisher sind wir von einem Typ-IIb QCMAS (vgl. Abschnitt 2.4.2) ausgegangen. In der Praxis ist es jedoch unwahrscheinlich, dass Verschränkung immer in ausreichender Menge zur Verfügung steht. Geht man davon aus, dass jedes verschränkte Paar von einer Seite zuerst erzeugt und verteilt werden muss, werden die meisten Vorteile, die durch die Ressource entstehen wieder zunichte gemacht. Beim Ansatz mit Superdense Coding, muss für eine klassische n-bit Nachricht zum Beispiel $\frac{n}{2}$ Quantenbits versendet und $\frac{n}{2}$ verschränkte Paare erzeugt und verteilt werden, was noch einmal das Versenden von $\frac{n}{2}$ Qubits erfordert. Dadurch könnte eine n-bit Nachricht, durch das Übertragen von n Qubits, versendet werden. Was auch durch die Triviallösung, das Versenden von Qubits, die sich nicht in Superposition befinden, erreicht werden könnte. Deshalb ist die Annahme über ein Typ-IIb QCMAS mit Vorsicht zu genießen.

Lässt sich die Koordination weiter verbessern?

Bei der Koordination gibt es gleich ein Reihe von Hindernissen. Für einige Probleme gibt es eine eine direkte Quantenlösung (z.B. Führerwahl und verteilter Konsens vgl. [HP06, 4, 5]). Diese baut jedoch auf Verschränkung auf und verliert in einem Typ-IIa QCMAS an Effizienz. So kann ein verteilter Konsens zwar getroffen werden, anstelle n Qubits könnten jedoch auch einfach n klassische Bits verteilt werden, die den allgemeinen Konsens beinhalten.
Im Klassischen wird Koordination entweder durch initiales Wissen oder durch Kommunikation von Informationen erreicht. Durch die nahe Anlehnung der hier durchgeführten Quantenumsetzungen an die klassischen Protokolle lässt sich die Koordination der Agenten nicht vom Rest der Kommunikation trennen. So löst das Empfangen bestimmter Nachrichten im TRACONET Protokoll durch den Listener die entsprechenden Aktionen aus, die nötig sind, um die empfangenen Nachrichten zu verarbeiten. Dadurch werden die Agenten durch die Kommunikation auch koordiniert.
Die Tatsache, dass quantenmechanische Messungen probabilistisch sind, im Zusammenhang mit dem No-Cloning-Theorem [WZ82] führt dazu, dass durch Verschränkung allein keine Informationsübertragung möglich ist. Erst im Zusammenhang mit einem klassischen Kanal oder einem übertragenem Quantenzustand (2.2.4) lassen sich Informationen übertragen. Aus diesem Grund konnte hier auch keine bessere Verwendung für das durch die Verhandlungen aufgebaute Netzwerk aus Verschränkungen gefunden werden, als die Quantenkommunikation.
Selbst wenn durch einen anderen Ansatz die Koordination der Agenten allein durch Ver-

schränkung in konstanter Zeit durchgeführt werden würde, müssten trotzdem die Informationen über die Verhandlungen, Kosten der Aufgaben und die Aufgaben und Ergebnisse selbst übertragen werden. Was aber hier alleine schon zum Koordinieren der Agenten ausreichend ist.
Für das Übertragen von Informationen, das bei den hier betrachteten Protokollen essenziell ist, führt Holevo [Hol73], [NC00, 12.1.1] den Beweis, dass in n Quantenbits nur bis zu n klassische Bits an Informationen extrahiert werden können. Durch mehrere Zustände in Superposition lassen sich zwar mehr Informationen übertragen, durch eine Messung lässt sich jedoch immer nur einer der Zustände auslesen.
Durch diese Ergebnisse kann gesagt werden, dass, um eine weitere Verbesserung der Koordination zu erhalten, eine Quantenumsetzung sich nicht zu nah an dem klassischen Protokoll orientieren sollte. Der bekannte Quantensuchalgorithmus von Grover hat wenig von dem Algorithmus gemein, den man im klassischen Fall verwenden würde, um eine solche Suche durchzuführen. Darin liegt jedoch auch die Schwierigkeit einen solchen Algorithmus für ein bestimmtes Problem zu finden.

Wo liegen dennoch die Vorteile in der Quantenumsetzung?

Der Benefit von Quantenalgorithmen im Bereich der Kommunikation und Koordination liegt an anderer Stelle als an geringerer Komplexität. Die Quantenauktionen bewahren die Vertraulichkeit der Gebote, so kennt der Manager nur das Gebot des Gewinners einer Auktion, nicht aber das aller anderen Bieter. Durch die Quantenbroadcast-Übertragungen wird die Sicherheit der damit übertragenen Informationen gewährleistet. Hier lässt sich sagen: Was Quantenkommunikation im Bereich des Multicast wegen dem No-Cloning-Theorem problematisch macht, ist genau das, was sich im Bereich der Sicherheit als überlegen herausstellt. Verschränkungen bieten, selbst wenn sie verteilt werden müssen immer noch den Vorteil, dass die Verteilung (theoretisch) zu einem früheren Zeitpunkt stattfinden kann. Dadurch wäre es zum Beispiel möglich, ein rein auf Verschränkung basiertes Notfallprotokoll auch dann auszuführen, wenn andere Kommunikation nicht möglich ist. Auch lässt sich somit ein Überlasten des Netzwerkes vermeiden, indem praktisch die Hälfte des Kommunikationsaufwandes bereits im Vorfeld durchgeführt wird. Auch kann die Zufälligkeit eines gemessenen Ergebnisses ein Vorteil sein, so ist es im klassischen ein nicht-triviales Problem eine richtige Zufälligkeit zu erzeugen.
Zusammenfassend läßt sich sagen:

- Es wurde keine Verbesserung im Bereich der Berechnung oder Kommunikation erreicht.
- In der Praxis muss von einem Typ-IIa System ausgegangen werden, wodurch die meisten der Quantenprimitive ihren Komplexitätsvorteil verlieren.
- TRACONET wird durch das Versenden der notwendigen Informationen koordiniert, die auch im TRACONET-Q übertragen werden müssen.
  - Deshalb muss, um wirkliche Verbesserungen zu erreichen, ein Quantenalgorithmus gefunden werden, der nicht an das klassische Protokoll angelehnt ist.
- Das No-Cloning-Theorem macht Quantenkommunikation zwar schwieriger, bringt im Bereich der Sicherheit jedoch auch Vorteile.

- Auch wenn Verschränkung durch die Verteilung nicht direkt Kommunikation spart, kann dies doch auf Vorrat geschehen.

Kapitel 5

Implementierung

In diesem Kapitel wird die Simulation der oben beschriebenen Quantenalgorithmen betrachtet. Die Programmierung von KCA erfolgte in C++. Die in KCA-Q betrachteten Algorithmen wurden mit *libquantum* (vgl. Abschnitt 2.3.2) umgesetzt, das dazu in C++ eingebunden wurde.
Alle Implementierungen sind auf der CD-Rom zu dieser Arbeit zu finden. Um die Protokolle zu implementieren wurden zuerst einige Grundlagen (*qbasics.cpp*) umgesetzt, die der Simulator nicht besitzt. Danach wurden die klassischen Komponenten von KCA und TRACONET geschrieben und mit den Quantenalgorithmen erweitert. Für KCA-Q sind das die logischen (*qlogic.cpp*) und arithmetischen (*qadder.cpp*) Quantenschaltkreise aus Abschnitt 2.2.3. Außerdem die in Abschnitt 3.2.2 beschriebenen Algorithmen zur Surplusberechnung (*qsurplus.cpp*), die Suchalgorithmen zum Finden des Maximums (*qmaxfind.cpp*) und die Quantenführerwahl (*qleader.cpp*). Für TRACONET-Q wurden die Quantenauktionen (*qauction.cpp*) implementiert. Im folgenden werden die wichtigsten umgesetzten Funktionen spezifiziert und beschrieben.

Umsetzung der Grundlagen

```
extern void print_qreg(quantum_reg *reg);
```

Diese Funktion gibt den aktuellen Zustand des Quantenregister `reg` in Binärdarstellung über den Standartausgabestrom aus.

```
extern void remove_controlbit(int pos, quantum_reg *reg);
```

Das Quantenregister `reg` wird durch Messung des Qubits an Position `pos` gekürzt. Um vorhandene Superpositionen nicht zu zerstören, muss das Qubit vorher gereinigt werden.

```
extern void q_unbounded_toffoli(int controlling, int target,
quantum_reg *reg, int controls[]);
```

Es wird die Operation $0nxp$-CNOT(`controls` $\rightarrow$ `target`) auf `reg` ausgeführt. Wobei `controlling` die Anzahl x der Kontrollqubits und die Länge von `controls` angibt. Das Array `controls[]`

beinhaltet die Positionen der Kontrollbits.

```
extern void q_cust_cnot(int ncontrolling, int pcontrolling,
int target, quantum_reg *reg, int controls[]);
```

Diese Funktion führt die Operation *ynxp*-CNOT(`controls` → `target`) auf `reg` aus. Wobei `ncontrolling` die Anzahl y der negativen und `pcontrolling` die Anzahl x der positiven Kontrollbits angibt. Die ersten `ncontrolling` Einträge von `controls[]` sind die Positionen der negativen Kontrollbits. Die folgenden `pcontrolling` Einträge die der positiven Kontrollbits.

Umsetzung der Logik-Schaltkreise

```
extern void q_nbit_controlled_or(int controlbit, int number,
int target, quantum_reg *reg, int bits[]);
```

Diese Funktion setzt den n-bit Oder Schaltkreis aus Abb. 2.2 um. Der einzige Unterschied ist, dass der Schaltkreis durch das Qubit an Position `controlbit` kontrolliert wird. Die Variable `number` gibt die Anzahl der Oder-Bits an, das Array `bits[]` die Positionen. Das Ergebnis wird auf Qubit `target` geschrieben.

```
extern void q_nbit_controlled_eq(int controlbit, int number,
int starta, int startb, int target, quantum_reg *reg);
```

Hier wird der Gleichheitstest von Abb. 2.3 auf `reg` ausgeführt. Auch dieser wird durch Qubit `controlbit` kontrolliert und das Ergebnis auf `target` geschrieben. Die Variablen `starta` und `startb` sind die Positionen der ersten, der jeweils insgesamt `number` Quantenbits, aus denen die zu vergleichenden Binärstrings bestehen.

```
extern void q_nbit_controlled_less(int controlbit, int number,
int starta, int startb, int target, quantum_reg *reg);
```

Diese Funktion führt den Kleiner-Test (vgl. Abb. 2.4) auf `reg` durch. Die Variablen haben die gleiche Funktion wie bei `q_nbit_controlled_eq`.

```
extern void q_nbit_compl_controlled_less(int controlbit, int number,
int starta, int startb, int target, quantum_reg *reg);
```

Diese Funktion führt einen Kleiner-Test durch. Die Werte der Quantenregisterabschnitte, beginnend von `starta` und `startb` an, werden nicht als normale Binärzahlen, sondern als Two's Complement Zahlen interpretiert werden.

Umsetzung der Addierer und Subtrahierer

```
extern void q_3bit_adder(int cin, int a, int b, int t0, int t1,
quantum_reg *reg);
extern void q_3bit_controlled_adder(int cbit, int cin, int a, int b,
int t0, int t1, quantum_reg *reg);
```

Diese Funktionen setzen genau den 3-Bit Addierer aus Abb. 2.5 um. Im zweiten Fall wird die Operation durch `cbit` kontrolliert.

```
extern void q_2bit_adder(int a, int b, int t0, int t1, quantum_reg *reg);
extern void q_2bit_controlled_adder(int cbit, int a, int b, int t0,
int t1, quantum_reg *reg);
```

Analog zum 3-bit Addierer wurde hier der 2-bit Addierer aus Abb. 2.6 implementiert.

```
extern void q_nbit_adder(int bits, int starta, int startb,
int startt, quantum_reg *reg);
extern void q_nbit_compl_adder(int bits, int starta, int startb,
int startt, quantum_reg *reg);
extern void q_nbit_controlled_adder(int cbit, int bits, int starta,
int startb, int startt, quantum_reg *reg);
```

Für die n-bit Adder (vgl. Alg. 1) gibt `starta` und `startb` die Startposition und `bits` die Länge der Quantenregisterabschnitte von `reg` für die beiden Binärwerte an, die addiert werden sollen. Das Ergebnis wird beginnend von `startt` auf `reg` geschrieben und hat eine Länge von `bits`+1. In der zweiten Version werden die Binärwerte als Two's Complement Zahlen interpretiert. In der dritten Version ist die Berechnung von einem Kontrollbit `cbit` abhängig.

```
extern void q_nbit_subtractor(int bits, int starta, int startb,
int startt, quantum_reg *reg);
extern void q_nbit_controlled_subtractor(int cbit, int bits, int starta,
int startb, int startt, quantum_reg *reg);
```

Die Funktionen für die Subtrahierer (vgl. Alg. 2) sind wie die der Addierer aufgebaut. In beiden Versionen werden alle Werte als Two's Complement Zahlen interpretiert.

Umsetzung der Surplusberechnung

```
extern void q_compute_v(vector<KAgent> ag, int tpos,  int cpos, quantum_reg *reg);
```

Diese Funktion berechnet die charakteristische Funktion. Im Grunde wird hier Algorithmus 15 implementiert. Die Variable `cpos` gibt dabei die Position der Koalitionskodierung auf dem

Quantenregister `reg` an. Der Beginn des Quantenregisterabschnittes, auf den das Ergebnis geschrieben werden soll, wird durch `tpos` angegeben. Für die Berechnung ist eine klassische Liste von allen Agenten nötig, die durch `ag` geliefert wird. Bei der Berechnung der lw-Werte kam es bei der Implementierung zu ersten Problemen mit dem Quantensimulator (siehe unten *Diskussion*). Aus diesem Grund wurde dieser Schritt nicht wie in dem Quantenalgorithmus angegeben umgesetzt, sondern durch die Annahme ersetzt, dass die lw-Werte bereits als Quanteninformation vorliegen.

```
extern void q_compute_u(vector<KAgent> ags, PayoffD pd, int tpos,
int cpos, quantum_reg *reg);
```

Hier werden die Summen der Auszahlungen der Agenten für eine Koalition, wie in Algorithmus 16 beschrieben, berechnet. Die Variable `pd` liefert die klassischen Auszahlungswerte der Agenten. Die Variablenbelegung ist ansonsten wie bei `q_compute_v`.

```
extern void q_compute_excess(vector<KAgent> ags, PayoffD pd, int tpos,
int cpos, quantum_reg *reg);
```

Diese Funktion verwendet `q_compute_v` und `q_compute_u` und berechnet zu den Koalitionen von `cpos` die jeweiligen Excesswerte, indem sie die beiden Quantenregisterabschnitte mit den Werten $v(\mathbf{C})$ und $u(\mathbf{C})$ voneinander subtrahiert.

Umsetzung der Suchalgorithmen

```
extern void grover_geq(KCA_Game game, int k, quantum_reg *reg);
```

In dieser Funktion wird der Suchalgorithmus implementiert. Das Quantenregister zu diesem Zeitpunkt besteht nur aus der Superposition von Koalitionen, wie durch Algorithmus 14 beschrieben. Intern wird eine Orakelfunktion verwendet, die den Wert des aktuellen `k` der Maximumsuche kodiert und den Excesswert der Koalitionen berechnet. Danach wird ein Kleiner-Test durchgeführt und ein angefügtes Orakelbit gesetzt. Durch mehrmaliges Ausführen dieser Funktion wird so, wie in Algorithmus 4 beschrieben, das Maximum gefunden.

Umsetzung der Quantenführerwahl

```
extern quantum_reg q_get_W_state(int agnum);
```

Die Funktion erzeugt ein Quantenregister mit einem W-Zustand. Die Größe ist entsprechend der Agentenanzahl `agnum`. Bei der Umsetzung wurde die Implementierung des Trugenberger Speichers [Tru01] von [Sch07, A.5] verwendet. Dazu wurden auch die in [Sch07, A.5] beschriebenen Veränderungen an *libquantum* durchgeführt.
Die Umsetzung der Führerwahl ist in *coalition.cpp* in der Funktion

```
void q_decideNewCoalitionLeader();
```

implementiert. Sie ist Teil einer Klasse *Coalition*, die eine Koalition repräsentiert.

Umsetzung der Quantenauktion

Die Klasse *QAuction* repräsentiert eine Quantenauktion. Sie verwaltet den Quantenzustand, auf den die Bieter den Operator U und der Manager die Iterationsberechnungen anwendet.

```
quantum_matrix compute_U(int bid);
quantum_matrix compute_Df(int d);
quantum_matrix compute_Pf(int d);
```

Mit diesen Funktionen werden die Matrizen für die aktuelle Iterationsrunde neu berechnet. Die Variable `bid` ist dabei der klassische Wert eines Gebots für einen Agenten. Alle weiteren benötigten Variablen werden von der Klasse *QAuction* bereitgestellt.

```
void apply_U(bool transpose);
```

Die Funktion wendet den Bietoperator aller bietenden Agenten an. Ist `transpose` true wird der komplex konjugierte, transponierte Operator $U^\dagger$ angewendet.

```
void nextRound(int d);
```

Hier wird eine komplette Iterationsrunde durchgeführt. Die Bietenden Agenten wenden den Operator U an und der Manager die Matrizen Df und Pf (vergleiche Abschnitt 2.2.5 Quantenauktionen).

```
void computeResult();
TAgent getWinner();
int getWinBid();
```

Diese Funktionen werten am Ende der letzten Iterationsrunde das Messergebnis des Quantenregisters aus und geben das Ergebnis zurück.

Alle Quantenalgorithmen, sowohl für KCA-Q als auch für TRACONET-Q wurden in einfach klassische KCA bzw. TRACONET Protokolle eingebunden, damit sie sowohl klassisch, als auch als Quantenversion ausgeführt werden können.

Diskussion

Ein Quantensystem kann, wie in Abschnitt 2.3.2 erwähnt, nicht durch ein klassisches System effizient simuliert werden. Die Implementierung der Quantenalgorithmen für das KCA Protokoll wurde vor allem durch die obere Grenze der Quantenbits behindert, die *libquantum* simulieren kann. In Abschnitt 3.2.3 wurde gezeigt, dass diese Anzahl gerade bei der Berechnung der *lw*-Werte quadratisch zur Anzahl der Agenten ist. Erste Annahmen über eine Simulation für eine geringe Anzahl von Agenten stellten sich dennoch wegen der Größe der Konstanten als nicht durchführbar heraus, da bei der Berechnung der *lw*-Werte die Listen

der Aufgaben von einem Agenten kodiert werden müssen. Selbst bei starker Beschränkung (limitierte Anzahl an Aufgaben, nur ganzzahlige Werte, wenig Agenten) wurde die Schwelle von 64 Quantenbits schnell überschritten.
Gespräche mit den Entwicklern von *libquantum* über eine Erweiterung der Anzahl der Quantenbits ergaben, dass eine solche Erweiterung zu aufwendig ist, um sie im Rahmen dieser Arbeit selbst vorzunehmen. Auch Tests mit anderen Quantensimulatoren versprachen keine Besserung. Aus diesem Grund muss die Berechnung der lw-Werte aus dem Klassischen erfolgen.
Bei der Umsetzung der Quantenauktion für größere Matrizen (ab 64x64 Matrizen) wirkten sich die Rundungsfehler, durch die Iterationen der Verhandlungen und die Approximationen des Matrixexponentials, so stark aus, dass das Messergebnis davon maßgeblich beeinflusst wurde.
Wegen dieser Schwierigkeiten handelt es sich bei der Implementierung um die Simulation der in den Abschnitten 3.2.4 und 4.2.4 beschriebenen Beispiele. Das Protokoll kann nicht mit jeder beliebigen Anzahl an Agenten und Aufgaben ausgeführt werden.

Kapitel 6

Ausblick und Zusammenfassung

In diesem Kapitel wird über weitere Forschungsfelder, der in der Arbeit behandelten Themen, motiviert und spekuliert, sowie eine kurze Zusammenfassung der Arbeit und ihrer Ergebnisse gegeben.

6.1 Ausblick und zukünftige Arbeiten

Diese Arbeit lässt ein breites Spektrum an Ansätzen für spätere Forschungen offen, um weitere Antworten auf die in Abschnitt 3.4 und 4.4 gestellten Fragen zu finden.

Auf der Seite von lokalen parallelen Berechnungen (siehe offene Fragen in Abschnitt 3.4) kann versucht werden, den Quantenzustand nicht wie hier beschrieben, nach der Anwendung eines Quantensuchalgorithmus, zu messen, sondern die Berechnungen auf der Quantenebene weiterzuführen. Eine erfolgversprechende Möglichkeit wäre die Entwicklung/ das Finden eines Quantenalgorithmus für die Bildung Kernel-stabiler Koalitionen oder des TRACONET Protokolls, der nicht an die klassischen Versionen angelehnt ist, sondern von Anfang an auf abstrakte Weise das Ziel erreicht. In einem solchen Protokoll könnte unter anderem die Holevoschranke [NC00, 12.1.1] umgangen werden. Auch die Fortführung der hier erstellten und verwendeten Schaltkreise für die parallele Berechnung klassischer Werte bis hin zu einer Quanten-ALU scheint vielversprechend.
Im Bereich der Koordination durch Verschränkung fehlen noch Forschungsvorarbeiten. Viele theoretischen Ansätze werden in der Praxis auf Grund von nicht durchführbaren Annahmen (ausreichend verschränkte Paare) nicht umsetzbar sein.
Im Bereich der Simulation stößt man bei dem Thema Quantenmultiagentensysteme schnell an die Grenze des Machbaren. Wegen der in Kapitel 2.3.2 und 5 beschriebenen Probleme ist der Einsatz einer großen Anzahl von Quantenbits nicht möglich. Dies liegt nicht allein an der exponentiellen Laufzeit. Der Quantensimulator kann Zahlenwerte, wie sie für solche großen Quantenregisterzustände nötig sind, nicht handhaben. Damit ist die Beschränkung auf 64 Quantenbits (bei *libquantum*) das Maximum. An dieser Stelle wäre ein Quantensimulator wünschenswert, der eine höhere Anzahl an Quantenbits simulieren kann, solange sie sich nicht in absoluter Superposition befindet (zum Beispiel 20 parallele Werte mit 300 Quantenbits).

6.2 Zusammenfassung

In dieser Arbeit wurden erste Möglichkeiten untersucht, klassische Probleme aus dem Bereich der Multiagentensysteme mit Hilfe von Quantenagenten umzusetzen und zu verbessern. Es wurde der Algorithmus zur Kernel-stabilen Koalitionsbildung aus dem Bereich der Spieltheorie und das TRACONET Protokoll aus dem Bereich der verteilten Problemlösung betrachtet. Dabei lag das Augenmerk auf der Kommunikation und Koordination der Agenten.
Im ersten Fall wurde die Idee verfolgt, durch mehr lokale Operationen der einzelnen Agenten, den Austausch von Informationen zu reduzieren. Dazu wurde versucht mit Hilfe der Quantenparallelität die rechen-aufwendigsten Schritte zu verbessern, was nur hinreichend, nicht für eine asymptotische Stufe, gelang. Trotzdem wird durch den Ansatz, wegen eines Quantensuchalgorithmus, Verbesserungen erreicht, die im klassischen Fall nicht möglich sind.
Bei dem zweiten Protokoll wurde eine Reihe von Quantenprimitiven untersucht, um eine Verbesserung im Bereich der Kommunikation und Koordination zu erlangen. Dabei wurden Quantenauktionen für die Verhandlungen umgesetzt und der Informationsaustausch zwischen den Agenten durch sichere Quantenkommunikationsprotokolle verbessert. Es stellte sich heraus, dass im Bereich der Komplexität nur schwer Verbesserungen zu finden sind, da der Aufwand für die Verteilung von verschränkten Quantenzuständen, die als Grundlage der meisten Anwendungen im Bereich der Quantenkoordination und -kommunikation dient, alle erreichten Verbesserungen wieder aufhob. Auch die Holevoschranke und das No-Cloning-Theorem zusammen mit den probabilistischen Messungen sind ein Hindernis. Trotzdem besitzen beide Quantenprotokolle genau dadurch Eigenschaften, die durch klassische Berechnungen nicht erreicht werden können.

Anhang A

Beispiele

Die Beispiele der in der Arbeit vorgestellten Quantenalgorithmen wurden gekürzt, um die Übersicht zu gewährleisten. So wurden alle Superpositionen entfernt und das Beispiel nur mit einem der vielen parallelen Zustände durchgeführt. Ausführlichere Beispiele, die alle Superpositionszustände betrachten sind auf der CD-Rom der Arbeit in dem Dokument *examples.pdf* zu finden. Im folgenden wird für jedes gekürzte Beispiel der betrachtete Zustand angegeben und einen Verweis auf das Beispieldokument gegeben.

A.1 Beispiele für die parallele Berechnung klassischer Werte

In diesem Anhang werden Beispiele für die in Abschnitt 2.2.3 vorgestellten Schaltkreise und Algorithmen gegeben.

A.1.1 3-qubit Oder-Schaltkreis

Vergleiche hierzu Abbildung 2.2.

$$
\begin{aligned}
|\psi\rangle \quad &= \quad \frac{1}{\sqrt{3}}(|\overbrace{000}^{a_{3,2,1}}\rangle + |\overbrace{101}^{a_{3,2,1}}\rangle + |\overbrace{111}^{a_{3,2,1}}\rangle) \otimes |\overbrace{0}^{c}\rangle \\
&= \quad \frac{1}{\sqrt{3}}(|0000\rangle + |1010\rangle + |1110\rangle) \\
&\overset{\text{3n0p-CNOT}(a_3,a_2,a_1 \rightarrow \text{c})}{\rightarrow} \quad \frac{1}{\sqrt{3}}(|0001\rangle + |1010\rangle + |1110\rangle) \\
&\overset{\text{Sigma-X(c)}}{\rightarrow} \quad \frac{1}{\sqrt{3}}(|0000\rangle + |1011\rangle + |1111\rangle)
\end{aligned}
$$

A.1.2 3-qubit Gleichheit Test

Der Schaltkreis testet zwei Quantenregisterabschnitte (a,b) auf ihre Gleichheit. Vergleiche hierzu Abbildung 2.3. Dieses Beispiel wurde gekürzt, nur der Start- und Endzustand sind hier aufgeführt. Das komplette Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu

finden.

$$
\begin{aligned}
& |\psi\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|\overbrace{011}^{a_{3,2,1}}\overbrace{111}^{b_{3,2,1}}\rangle + |\overbrace{000}^{a_{3,2,1}}\overbrace{001}^{b_{3,2,1}}\rangle + |\overbrace{010}^{a_{3,2,1}}\overbrace{010}^{b_{3,2,1}}\rangle) \otimes |\overbrace{000:0}^{t_{3,2,1}:c}\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|011{:}111{:}000{:}0\rangle + |000{:}001{:}000{:}0\rangle + |010{:}010{:}000{:}0\rangle) \\
& \vdots \\
\rightarrow \quad & \frac{1}{\sqrt{3}}(|011{:}111{:}000{:}0\rangle + |000{:}001{:}000{:}0\rangle + |010{:}010{:}000{:}1\rangle)
\end{aligned}
$$

A.1.3 3-qubit Größer Test

Der Schaltkreis testet ob die Zahl von Quantenregisterabschnitt a größer als die in Quantenregisterabschnitt b ist. Vergleiche hierzu Abbildung 2.4. Dieses Beispiel wurde gekürzt, nur der Start- und Endzustand sind hier aufgeführt. Das komplette Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{aligned}
& |\psi\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|\overbrace{101}^{a_{3,2,1}}\overbrace{111}^{b_{3,2,1}}\rangle + |\overbrace{010}^{a_{3,2,1}}\overbrace{001}^{b_{3,2,1}}\rangle + |\overbrace{110}^{a_{3,2,1}}\overbrace{110}^{b_{3,2,1}}\rangle) \otimes |\overbrace{00000:0}^{t_{5,4,3,2,1}:c}\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|101{:}111{:}00000{:}0\rangle + |010{:}001{:}00000{:}0\rangle + |110{:}110{:}00000{:}0\rangle) \\
& \vdots \\
\rightarrow \quad & \frac{1}{\sqrt{3}}(|101{:}111{:}00000{:}0\rangle + |010{:}001{:}00000{:}1\rangle + |110{:}110{:}00000{:}0\rangle)
\end{aligned}
$$

A.1.4 3-qubit Addierer

Vergleiche hierzu Abbildung 2.5. Dieses Beispiel wurde gekürzt, nur der Start- und Endzustand sind hier aufgeführt. Das komplette Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{aligned}
& |\psi\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|\overbrace{101}^{a,b,c_{in}}\rangle + |\overbrace{010}^{a,b,c_{in}}\rangle + |\overbrace{111}^{a,b,c_{in}}\rangle) \otimes |\overbrace{00}^{s_1,s_0}\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|101{:}00\rangle + |010{:}00\rangle + |111{:}00\rangle) \\
& \vdots \\
\rightarrow \quad & \frac{1}{\sqrt{3}}(|101{:}10\rangle + |010{:}01\rangle + |111{:}\mathbf{1}1\rangle)
\end{aligned}
$$

A.1.5 n-qubit Addierer

Der Algorithmus addiert zwei Zahlen von einem Quantenregisterabschnitt miteinander und schreibt das Ergebnis auf einen anderen "leeren" Quantenregisterabschnitt. Vergleiche hierzu Algorithmus 1. Im Beispiel wird eine Addition mit Two's Complement Zahlen gezeigt. Die zusätzlichen Bits $s[n+1]$ und c_{se} werden direkt an die Quantenregister s und c angehängt. Dieses Beispiel wurde gekürzt, nur der Start- und Endzustand sind hier aufgeführt. Das komplette Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{aligned}
& |\psi\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|\overbrace{01}^{a_{1,0}=1}\ \overbrace{10}^{b_{1,0}=-2}\rangle + |\overbrace{11}^{a_{1,0}=-1}\ \overbrace{01}^{b_{1,0}=1}\rangle + |\overbrace{10}^{a_{1,0}=-2}\ \overbrace{11}^{b_{1,0}=-1}\rangle) \otimes |\overbrace{0000:000:00}^{s_{3,2,1,0}:c_{2,1,0}:a_s e,b_s e}\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|01{:}10{:}0000{:}000{:}00\rangle + |11{:}01{:}0000{:}000{:}00\rangle + |10{:}11{:}0000{:}000{:}00\rangle) \\
& \vdots \\
\rightarrow \quad & \frac{1}{\sqrt{3}}(|01{:}10{:}0\overbrace{111}^{-1}{:}000{:}00\rangle + |11{:}01{:}0\overbrace{000}^{0}{:}000{:}\mathbf{0}0\rangle + |10{:}11{:}0\overbrace{101}^{-3}{:}000{:}\mathbf{0}0\rangle)
\end{aligned}
$$

A.1.6 n-qubit Subtrahierer

Der Algorithmus subtrahiert zwei Zahlen von einem Quantenregisterabschnitt miteinander und schreibt das Ergebnis auf einen anderen "leeren" Quantenregisterabschnitt. Vergleiche hierzu Algorithmus 2. Dieses Beispiel wurde gekürzt, nur der Start- und Endzustand sind hier aufgeführt. Das komplette Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{aligned}
& |\psi\rangle \\
= \quad & \frac{1}{\sqrt{3}}(|\ \overbrace{001}^{a_{1,0}=1}\ \overbrace{110}^{b_{1,0}=-2}\ \rangle + |\ \overbrace{111}^{a_{1,0}=-1}\ \overbrace{001}^{b_{1,0}=1}\rangle + |\ \overbrace{111}^{a_{1,0}=-1}\ \overbrace{101}^{b_{1,0}=-3}\ \rangle) \otimes |\ \overbrace{0000:000:10}^{s_{3,2,1,0}:c_{2,1,0}:ctrl_1,ctrl_2}\ \rangle \\
= \quad & \frac{1}{\sqrt{3}}(|001{:}110{:}0000{:}000{:}10\rangle + |111{:}001{:}0000{:}000{:}10\rangle + |111{:}101{:}0000{:}000{:}10\rangle) \\
& \vdots \\
\rightarrow \quad & \frac{1}{\sqrt{3}}(|001{:}110{:}\overbrace{0011}^{3}{:}000{:}00\rangle + |111{:}001{:}\overbrace{1110}^{-2}{:}000{:}00\rangle + |111{:}101{:}\overbrace{0010}^{2}{:}000{:}00\rangle)
\end{aligned}
$$

A.2 Beispiel für einen Kernel-stabilen Koalitionsalgorithmus

A **n**		$\{a_1, a_2, a_3\}$ 3	
	a_1	a_2	a_3
Angebot	ws_1 kostet 1	ws_2 kostet 1 ws_4 kostet 2	ws_3 kostet 1
Nachfrage	ws_2 kostet 2	ws_3 kostet 2	ws_1 kostet 3
Zeile			
2	$(\{\{a_1\},\{a_2\},\{a_3\}\},(v(\{a_1\}),v(\{a_2\}),v(\{a_3\})=0))$		
3	Führer: a_1	Führer: a_2	Führer: a_3
4	Führungsliste: $[a_2 > a_1 > a_3]$		
5	sende ws_2 an a_2,a_3	sende ws_3 an a_1,a_3	sende ws_1 an a_1,a_2
6	empfange ws_3 und ws_1	empfange ws_2 und ws_1	empfange ws_2 und ws_3
8	sende ws_1 an a_2,a_3	sende ws_2 an a_1,a_3	sende ws_3 an a_1,a_2
9	empfange ws_2 und ws_3	empfange ws_1 und ws_3	empfange ws_1 und ws_2
11 $C = \{a_1\}$	$lw_{a_1}(C) = 0$	$lw_{a_2}(C) = -1$	$lw_{a_3}(C) = 3$
11 $C = \{a_2\}$	$lw_{a_1}(C) = 2$	$lw_{a_2}(C) = 0$	$lw_{a_3}(C) = -1$
11 $C = \{a_3\}$	$lw_{a_1}(C) = -1$	$lw_{a_2}(C) = 2$	$lw_{a_3}(C) = 0$
11 $C = \{a_1, a_2\}$	$lw_{a_1}(C) = 2$	$lw_{a_2}(C) = -1$	$lw_{a_3}(C) = 2$
11 $C = \{a_1, a_3\}$	$lw_{a_1}(C) = -1$	$lw_{a_2}(C) = 1$	$lw_{a_3}(C) = 3$
11 $C = \{a_2, a_3\}$	$lw_{a_1}(C) = 1$	$lw_{a_2}(C) = 2$	$lw_{a_3}(C) = -1$
11 $C = \{a_1, a_2, a_3\}$	$lw_{a_1}(C) = 1$	$lw_{a_2}(C) = 1$	$lw_{a_3}(C) = 2$
12 $\forall C$	sende $lw_{a_1}(C)$ an a_2,a_3	sende $lw_{a_2}(C)$ an a_1,a_3	sende $lw_{a_3}(C)$ an a_1,a_2
14 $\forall C \forall a$	empfange alle $lw_a(C)$		
19 $C = \{a_1, a_2\}$	$S^* = \{\{a_1, a_2\},\{a_3\}\}, u^* =< 1,0,0 >$		
20	a_1 sendet an a_2, a_2 sendet an a_1		
19 $C = \{a_1, a_3\}$	$S^* = \{\{a_1, a_3\},\{a_2\}\}, u^* =< 1,0,1 >$		
20	a_1 sendet an a_3, a_3 sendet an a_1		
19 $C = \{a_2, a_3\}$	$S^* = \{\{a_2, a_3\},\{a_1\}\}, u^* =< 0,0,1 >$		
20	a_2 sendet an a_3, a_3 sendet an a_2		
24	von $a_2 \rightarrow < 1,0,0 >$ von $a_3 \rightarrow < 1,0,1 >$	von $a_1 \rightarrow < 1,0,0 >$ von $a_3 \rightarrow < 0,0,1 >$	von $a_1 \rightarrow < 1,0,1 >$ von $a_2 \rightarrow < 0,0,1 >$
25	wählt $< 1,0,1 >$	wählt $< 1,0,0 >$	wählt $< 1,0,1 >$
28	Informieren aller Führer über den angenommenen Vorschlag		
29	empfängt Vorschläge $< 1,0,1 >$, $< 1,0,0 >$ und $< 1,0,1 >$		
33	wählt $< 1,0,1 >$	wählt $< 1,0,1 >$	wählt $< 1,0,1 >$
45	Führer: a_1		Führer: a_1
49	an a_2 Führer: a_1		
51		von a_1 Führer: a_1	
57	Nächste Runde $\rightarrow$ Zeile 16		

Zeile	a_1	a_2	a_3
aktuelle Konfiguration	$(\{\{a_1, a_3\}, \{a_2\}\}, (v(\{a_1, a_3\}) = 2, v(\{a_2\}) = 0))$		
16		springe nach 41	
19 $C = \{a_1, a_2, a_3\}$ 20	$S^* = \{\{a_1, a_2, a_3\}\}, u^* =< 1.5, 1, 1.5 >$ a_1 sendet an a_2, a_2 sendet an a_1		
25	wählt $< 1.5, 1, 1.5 >$	wählt $< 1.5, 1, 1.5 >$	
28 28	Informieren alle Führer über den angenommenen Vorschlag		
29	empfängt Vorschläge $< 1.5, 1, 1.5 >$, $< 1.5, 1, 1.5 >$		
33	wählt $< 1.5, 1, 1.5 >$	wählt $< 1.5, 1, 1.5 >$	
40	informiere a_3		
41			empfange $< 1.5, 1, 1.5 >$
45	Führer: a_2	Führer: a_2	Führer: a_2
54	Große Koalition		

A.3 Beispiele für den KCA-Q

Falls nicht anders angesprochen, werden die Werte aus Anhang A.2 verwendet. Für die Konstanten gilt $l = 2$. Des Weiteren werden 3 Qubits für die Kodierung der Aufgaben-Id und 4 Qubits für die Kodierung der Kosten verwendet.

A.3.1 Berechne Angebotsliste (Algorithmus 7)

Dieses Beispiel betrachtet nur die Koalition $\{a_1, a_2, a_3\}$, ein ausführlicheres Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{aligned}
& |\phi\rangle \\
= \quad & |\overbrace{111}^{\{a_1,a_2,a_3\}}\rangle \otimes |\overbrace{100{:}001{:}0001;000{:}000{:}0000}^{\mathbf{O_a1}}\rangle \otimes |\overbrace{010{:}010{:}0001;010{:}100{:}0010}^{\mathbf{O_a2}}\rangle \\
& \otimes |\overbrace{001{:}011{:}0001;000{:}000{:}0000}^{\mathbf{O_a3}}\rangle \otimes |\overbrace{000{:}000{:}0000;000{:}000{:}0000}^{\psi_\mathbf{a1}}\rangle \\
& \otimes |\overbrace{000{:}000{:}0000;000{:}000{:}0000}^{\psi_\mathbf{a2}}\rangle \otimes |\overbrace{000{:}000{:}0000;000{:}000{:}0000}^{\psi_\mathbf{a3}}\rangle \\
= \quad & |111;100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;001{:}011{:}0001;000{:}000{:}0000; \\
& 000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000\rangle \\
\text{Z. 2 } a_1 \rightarrow \quad & |111;100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;001{:}011{:}0001;000{:}000{:}0000; \\
& \mathbf{100{:}001{:}0001};000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000\rangle \\
\text{Z. 2 } a_2 \rightarrow \quad & |111;100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;001{:}011{:}0001;000{:}000{:}0000; \\
& 100{:}001{:}0001;000{:}000{:}0000;\mathbf{010{:}010{:}0001;010{:}100{:}0010};000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000\rangle \\
\text{Z. 2 } a_3 \rightarrow \quad & |111;100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;001{:}011{:}0001;000{:}000{:}0000; \\
& 100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;\mathbf{001{:}011{:}0001;000{:}000{:}0000};000{:}000{:}0000;000{:}000{:}0000\rangle
\end{aligned}
$$

A.3.2 Suche Angebote für Nachfragen (Algorithmus 8)

In diesem Beispiel werden die Werte aus dem vorherigen Beispiel weiter übernommen. Wir betrachten die Nachfragen von Agent a_1 und füllen die *val* Quantenregister mit den Kosten der Nachfragen, falls in der Koalition ein Angebot für die Nachfrage existiert. Dieses Beispiel betrachtet nur die Koalition $\{a_1, a_2, a_3\}$, ein ausführlicheres Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{aligned}
& |\phi\rangle \\
= \quad & |\overbrace{111}^{\{a1,a2,a3\}};\overbrace{100{:}010{:}0010;000{:}000{:}0000}^{\mathbf{T_{a1}}};\overbrace{00{:}00{:}00{:}00{:}00{:}00;}^{tmp};\overbrace{0000{:}0000}^{val};\overbrace{00}^{c}; \\
& 100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;001{:}011{:}0001;000{:}000{:}0000\rangle \\
\text{Z. 2 } i = 1 \rightarrow \quad & |111;100{:}010{:}0010;000{:}000{:}0000;00{:}\mathbf{1}0{:}00{:}00{:}00{:}00;0000{:}0000;00; \\
& 100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;001{:}011{:}0001;000{:}000{:}0000\rangle \\
\text{Z. 5 } i = 1 \rightarrow \quad & |111;100{:}010{:}0010;000{:}000{:}0000;00{:}10{:}00{:}00{:}00{:}00;0000{:}0000;\mathbf{1}0; \\
& 100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;001{:}011{:}0001;000{:}000{:}0000\rangle
\end{aligned}
$$

Z. 6 $i = 1$ $\rightarrow$ |111;100:010:0010;000:000:0000;00:10:00:00:00:00;**0010**:0000;10;
100:001:0001;000:000:0000;010:010:0001;010:100:0010;001:011:0001;000:000:0000⟩

$i = 2$ $\rightarrow$ keine Änderung da keine 2. Nachfrage vorhanden

Z. 10 $\rightarrow$ |111;100:010:0010;000:000:0000;00:**00**:00:00:00:00;0010:0000;**00**;
100:001:0001;000:000:0000;010:010:0001;010:100:0010;001:011:0001;000:000:0000⟩

A.3.3 Summiere die Kosten der Nachfragen (Algorithmus 9)

In diesem Beispiel werden die *val* Quantenregister aufaddiert. Da hier $l = 2$ wird nur eine Stufe benötigt, um dies zu erledigen. Die anderen Quantenregister werden nicht beeinflußt und deshalb nicht dargestellt. Dieses Beispiel betrachtet nur die Koalition $\{a_1, a_2, a_3\}$, ein ausführlicheres Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$|\phi\rangle$

$= \quad | \overbrace{111}^{\{a_1,a_2,a_3\}} ; \overbrace{0010{:}0000}^{val}; \overbrace{00000}^{tmp}\rangle$

Z. 1 Stufe 1 $\rightarrow$ $| \overbrace{111}^{\{a_1,a_2,a_3\}} ; \overbrace{0010{:}0000}^{val}; \overbrace{\mathbf{00010}}^{tmp}\rangle)$

Z. 6 $\rightarrow$ keine Zwischenstufen vorhanden

A.3.4 Berechne Liste von gleichen Angeboten (Algorithmus 10)

In diesem Beispiel werden Listen von gleichen Angeboten erstellt. Dabei wird von den Angeboten $\mathbf{O}_a$ ausgegangen. Das Quantenregister All$\mathbf{O}(\mathbf{C})$ hat veränderte Werte von a_3, da dieses Beispiel bei den bisherigen Werten keine gleichen Angebote aufweisen würde. Dieses Beispiel betrachtet nur die Koalition $\{a_1, a_2, a_3\}$, ein ausführlicheres Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$|\phi\rangle = \quad | \overbrace{111}^{\{a_1,a_2,a_3\}} ; \overbrace{100{:}001{:}0001;000{:}000{:}0000}^{\mathbf{O}_a}; \overbrace{0}^{c};$
$\overbrace{100{:}001{:}0001;000{:}000{:}0000;010{:}010{:}0001;010{:}100{:}0010;001{:}011{:}0001;001{:}001{:}0010;}^{\text{All}\mathbf{O}(\mathbf{C})}$
$\overbrace{000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000}^{\mathbf{gO_1}}; \overbrace{000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000}^{\mathbf{gO}}\rangle$

Z. 2 $i = 1, j = 1$ $\rightarrow$ |111; 100:001:0001;000:000:0000;**1**;
100:001:0001;000:000:0000;010:010:0001;010:100:0010;001:011:0001;001:001:0010;
000:000:0000;000:000:0000;000:000:0000;000:000:0000;000:000:0000;000:000:0000⟩

Z. 4 $i = 1, j = 1$ $\rightarrow$ |111; 100:001:0001;000:000:0000;1;
100:001:0001;000:000:0000;010:010:0001;010:100:0010;001:011:0001;001:001:0010;
100:001:0001;000:000:0000;000:000:0000;000:000:0000;000:000:0000;000:000:0000⟩

Z. 7 $i = 1, j = 1$ $\rightarrow$ |111; 100:001:0001;000:000:0000;**0**;
100:001:0001;000:000:0000;010:010:0001;010:100:0010;001:011:0001;001:001:0010;

100:001:0001;000:000:0000;000:000:0000;000:000:0000;000:000:0000;000:000:0000⟩

Z. 2, 4, 7 $i = 1, j = 2$bis5 $\rightarrow$ keine Veränderung

Z. 2 $i = 1, j = 6$ $\rightarrow$ |111; 100:001:0001;000:000:0000;**1**;
100:001:0001;000:000:0000;010:010:0001;010:100:0010;001:011:0001;001:001:0010;
100:001:0001;000:000:0000;000:000:0000;000:000:0000;000:000:0000;000:000:0000⟩

Z. 4 $i = 1, j = 6$ $\rightarrow$ |111; 100:001:0001;000:000:0000;1;
100:001:0001;000:000:0000;010:010:0001;010:100:0010;001:011:0001;001:001:0010;
100:001:0001;000:000:0000;**001:001:0010**;000:000:0000;000:000:0000;000:000:0000⟩

Z. 7 $i = 1, j = 6$ $\rightarrow$ |111; 100:001:0001;000:000:0000;**0**;
100:001:0001;000:000:0000;010:010:0001;010:100:0010;001:011:0001;001:001:0010;
100:001:0001;000:000:0000;001:001:0010;000:000:0000;000:000:0000;000:000:0000⟩

Z. 7 $i = 2$ $\rightarrow$ keine Veränderung

A.3.5 Suche Angebot mit minimalen Kosten (Algorithmus 11)

In diesem Beispiel wird aus den Listen der gleichen Angebote die ausgewählt, die die minimalen Kosten besitzen. Dieses Beispiel betrachtet nur die Koalition $\{a_1, a_2, a_3\}$, ein ausführlicheres Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$|\phi\rangle =$ | $\overbrace{111}^{\{a_1,a_2,a_3\}}$; $\overbrace{000:000:0000;000:000:0000}^{min_i}$; $\overbrace{000:000:0000}^{tmp}$; $\overbrace{0:0:0:0:0}^{nzb,bsa,za,c1,c}$;
$\overbrace{100:001:0001;000:000:0000;001:001:0010}^{\mathbf{l}_1}$; $\overbrace{000:000:0000;000:000:0000;000:000:0000}^{\mathbf{l}_2}$⟩

Z. 4 bis 8 $i = 1, j = 2$ $\rightarrow$ |111; 000:000:0000;000:000:0000;000:000:0000;**0:1:0:1:0**;
100:001:0001;000:000:0000;001:001:0010;000:000:0000;000:000:0000;000:000:0000⟩

Z. 9 $i = 1, j = 2$ $\rightarrow$ keine Änderung

Z. 17 bis 21 $i = 1, j = 2$ $\rightarrow$ |111; 000:000:0000;000:000:0000;000:000:0000;**0:0:0:0:0**;
100:001:0001;000:000:0000;001:001:0010;000:000:0000;000:000:0000;000:000:0000⟩

Z. 4 bis 8 $i = 1, j = 3$ $\rightarrow$ |111; 000:000:0000;000:000:0000;000:000:0000;**1:0:0:0:0**;
100:001:0001;000:000:0000;001:001:0010;000:000:0000;000:000:0000;000:000:0000⟩

Z. 9 $i = 1, j = 3$ $\rightarrow$ keine Änderung

Z. 17 bis 21 $i = 1, j = 3$ $\rightarrow$ |111; 000:000:0000;000:000:0000;000:000:0000;**0:0:0:0:0**;
100:001:0001;000:000:0000;001:001:0010;000:000:0000;000:000:0000;000:000:0000⟩

Z. 23 $i = 1$ $\rightarrow$ |111; **100:001:0001**;000:000:0000;000:000:0000;0:0:0:0:0;
100:001:0001;000:000:0000;001:001:0010;000:000:0000;000:000:0000;000:000:0000⟩

$i = 2$ $\rightarrow$ keine Änderung

A.3.6 Berechne Liste von optimalen Angeboten (Algorithmus 12)

In diesem Beispiel wird eine Liste von optimalen Angeboten erstellt, die zu Agent a_1 gehören. Dieses Beispiel betrachtet nur die Koalition $\{a_1, a_2, a_3\}$, ein ausführlicheres Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{array}{ll}
 & |\overbrace{111}^{\{a_1,a_2,a_3\}};\overbrace{100{:}001{:}0001;000{:}000{:}0000}^{min_i};\overbrace{000{:}000{:}0000;000;000;0000}^{\mathbf{optO}};\overbrace{100}^{a_1};\overbrace{0}^{c}\rangle \\
|\phi\rangle = & \\
\text{Z. } 2 \xrightarrow{i=1} & |111;100{:}001{:}0001;000{:}000{:}0000;000{:}000{:}0000;000{:}000{:}0000;100;\mathbf{1}\rangle \\
\text{Z. } 4 \xrightarrow{i=1} & |111;100{:}001{:}0001;000{:}000{:}0000;\mathbf{100{:}001{:}0001};000{:}000{:}0000;100;1\rangle \\
\text{Z. } 6 \xrightarrow{i=1} & |111;100{:}001{:}0001;000{:}000{:}0000;100{:}001{:}0001;000{:}000{:}0000;100;\mathbf{0}\rangle \\
i=2 \xrightarrow{} & \text{keine Veränderung}
\end{array}
$$

A.3.7 Subtrahieren der Summen, um $lw_a(C)$ zu erhalten

Dieses Beispiel betrachtet nur die Koalition $\{a_1, a_2, a_3\}$, ein ausführlicheres Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden. Unter Verwendung der Werte von Beispiel aus Anhang A.2 für Agent a_1 ist der Quantenzustand:

$$
|\phi\rangle = |\overbrace{111}^{\{a_1,a_2,a_3\}};\overbrace{0010}^{\sum_{x\in\mathbf{R}_a(\{a_1,a_2,a_3\})} w_a(x)};\overbrace{0001}^{\sum_{x\in\mathbf{E}_a(\{a_1,a_2,a_3\})} c_a(x)}\rangle
$$

Nach dem Hinzufügen eines Ergebnisregisters und dem Durchführen der Subtraktion (siehe dazu Anhang A.1.6) ist der resultierende Zustand:

$$
|\phi\rangle = \frac{1}{\sqrt{8}}(|\overbrace{111}^{\{a_1,a_2,a_3\}};\overbrace{0010}^{\sum_{x\in\mathbf{R}_a(\{a_1,a_2,a_3\})} w_a(x)};\overbrace{0001}^{\sum_{x\in\mathbf{E}_a(\{a_1,a_2,a_3\})} c_a(x)};\overbrace{00001}^{lw_{a_1}(\{a_1,a_2,a_3\})}\rangle
$$

A.3.8 Erstellen der Koalitionen (Algorithmus 14)

Das Beispiel erzeugt die in Algorithmus 14 beschriebenen Quantenzustände. Die Anzahl der Agenten beläuft sich auf 4 und betrachtet wird das Agentenpaar $(a2, a3)$. Dieses Beispiel wurde gekürzt, nur der Start- und Endzustand sind hier aufgeführt. Das komplette Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{array}{ll}
 & |\overbrace{0110}^{(a2,a3)};\overbrace{0000;0000}^{\mathbf{C}_1,\mathbf{C}_2}\rangle \\
|\phi\rangle = & \vdots \\
\rightarrow & \frac{1}{\sqrt{4}}(|0110;0010;0100\rangle + |0110;1010;1100\rangle + |0110;0011;0101\rangle + \\
 & |0110;1011;1101\rangle)
\end{array}
$$

A.3.9 Berechnen der charakteristischen Funktion (Algorithmus 15)

Hier die Werte für die charakteristische Funktion, wie in Algorithmus 15 beschrieben, berechnet. Um die Größe der Quantenregister überschaubar zu halten ist die Zahl der Agenten 3. Des Weiteren wird hier das Agentenpaar $(a2, a3)$ betrachtet. $\mathbf{C}$ ist eines der Koalitionsregister

aus Algorithmus 14. Dieses Beispiel wurde gekürzt, nur der Start- und Endzustand sind hier aufgeführt. Das komplette Beispiel ist auf der CD-Rom (*examples.pdf*) zur Arbeit zu finden.

$$
\begin{aligned}
|\phi\rangle = \quad & \tfrac{1}{\sqrt{2}}(|\overbrace{010}^{\mathbf{C}};\overbrace{000;0000}^{lw_a1};\overbrace{000;0000}^{lw_a2};\overbrace{000;0000}^{lw_a1};\overbrace{0000;0000;0000}^{tmp};\overbrace{000000}^{res}\rangle \\
& +|110;000;0000;000;0000;000;0000;0000;0000;0000;000000\rangle) \\
& \vdots \\
\rightarrow \quad & \tfrac{1}{\sqrt{2}}(|010;\mathbf{000};0000;000;0000;000;0000;0000;0000;0000;111111\rangle \\
& +|110;\mathbf{000};0000;000;0000;000;0000;0000;0000;0000;000001\rangle)
\end{aligned}
$$

A.3.10 Subtrahieren der Werte $v(C)$ und $u(C)$

Unter Verwendung der Werte von Beispiel aus Anhang A.2 für Agentenpaar (a_2, a_3) mit initialem $u(\mathbf{C})$ ist der Quantenzustand:

$$
\begin{aligned}
|\phi\rangle = \quad & \tfrac{1}{\sqrt{2}}(|\overbrace{010}^{\mathbf{C}};\overbrace{000000}^{u(\mathbf{C})};\overbrace{000000}^{v(\mathbf{C})}\rangle \\
& +|011;000000;000001\rangle
\end{aligned}
$$

Nach dem Hinzufügen eines Ergebnisregisters und dem Durchführen der Subtraktion (siehe dazu Anhang A.1.6) ist der resultierende Zustand:

$$
\begin{aligned}
|\phi\rangle = \quad & \tfrac{1}{\sqrt{2}}(|\overbrace{010}^{\mathbf{C}};\overbrace{000000}^{u(\mathbf{C})};\overbrace{000000}^{v(\mathbf{C})};\overbrace{0000000}^{res};\rangle \\
& +|011;000000;000000;0000001;\rangle
\end{aligned}
$$

A.3.11 Finden des Maximalen Excesswertes

Da Algorithmus 4 eine Mindestgröße benötigt, was die Anzahl verschiedener Zustände in Superposition angeht, wird hier wieder ein System von 4 Agenten und das Agentenpaar (a_2, a_3) betrachtet. Algorithmus 14 liefert den Zustand:

$$\tfrac{1}{\sqrt{4}}(|0110;0010;0100\rangle + |0110;1010;1100\rangle + |0110;0011;0101\rangle + |0110;1011;1101\rangle)$$

Wir betrachten $\mathbf{C_1}$ also ist der Startzustand für Algorithmus 4:

$$\tfrac{1}{\sqrt{4}}(|0010;0\rangle + |1010;0\rangle + |0011;0\rangle + |1011;0\rangle)$$

Um das Initiale y herauszufinden werden die Excesswerte berechnet und der Zustand danach gemessen, um einen zufälligen Excesswert zu erhalten.

$$
\begin{aligned}
|\phi\rangle = \quad & \tfrac{1}{\sqrt{4}}(|0010;0\rangle + |1010;0\rangle + |0011;0\rangle + |1011;0\rangle) \\
\overset{\text{Excess}}{\rightarrow} \quad & \tfrac{1}{\sqrt{4}}(|000001;0010;0\rangle + |111111;1010;0\rangle + |000010;0011;0\rangle + |000011;1011;0\rangle) \\
\overset{\text{Messen}}{\rightarrow} \quad & 000010;0011;0 \\
\rightarrow \quad & y = 2
\end{aligned}
$$

Jetzt wird eine Grover-Suche mit $y = 1$ durchgeführt, die Zeilenangaben beziehen sich auf Algorithmus 3.

$\lvert\phi\rangle =$	$\frac{1}{\sqrt{4}}(\lvert 0010;0\rangle + \lvert 1010;0\rangle + \lvert 0011;0\rangle + \lvert 1011;0\rangle)$
Z. 1 $\rightarrow$	$\frac{1}{\sqrt{8}}(\lvert 0010;0\rangle + \lvert 1010;0\rangle + \lvert 0011;0\rangle + \lvert 1011;0\rangle - \lvert 0010;1\rangle - \lvert 1010;1\rangle - \lvert 0011;1\rangle - \lvert 1011;1\rangle)$
Z. 4 Werte berechnen $\rightarrow$	$\frac{1}{\sqrt{8}}(\lvert 000010;000001;0010;0\rangle + \lvert 000010;111111;1010;0\rangle$ $+\lvert 000010;000010;0011;0\rangle + \lvert 000010;000011;1011;0\rangle$ $-\lvert 000010;000001;0010;1\rangle - \lvert 000010;111111;1010;1\rangle$ $-\lvert 000010;000010;0011;1\rangle - \lvert 000010;000011;1011;1\rangle)$
Z. 4 Subtrahieren $\rightarrow$	$\frac{1}{\sqrt{8}}(\lvert 0000001;000010;000001;0010;0\rangle$ $+\lvert 0000011;000010;111111;1010;0\rangle$ $+\lvert 0000000;000010;000010;0011;0\rangle$ $+\lvert 1111111;000010;000011;1011;0\rangle$ $-\lvert 0000001;000010;000001;0010;1\rangle$ $-\lvert 0000011;000010;111111;1010;1\rangle$ $-\lvert 0000000;000010;000010;0011;1\rangle$ $-\lvert 1111111;000010;000011;1011;1\rangle)$
Z. 4 Bit setzen $\rightarrow$	$\frac{1}{\sqrt{8}}(\lvert 0000001;000010;000001;0010;0\rangle$ $+\lvert 0000011;000010;111111;1010;0\rangle$ $+\lvert 0000000;000010;000010;0011;0\rangle$ $+\lvert 1111111;000010;000011;1011;\mathbf{1}\rangle$ $-\lvert 0000001;000010;000001;0010;1\rangle$ $-\lvert 0000011;000010;111111;1010;1\rangle$ $-\lvert 0000000;000010;000010;0011;1\rangle$ $-\lvert 1111111;000010;000011;1011;\mathbf{0}\rangle)$
Z. 4 Reinigen $\rightarrow$	$\frac{1}{\sqrt{8}}(\lvert 0010;0\rangle + \lvert 1010;0\rangle + \lvert 0011;0\rangle + \lvert 1011;1\rangle - \lvert 0010;1\rangle - \lvert 1010;1\rangle - \lvert 0011;1\rangle - \lvert 1011;0\rangle)$
Z. 5 $\rightarrow$	keine Veränderung
Z. 6 $\rightarrow$	$\frac{1}{\sqrt{8}}(-\lvert 0010;0\rangle + \lvert 1010;0\rangle + \lvert 0011;0\rangle - \lvert 1011;1\rangle + \lvert 0010;1\rangle - \lvert 1010;1\rangle - \lvert 0011;1\rangle + \lvert 1011;0\rangle)$
Z. 7 $\rightarrow$	$\frac{1}{\sqrt{4}}(\lvert 1010;0\rangle - \lvert 0011;0\rangle - \lvert 1010;1\rangle + \lvert 0011;1\rangle)$
Z. 8 $\rightarrow$	keine Veränderung
Z. 9 $\rightarrow$	$\frac{1}{\sqrt{8}}(-\lvert 0010;0\rangle + \lvert 1010;0\rangle + \lvert 0011;0\rangle + \lvert 1011;1\rangle + \lvert 0010;1\rangle - \lvert 1010;1\rangle - \lvert 0011;1\rangle - \lvert 1011;0\rangle)$
Z. 10 $\rightarrow$	keine Veränderung
Z. 11 $\rightarrow$	$\frac{1}{\sqrt{2}}(\lvert 1011;1\rangle - \lvert 1011;0\rangle)$
Excess berechnen $\rightarrow$	$\frac{1}{\sqrt{2}}(\lvert 000011;1011;1\rangle - \lvert 000011;1011;0\rangle)$
Z. 13 $\rightarrow$	000011;1011;1
$\rightarrow$	$y = 2$

A.4 Beispiel für TRACONET

Hier wird ein Beispiel für das klassische TRACONET Protokoll gegeben. Betrachtet werden Fahrradkuriere in einem Blockstadtsystem, die sich gegenseitig Aufträge überlassen, wenn diese für den Auftraghalter nicht rentabel sind. Dabei hat jeder Auftrag einen Abhol- und einen Zielort. Im Beispiel betrachten wir drei Transportzentren, die jeweils einen Kurier besitzen. Der lokale Optimierer der einzelnen Zentren ist intuitiv, so werden als Kosten die Entfernungen genommen, die die Kuriere zurücklegen müssen. Jeder Kurier betrachtet für ein abzugebendes Gebot sowohl die Menge der Aufgaben die er im Moment besitzt, als auch die, die er nach Abgabe seiner aktuell laufenden Ausschreibungen noch hat. Er wählt als Gebotspreis den schlechteren der beiden Fälle, um sicher zu gehen, dass ihm das Gebot kein Nachteil bringt.

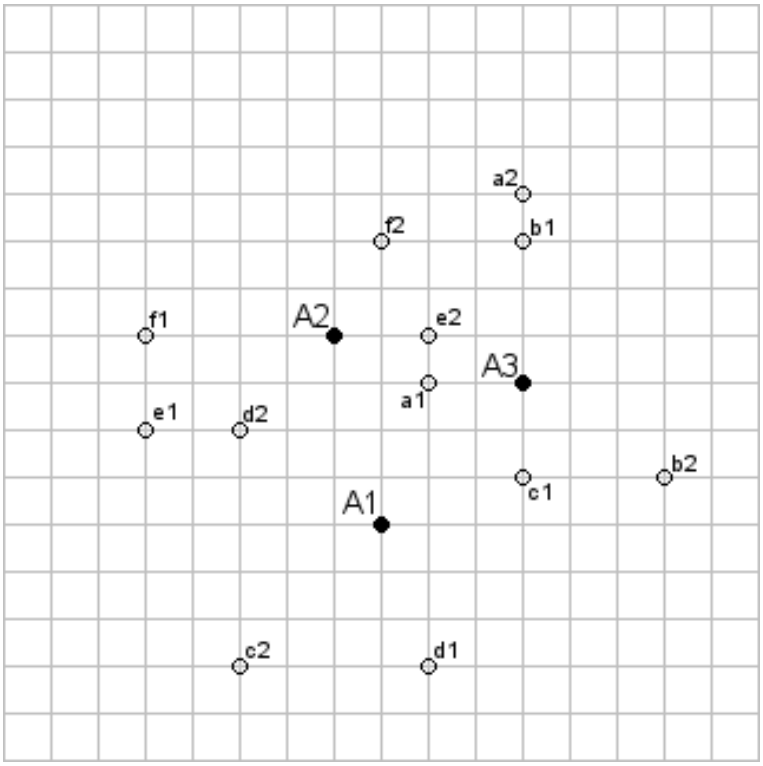


Abbildung A.1: Lageplan der Zentren und Lieferungen

Abbildung A.1 zeigt die Lage der drei Zentren (A_1, A_2, A_3) und die Aufträge $\{a, b, c, d, e, f\}$, wobei der Index 1 den Abholpunkt und Index 2 den Zielort angibt. Die Zentren versuchen ihre ungünstigsten Aufträge aus der Route ihres Kuriers zu entfernen und einem anderen Zentrum anzubieten.

A		$\{A_1, A_2, A_3\}$	
n		3	
	A_1	A_2	A_3
Aufträge	c, f	e, b	a, d
Route **Kosten der Route**	$R = [A_1 \to c_1 \to f_1 \to f_2 \to c_2 \to A_1]$ $C(R) = 4 + 8 + 7 + 9 + 6 = 34$	$R = [A_2 \to e_1 \to e_2 \to b_1 \to b_2 \to A_2]$ $C(R) = 6 + 8 + 4 + 8 + 10 = 36$	$R = [A_3 \to a_1 \to d_1 \to d_2 \to a_2 \to A_3]$ $C(R) = 2 + 6 + 9 + 11 + 4 = 32$
Bisher keine Nachrichten empfangen.			
Ausschreibungen **ungünstigster Auftrag** **Kosten ohne Ausschreibung**	$c = 20, f = 22 \to As(f, 14)$ $C(R') = 20$	$e = 16, b = 22 \to As(b, 20)$ $C(R') = 16$	$a = 12, d = 24 \to As(d, 20)$ $C(R') = 14$
Gebot für $As(f, 14)$ **Kosten mit aktueller Ausschreibung** **Kosten ohne aktuelle Ausschreibung** **Gebotsabgabe**		$C(R \cup \{f\}) = 36$ $C(R' \cup \{f\}) = 22$ $G(f, 6)$	$C(R \cup \{f\}) = 34$ $C(R' \cup \{f\}) = 28$ $G(f, 14)$
Gebot für $As(b, 20)$ **Kosten mit aktueller Ausschreibung** **Kosten ohne aktuelle Ausschreibung** **Gebotsabgabe**	$C(R \cup \{b\}) = 48$ $C(R' \cup \{b\}) = 36$ $G(b, 16)$		$C(R \cup \{b\}) = 42$ $C(R' \cup \{b\}) = 22$ $G(b, 10)$
Gebot für $As(d, 20)$ **Kosten mit aktueller Ausschreibung** **Kosten ohne aktuelle Ausschreibung** **Gebotsabgabe**	$C(R \cup \{d\}) = 42$ $C(R' \cup \{d\}) = 24$ $G(d, 8)$	$C(R \cup \{d\}) = 44$ $C(R' \cup \{d\}) = 36$ $G(d, 20)$	
Evaluation	Zuschlag(A_2) Ablehnung(A_3)	Zuschlag(A_3) Ablehnung(A_1)	Zuschlag(A_1) Ablehnung(A_2)
Route **Kosten der Route**	$R = [A_1 \to c_1 \to d_1 \to c_2 \to d_2 \to A_1]$ $C(R) = 4 + 6 + 4 + 5 + 5 = 24$	$R = [A_2 \to f_1 \to f_2 \to e_1 \to e_2 \to A_2]$ $C(R) = 3 + 7 + 2 + 8 + 2 = 22$	$R = [A_3 \to a_1 \to a_2 \to b_1 \to b_2 \to A_3]$ $C(R) = 2 + 6 + 1 + 8 + 5 = 22$
Ausschreibungen **ungünstigster Auftrag** **Kosten ohne Ausschreibung**	$c = 20, d = 18 \to As(c, 6)$ $C(R') = 18$	$e = 16, f = 14 \to As(e, 8)$ $C(R') = 14$	$a = 12, b = 16 \to As(b, 10)$ $C(R') = 12$
Gebot für $As(c, 6)$		kein günstiges Gebot	kein günstiges Gebot
Gebot für $As(e, 8)$	kein günstiges Gebot		kein günstiges Gebot
Gebot für $As(b, 10)$	kein günstiges Gebot	kein günstiges Gebot	

A.5 Beispiel für TRACONET-Q

A.5.1 Quantenbroadcast-Protokoll

Dies ist ein Beispiel für die Quantenbroadcast-Kommunikation aus Abbildung 4.1. Hier betrachten wir die reine Kommunikation ohne die Authentifizierung und Absicherung der Übertragung. Mit den Authentifizierungsschritten (5a-f) kann der Manager die Korrektheit der Paarschlüssel, mit der Absicherung der Übertragung (Schritte 11, 12, 13) die Sicherheit bei der Rücksendung der Partikel in Schritt 7 überprüfen. Der Manager M sendet eine 2-bit Nachricht an die Agenten A_1 und A_2. Somit ist $N = 1$ und $r = 2$. In Schritt 1 wählt M einen der 3-Partikel GHZ Zustände.

$$\Psi_1 = \tfrac{1}{\sqrt{2}} \cdot (|\overbrace{0}^{M}\overbrace{0}^{A_1}\overbrace{0}^{A_2}\rangle + |\overbrace{1}^{M}\overbrace{1}^{A_1}\overbrace{1}^{A_2}\rangle)$$

$$\Psi_2 = \tfrac{1}{\sqrt{2}} \cdot (|001\rangle + |110\rangle)$$
$$\Psi_3 = \tfrac{1}{\sqrt{2}} \cdot (|010\rangle + |101\rangle)$$
$$\Psi_4 = \tfrac{1}{\sqrt{2}} \cdot (|100\rangle + |011\rangle)$$
$$\Psi_5 = \tfrac{1}{\sqrt{2}} \cdot (|000\rangle - |111\rangle)$$
$$\Psi_6 = \tfrac{1}{\sqrt{2}} \cdot (|001\rangle - |110\rangle)$$
$$\Psi_7 = \tfrac{1}{\sqrt{2}} \cdot (|010\rangle - |101\rangle)$$
$$\Psi_8 = \tfrac{1}{\sqrt{2}} \cdot (|100\rangle - |011\rangle)$$

Ohne Beschränkung der Allgemeinheit wählt M Ψ_1. Schritt 2 wendet M entsprechend der Paarschlüssel $K_{Mi}(i \in \{1,2\})$ die Operationen I, H auf die Partikel von A_i an. Ohne Beschränkung der Allgemeinheit sei $K_{M1} = 1$ und $K_{M2} = 0$.

$$(I \otimes \overbrace{H}^{K_{M1}=1} \otimes \overbrace{I}^{K_{M2}=0})\Psi_1 = \frac{1}{2} \cdot (|000\rangle + |010\rangle + |101\rangle - |111\rangle) \tag{A.1}$$

In Schritt 3 sendet M die A_i-Sequenz (hier nur aus einem Partikel bestehend) an A_i. Beide Agenten A_1 und A_2 wenden die gleichen Operatoren entsprechend ihrem Schlüssel K_{Mi} auf das erhaltene Partikel an. Das Gesamtsystem verhält sich dabei wie folgt:

Operation von A_1:

$$\begin{aligned}(I \otimes \overbrace{H}^{K_{M1}=1} \otimes I)(\frac{1}{2} \cdot (|000\rangle + |010\rangle + |101\rangle - |111\rangle)) &= \frac{1}{\sqrt{2}} \cdot (|000\rangle + |111\rangle) \\ &= \Psi_1 \end{aligned} \tag{A.2}$$

Operation von A_2:

$$(I \otimes I \otimes \overbrace{I}^{K_{M2}=0})\Psi_1 = \Psi_1 \tag{A.3}$$

Nach der Authentifizierung wählen A_1 und A_2 in Schritt 6 einen zufälligen String s (hier bestehend aus nur einem Bit). Ohne Beschränkung der Allgemeinheit sei $s_1 = 0$ und $s_2 = 1$. Entsprechend dem String wenden sie die Operatoren $I, i\sigma_y$ auf ihren Partikel an.
Operation von A_1:

$$(I \otimes \overbrace{I}^{s_1=0} \otimes I)\Psi_1 \quad = \quad \Psi_1 \tag{A.4}$$

Operation von A_2:

$$(I \otimes I \otimes \overbrace{i\sigma_y}^{s_2=1})\Psi_1 \quad = \quad \frac{1}{\sqrt{2}} \cdot (-|001\rangle + |110\rangle) \tag{A.5}$$

In Schritt 7 senden beide ihren Partikel an M. In Schritt 9 kodiert M seine 2-bit Nachricht anhand der Operatoren $\{I, \sigma_x, i\sigma_y, \sigma_z\}$ in der M-Sequenz. Ohne Beschränkung der Allgemeinheit sei die Nachricht 01.
Operation von M:

$$(\overbrace{\sigma_x}^{01} \otimes I \otimes I)\frac{1}{\sqrt{2}} \cdot (-|001\rangle + |110\rangle) \quad = \quad \frac{1}{\sqrt{2}} \cdot (-|101\rangle + |010\rangle) \tag{A.6}$$

In Schritt 10 misst M den Quantenzustand in der GHZ Basis, das Ergebnis ist Ψ_7. Er sendet das Ergebnis und den Ausgangszustand Ψ_1 an A_1 und A_2. Diese können anhand folgender Tabelle für den Ausgangszustand Ψ_1 den verwendeten Operator und somit die Nachricht mit Hilfe ihres Strings ablesen.

s_1	s_2	Messergebnis	Kodierungsoperator
0	0	$\frac{1}{\sqrt{2}} \cdot (\|000\rangle + \|111\rangle) = \Psi_1$	I
0	1	$\frac{1}{\sqrt{2}} \cdot (\|110\rangle - \|001\rangle) = -\Psi_6$	
1	0	$\frac{1}{\sqrt{2}} \cdot (\|101\rangle - \|010\rangle) = -\Psi_7$	
1	1	$\frac{1}{\sqrt{2}} \cdot (\|100\rangle + \|011\rangle) = \Psi_4$	
0	0	$\frac{1}{\sqrt{2}} \cdot (\|100\rangle + \|011\rangle) = \Psi_4$	σ_x
0	1	$\frac{1}{\sqrt{2}} \cdot (\|010\rangle - \|101\rangle) = \Psi_7$	
1	0	$\frac{1}{\sqrt{2}} \cdot (\|001\rangle - \|110\rangle) = \Psi_6$	
1	1	$\frac{1}{\sqrt{2}} \cdot (\|000\rangle + \|111\rangle) = \Psi_1$	
0	0	$\frac{1}{\sqrt{2}} \cdot (\|011\rangle - \|100\rangle) = -\Psi_8$	$i\sigma_y$
0	1	$\frac{1}{\sqrt{2}} \cdot (\|010\rangle + \|101\rangle) = \Psi_3$	
1	0	$\frac{1}{\sqrt{2}} \cdot (\|001\rangle + \|110\rangle) = \Psi_2$	
1	1	$\frac{1}{\sqrt{2}} \cdot (\|000\rangle - \|111\rangle) = -\Psi_5$	
0	0	$\frac{1}{\sqrt{2}} \cdot (\|000\rangle - \|111\rangle) = \Psi_5$	σ_z
0	1	$\frac{1}{\sqrt{2}} \cdot (-\|110\rangle - \|001\rangle) = -\Psi_2$	
1	0	$\frac{1}{\sqrt{2}} \cdot (-\|101\rangle - \|010\rangle) = -\Psi_3$	
1	1	$\frac{1}{\sqrt{2}} \cdot (-\|100\rangle + \|011\rangle) = -\Psi_8$	

Für das Messergebnis Ψ_7 und dem String $s_1 = 0$ kann der gesuchte Operator nur σ_x sein.

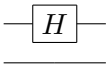
A.5.2 TRACONET-Q Verhandlung

Wegen der Größe wird hier nur die Verhandlung einer Aufgabe in einem System aus 4 Agenten betrachtet. A_1 ist der Manager der Aufgabe und schreibt diese in *q-ausschreiben()* zu dem von ihm verrechneten maximalen Kosten von $c_{max} = 12$ aus. Für den Quantenbroadcast siehe A.5.1. Dazu verteilt er einen Quantenzustand an alle drei Agenten:

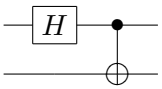
$$\begin{aligned} |\Psi_{init}\rangle &= |\psi^2_{init}\rangle \otimes |\psi^3_{init}\rangle \otimes |\psi^4_{init}\rangle \\ &= |000000\rangle \end{aligned} \tag{A.7}$$

$$\tag{A.8}$$

In *q-bieten* evaluieren A_2, A_3 und A_4 die Ausschreibung. A_2 kann die Aufgabe zu Kosten von $c_{bid} = 10$ übernehmen und kodiert deshalb die Differenz in seinen Quantenoperator.

$U_2 =$ 

A_2 wendet den Operator auf $|\psi^2_{init}\rangle$ an und erhält $|\psi^2_0\rangle = \frac{|00\rangle+|10\rangle}{\sqrt{2}}$.
A_3 ist an der Aufgabe nicht interessiert und unternimmt nichts. A_4 berechnet $c_{bid} = 9$ und erstellt einen Quantenoperator.

$U_4 =$ 

Der resultierende Zustand aus $U_4 \cdot |\psi^4_{init}\rangle$ ist $|\psi^2_0\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$.
A_2 und A_4 senden ihre Quantenbits wieder an A_1 zurück. Das Quantensystem ist somit beim Manager in Zustand:

$$|\Psi_0\rangle = \frac{1}{2} \cdot (|0000\rangle + |0011\rangle + |1000\rangle + |1011\rangle) \tag{A.9}$$

Es beginnen die Verhandlungsrunden, in denen die Auktionsiterationen durchgeführt werden. Die Diagonalmatrix $W = diag(0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4)$ resultierend aus der Hammingdistanz der Gebote zu $|0000\rangle$.
Die Diagonalmatrix $H_p = diag(0, -1, -2, -3, -1, 0, 0, 0, -2, 0, 0, 0, -3, 0, 0, 0)$ sind die negativen Gewinnergebote in den möglichen Endzuständen, bzw 0 für nicht mögliche Endzustände. Daraus ergeben sich $D(f) = \exp(-i\Delta(1-f)W)$ und $P(f) = \exp(-i\Delta f H_p)$. In jeder Iterationsrunde wird der Zustand des Systems auf $\Psi_s = UD(f)U^\dagger P(f)\Psi_{s-1}$ aktualisiert. Nach 20 Iterationsrunden ist das System ungefähr in Zustand:

$$\begin{aligned} |\Psi_{20}\rangle &= -0,007 - 0,015i \cdot |0000\rangle + 0.708 + 0.675i \cdot |0011\rangle \\ &\quad -0,106 - 0,179i \cdot |1000\rangle - 0,007 - 0,015i|1011\rangle \end{aligned} \tag{A.10}$$

Eine Messung ergibt mit hoher Wahrscheinlichkeit das Ergebnis 0011 und somit den Gewinner der Auktion A_4 bekannt. A_1 sendet Ablehnungen an A_2 und A_3 und eine Gewinnernachricht mit der entsprechenden Aufgabe an A_4.

Literaturverzeichnis

[ACL99] M. Ardehali, H. F. Chau und Hoi-Kwong Lo, *Efficient Quantum Key Distribution*, arXiv:quant-ph/9803007v4, Los Alamos National Laboratory/Cornell University, 1999 [Accessed 3 Jan 2008].

[AG91] Y. Afek und E. Gafni, *Time and Message Bounds for Election in Synchronous and Asynchronous Complete Networks*, Symposium on Principles of Distributed Computing, 1991,
http://citeseer.ist.psu.edu/afek91time.html [Accessed 3 Jan 2008].

[AK99] A. Ahuja und S. Kapoor, *A Quantum Algorithm for finding the Maximum*, arXiv:quant-ph/9911082v1, Los Alamos National Laboratory/Cornell University, 1999 [Accessed 3 Jan 2008].

[BB84] C. H. Bennett und G. Brassard, *Quantum cryptography: Public-key distribution and coin tossing*, Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, India, 1984.

[BBC+93] C. H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres und W. Wootters, *Teleporting an unknown quantum state via dual classical and EPR channels*, Physical Review Letters, Vol. 70, American Physical Society, 1993.

[BCD97] H. Buhrman, R. Cleve und W. van Dam, *Quantum Entanglement and Communication Complexity*, arXiv:quant-ph/9705033v1, Los Alamos National Laboratory/Cornell University, 1997 [Accessed 3 Jan 2008].

[BCS03] S. Bettelli, T. Calarco und L. Serafini, *Toward an Architecture for Quantum Programming*, The European Physical Journal D, Vol. 25, Springer, 2003.

[BK04] B. Blankenburg und M. Klusch, *On Safe Kernel Stable Coalition Forming among Agents*, Autonomous Agents and Multiagent Systems, AAMAS 2004, Proceedings of the Third International Joint Conference on, IEEE Computer Society, 2004.

[BPM+97] D. Bouwmeester, J. Pan, K. Mattle, M. Eibl, H. Weinfurter und A. Zeilinger, *Experimental quantum teleportation*, Nature, Vol. 390, Macmillan Publishers Ltd., 1997.

[BVK98] S. Bose, V. Vedral und P. L. Knight, *A Multiparticle Generalization of Entanglement Swapping*, Physical Review A, Vol. 57, American Physical Society, 1998.

[BW92] C. H. Bennett und S. J. Wiesner, *Communication via one- two-particle operators on Einstein-Podolsky-Rosen states*, Physical Review Letters, Vol 69, N0. 20, American Physical Society,1992.

[BZW98] W. Brenner, R. Zarnekow und H. Wittig, *Intelligent Software Agents: Foundations and Applications*, Springer Verlag, 1998.

[Deu84] D. Deutsch, *Quantum theory, the Chruch-Turing principle and the universal quantum computer*, Proceedings of the Royal Society of London Ser. A 400, Royal Society Publishing, 1985.

[DFJN97] J. E. Doran, S. Franklin, N. R. Jennings und T. J. Norman, *On Cooperation in Multi-Agent Systems*, The Knowledge Engineering Review 12(3), Cambridge University Press, 1997.

[DL91] E. H. Durfee und V. R. Lesser, *Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21 No. 5, 1991.

[DM65] M. Davis und M. Maschler, *The kernel of a cooperative game*, Naval Research Logistics Quarterly, Vol. 12, I. 3, Wiley, 1965.

[DVC00] W. Dür, G. Vidal und J. I. Cirac, *Three qubits can be entangled in two inequivalent ways*, Physical Review A, Vol. 62, 062314, American Physical Society, 2000.

[Fey82] R. P. Feynman, *Simulating Physics with Computers*, International Journal of Theoretical Physics, Vol.21, No. 6/7, Springer, 1982.

[GC97] N. A. Gershenfeld und I. L. Chuang, *Bulk Spin-Resonance Quantum Computation*, Science Vol. 275, American Association for the Advancement of Science, 1997.

[GHF+07] S. Guha, T. Hogg, D. Fattal, T. Spiller und R. G. Beausoleil, *Quantum Auctions using Adiabatic Evolution: The Corrupt Auctioneer and Circuit Implementations*, arXiv:0707.2051v1 [quant-ph], Los Alamos National Laboratory/Cornell University, 2007 [Accessed 3 Jan 2008].

[GHN+97] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers und R.Evans, *Software Agents: A review*, Techncal report, TCS-CS-1997-06. Trinity College Dublin, Broadcom Eireann Research Ltd., 1997,
http://citeseer.ist.psu.edu/green97software.html, [Accessed 3 Jan 2008].

[Gro96] L. K. Grover, *A fast quantum mechanical algorithm for database search*, Proceedings, 28th ACM. Symposium on Theory of Computing (STOC), 1996.

[HBWC99] A.L.G. Hazyzelden, J. Bigham, M. Wooldridge, und L.G. Cuthbert, *Future communication networks using software agents*, In A.L.G. Hayzelden und J. Bigham, editors, Software Agents for Future Communciation Systems, chapter 1. Springer-Verlag, 1999.

[HHC07] T. Hogg, P. Harsha und K. Chen, *Quantum Auctions*, arXiv:0704.0800v1 [quant-ph], Los Alamos National Laboratory/Cornell University, 2007 [Accessed 3 Jan 2008].

[HHR+05] H. Häffner, W. Hänsel1, C. F. Roos, J. Benhelm, D. Chek-al-kar, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür und R. Blatt, *Scalable multiparticle entanglement of trapped ions*, Nature, Vol. 438, Macmillan Publishers Ltd., 2005.

[HI91] M. J. Holler und G. Illing, *Einführung in die Spieltheorie*, Springer-Verlag, Heidelberg, 1991.

[Hol73] A. S. Holevo, *Some estimates for information quantity transmitted by quantum communication channel*, Problems of Information Transmission, Vol. 9, No. 3, MAIK Nauka/Interperiodica, 1973 (rus).

[HP06] E. D'Hondt und P. Panangaden, *The Computational Power of the W and GHZ states*, arXiv:quant-ph/0412177v2, Los Alamos National Laboratory/Cornell University, 2006 [Accessed 3 Jan 2008].

[KB98] A. Karlsson und M. Bourennane, *Quantum teleportation using three-particle entanglement*, Physical Review A, Vol. 58, No. 6, American Physical Society, 1998.

[Klu04] M. Klusch, *Toward Quantum Computational Agents*, In: Agents and Computational Autonomy. M Nickles, M Rovatsos, G Weiss (eds.), Lecture Notes in Artificial Intelligence series, Vol. 2969, Springer, 2004.

[KP95] J. Keller und W. J. Paul, *Hardware Design*, Teubner-Texte zur Informatik, Vol. 15, 2. Auflage Teubner, 1995.

[KR84] J. P. Kahan und A. Rapoport, *Theories of Coalition Formation*, Lawrence Erlbaum Associates, Hillsdale, 1984.

[KS96] M. Klusch und O. Shehory, *A Polynomial Kernel-Oriented Coalition Algorithm for Rational Information Agents*, Proc. Second International Conference on Multi-Agent Systems, AAAI Press, 1996.

[MP93] J. P. Müller und M. Pischel, *The Agent Architecture InteRRap: Concept and Application*, Technical Report RR-93-26, German Research Center for Artificial Intelligence, Saarbrücken Germany, 1993,
http://citeseer.ist.psu.edu/227024.html, [Accessed 3 Jan 2008].

[MP00] S. M. Mueller und W. J. Paul, *Computer architecture: complexity and correctness*, Springer, 2000.

[NC00] M. A. Nielsen und I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge Univ. Press, Cambridge, 2000.

[Oem00] B. Oemer, *Quantum Programming in QCL*, Master Thesis, Technical University of Vienna, Computer Science Department, Vienna, Austria, 2000.

[PSK03] S. P. Pal, S. K. Singh und S. Kumar, *Multi-partite Quantum Entanglement versus Randomization: Fair and Unbiased Leader Election in Networks*, arXiv:quant-ph/0306195v1, Los Alamos National Laboratory/Cornell University, 2003 [Accessed 3 Jan 2008].

[Ric07] W. Richter, *Rechenwunder, bitte warten*, Financial Times Deutschland, 14. Februar 2007.

[Rin98] J. Rink, *Quäntchen für Quäntchen, Fortschritte in der Quanteninformationsverarbeitung*, c't Magazin für Computertechnik, Heft 16, 1998.

[SAC+06] K. M. Svore, A. V. Aho, A. W. Cross, I. Chuang und I. L. Markov, *A Layered Software Architecture for Quantum Computing Design Tools*, IEEE Computer Society, Vol. 39, No. 1, 2006.

[San93] T. Sandholm, *An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations*, Eleventh National Conference on Artificial Intelligence, AAAI Press, 1993.

[Sch07] R. Schubotz, *Programming and Simulation of Quantum Agents*, Master Thesis, Saarland University, Department of Computer Science, Saarbrücken, Germany, 2007.

[Sel04] P. Selinger, *Towards a Quantum Programming Language*, Mathematical Structures in Computer Science, Vol. 14, No. 4, Cambridge University Press, 2003.

[Sho94] P. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, Proceedings, 35th Annual Symposium on Foundations of Computer Science, IEEE Press, Los Alamitos, CA, 1994.

[SIG+05] V. Scarani, S. Iblisdir, N. Gisin und A. Acin, *Quantum cloning*, Review of Modern Physics, Vol. 77, No. 1225, American Physical Society, 2005.

[Smi80] R. G. Smith, *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver*, IEEE Transactions on Computers, Vol. C-29, No. 12, 1980.

[SDP+96] K. Sycara, K. Decker, A. Pannu, M. Williamson und D. Zeng. *Distributed intelligent agents*, IEEE Expert Vol. 11, I. 6, 1996.

[SPG06] A. J. Short, S. Popescu und N. Gisin, *Entanglement swapping for generalized nonlocal correlations*, Physical Review A, Vol. 73, 012101, American Physical Society, 2006.

[Ste97] A. Steane, *Quantum computing*, arXiv:quant-ph/9708022v2, Los Alamos National Laboratory/Cornell University, 1997 [Accessed 3 Jan 2008].

[Tru01] C. A. Trugenberger, *Probabilistic Quantum Memories*, Physical Review Letters, Vol. 87, 067901, American Physical Society, 2001.

[Wal99] J. Wallace *Quantum Computer Simulators - A Review Version 2.1*, 1999, http://citeseer.ist.psu.edu/399644.html, [Accessed 3 Jan 2008].

[WRZ05] P. Walther, K. J. Resch und A. Zeilinger, *Local Conversion of Greenberger-Horne-Zeilinger States to Approximate W States*, Physical Review Letters, Vol. 94, 240501, American Physical Society, 2005.

[WZ82] W. K. Wootters und W. H. Zurek, *A single quantum cannot be cloned*, Nature, Vol. 299, No. 5886, Macmillan Publishers Ltd., 1982.

[WZT07] J. Wang, Q. Zhang und C. Tang, *Quantum broadcast communication*, Chinese Physics, Vol. 16, Chin. Phys. Soc./IOP Publishing Ltd., 2007.

[Yam05] S. Yamashita, *How to Utilize Grover Search in General Programming*, Laser Physics, Vol. 16, No. 4, Springer, 2006.

[YCH04] C.-P. Yang, S.-I Chu und S. Han, *Efficient many-party controlled teleportation of multiqubit quantum information via entanglement*, Physical Review A, Vol. 70, 022329, American Physical Society, 2004.

[Zal99] C. Zalka, *Grover's quantum searching algorithm is optimal*, Physical Review A, Vol. 60, American Physical Society, 1999.