Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Bachelorarbeit Informatik

Analyse von Strategien zur Taskabarbeitung durch rekonfigurierbare Agenten

Mirko Pinseler

Studiengang Informatik

Leipzig, 15.10.2012

Betreuer

Martin Middendorf

Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Mirko Pinseler:
Analyse von Strategien zur Taskabarbeitung durch rekonfigurierbare Agenten

Bachelorarbeit Informatik
Universität Leipzig

Inhaltsverzeichnis

Kapitel 1

Einführung und Definition der Konfigurationsräume

1.1 Einleitung

Der Gegenstand dieser Arbeit ist von einem Organic Computing System inspiriert. Ein Organic Computing System beschreibt ein "technisches System, das sich dynamisch an geänderte Umweltbedingungen anpasst." [Muller-Schloer, 2004]. Um diese Anforderung zu erfüllen bedient sich Organic Computing Verfahren, deren Vorbilder sich in der Natur wiederfinden. Organic Computing Systeme weisen eigenständiges organisiertes Verhalten ohne erxterne Kontrolle auf, ähnlich diversen Organisationsformen in der Natur, z.B. einem Ameisenstaat. Die Selbstorganisation im Organic Computing wird von den sogenannten "self-x" Eigenschaften begleitet, z.B. self-healing oder self-optimization [Würtz, 2008]. Oft bestehen Organic Computing Systeme u.a. aus autonomen Untereinheiten, sogenannten Agenten. Agenten sind in sich abgeschlossene Systeme, die mit ihrer Umwelt interagieren können und so zur Lösung einer bestimmten Aufgabe oder eines Problems beitragen. Inspiration dieser Arbeit waren rekonfigurierbare Helfer-Agenten, wie sie in [Merkle et al., 2008] untersucht werden. Diese passen sich auf hardware-Ebene an gestellte Aufgaben an. Es sind Helfer, die für andere Komponenten des Organic Computing Systems bestimmte Services bereitstellen. Die Helfer lösen bestimmte Tasks, die von den Arbeiterkomponenten des OC Systems an sie gerichtet wurden. Dabei ist von Interesse, wie die Helfer organisiert sein müssen, um möglichst geringe Rekonfigurationskosten zu erzeugen. Diese Arbeit nimmt eine theoretische Analyse zweier möglicher Organisationsformen von rekonfigurierbaren Agenten vor. Dabei ist von Interesse, unter welchen Umständen die Rekonfigurationskosten möglichst gering ausfallen, bei zufälligen eintreffenden Tasks. Zu Beginn werden die Konfigurationsräume partitionierter und unpartitionierter Kreis definiert. In diesen Konfigurationsräumen treffen die Tasks ein und bewegen sich die Agenten.

Anschließend werden die auftretenden Kosten für verschiedene zeitliche Abstände, in denen die Tasks eintreffen, untersucht. Die Tasks treffen in variablen und festen Zeitabständen ein. Es werden jeweils die Maximalkosten und die Durchschnittskosten für die zwei Konfigurationsräume berechnet und die Ergebnisse verglichen.

1.2 Kreisgraph und partitionierter Kreisgraph

Die Konfigurationsmöglichkeiten von Agenten werden mittels eines Konfigurationsgraphen modelliert. Alle möglichen Konfigurationen eines Agenten können auf einen Graph abgebildet werden. Die Knoten des Konfigurationsgraphen entsprechen den Konfigurationen. Die Kanten zu Nachbarknoten modellieren die Nachbarkonfigurationen, zu denen sich innerhalb eines Schrittes konfiguriert werden kann. Ein Agent befindet sich auf genau einem Knoten im Konfigurationsgraph. Auch einem zu lösendem Task wird ein Knoten zugeordnet. Der dem Task zugeordnete Knoten entspricht der Konfiguration, die am besten für die Lösung des Tasks geeignet ist. Ein Agent muss sich entlang des Konfigurationsgraphen auf den entsprechenden Knoten bewegen, also in den entsprechenden Zustand konfigurieren, um den Task zu lösen. Es können auf einem Konfigurationsgraph mehrere Agenten vorhanden sein. Ein Task wird immer von genau einem Agenten gelöst. Es wird angenommen, dass für jede Rekonfiguration die gleichen Kosten anfallen, im Graph existieren keine individuellen Kantengewichte. Die Anzahl an Rekonfigurationen um einen Task zu lösen stellt somit ein Maß für die entstehenden Kosten dar. Die eigentliche Lösung der Tasks ist kostenlos. Sobald ein Agent einen Task erreicht, gilt dieser als gelöst. Für ein Beispiel eines Konfigurationsgraphen siehe Abbildung 1.1.
Die Konfigurationsgraphen, auf die sich in dieser Arbeit beschränkt wird, sind der Kreisgraph bzw. der partitionierte Kreisgraph, jeweils mit zwei Agenten besetzt. Im Kreisgraph besitzt jeder Knoten genau zwei Nachbarknoten, d.h. ein Agent hat zwei mögliche Nachbarkonfigurationen. Im partitionierten Kreisgraph gibt es zwei gleich große Partitionen, zwei Halbkreise, mit jeweils einem Agent. Ein Agent darf seine Partition nicht verlassen. In den folgenden Kapiteln wird untersucht, welche Kosten auf beiden Konfigurationsräumen anfallen, wenn zu bestimmten Zeitpunkten zufällige Tasks eintreffen, die gelöst werden müssen.

1.3 Stetiger Konfigurationsraum

Um die mathematischen Berechnungen zu vereinfachen wird anstatt des diskreten Kreisgraphen der stetige Kreis betrachtet. Die Länge des Kreises entspricht der Anzahl an Knoten im diskreten Graph. Dadurch muss bei Be-

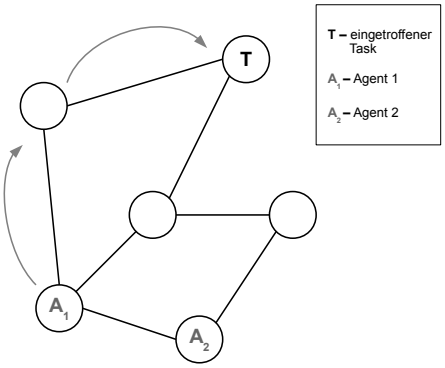

Abbildung 1.1: Konfigurationsgraph. Der nähere Agent bewegt sich zum Task.

rechnungen keine Unterscheidung in gerade bzw. ungerade Knotenanzahlen vorgenommen werden. Es ist zwar unmöglich im stetigen Kreis jeder Postition eine konkrete Konfiguration zuzuordnen, jedoch gestaltet sich die Berechnung von Durchschnitts- und Maximalkosten wesentlich einfacher. Der Abstand zwischen Agent und Task entspricht den Rekonfigurationskosten. Der Abstand zweier Agenten entspricht den Kosten, die anfallen würden um einen Agent genau wie den anderen zu konfigurieren. Die Durchschnitts- und Maximalkosten vom diskreten Kreisgraph und vom stetigen Kreis unterscheiden sich nur bei sehr kleiner Kreislänge wesentlich (Kreisgraph weniger als 10 Knoten).

Kapitel 2

Optimale zeitliche Taskabstände

Dieses Kapitel untersucht die Kosten, die entstehen, wenn die zeitlichen Abstände der Tasks optimal sind. Das bedeutet, dass ein neuer Task genau dann eintrifft, wenn der vorige bearbeitet wurde. Dadurch stehen für einen neuen Task immer beide Agenten zu Verfügung.

2.1 Unpartitionierter Kreisgraph

2.1.1 Taskfolgen maximaler Kosten

Zuerst werden Taskfolgen betrachtet, die auf dem unpartitionierten Kreis maximale Kosten erzeugen. Eine Folge von Tasks erzeugt genau dann maximale Kosten auf dem Kreis, wenn jeder Task die längere Strecke zwischen beiden Agenten halbiert. Das soll im Folgenden gezeigt werden.

Definition 1. Es seien $k_i,\ i \in \mathbb{N}$ die Kosten, die ein eintreffender Task t_i verursacht. $k_{i\,\max}$ bezeichnet die maximalen Kosten, die ein eintreffender Task in Schritt i verursachen kann, also die Halbierung der längeren Distanz zwischen beiden Agenten auf dem Kreis.

Lemma 2. *Sei $t_1, t_2, \ldots, t_m$ eine Taskfolge mit maximalen Kosten. Dann gilt $k_m = k_{m\,\max}$.*

Beweis. Annahme $k_m < k_{m\,\max}$. Dann könnte man als letzten Task einen Task wählen, der $k_{m\,\max}$ Kosten erzeugt. Die Gesamtkosten dieser Folge wären größer, als die der Ausgangsfolge. Da die Ausgangsfolge maximale Kosten aufweist, ergibt sich ein Widerspruch. □

Lemma 3. *Sei $t_1, t_2, \ldots, t_m$ eine Taskfolge, wobei gilt $k_i = k_{i\,\max},\ i \geq 2$. Vor Eintreffen des ersten Tasks befinden sich beide Helfer auf der gleichen Position. Die Gesamtkosten werden maximal, wenn auch $k_1 = k_{1\,\max}$.*

Beweis. Es gilt folgende Rekursionsformel:

$$k_i = \frac{n - k_{i-1}}{2},\ i \geq 2 \tag{1.1}$$

Die vorhergehenden Kosten k_{i-1} sind der aktuellen kürzere Abstand zwischen beiden Agenten, da die Bedingung $k_i = k_{i\max}$ für $i \geq 2$ gilt. $(n - k_{i-1})/2$ ist somit der längere Abstand halbiert, was $k_i = k_{i\max},\ i \geq 2$ fordert. Somit hängen die Gesamtkosten nur von k_1 ab.
Im Folgenden wird gezeigt, dass die Gesamtkosten maximal werden, wenn k_1 maximal wird. k_i kann dargestellt werden, als

$$k_i = \sum_{l=1}^{i-1} \frac{(-1)^{l-1}}{2^l} n + \left(-\frac{1}{2}\right)^{i-1} k_1,\ i \geq 2 \tag{1.2}$$

Beweis (vollständige Induktion):
Induktionsvoraussetzung: $i = 2$

$$k_2 = \frac{n - k_1}{2} = \frac{n}{2} - \frac{k_1}{2} = \frac{1}{2}n + \left(-\frac{1}{2}\right)^1 k_1$$

Induktionsbehauptung: Obige Gleichung gilt für ein $i \in \mathbb{N}$. Also gilt sie auch für jedes $i + 1$:

$$k_{i+1} = \sum_{l=1}^{i} \frac{(-1)^{l-1}}{2^l} n + \left(-\frac{1}{2}\right)^i k_1$$

Induktonsschritt:

$$\begin{aligned}
k_{i+1} &= \frac{n - k_i}{2} \\
&= \frac{n}{2} - \frac{n}{2} \sum_{l=1}^{i-1} \frac{(-1)^{l-1}}{2^l} - \frac{\left(-\frac{1}{2}\right)^{i-1} k_1}{2} \\
&= \frac{n}{2} - \frac{n}{2} \sum_{l=1}^{i-1} \frac{(-1)^{l-1}}{2^l} + \left(-\frac{1}{2}\right)^i k_1 \\
&= \frac{n}{2} - n \sum_{l=1}^{i-1} \frac{(-1)^{l-1}}{2^{l+1}} + \left(-\frac{1}{2}\right)^i k_1 \\
&= \frac{n}{2} + n \sum_{l=1}^{i-1} \frac{(-1)^{l}}{2^{l+1}} + \left(-\frac{1}{2}\right)^i k_1 \\
&= \sum_{l=0}^{i-1} \frac{(-1)^{l}}{2^{l+1}} n + \left(-\frac{1}{2}\right)^i k_1
\end{aligned}$$

Indexverschiebung:

$$= \sum_{l=1}^{i} \frac{(-1)^{l-1}}{2^l} n + \left(-\frac{1}{2}\right)^i k_1$$

Aus Gleichung 1.2 folgt, dass für alle i nur $(-1/2)^{i-1} k_1$ von k_1 beeinflusst wird. Der nicht von k_1 abhängige Teil muss nicht beachtet werden, da er konstant ist. Die Summe aller von k_1 abhängigen Teile muss maximal werden, damit die Gesamtkosten maximal werden. Diese Summe wird mit $k_1 - Gesamtkosten$ bezeichnet:

$$k_1 - Gesamtkosten = \sum_{i=2}^{m} \left(-\frac{1}{2}\right)^{i-1} k_1 + k_1 \tag{1.3}$$

$$= \sum_{l=1}^{m} \left(-\frac{1}{2}\right)^{l-1} k_1 \tag{1.4}$$

Da $\sum_{l=1}^{m} (-1/2)^{l-1}$ für alle $m \in \mathbb{N}$ größer Null ist, werden die $k_1 - Gesamtkosten$ genau dann maximal, wenn k_1 maximal ist. Es folgt $k_1 = k_{1\,\max}$. □

Lemma 4. *Sei eine Taskfolge $t_1, t_2, \ldots, t_m$ gegeben, wobei vor Eintreffen des ersten Tasks beide Helfer auf verschiedenen Positionen weilen. Weiterhin gelte $k_i = k_{i\,\max},\ i \geq 2$. Die Gesamtkosten werden maximal, wenn $k_1 = k_{1\,\max}$.*

Beweis. Sei d der kürzere Abstand beider Helfer, bevor der erste Task eintrifft. Es werden die Indizes für die Taskkosten verschoben. Also sind die Kosten von t_i gleich k_{i-1}. Es folgt:

$$0 \leq k_0 \leq \frac{n-d}{2}$$

$$k_1 = \frac{k_0 + d}{2}$$

oder:

$$k_1 = \frac{n - (k_0 + d)}{2}$$

Je nach der Größe von k_0 gilt die erste oder zweite Gleichung für k_1. Es gilt weiterhin die Rekursionsgleichung 1.1. Ab k_2 ergeben sich die Kosten identisch zum Fall in Lemma 3. Also gilt Formel 1.4:

$$k_1 - Gesamtkosten = \sum_{l=1}^{m} (-1/2)^{l-1} k_1$$

Es können zwei Fälle unterschieden werden.

Fall 1: $0 \leq k_0 \leq (n/2) - d$
Es kommt die zweite Gleichung von k_1 zur Anwendung, da diese die längere Distanz beider Agenten halbiert.

$$k_1 = \frac{n - (k_0 + d)}{2}$$

Somit:

$$k_1 - Gesamtkosten = \sum_{l=1}^{m} \left(-\frac{1}{2}\right)^{l-1} \frac{n - (k_0 + d)}{2}$$

Die $k_0 - Gesamtkosten$ sind analog zu den $k_1 - Gesamtkosten$ definiert. Um diese zu erhalten werden die von k_0 unabhängigen Summanden entfernt und k_0 addiert:

$$\begin{aligned} k_0 - Gesamtkosten &= \sum_{l=1}^{m} \frac{-\left(-\frac{1}{2}\right)^{l-1} k_0}{2} + k_0 \\ &= \sum_{l=1}^{m} \left(-\frac{1}{2}\right)^{l} k_0 \end{aligned}$$

Fall 2: $n/2 - d < k_0 \leq (n - d)/2$
Es kommt die erste Gleichung von k_1 zum Einsatz, da jetzt diese die längere Distanz beider Agenten halbiert.

$$k_1 = \frac{k_0 + d}{2}$$

Somit:

$$k_1 - Gesamtkosten = \sum_{l=1}^{m} \left(-\frac{1}{2}\right)^{l-1} \frac{k_0 + d}{2}$$

Daraus folgt:

$$\begin{aligned} k_0 - Gesamtkosten &= \sum_{l=1}^{m} \frac{\left(-\frac{1}{2}\right)^{l-1} k_0}{2} + k_0 \\ &= \sum_{l=1}^{m} \frac{(-1)^{l-1}}{2^l} k_0 + k_0 \end{aligned}$$

Es bleibt zu klären, für welchen Wert von k_0 zwischen 0 und $(n-d)/2$ die Kosten maximal werden. Für die Reihe aus Fall 1 ergeben sich die von k_0 abhängigen Kosten als $k_0 - (k_0/2) + (k_0/4) - \ldots$.
Die von k_0 abhängigen Kosten aus Fall 2 ergeben sich als $k_0 + (k_0/2) - (k_0/4) + (k_0/8) - \ldots$. Es ist sofort zu erkennen, dass die zweite Summe größer ist. Es befinden sich am Anfang zwei positive Summanden und k_0 ist in Fall 2 größer als in Fall 1. Die $k_0 - Gesamtkosten$ von Fall 2 werden genau dann maximal, wenn $k_0 = k_{0\,\max}$, da $\sum_{l=0}^{m} (-1)^{l-1}/2^l + 1$ immer positiv ist.
Die Verwendung der ursprünglichen Indizes führt zu $k_1 = k_{1\,\max}$ □

Lemma 5. *Sei $t_1, t_2, \ldots, t_m$ eine Folge von Tasks mit maximalen kosten. Dann sind alle Teilfolgen $t_i, t_{i+1}, \ldots, t_m,\ 0 \leq i \leq m$ Taskfolgen mit maximalen Kosten.*

Beweis. Annahme die hintere Teilfolge $t_i, t_{i+1}, \ldots, t_m$ hätte keine maximalen Kosten. Dann könnte man diese durch eine Teilfolge maximaler Länge ersetzen und hätte höhere Gesamtkosten. Widerspruch zur Annahme. □

Satz 6. *Die Kosten einer Folge $t_1, t_2, \ldots, t_m$ von Tasks seien maximal. Dann folgt, dass $k_i = k_{i\,\max},\ 0 \leq i \leq m$. D.h. jeder eintreffende Task halbiert die längere Distanz zwischen beiden Helfern.*

Beweis. Induktionsvoraussetzung: Aus Lemma 2 folgt, dass $k_m = k_{m\,\max}$.

Induktionsbehauptung: Wenn die letzten s Tasks der Taskfolge $t_1, t_2, \ldots, t_m$ die Kosten $k_i = k_{i\,\max}$ aufweisen, so weisen auch die letzten $s+1$ Tasks die Kosten $k_i = k_{i\,\max}$ auf.

Induktionsschritt: Aus Lemma 5 folgt, dass die Teilfolge der letzten $s+1$ Tasks maximale Kosten verursacht. Aus Lemma 4 folgt somit, dass auch $k_{m-(s+1)} = k_{m-(s+1)\,\max}$.
Dieser induktive Vorgang setzt sich lässt sich fortsetzen bis Task t_1, sodass die Kosten jedes Tasks $k_{i\,\max}$ entsprechen. □

2.1.2 Maximale Kosten

Im vorigen Abschnitt wurde das Verfahren gezeigt, um Taskfolgen mit maximalen Kosten zu erzeugen. In diesem Abschnitt sollen die maximalen Kosten berechnet werden.

Satz 7. *Seien vor Eintreffen des ersten Tasks beide Agenten auf unterschiedlichen Positionen und kennzeichne d den kürzeren Abstand zwischen beiden.*

Dann berechnen sich die Kosten einer maximal teuren Taskfolge der Länge i nach folgender Gleichung:

$$K_i = \frac{1}{9}n\left(3i - \left(-\frac{1}{2}\right)^i + 1\right) + \frac{1}{3}d\left(\left(-\frac{1}{2}\right)^i - 1\right) \tag{1.5}$$

Beweis. (Vollständige Induktion)

Induktionsvoraussetzung: $i = 1$

$$K_1 = \frac{n-d}{2} = \frac{9}{9*2}n - \frac{3}{3*2}d$$

Induktionsbehauptung: Die Gleichung gilt für ein $i \in \mathbb{N}$. Also gilt sie auch für jedes $i+1$.

$$\begin{aligned} K_{i+1} &= \frac{1}{9}n\left(3\left(i+1\right) - \left(-\frac{1}{2}\right)^{i+1} + 1\right) + \frac{1}{3}d\left(\left(-\frac{1}{2}\right)^{i+1} - 1\right) \\ &= \frac{1}{9}n\left(3i - \left(-\frac{1}{2}\right)^{i+1} + 4\right) + \frac{1}{3}d\left(\left(-\frac{1}{2}\right)^{i+1} - 1\right) \end{aligned}$$

Induktionsschritt:

$$\begin{aligned} K_{i+1} &= K_i + k_{i+1} \\ &= \frac{1}{9}n\left(3i - \left(-\frac{1}{2}\right)^i + 1\right) + \frac{1}{3}d\left(\left(-\frac{1}{2}\right)^i - 1\right) + k_{i+1} \end{aligned} \tag{1.6}$$

Für k_{i+1} ergibt sich aus Gleichung 1.2 mit $k_1 = (n-d)/2$:

$$\begin{aligned} k_{i+1} &= n\sum_{l=1}^{i}\frac{(-1)^{l-1}}{2^l} + \left(-\frac{1}{2}\right)^i\frac{n-d}{2} \\ &= n\sum_{l=1}^{i}\frac{(-1)^{l-1}}{2^l} + \left(-\frac{1}{2}\right)^i\frac{n}{2} - \left(-\frac{1}{2}\right)^i\frac{d}{2} \\ &= n\sum_{l=1}^{i}\frac{(-1)^{l-1}}{2^l} - \left(-\frac{1}{2}\right)^{i+1}n - \left(-\frac{1}{2}\right)^i\frac{d}{2} \\ &= n\sum_{l=1}^{i+1}\frac{(-1)^{l-1}}{2^l} - \left(-\frac{1}{2}\right)^i\frac{d}{2} \\ &= n\sum_{l=0}^{i}\frac{(-1)^{l}}{2^{l+1}} - \left(-\frac{1}{2}\right)^i\frac{d}{2} \qquad (1.7) \\ &= \frac{1}{6}n\left(-\frac{1}{2}\right)^i + \frac{1}{3}n - \left(-\frac{1}{2}\right)^i\frac{d}{2} \qquad (1.8) \end{aligned}$$

Der Beweis über die Korrektheit des letzten Umformungsschrittes erfolgt im Anschluss. Es folgt aus Gleichung 1.6 und 1.8:

$$\begin{aligned}
K_{i+1} &= \frac{1}{9}n\left(3i - \left(-\frac{1}{2}\right)^i + 1\right) + \frac{1}{6}n\left(-\frac{1}{2}\right)^i + \frac{1}{3}n \\
&\quad + \frac{1}{3}d\left(\left(-\frac{1}{2}\right)^i - 1\right) - \left(-\frac{1}{2}\right)^i \frac{d}{2} \\
&= \frac{3}{9}ni - \frac{1}{9}n\left(-\frac{1}{2}\right)^i + \frac{1}{6}n\left(-\frac{1}{2}\right)^i + \frac{4}{9}n \\
&\quad + \frac{1}{3}d\left(-\frac{1}{2}\right)^i + \left(-\frac{1}{2}\right)^{i+1} d - \frac{1}{3}d \\
&= \frac{3}{9}ni + n\left(-\frac{1}{2}\right)^{i+1}\left(-\frac{1}{9} * (-2) + \frac{1}{6} * (-2)\right) + \frac{4}{9}n \\
&\quad + d\left(-\frac{1}{2}\right)^{i+1}\left(\frac{1}{3} * (-2) + 1\right) - \frac{1}{3}d \\
&= \frac{1}{9}n\left(3i - \left(-\frac{1}{2}\right)^{i+1} + 4\right) + \frac{1}{3}d\left(\left(-\frac{1}{2}\right)^{i+1} - 1\right)
\end{aligned}$$

□

Beweis. Per vollständiger Induktion wird bewiesen, dass 1.7 in 1.8 überführt werden kann. Es reicht zu zeigen, dass

$$n\sum_{l=0}^{i}\frac{(-1)^l}{2^{l+1}} = \frac{1}{6}n\left(-\frac{1}{2}\right)^i + \frac{1}{3}n \tag{1.9}$$

da $-(-1/2)^i * (d/2)$ auf beiden Seiten der Gleichung vorkommt.

Induktionsvoraussetzung: $i = 1$

$$n\sum_{l=0}^{1}\frac{(-1)^l}{2^{l+1}} = \frac{1}{2}n - \frac{1}{4}n = \frac{1}{4}n = -\frac{1}{12}n + \frac{1}{3}n$$

Induktionsbehauptung: 1.9 gilt für ein $i \in \mathbb{N}$. Also gilt die Gleichung auch für jedes $i + 1$:

$$n\sum_{l=0}^{i+1}\frac{(-1)^l}{2^{l+1}} = \frac{1}{6}n\left(-\frac{1}{2}\right)^{i+1} + \frac{1}{3}n$$

Induktionsschritt:

$$
\begin{aligned}
n\sum_{l=0}^{i+1}\frac{(-1)^l}{2^{l+1}} &= n\sum_{l=0}^{i}\frac{(-1)^l}{2^{l+1}} + \frac{(-1)^{i+1}}{2^{i+2}}n \\
&= \frac{1}{6}n\left(-\frac{1}{2}\right)^i + \frac{1}{3}n + \left(-\frac{1}{2}\right)^{i+1}\frac{1}{2}n \\
&= \left(-\frac{1}{2}\right)^{i+1} n\left(\frac{1}{6} * (-2) + \frac{1}{2}\right) + \frac{1}{3}n \\
&= \frac{1}{6}n\left(-\frac{1}{2}\right)^{i+1} + \frac{1}{3}n
\end{aligned}
$$

□

2.1.3 Durchschnittliche Kosten

Wenn die Tasks rein zufällig eintreffen, sind die entstehenden Kosten geringer als im schlechtesten Fall. Es ist zu klären, wie hoch die erwarteten Kosten pro Task ausfallen, wenn alle Tasks zufällig und gleichverteilt auf dem Kreis eintreffen. Zufallsgröße K bezeichnet die entstehenden Kosten durch einen Task. Um den Erwartungswert dieser Zufallgröße zu berechnen benötigt man ihre Verteilungsfunktion. Die Verteilungsfunktion von K gibt an mit welcher Wahrscheinlichkeit die Kosten eines Tasks unter einem bestimmten Wert k liegen. Die Wahrscheinlichkeit, dass der Wert der Zufallsgröße K kleiner als ein bestimmter Wert k ist, wird durch die zweidimensionale geometrische Wahrscheinlichkeit berechnet [Clauß et al., 2004].

Siehe Abbildung 2.1. Im Koordinatenursprung befindet sich Agent 1. Auf der x-Achse wird der Abstand des Agenten 2 von Agent 1 abgetragen, auf der y-Achse der Abstand des eintreffenden Tasks zu Agent 1. Jeder Punkt in diesem Koordinatensystem repräsentiert zwei Abstände Task und Agent 2 zu Agent 1. Die maximale Distanz zwischen den Agenten bzw. Agent und Task beträgt n, da die Entfernung von Agent 1 auf dem Kreis nur in eine Richtung ermittelt wird. Der mögliche Bereich ist als Quadrat markiert. Die Wahrscheinlichkeit, dass der Abstand eines Tasks zu einem der beiden Agenten kleiner als Wert k ist, kann als Verhältnis der Flächen in diesem Quadrat interpretiert werden. Der Task muss entweder in dem unteren Streifen, den die untere gestrichelte Linie mit der x-Achse einschließt bzw. in dem oberen Streifen, den die obere gestrichelte Linie mit der Oberseite des Quadrats einschließt eintreffen, dann ist sein Abstand zu Agent 1 kleiner als k. Es gibt den oberen und unteren Streifen, da der Task rechts oder links von Agent 1 eintreffen kann. Eine weitere Möglichkeit ist, dass der Task in dem Diagonalen Streifen, der von beiden

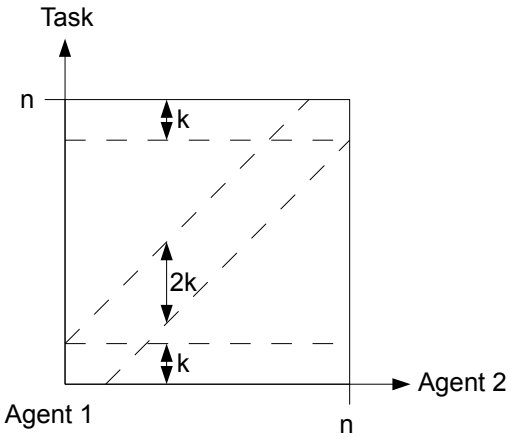

Abbildung 2.1: Geometrische Wahrscheinlichkeit

diagonalen gestrichelten Linien eingeschlossen wird, eintrifft. Dann ist sein Abstand zu Agent 2 kleiner als z. Wenn z.B. Agent 2 den Abstand $n/2$ zu Agent 1 aufweist, muss der Task auf der y-Achse in den Intervallen $[0;k]$, $[n-k;n]$ oder $[n/2-k;n/2+k]$ liegen. In dem Bereich, wo sich die genannten Streifen schneiden ist der Abstand des Taks zu beiden Agenten kleiner als k. Die Wahrscheinlichkeit, dass der Abstand vom Task zu den Agenten kleiner als k ist, ergibt sich aus dem Verhältnis des oberen, unteren und diagonalen Streifens zur Gesamtfläche des Quadrats.

Es ergibt sich folgenden Gleichung für die Verteilungsfunktion:

$$\begin{aligned} F_K(k) = P\left(K \leq k\right) &= \frac{2nk + \left(n-2k\right)n - \left(n-2k\right)^2}{n^2} \\ &= -\frac{4}{n^2}k^2 + \frac{4}{n}k \end{aligned} \tag{1.10}$$

Um den Erwartungswert zu berechnen muss die Dichtefunktion $f(k)$ berechnet werden. Dies geschieht durch Differentiation der Verteilungsfunktion.

$$f(k) = F'_K(k) = \frac{4}{n} - \frac{8}{n^2}k$$

Der Erwartungswert EK von K berechnet sich wie folgt:

$$\begin{aligned} EK &= \int k f(k)\, dk \\ &= \int_0^{n/2} k(\frac{4}{n} - \frac{8}{n^2}k)dk \\ &= \left[-\frac{8}{3n^2}k^3 + \frac{2}{n}k^2\right]_0^{n/2} \\ &= -\frac{8}{3n^2}\left(\frac{n}{2}\right)^3 + \frac{2}{n}\left(\frac{n}{2}\right)^2 \\ &= -\frac{1}{3}n + \frac{n}{2} \\ &= \frac{1}{6}n \end{aligned}$$

Im Durchschnitt kostet ein Task also 1/6 der Kreislänge.

2.2 Partitionierter Kreis

Im Vergleich zum unpartitionierten Kreis soll untersucht werden, welche Kosten sich ergeben, wenn man den Kreis partitioniert. Es wird der Fall untersucht, dass der Kreis in zwei gleich große Hälften geteilt wird. In jeder Hälfte befindet sich ein Agent und es ist keinem Agenten möglich, seine Hälfte zu verlassen. Damit zerfällt der Kreis in zwei Strecken der Länge $n/2$. Bevor ein Task auf einer konkreten Position eintreffen kann, muss entschieden werden, in welcher Partition das geschieht. Es wird angenommen, dass dafür keine Kosten anfallen.

2.2.1 Maximale Kosten

Da es sich um zwei unabhängige Halbkreise handelt, lassen sich maximal teure Taskfolgen einfach generieren. Um maximale Kosten zu erzeugen pendelt ein Agent immer zwischen den zwei Endpunkten des Halbkreises. Um auch die Kosten des ersten Tasks zu maximieren, muss dieser an dem Endpunkt eintreffen, der am weitesten von einem Task entfernt ist. Sei d der längste Abstand eines Agenten von einem seiner Endpunkte. Dann können die maximalen Kosten K_i nach Schritt i berechnet werden.

$$K_i = d + (i-1)\,\frac{n}{2}$$

Für den ersten Schritt fallen Kosten d an, für jeden weiteren Schritt Kosten $n/2$.

2.2.2 Durchschnittliche Kosten

Satz 8. *Es treffen Tasks zufällig und gleichverteilt auf einem Halbkreis der Länge $n/2$ ein, der mit einem Agenten besetzt ist. Die durchschnittlichen Kosten pro Task sind $n/6$.*

Beweis. Der Abstand eines Agenten zu einem Endpunkt der Strecke sei durch die Zufallsgröße X bezeichnet. Da alle Tasks hintereinander gleichverteilt und zufällig eintreffen und der Agent sich zu diesen Tasks bewegt, ist X eine Zufallsgröße. Die Abbildung

$$\begin{aligned} g(X) &= \frac{x}{\frac{n}{2}} * \frac{x}{2} + \frac{\frac{n}{2} - x}{\frac{n}{2}} * \frac{\frac{n}{2} - x}{2} \\ &= \frac{2}{n}x^2 - x + \frac{n}{4} \end{aligned}$$

stellt die erwarteten Kosten von einem zufällig eintreffenden Task dar, für ein bestimmtes x. Es muss der Erwartungswert über alle x berechnet werden. Der Erwartungswert einer Abbildung einer stetigen Zufallsgröße berechnet durch die Transformationsformel [Dehling and Haupt, 2003, S.174 Transformationsformel].

$$E(g(X)) = \int g(x) f_X(x)\, dx$$

Die Verteilungsfunktion von X lautet $F_X(x) = P(X \leq x) = (2/n)x$, da alle Tasks gleichverteilt auf einer Strecke der Länge $n/2$ eintreffen. Also ist die Dichtefunktion gegeben durch $f_X(x) = F'_X(x) = 2/n$. Für den Erwartungswert von $g(X)$ folgt:

$$\begin{aligned} E(g(X)) &= \int_0^{n/2} \frac{2}{n}\left(\frac{2}{n}x^2 - x + \frac{n}{4}\right) dx \\ &= \left[\frac{4}{3n^2}x^3 - \frac{1}{n}x^2 + \frac{1}{2}x\right]_0^{n/2} \\ &= \frac{1}{6}n \end{aligned}$$

□

Die erwarteten Kosten beim partitionierten Kreis sind also identisch zu den erwarteten Kosten beim unpartitionierten Kreis, besetzt mit zwei Agenten. Einen von beiden Halbkreisen zu betrachten ist ausreichend. Da sich auf auf dem anderen Halbkreis der Agent ebenfalls an zufälliger Position befindet und Tasks auch dort gleichverteilt eintreffen, ergeben sich die selben erwarteten Kosten pro Task.

2.3 Vergleich

Beide Modelle weisen die gleichen druchschnittlichen Kosten pro Task auf. Wenn lange Taskfolgen ohne Unterbrechung bearbeitet werden sollen, ist es egal, ob es auf einem partitionierten oder einem unpartitionierten Kreis stattfindet. Es würden sich bei Testsystemen beider Modelle nahezu die gleichen durchschnittlichen Kosten pro Task ergeben und damit auch nahezu die gleichen Gesamtkosten. Erst wenn die maximalen Kosten einer Taskfolge eine Rolle spielen, ist eine Unterscheidung der Modelle wichtig.

Um die Maximalkosten zu Vergleichen bietet es sich an, bei beiden Modellen die Durchschnittlichen Kosten pro Task im worst case zu bestimmen. Dieser berechnet sich beim unpartitionierten Kreis durch folgenden Grenzwert

$$\begin{aligned}
\lim_{i\to\infty} \frac{K_i}{i} &= \lim_{i\to\infty} \frac{\frac{1}{9}n\left(3i - \left(-\frac{1}{2}\right)^i + 1\right) + \frac{1}{3}d\left(\left(-\frac{1}{2}\right)^i - 1\right)}{i} \\
&= \lim_{i\to\infty} \frac{\frac{1}{3}ni - \frac{1}{9}n\left(-\frac{1}{2}\right)^i + \frac{1}{9}n + \frac{1}{3}d\left(-\frac{1}{2}\right)^i - \frac{1}{3}d}{i} \\
&= \frac{1}{3}n + \lim_{i\to\infty} \frac{-\frac{1}{9}n\left(-\frac{1}{2}\right)^i + \frac{1}{9}n + \frac{1}{3}d\left(-\frac{1}{2}\right)^i - \frac{1}{3}d}{i} \\
&= \frac{1}{3}n
\end{aligned}$$

Beim partitionierten Kreis ergeben sich die asymptotischen Kosten aus

$$\begin{aligned}
\lim_{i\to\infty} \frac{K_i}{i} &= \lim_{i\to\infty} \frac{d + (i-1)\frac{n}{2}}{i} \\
&= \frac{1}{2}n + \lim_{i\to\infty} \frac{d - \frac{n}{2}}{i} \\
&= \frac{1}{2}n
\end{aligned}$$

Während die Kosten beim unpartitionierten Kreis pro Task im schlechtesten Fall immer kleiner $n/2$ sind, betragen sie beim partitionierten immer $n/2$, bis auf den ersten Task. Im Durchschnitt ist beim worst case des unpartitionierten Kreis jeder Task $n/6$ preiswerter als beim partitionierten Graph. Wenn in einem System kurze Taskfolgen bearbeitet werden sollen, ist es günstiger den unpartitionierten Kreis zu wählen. Die Kosten für eine kurze Taskfolge können sich dem worst case annähern, oder diesem entsprechen. Diese Maximalkosten sind beim unpartitionierten Kreis deutlich geringer als beim partitionierten, sodass die Kostenobergrenze für eine kurze Taskfolge beim unpartitionierten Kreis tiefer liegt. Abbildung 2.2 stellt die Kosten grafisch dar. Es wurde angenommen, dass zu Beginn beide Agenten auf dem unpartitionierten Kreis den

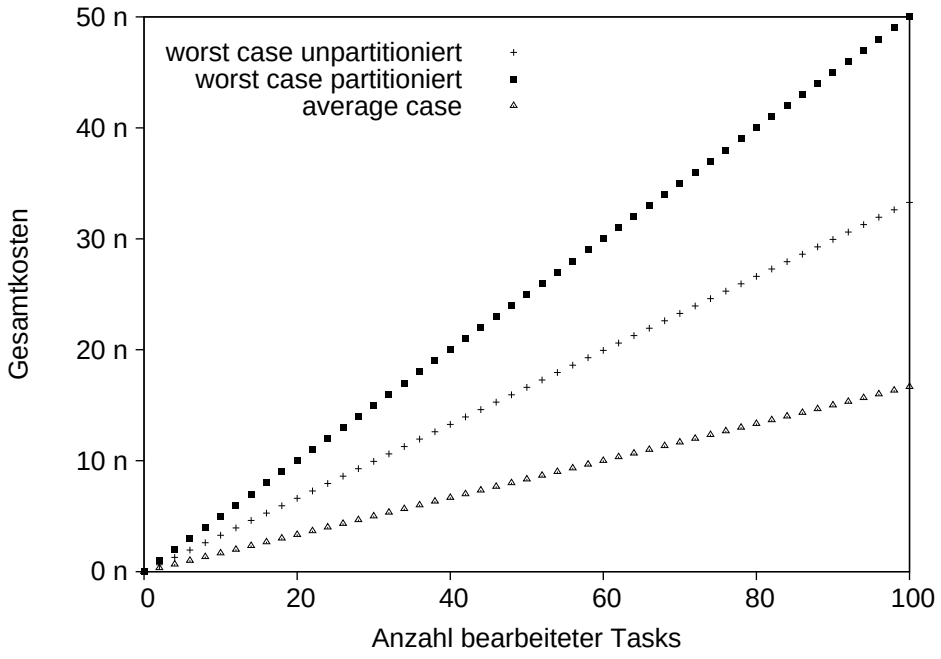

Abbildung 2.2: Vergleich der worst-case- und erwarteten Durchschnittskosten

Abstand $n/2$ haben und auf dem partitionierten Kreis beide Agenten in der Mitte ihres Halbkreises starten.

Kapitel 3

Feste zeitliche Taskabstände

In diesem Kapitel soll untersucht werden, wie sich der unpartitionierte und partitionierte Kreisgraph verhalten, wenn Tasks in festen zeitlichen Abständen eintreffen. Die zugrunde liegenden Modelle sind die gleichen. Auf dem unpartitionierten Kreis befinden sich zwei Agenten, auf dem partitionierten auch, je einer pro Partition. Je nachdem wie lang das Zeitintervall ist, nach dem der nächste Tasks eintrifft, kann es passieren, dass einer oder beide Agenten noch mit der Lösung eines vorher eingetroffenen Tasks beschäftigt sind. Zuerst wird das Verhalten untersucht, wenn diese Tasks einfach abgelehnt werden. Dass heißt, wenn ein Task eintrifft während alle Agenten beschäftigt sind, wird er verworfen. Im Anschluss werden Strategien untersucht, die jeden Task berücksichtigen.

3.1 Unpartitionierter Kreis

Zuerst werden die entstehenden Kosten auf dem unpartitionierten Kreis untersucht. Damit sich eine Betrachtung des Problems vom vorigen Fall optimaler zeitlicher Taskabstände unterscheidet, muss die Zeit t zwischen den Tasks kleiner als $n/2$ sein. Wäre das nicht der Fall würde jeder neue Task erst dann eintreffen wenn beide Agenten ruhen, da auf dem unpartitionierten Kreis keine Kosten größer als $n/2$ entstehen können. Damit würde sich im Verhalten kein Unterschied zu optimalen Taskabständen ergeben, außer dass evtl. zwischen zwei Tasks beide Agenten für kurze Zeit ruhen.

3.1.1 Maximal teure Taskfolgen

Durch konstante Taskzwischenzeiten entstehen im worst case teurere Taskfolgen. Es kann sein, dass ein Agent noch in Bewegung ist, während der nächste Task eintrifft. Dadurch ergeben sich die Kosten, als wäre nur ein Agent auf dem Kreis. Im schlechtesten Fall trifft ein Task dann genau gegenüber mit den

Kosten $n/2$ ein.

Sei d der kürzere Abstand beider Agenten. Wenn zu Beginn gilt

$$t < \frac{n-d}{2} \tag{1.1}$$

ergibt sich eine maximal teure Taskfolge mit folgenden Kosten.

Satz 9. *Die Maximalkosten K_i nach Task i, wenn obige Bedinung gilt, betragen*

$$K_i = \frac{n-d}{2} + (i-1)\,\frac{n}{2} \tag{1.2}$$

Beweis. Der erste Task verursacht Kosten $(n-d)/2$, das sind die maximalen Kosten, die im ersten Schritt möglich sind, da beide Agenten zur Verfügung stehen. Nach Zeit t trifft der nächste Task ein. Durch Bedingung 1.1 ist der eine Agent noch mit der Lösung des ersten Tasks beschäftigt, während der nächste eintrifft. Damit ergeben sich für den nächsten Task die Kosten $n/2$, also die maximal möglichen. Für jeden weiteren Task der angenommen wird steht immer nur ein Agent zur Verfügung. Der jeweils andere Agent ist noch beschäftigt, denn beide Agenten benötigen für jeden Task genau gleich lang, allerdings zeitlich verschoben. So ergeben sich ab dem zweiten Schritt die Kosten $n/2$, das macht den zweiten Summanden aus. □

Allerdings gilt Bedingung 1.1 nicht immer, abhängig vom gewählten t und d. In diesem Fall sind die Maximalkosten nicht so einfach zu bestimmen, da nach dem ersten Task wieder beide Agenten ruhen und nicht in obigen Rhythmus übergehen. Eintreffende Tasks könnten die Distanz d so lange halbieren, bis Bedingung 1.1 erfüllt wird und anschließend erzeugt wieder jeder Schritt die Kosten $n/2$. Allerdings stellt dieses Vorgehen für kurze Taskfolgen nicht den worst case dar, denn die ersten Schritte erzeugen sehr geringe Kosten. Die exakte Berechnung der Maximalkosten wenn Bedingung 1.1 nicht gilt wurde nicht vorgenommen. Allerdings sind die Maximalkosten in jedem Fall kleiner als die Kosten, die Gleichung 1.2 angibt. Gleichung 1.2 stellt eine Obergrenze dar, da jeder Schritt die maximal möglichen Kosten erzeugt. Im folgenden werden die Maximalkosten mit dieser Gleichung abgeschätzt.

3.1.2 Durchschnittliche Taskfolgen

Die durchschnittlichen Taskkosten im Modell mit festen Taskzwischenzeiten zu berechnen erfordert mehr Aufwand, als bei optimalen zeitlichen Taskabständen. Die erwarteten Kosten für einen eintreffenden Task unterscheiden sich je nachdem, ob beide Agenten in Ruhe sind oder nur einer. Wenn sich beide Agenten bewegen wird der Task abgewiesen, deswegen fließt dieser Fall nicht

in die Kostenberechnung ein.

Die Tasks befinden sich entweder beide in Ruhe oder nur einer. Gerade bei kurzen Taskzwischenzeiten werden sehr lange Taskfolgen bearbeitet, in denen jeweils nur ein Agent vorhanden ist, weil der andere beschäftigt ist. Wenn beide Agenten in Ruhe sind bezeichnet die Zufallsgröße T die Anzahl von Tasks, die bearbeitet werden, bis beide Agenten wieder gleichzeitig in Ruhe verharren. Es wird der Erwartungswert ET von T berechnet. Der erwartete Wert ist abhängig vom gewählten t und n.

Beide Agenten befinden sich nach einem Task wieder in Ruhe, wenn der eintreffende Task Kosten kleiner t verursacht. Aus der Verteilungsfunktion 1.10 folgt die Wahrscheinlichkeit

$$P(K < t) = P_1 = -\frac{4}{n^2}t^2 + \frac{4}{n}t$$

Beide Agenten befinden sich nach zwei Tasks wieder in Ruhe, wenn die Kosten des ersten Tasks größer t sind und die Kosten des zweiten genau die Höhe haben, dass beide Agenten im gleichen Zeitintervall ihre Tasks beenden. Dass heißt, der zweite Task muss auf dem richtigen Abschnitt der Länge t eintreffen. Es ergibt sich die Wahrscheinlichkeit

$$P_2 = \left(1 + \frac{4}{n^2}t^2 - \frac{4}{n}t\right)\frac{2t}{n}$$

Der erste Faktor ist die Wahrscheinlichkeit, dass die Kosten des ersten Tasks größer t sind, der zweite Faktor, dass Task zwei rechts oder links von Agent 2 im richtigen t-langen Abschnitt auf dem Kreis eintrifft.

Beide Agenten befinden sich nach drei Tasks wieder in Ruhe, wenn der erste Task länger als t benötigt, der zweite nicht auf den zwei entsprechenden t-langen Abschnitten eintrifft, aber dafür der Dritte.

$$P_3 = \left(1 + \frac{4}{n^2}t^2 - \frac{4}{n}t\right)\left(1 - \frac{2t}{n}\right)\frac{2t}{n}$$

Dies setzt sich fort. Für jede weitere Wahrscheinlichkeit muss $(1 - 2t/n)$ multipliziert werden. Dieser Faktor drückt aus, dass ein Task auf dem falschen Abschnitt eintrifft und beide Agenten nicht gleichzeitg ihre Tasks lösen. Folgende Gleichung berechnet den Erwartungswert ET, indem die Taskanzahl mit der entsprechenden Wahrscheinlichkeit multipliziert wird.

$$\begin{aligned} ET &= \sum_{i=0}^{\infty} iP_i \\ &= \frac{-4}{n^2}t^2 + \frac{4}{n}t + \frac{2t}{n}\left(1 + \frac{4}{n^2}t^2 - \frac{4}{n}t\right)\sum_{i=0}^{\infty}(i+2)\left(1 - \frac{2t}{n}\right)^i \end{aligned} \tag{1.3}$$

Es ist der Grenzwert der unendlichen Reihe zu berechnen. Setze $x = (1 - 2t/n)$. Es gilt $0 < x < 1$, da $0 < t < n/2$.

$$\begin{aligned}\sum_{i=0}^{\infty} (i+2) \left(1 - \frac{2t}{n}\right)^i &= \sum_{i=0}^{\infty} (i+2)\, x^i \\ &= \sum_{i=0}^{\infty} (i+1)\, x^i + \sum_{i=0}^{\infty} x^i\end{aligned}$$

Der Grenzwert der zweiten unendlichen Reihe (geometrische Reihe) lautet

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \tag{1.4}$$

Dieser wird zur Berechnung des Grenzwertes der ersten unendlichen Reihe genutzt.

$$\begin{aligned}f(x) &= \frac{1}{1-x} = \sum_{i=0}^{\infty} x^i \\ f'(x) &= \frac{1}{(1-x)^2} = \sum_{i=0}^{\infty} i x^{i-1} = \sum_{i=0}^{\infty} (i+1)\, x^i \end{aligned} \tag{1.5}$$

Damit folgt aus 1.4 und 1.5

$$\sum_{i=0}^{\infty} (i+2)\, x^i = \frac{1}{1-x} + \frac{1}{(1-x)^2}$$

Substitution von x:

$$\begin{aligned}&= \frac{n^2}{4t^2} + \frac{n}{2t} \\ &= \frac{n}{2t}\left(\frac{n}{2t} + 1\right)\end{aligned}$$

Eingesetzt in 1.3 ergibt sich der Erwartungswert ET:

$$\begin{aligned}ET &= \frac{-4}{n^2}t^2 + \frac{4}{n}t + \frac{2t}{n}\frac{n}{2t}\left(\frac{n}{2t} + 1\right)\left(1 + \frac{4}{n^2}t^2 - \frac{4}{n}t\right) \\ &= 1 - 2 + 2\frac{t}{n} + \frac{n}{2t} \\ &= 2\frac{t}{n} + \frac{n}{2t} - 1\end{aligned}$$

Der Erwartungswert ET ist so zu interpretieren, dass auf einen Task bei dem beide Agenten ruhen im Schnitt $ET - 1$ Tasks folgen, mit nur einem ruhenden

Agenten. Von ET muss ein Task abgezogen werden, da bei Eintreffen des ersten Tasks beide Agenten ruhen. Die durchschnittlichen Kosten, wenn beide Agenten zur Verfügung stehen wurden in 2.1.3 berechnet und betragen $n/6$. Die Durchschnittskosten, wenn ein Agent zur Verfügung steht, betragen $n/4$, da die Tasks gleichverteilt mit Kosten zwischen 0 und $n/2$ eintreffen. Auf einen Task mit den erwarteten Kosten $n/6$ kommen also im Schnitt $EZ - 1$ Tasks mit den erwarteten Kosten $n/4$. Multipliziert man die erwarteten Taskanzahlen mit den erwarteten Kosten und teilt durch die Gesamtanzahl an Tasks erhält man den Erwartungswert EK der Kosten pro Task.

$$EK = \frac{\frac{n}{6} + \left(2\frac{t}{n} + \frac{n}{2t} - 2\right)\frac{n}{4}}{2\frac{t}{n} + \frac{n}{2t} - 1}$$

3.2 Partitionierter Kreis

Umfangreichen Berechnungen beim partitionierten Kreis sind unnötig. Es ergeben sich die gleichen Kosten für maximale und durchschnittliche Taskfolgen wie bei optimalen zeitlichen Taskabständen auf dem partitionierten Kreis (siehe Abschnitte 2.2.1 und 2.2.2). Da beide Agenten voneinander unabhängige Strecken bearbeiten, können sie sich nicht gegenseitig beeinflussen. Es treten keine Effekte wie beim unpartitionierten Kreis auf. Allerdings kommt es durch das konstante Zeitintervall zwischen den Tasks zur Taskablehnung, sodass wie auch beim unpartitionierten Kreis nicht alle Tasks bearbeitet werden.

3.3 Vergleich

Im worst case verhalten sich beide Modelle ähnlich. Nur der erste Schritt kann sich unterscheiden. Wenn am Anfang auf dem unpartitionierten Kreis beide Agenten gegenüber stehen und auf dem partitionierten Kreis beide Agenten in der Mitte ihres Segments, ergeben sich auch für den ersten Schritt identische Kosten von $n/4$.
Die durchschnittliche Taskfolge unterscheidet sich bei beiden Modellen. Während auf dem partitionierten Kreis jeder Task im Schnitt $n/6$ kostet, sind die Kosten auf dem unpartitionierten Kreis pro Task abhängig von t. Um zu ermitteln zwischen welchen Werten sich diese Kosten bewegen werden die Grenzen betrachtet.

An der Grenze $t \to n/2$ ergibt sich

$$\begin{aligned}\lim_{t \to n/2} EK &= \frac{\frac{n}{6} + (1 + 1 - 2)\frac{n}{4}}{1 + 1 - 1} \\ &= \frac{n}{6}\end{aligned}$$

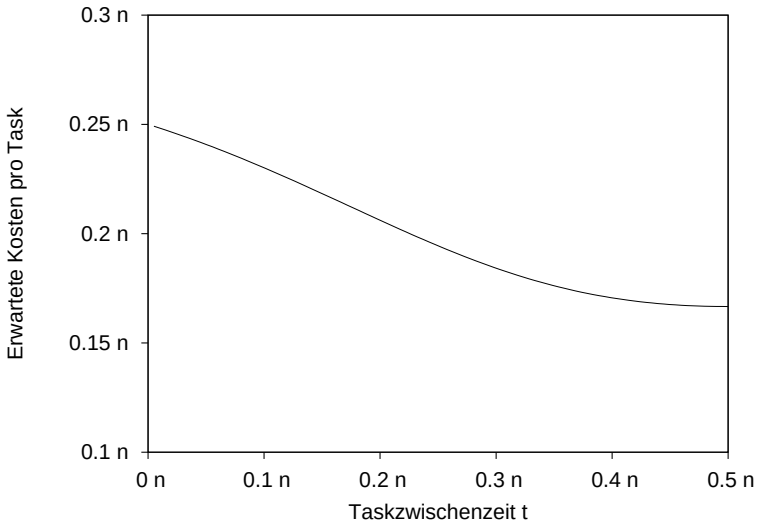

Abbildung 3.1: Durchschnittskosten des unpartitionierten Kreises in Abhängigkeit von t.

da für $t = n/2$ stets beide Agenten ruhen würden wenn der nächste Task eintrifft. Für $t \to 0$ ergibt sich der Grenzwert

$$\begin{aligned}\lim_{t\to 0} EK &= \lim_{t\to 0} \frac{\frac{1}{t}\left(\frac{nt}{6} + \frac{t^2}{2} + \frac{n^2}{8} - \frac{nt}{2}\right)}{\frac{1}{t}\left(2\frac{t^2}{n} + \frac{n}{2} - 1t\right)} \\ &= \frac{\frac{n^2}{8}}{\frac{n}{2}} \\ &= \frac{n}{4}\end{aligned}$$

da für $t = 0$ nie beide Agenten für genau einen Task zur Verfügung stehen könnten.

Die durchschnittlichen Kosten pro Task bewegen sich also zwischen $n/6$ und $n/4$. Der partitionierte Kreis ist in diesem Fall das bessere Modell. Die durchschnittlichen Kosten des unpartitionierten Kreises nähern sich denen des partitionierten bei langen Taskzwischenzeiten zwar an, dessen Durchschnittskosten werden aber nie erreicht.
Abbildung 3.1 veranschaulicht die erwarteten Kosten in Abhängigkeit von der Taskzwischenzeit t.

3.4 Keine Taskablehnung

Wenn alle Taks, die eintreffen, bearbeitet werden sollen, ist das vorherige Vorgehen nicht korrekt. Es ist nicht mehr zulässig, einen Task einfach zu verwerfen, wenn gerade beide Agenten beschäftigt sind. Alle Tasks, die nicht unmittelbar bearbeitet werden können, werden in einem Zwischenspeicher abgelegt. Dieser muss eine entsprechende Größe aufweisen, um bei zuvielen abgelehnten Tasks nicht überzulaufen. Um den Inhalt des Zwischenspeichers zu verkleinern bieten sich verschiedene Lösungsanätze an. Im Folgenden werden einige mögliche Strategien vorgestellt.

3.4.1 Lösen eines Zusatzttasks

Die erste betrachtete Möglichkeit ist, es zuzulassen, dass genau ein weiterer Task pro Zeitintervall gelöst wird. Wenn der aktuelle Task gelöst wurde, und es ist noch genug Zeit übrig bis der nächste Task eintrifft, kann ein weiterer aus dem Zwischenspeicher gelöst werden. Dazu muss sich ein Task im Zwischenspeicher befinden, der sich innerhalb der verbleibenden Zeit erreichen lässt. Dieses Modell ist nur erfolgreich, solange im Durchschnitt weniger als ein Task pro gelöstem Task abgelehnt wird. Ansonsten wächst die Zahl der zwischengespeicherten Tasks immer weiter an.

3.4.2 Lösen der Tasks “im Vorbeigehen”

Da für das eigentliche Ausführen der Tasks keine Kosten anfallen, wäre eine weitere Möglichkeit, alle zusätzlichen Tasks zu lösen, die auf der zurückzulegenden Strecke liegen. Muss sich ein Agent aufgrund eines angenommenen Tasks von Punkt A nach Punkt B bewegen, so befinden sich evtl. Tasks im Zwischenspeicher, die genau auf diesem Abschnitt liegen. Der Agent kann alle Tasks dieser Strecke im Zwischenspeicher bearbeiten, sobald er den entsprechenden Punkt passiert. Diese Möglichkeit profitiert stark von den idealisierten Vorannahmen. Man kann nicht davon ausgehen, dass durch das Lösen zahlreicher zusätzlicher Tasks keine weiteren Kosten entstehen. Außerdem hat diese Methode den Nachteil, dass je näher ein abgelehnter Task am Rand liegt, beim partitionierten Kreis, desto länger wird er ungelöst bleiben, da erst ein Task angenommen werden muss, der noch näher am Rand liegt. Wenn der Abstand zwischen abgelehntem Task und Rand gegen Null geht, so wird dieser evtl. niemals gelöst werden. Dieses Problem tritt nur auf dem partitionierten Kreis auf, da nur es nur dort Ränder des Konfigurationsraumes gibt.

3.4.3 Hybridmethode

Beide letztgenannte Methoden zur Vermeidung des Aufstauens von Tasks haben ihre Nachteile. Die erste Möglichkeit ist für zu kurze Taskintervalle nicht zu gebrauchen, da der Speicher überlaufen würde. Die zweite Strategie kann Tasks an den Rändern des Konfigurationsraumes nicht effizient lösen. Kombiniert man beide Möglichkeiten, heben sie ihre Nachteile gegenseitig auf. Ein Hybridvorgehen beider letztgenannter Methoden würde zulassen, dass Tasks "im Vorbeigehen" gelöst werden können. Außerdem, wenn der Zieltask erreicht wurde und es ist noch Zeit übrig, dass noch weitere Tasks gelöst werden. Auch der unpartitionierte Kreis profitiert von dieser Methode im Vergleich zu 3.4.2, da die verbleibende Restzeit genutzt wird.
In der Praxis wären einige Faktoren zu beachten, die die Effizienz dieses und der vorher genannten Verfahren deutlich schmälern würden. Der Agent müsste ständig mit dem Zwischenspeicher kommunizieren, um festzustellen, ob mit der aktuellen Konfiguration zusätzliche Tasks gelöst werden können. Die zusätzlich zu lösenden Tasks müssten zwischen Speicher und Agent ausgetauscht werden. Nach dem Lösen des eigentlichen Zieltasks müsste der Agent berechnen, ob in der verbleibenden Zeit noch zusätzliche Tasks erreicht werden können. Auch dazu ist Kommunikation mit dem Speicher vonnöten. Man kann nicht davon ausgehen, dass die gesamte Kommunikation mit dem Speicher und die Zusatzberechnungen keine Kosten verursachen.
Konkrete Kostenberechnungen waren nicht möglich, da die durchschnittliche Anzahl abgelehnter Tasks nicht ermittelt werden konnte.

3.4.4 Periodisches Wandern

Die letzte Möglichkeit mit den Tasks umzugehen ist die simpelste. Sie behält die Vorteile der vorigen Methode bei, aber das Vorgehen ist einfacher. Beide Agenten laufen einfach periodisch den gesamten Konfigurationsraum ab und lösen auf dem Weg alle zwischengespeicherten Tasks. Auf dem unpartitionierten Kreis stehen sich beide Agenten genau gegenüber, also durch $n/2$ getrennt und bewegen sich beide in die gleiche Richtung.

Satz 10. *Auf dem unpartitionierten Kreis bewegen sich beide Agenten in die gleiche Richtung. Die maximalen und durchschnittlichen Kosten werden minimal, wenn sich beide Agenten genau gegenüber stehen.*

Beweis. Der kürzere Abstand beider Agenten beträgt d und der längere $n-d$. Die Abstände zwischen beiden Agenten sind konstant, da sich beide mit der gleichen Geschwindigkeit rekonfigurieren. Die maximal möglichen Kosten sind der längere Abstand beider Agenten, also $n-d$. Dieser Abstand wird minimal für $d = n/2$.

Die durchschnittlichen Kosten eines eintreffenden Tasks berechnen sich fogendermaßen:

$$\begin{aligned} EK\,(d) &= \frac{d}{n}\frac{d}{2} + \frac{n-d}{n}\frac{n-d}{2} \\ &= \frac{1}{2n}d^2 + \frac{n^2 - 2nd + d^2}{2n} \\ &= \frac{1}{n}d^2 - d + \frac{n}{2} \end{aligned} \tag{4.6}$$

Die quadratische Gleichung 4.6 hat ihr Minimum bei $d = n/2$. Also sind auch die Durchschnittskosten minimal, wenn sich beide Agenten gegenüber stehen. Die Durchschnittskosten betragen $n/4$. □

Satz 11. *Die durchschnittlichen Kosten auf dem partitionierten Kreis betragen $n/3$ pro Task.*

Beweis. Sei x der Abstand des Agenten zu dem Endpunkt der Strecke, auf den sich der Agent zubewegt. x ist keine Zufallsgröße, da der Agent periodisch wandert. Die erwarteten Kosten $k(x)$ eines zufällig und gleichverteilt eintreffenden Tasks berechnen sich wie folgt:

$$\begin{aligned} k(x) &= \frac{x}{\frac{n}{2}} * \frac{x}{2} + \frac{\frac{n}{2} - x}{\frac{n}{2}} \left(\frac{\frac{n}{2} - x}{2} + 2x \right) \\ &= -\frac{2x^2}{n} + x + \frac{n}{4} \end{aligned}$$

Das erste Produkt stellt die Wahrscheinlichkeit, dass der Task auf x eintrifft multipliziert mit den erwarteten Kosten dar. Das zweite Produkt berechnet den Fall, dass der Task auf dem anderen Abschnitt $n/2 - x$ eintrifft. Zu den erwarteten Kosten muss $2x$ addiert werden, da der Agent erst bis zum Rand läuft, dann zurück, und dann erst im entsprechenden Abschnitt ist. Um die erwarteten Kosten EK über alle x zu erhalten muss über den Konfigurationsraum integriert werden und durch $n/2$ geteilt werden.

$$\begin{aligned} EK &= \frac{\int_0^{n/2} \left(-\frac{2}{n}x^2 + x + \frac{n}{4} \right)\, dx}{\frac{n}{2}} \\ &= \frac{2}{n} \int_0^{n/2} \left(-\frac{2}{n}x^2 + x + \frac{n}{4} \right)\, dx \\ &= \frac{2}{n} \left[\frac{-2}{3n}x^3 + \frac{1}{2}x^2 + \frac{n}{4}x \right]_0^{n/2} \\ &= \frac{n}{3} \end{aligned}$$

□

Die Durchschnittskosten sind als durchschnittliche Wartezeit zu interpretieren, bis der Agent beim Task eingetroffen ist. Beim periodischen Wandern ist der unpartitionierte Kreis im Vorteil, da es höchsten $n/2$ dauert, bis ein Task bearbeitet wird und jeder Task im Schnitt nur warten muss, bis die Strecke $n/4$ abgeschritten ist, statt $n/3$. Demnach kann der Zwischenspeicher kleiner ausfallen, da das Maximum an zwischengespeicherten Tasks kleiner ist als beim partitionierten Kreis.

Kapitel 4

Praxisrelevanz

Es konnte kein existierendes System gefunden werden, dass rekonfigurierbare Software- oder Hardware-Agenten auf einem Kreis rekonfiguriert. Die realen Konfigurationsräume sind deutlich komplexer, da die Agenten in der Lage sind wesentlich mehr Nachbarkonfigurationen, als zwei, zu erreichen. Ursprünglich sollten in dieser Arbeit verschiedene, auch komplexere Räume untersucht werden, jedoch erwies sich schon die Analyse des Kreises als sehr aufwendig, sodass die Barbeitung weiterer Räume den Rahmen gesprengt hätte.
Um den Konfigurationsraum eines Hardware-Agenten mit n rekonfigurierbaren Bits zu modellieren Bedarf es eines n-dimensionelen Hyperwürfels. Jeder Eckpunkt im Hyperwürfel entspricht genau einer Konfiguration. Zu jedem Eckpunkt gibt es genau einen Eckpunkt, der in n Schritten zu erreichen ist. Das ist die komplementäre Konfiguration, bei der jedes Bit invertiert wurde. Die Analyse eines solchen Konfigurationsraumes würde auf der untersten Ebene ansetzen. Allerdings ist zu bezweifeln, ob es überhaupt möglich ist, die Maximalkosten bzw. die erwarteten Durchschnittskosten zu berechnen. Vor allem wenn mehr als ein Agent beteiligt ist.
Für die meisten realen Systeme jedoch wäre der Hyperwürfel wahrscheinlich auch unangemessen, da die Anzahl verschiedener auftretender Tasks deutlich geringer ausfällt, als die Anzahl möglicher Konfigurationen. Es wird kein reales System geben, bei dem jede mögliche Konfiguration einem Task entspricht. Demnach würden viele Knoten des Hyperwürfels nie betreten werden, sodass die Berechnung von Durchschnitts- und Maximalkosten für solche Systeme keine Relevanz hätte. Systeme mit rekonfigurierbaren Agenten zur Taskabarbeitung haben wahrscheinlich alle individuelle Konfigurationsräume, in denen die Agenten sich bewegen. Je nach Anzahl der verschiedenen auftretenden Tasks und Konfigurationen, die von den Agenten durchlaufen werden, um sich von einem Task zum nächsten zu bewegen. Es müsste zu einem realen System erst der individuelle Konfigurationsraum ermittelt werden und anschließend könnten praxisrelevante Kostenberechnungen durchgeführt werden, sofern das möglich ist.

Zusammenfassung

Es wurden entstehende Kosten bei zufällig eintreffenden Tasks auf dem partitionierten und unpartitionierten Kreis, jeweils besetzt mit zwei Agenten, untersucht. Die entstehenden Durchschnittskosten für verschiedene zeitliche Abstände der Tasks, sowie die maximal möglichen Kosten waren von Interesse. Die Frage, ob sich ein partitionierter oder ein unpartitionierter Kreis performanter verhält konnte nicht eindeutig beantwortet werden. Die korrekte Wahl des Modells hängt stark davon ab, in welchen Zeitabständen die Tasks eintreffen und mit welchen Strategien die Agenten die Tasks bearbeiten. Treffen alle Tasks in optimalen Abständen ein, so Verhalten sich beide Modelle im Durchschnitt gleich, allerdings verursacht der partitionierte Kreis höhere Maximalkosten. Treffen Taks in festen Zeitabständen ein, sind die erwarteten Durchschnittskosten auf dem partitionierten Kreis geringer. Allerdings nur wenn Tasks abgelehnt werden dürfen, falls die Agenten gerade einen vorher eingetroffenen Task bearbeiten. Die maximal möglichen Kosten sind in beiden Modellen gleich. Müssen alle Tasks bearbeitet werden, z.B. indem Tasks zwischengespeichert werden und wandern die Agenten dazu den Rekonfigurationsraum periodisch ab, weist der unpartitionierte Kreis geringere Durchschnittskosten auf. Dadurch kann auch der Zwischenspeicher kleiner ausfallen als auf dem partitionierten Kreis. Welches Modell vorzuziehen ist hängt stark von den Bedingungen ab, unter denen die Taks eintreffen und mit welcher Strategie sie bearbeitet werden. Es lässt sich keine Prognose bezüglich komplexerer Konfigurationsräume abgeben, ob sich womöglich partitionierte Räume im Allgemeinen performanter verhalten, oder unparitionierte. Auch bei komplexeren Räumen wird es stark von den Bedingungen, unter denen die Tasks eintreffen, abhängen. Eine mathematische Berechnung der Kosten auf komplexeren Konfigurationsräumen dürfte sich jedoch deutlich schwieriger gestalten als im Falle des Kreises.

Literaturverzeichnis

[Clauß et al., 2004] Clauß, G., Finze, F., and Partzsch, L. (2004). *Grundlagen der Statistik*. Harri Deutsch Verlag.

[Dehling and Haupt, 2003] Dehling, H. and Haupt, B. (2003). *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Springer.

[Merkle et al., 2008] Merkle, D., Middendorf, M., and Scheidler, A. (2008). Self-adaptive worker-helper systems with self-organized task allocation. *Organic Computing*, pages 221–239.

[Muller-Schloer, 2004] Muller-Schloer, C. (2004). Organic computing-on the feasibility of controlled emergence. In *Hardware/Software Codesign and System Synthesis, 2004.*, pages 2–5. IEEE.

[Würtz, 2008] Würtz, R. (2008). *Organic computing*. Springer Verlag.

Abbildungsverzeichnis

Erklärung

"Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann".

Leipzig, 15.10.2012