Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Hausarbeit

Birger Kamp

Der wissende Aufzug

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Birger Kamp

Der wissende Aufzug

Hausarbeit eingereicht im Rahmen des Moduls Modellierung Dynamischer Systeme

im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer: Prof. Dr. Wolfgang Fohl

Eingereicht am: 05.07.16

Birger Kamp

Thema der Arbeit
Der wissende Aufzug

Stichworte
Aufzug, Fahrstuhl, Warteschlange, Scheduling

Kurzzusammenfassung
In mehrstockigen Gebäuden werden häufig Fahrstühle verwendet, um von einem Stockwerk in ein anderes Stockwerk zu gelangen. Ein Problem dabei ist, dass man meist nicht der einzige Nutzer des Fahrstuhls ist, sodass man sich die Ressource Fahrstuhl mit anderen Personen teilen muss. Dabei muss ein Fahrstuhl-Planungssystem die Reihenfolge der Stockwerk-Stops planen. Beim Warten auf den Fahrstuhl entsteht eine Wartezeit, die sich möglicherweise reduzieren lässt, wenn das Fahrstuhl-Planungssystem bereits von dem Fahrstuhl-Ruf weiß, bevor der Ruf überhaupt auftritt. Diese These wird von der folgenden Arbeit analyisiert und hinterfragt.

Inhaltsverzeichnis

1 Einleitung

Man nimmt es als selbstverständlich an, dass ein Fahrstuhl kommt, wenn man auf den *Rufen*-Knopf gedrückt hat. Tut man dies in Gebäuden, in denen ein System mehrere Fahrstühle koordiniert, hat man mit dem bloßen Knopfdrücken einen komplexen Prozess gestartet. Die folgende Arbeit verschafft einen Einblick in die Ablaufplanung eines Fahrstuhls und was es dabei zu beachten gibt. Außerdem wird eine These gestellt und mit einem einfach gehaltenen Modell beantwortet.

1.1 These der Arbeit

Ein einfacher Aufzug setzt sich erst dann in Bewegung, wenn er eine Anfrage erhalten hat. Jemand der den *Rufen*-Knopf drückt, muss also warten bis der Fahrstuhl ihn abholt. Fahrstuhlsysteme sollten grundsätzlich bemüht sein, diese Wartezeit möglichst kurz zu halten.

Daraus leitet sich die These dieser Arbeit ab: „Wenn das Fahrstuhl-System wüsste, dass zum Zeitpunkt x in der Zukunft, eine Anfrage auf dem Stockwerk a gestellt wird, dann müsste der Anfragesteller kürzer warten“. Ein Anwendungszenario wäre eine Hochschule bei der dem Fahrstuhlsystem bekannt ist, dass zur Pausenzeit um 9.30 Uhr auf Stockwerk 6 einige Personen zum Erdgeschoss fahren möchten.

1.2 Verwendete Metriken

Um die gestellte These anhand eines Modells evaluieren zu können, müssen einige Metriken bestimmt und im Verlauf der Ausarbeitung verglichen werden.

Die These selbst nennt die *Wartezeit* in als Metrik. Um die weiteren Auswirkungen der veränderten Ablaufplanung beurteilen zu können, werden außerdem der *Anfragen-Durchsatz* und die *Fahrtzeit* gemessen. Mit Fahrtzeit wird die Zeit bezeichnet, die ein aufgenommer Fahrgast im Fahrstuhl verbringt bis er auf seinem Ziel-Stockwerk aussteigt.

2 Grundlagen

Hinter der Planung der Fahrstühle stecken Theorien und algorithmische Probleme, die im Folgenden erläutert werden.

2.1 Scheduling-Theorie

Die theoretische Grundlage für eine Planung eines Fahrstuhl-Systems ist die Scheduling-Theorie. In der Scheduling-Theorie geht es laut Pinedo (2012) darum, dass der Zugriff auf eine Ressource zeitgesteuert auf die Anfragenden verteilt wird.

Ein Beispiel aus der Informatik für die Anwendung der Scheduling-Theorie sind Multi-Threading Prozessoren. Dabei sind die Threads diejenigen Objekte, die den Zugriff auf die Ressource Prozessor erfragen. Der Thread-Scheduler erlaubt jedem anfragenden Thread eine gewisse Zeitspanne den Prozessor zu benutzen. Anschließend erhält ein anderer Thread die Ressource für eine gewisse Zeit.

2.1.1 Online Probleme

Nach Manasse u. a. (1988) ist eine Fahrstuhlablaufplanung ein Problem, das in eine Unterart der Scheduling-Probleme einzuordnen ist, die *Online Probleme* genannt werden. Diese Probleme zeichnen sich dadurch aus, dass dem Scheduler zu einem Zeitpunkt nur eine Teilmenge aller anfallenden Anfragen bekannt ist. Zu jedem Zeitpunkt können weitere Anfragen dazu kommen. Daraus lässt sich folgern, dass der Scheduler nur bedingt optimal planen kann, da er nicht weiß welche Anfragen noch kommen werden.

2.1.2 Offline Probleme

Im Gegensatz zu den Online Problemen stehen die *Offline Probleme*. Diese sind ebenfalls eine Unterart der Scheduling-Probleme. Bei diesen Problemen sind dem Scheduler bereits alle anfallenden Anfragen bekannt, sodass der Scheduler optimal planen kann.

Ein Beispiel für diese Probleme sind Logistik-Unternehmen: Diese können bereits im Voraus die Fahrt der Lieferwagen für den nächsten Tag planen. Denn alle Lieferanfragen gehen bereits im Tag der Planung ein.

3 Modellierung

Das folgende Kapitel beschreibt wie das Modell erstellt wurde und die getroffenen Annahmen und Vereinfachungen.

3.1 Verwendete Tools

Die Modellierung wurde in der Programmiersprache Python vorgenommen. Diese eignet sich besonders für wissenschaftliche Zwecke, da es eine breite Auswahl an Libraries für diese Zwecke gibt.

Für die Simulation von fortschreitender Zeit und Events zu bestimmten Zeitpunkten wurde die Library *SimPy*[1] verwendet.

Um die Ergebnisse in Diagrammen zu visualisieren wurde die Library *matplotlib*[2] verwendet. Diese Library bietet eine MATLAB-ähnliche Schnittstelle an, sodass man ohne viel Zeitaufwand ein einfaches Linien-Diagramm konstruieren kann. Mit etwas mehr Aufwand sind auch komplexe 2D- und 3D-Diagramme möglich.

Außerdem wurde ein minimales Python-Interface in *curses*[3] geschrieben. Dies bietet zwar keinen Mehrwert in der Genauigkeit der Simulation, aber es bietet einen Überblick was während der Simulation passiert. Dadurch fallen einige Fehler eher auf als wenn man die Simulation blind ablaufen lässt. Das Interface unterstützt daher bei der Fehlerfindung und visuellen Verifizierung des Planungsablaufs.

3.2 Vereinfachungen und Annahmen

Dieses Modell wird sich auf das wesentliche des Fahrstuhl-Schedulings beschränken. Um den Aufwand der Modellierung im Rahmen einer Praktikumsaufgabe zu halten, wurden daher folgende Modellierungs-Entscheidungen getroffen:

1. Ein Fahrstuhl kann unendlich viele Personen beinhalten

[1] http://simpy.readthedocs.io/en/latest/
[2] http://matplotlib.org/
[3] https://docs.python.org/2/howto/curses.html

2. Es wird nicht beachtet ob und wieviele Personen bei einem Halt aussteigen
3. Obwohl *keine Personen* in den Fahrstühlen mitfahren, soll keine der Anfragen unnötig lange verzögert werden
4. Folgende Aktionen eines Fahrstuhls benötigen eine Zeiteinheit:
   - Türen öffnen
   - Türen schließen
   - Veränderung der Position um ein Stockwerk
5. Beim Rufen des Fahrstuhls kann der Fahrgast angeben, in welche Richtung er fahren möchte
6. Ein Fahrstuhlruf hat immer ein anderes Ziel-Stockwerk als das, auf dem der Ruf erfolgt
7. Es kann immer nur einen aktiven Fahrstuhlruf für ein Stockwerk geben

Die Vereinfachungen 1 und 2 sind auf den ersten Blick nicht relevant für die Planung der Aufzüge, in einigen Fällen ist es jedoch relevant. In der Realität kann ein Fahrstuhl nur endlich viele Personen aufnehmen. Sobald die konstruktionsbedingte maximale Personen-Anzahl eines Fahrstuhls überschritten wurde, geben moderne Fahrstühle ein Warnsignal, dass der Fahrstuhl zu voll beladen ist. In dem Fall müssen einige Personen aussteigen und auf den nächsten Fahrstuhl warten. Bezogen auf die Ablaufplanung hat dies den Effekt, dass aus einem Fahrstuhl-Ruf mehrere Rufe werden, da die wartenden Personen erneut den Ruf-Knopf betätigen. Dies ist jedoch ein Spezialfall und wird in diesem Modell nicht beachtet.

3.3 Das Modell

Das Wesentliche an diesem Modell ist, wann ein Aufzug in welchem Stockwerk hält und in welche Richtung er nach dem Halt weiterfährt. Abhängig davon werden dem Fahrstuhl Anfragen zugeordnet, die er zu bearbeiten hat.

Ein Fahrstuhl wird beschrieben durch:

- Die Queue Q der dem Fahrstuhl zugeordneten Anfragen
- Das Stockwerk $current\_floor$ in dem sich der Fahrstuhl aktuell befindet
- Die Richtung $r \in R; R = \{„hoch“, „runter“, „steht“\}$ in der sich der Fahrstuhl bewegt

- Der Status $s \in S; S = \{„frei“, „belegt“\}$ des Fahrstuhls

Alle vorhandenen Fahrstühle des Modells werden mit der Menge F bezeichnet.

Eine Anfrage wird beschrieben durch:

- Das Stockwerk $start\_floor$ auf dem nach dem Fahrstuhl gerufen wird
- Die Richtung r in die der Anfragende fahren möchte

In Abbildung 3.1 ist das erstellte Modell dargestellt. Es gibt eine Menge von zufällig generierten Fahrstuhl-Anfragen, die jeweils zu einem bestimmten Zeitpunkt t an den *ElevatorScheduler* gemeldet werden. Dies ist in der Realität der Zeitpunkt, an dem eine Person den Fahrstuhl per Knopfdruck ruft.

Der *ElevatorScheduler* prüft zunächst, ob einer der vorhandenen Fahrstühle frei ist oder ob einer der Fahrstühle in die gewünschte Richtung fährt und auf dem Weg in die Richtung an dem Stockwerk des Rufs vorbeikommt. Beschrieben durch: Sei $f \in F$ der zu prüfende Fahrstuhl, c die neue Anfrage und $check\_on\_way$ eine Funktion die prüft ob ein Stockwerk in Fahrtrichtung eines Fahrstuhls liegt, dann sind die fr c in Frage kommenden Fahrstühle F_c:

$$f \in F_c, F_c \subseteq F : s(f) = „frei“ \vee (r(f) = r(c) \wedge check\_on\_way(f, c)) \tag{3.1}$$

Falls eine der Bedingungen zutrifft, wird die Anfrage in die Queue des Fahrstuhls f eingereiht. Es wird berechnet, wann die neue Anfrage von diesem Fahrstuhl bearbeitet wird. Anschließend wird geprüft, wie die neue Anfrage die anderen Anfragen des Fahrstuhls beeinflusst. Aus der Bearbeitungszeit k_d des Fahrstuhls und der Beeinflussung k_l der anderen Anfragen in der Queue wird ein Wert berechnet, der die Kosten k der Anfrage bezogen auf den spezifischen Fahrstuhl f darstellt. Die neue Anfrage wird dem Fahrstuhl zugewiesen, der die geringsten Kosten aufweist. Beschrieben durch: Sei $q_n \in Q$ die Anfrage an n-ter Stelle in Q, $n = |Q(f)|$ und $k : k <= n$ die Position von c in Q:

$$k_d = \sum_{i=1}^{k} i = |q_i - q_{i-1}| \tag{3.2}$$

$$k_l = \sum_{i=k+1}^{n} i = |q_i - q_{i-1}| \tag{3.3}$$

$$k = k_d + k_l \tag{3.4}$$

Der Fahrstuhl selbst beinhaltet keine Planungslogik. Er fährt lediglich die Stockwerke ab, die in seiner Queue vom Planungssystem eingereiht wurden.

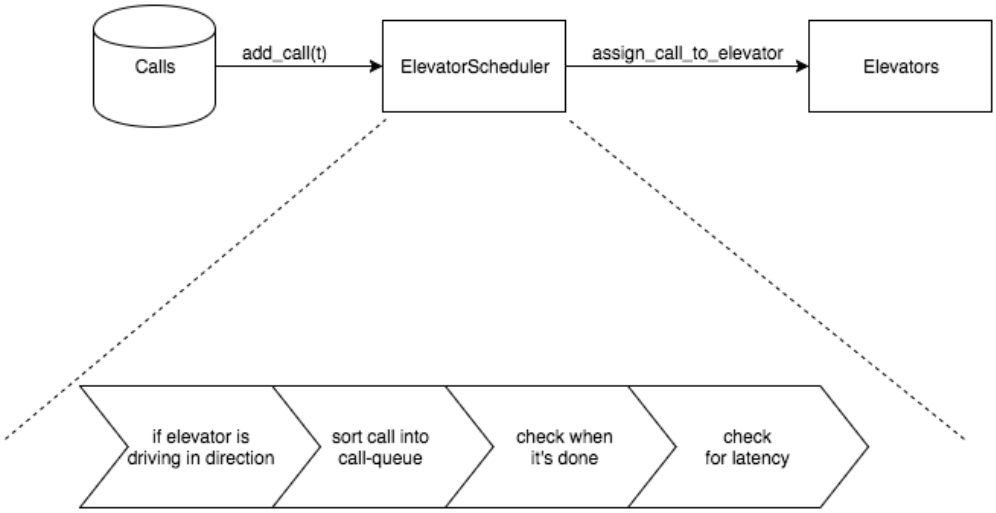

Abbildung 3.1: Modellierung einer Fahrstuhl-Ablaufplanung

Ein zukünftiger Fahrstuhlruf, der bereits vorher bekannt ist, unterscheidet sich von einem einfachen Fahrstuhlruf insofern, dass der zukünftige Ruf bereits 10 Zeiteinheiten bevor er auftritt vom Scheduler in seiner Planung berücksichtigt wird.

4 Auswertung der Ergebnisse

Im Folgenden werden die unterschiedlichen Fahrstuhl-Scheduler anhand der Metriken aus Abschnitt 1.2 miteinander verglichen. Dazu wurde eine Menge von zufälligen Fahrstuhl-Rufen zusammengestellt, die im einfachen Scheduler zu einem nahezu konstanten Durchsatz führen. Dadurch, dass eine der Metriken konstant ist, sind die Unterschiede zwischen den Metriken der Scheduler deutlicher zu sehen.

Für die Simulation wurde eine Zeitspanne von 200 Zeiteinheiten festgesetzt. Alle zwei Zeiteinheiten wird ein Fahrstuhl-Ruf generiert, der vom Scheduler anschließend behandelt wird.

4.1 Fahrstuhl ohne Kenntnis zukünftiger Anfragen

Im Folgenden werden die ermittelten Werte des einfachen Fahrstuhl-Schedulers dargestellt.

In Diagramm 4.1 ist der Durchsatz des einfachen Schedulers dargestellt. Es ist zu sehen, dass der einfache Scheduler nach Ablauf der Simulationszeit 89 Fahrstuhl-Rufe erledigt hat.

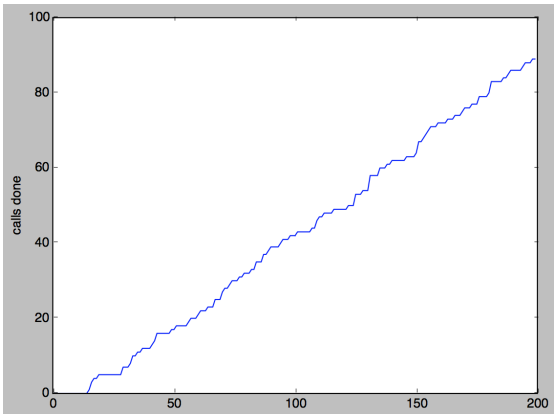

Abbildung 4.1: Durchsatz des einfachen Schedulers

In Abbildung 4.2 sind die durchschnittlichen Warte- und Fahrzeiten pro Zeiteinheit der Fahrstuhlrufe zu sehen.

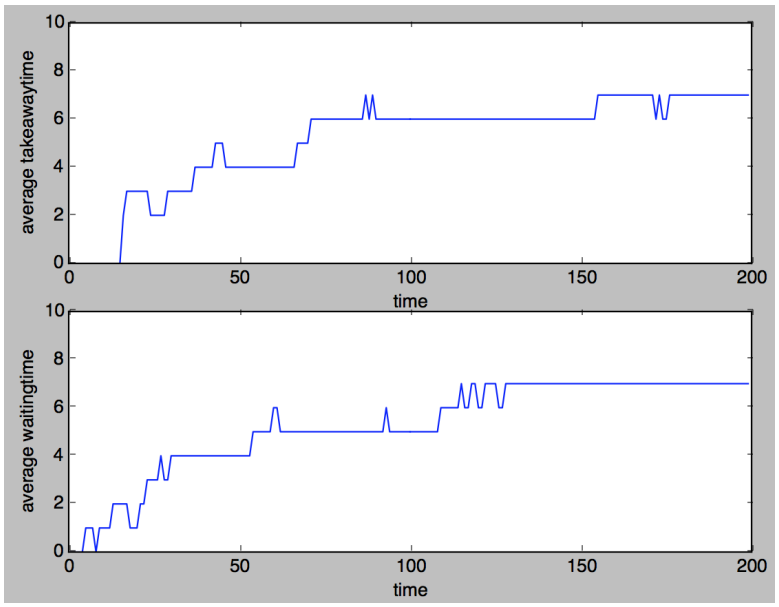

Abbildung 4.2: Zeiten des einfachen Schedulers

4.2 Fahrstuhl mit Kenntnis einiger zukünftiger Anfragen

Im Folgenden werden die ermittelten Werte des wissenden Fahrstuhl-Schedulers dargestellt.

In Diagramm 4.3 ist der Durchsatz des einfachen Schedulers dargestellt. Es ist zu sehen, dass der wissende Scheduler nach Ablauf der Simulationszeit 92 Fahrstuhl-Rufe erledigt hat.

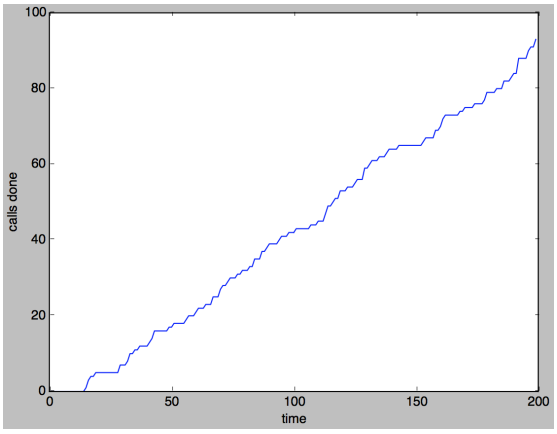

Abbildung 4.3: Durchsatz des wissenden Schedulers

In Abbildung 4.4 sind die durchschnittlichen Warte- und Fahrzeiten pro Zeiteinheit der Fahrstuhlrufe zu sehen.

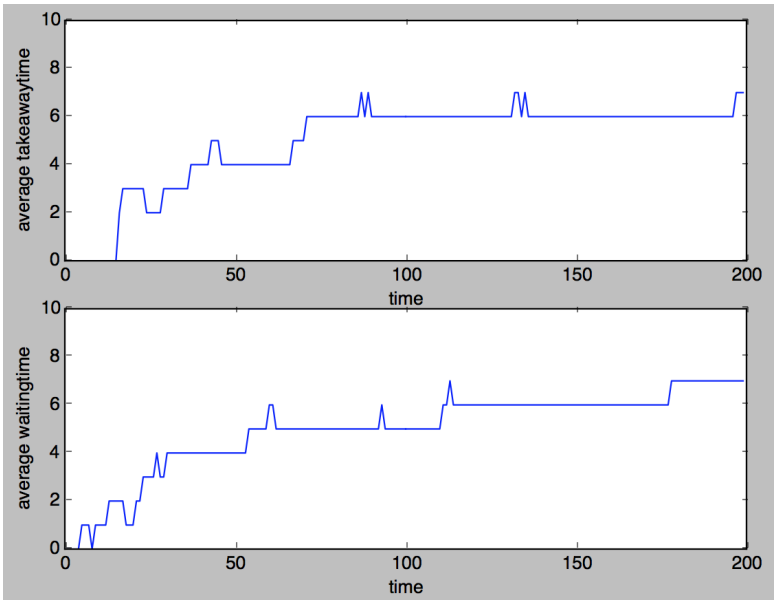

Abbildung 4.4: Zeiten des wissenden Schedulers

4.3 Betrachtung einzelner Anfragen

Einige zufällig ausgewählte Fahrstuhl-Anfragen wurden in der Simulation so konfiguriert, dass sie dem wissenden Scheduler vorzeitig bekannt sind. Für diese dedizierten Anfragen wurden die Metrik-Werte erfasst, die im Folgenden gezeigt werden.

Die Metrikwerte sind in der Tabelle 4.1 aufgeführt. Die Werte einer Zelle sind dabei folgendermaßen zu lesen: *a/b* wobei *a* der Wert ist, der im einfachen Scheduler erreicht wurde und *b* der Wert ist, der im wissenden Scheduler erreicht wurde. Dort ist außerdem zu sehen, zu welchem Zeitpunkt die Anfrage in das Scheduling des Schedulers eingegangen ist. Dieser Zeitpunkt ist in der Tabelle 4.1 in der Spalte *Scheduler-Zeit* geführt.

ID der Anfrage	Scheduler-Zeit	Wartezeit	Fahrzeit
52	104/94	115/105	131/119
59	118/108	134/129	140/135
74	148/138	163/149	176/158
83	166/156	166/167	169/170

Tabelle 4.1: Metrikwerte von ausgewählten Anfragen

4.4 Analyse

Nimmt man die Abbildungen dieses Kapitels für sich, haben sie keine Aussagekraft. Im Folgenden werden sie miteinander verglichen, um einen Wert aus ihnen zu ziehen.

Vergleicht man die Durchsatz-Abbildungen, dann sieht man, dass der wissende Scheduler einige Fahrstuhlrufe mehr beendet hat als der einfache Scheduler. Dies ist auf die Konstellation der zufällig erstellen Fahrstuhlrufe zurückzuführen und ist für diese Betrachtung nicht von Interesse. Simulationen mit anderen zufälligen Fahrstuhlruf-Konstellationen hat gezeigt, dass der einfache Scheduler ebenso einen höheren Durchsatz erreichen kann als der wissende Scheduler.

Keiner der beiden Scheduler schafft es alle generierten Fahrstuhl-Anfragen durchzuführen. Dies liegt daran, dass bis zum letzten Zeitschritt der Simulation neue Anfragen generiert werden. Für diese zuletzt generierten Anfragen ist nicht mehr ausreichend vorhanden, um sie erfolgreich durchzuführen.

Was außerdem auffällt ist, dass die Kurve anders verläuft. Bis zum Zeitpunk 104 verlaufen die Durchsatz-Kurven der beiden Scheduler gleichmäßig. Zu dem Zeitpunkt tritt der erste zukünftige Fahrstuhlruf auf. Man sieht in den Graphen, dass in der Kurve des wissenden Scheduler - im Vergleich zum einfachen Scheduler - an dieser Stelle weniger Rufe erledigt wurden. Dies liegt am veränderten Scheduling-Verhalten, das in in Abschnitt 3.3 erklärt wird. An den Zeitpunkten 118, 148 und 166 treten ebenfalls zukünftige Rufe auf. Zu diesen Zeitpunkten ist ebenfalls niedriges Wachstum in Abbildung 4.3 zu finden.

Ähnlich ist es in den Zeit-Graphen (Abbildungen 4.2 und 4.4). Auch dort verlaufen die Kurven der beiden Scheduler bis zum Auftreten des ersten zukünftigen Fahrstuhlrufs deckungsgleich. Ab dem Zeitpunkt 104 verläuft die Kurve des wissenden Schedulers anders als die des einfachen Schedulers. Die Durchschnittswerte der Warte- und Fahrzeiten des einfachen Schedulers wachsen nach dem Zeitpunkt 104 auf den Wert sieben an. Dieses Wachstum gibt es beim wissenden Schedulers auch, dort erfolgt das Wachstum allerdings erst später. Diese

Wachstums-Verzögerung ist nicht von Bedeutung, denn Simulationsversuche mit anderen zufälligen Fahrstuhlruf-Konstellationen haben gezeigt, dass der wissende Scheduler ebenso deutlich höhere Zeit-Werte als der einfache Scheduler erreichen kann.

In Tabelle 4.1 ist zu sehen, dass in drei der vier Fälle die dedizierten Anfragen in allen Zeit-Metriken niedrigere Werte erreicht haben. Dies ist so zu verstehen, dass die Personen kürzer auf den Fahrstuhl warten mussten und eine kürzere Fahrtzeit bis zu ihrem Ziel hatten.

5 Fazit

Die Simulations-Ergebnisse in Kapitel 4 zeigen, dass sich das Verhalten des Fahrstuhl-Schedulers verändert, sobald es Fahrstuhlrufe gibt, die bereits vor ihrem Auftreten bekannt sind. Dies ist wenig verwunderlich, denn dadurch dass sie bereits vor ihrem Erscheinen bekannt sind, können sie anders in die Bearbeitungsreihenfolge eingegliedert werden. Und es ergibt sich insgesamt ein besserer Verarbeitungs-Ablauf als bei einem einfachen Fahrstuhl-Scheduler.

5.1 Beantwortung der Arbeitsthese

Die Arbeitsthese in Abschnitt 1.1 kann mittels der Tabelle 4.1 beantwortet werden. Dort sind unter anderem die Wartezeiten der vorzeitig bekannten Fahrstuhl-Anfragen der beiden Scheduler gezeigt. Es ist zu sehen, dass in drei der vier Fälle die Wartezeit deutlich kürzer ist, falls der Fahrstuhl-Scheduler bereits vor auftreten des Rufs von ihm weiß. Im vierten Fall ist die Wartezeit 1 Zeiteinheit länger im wissenden Scheduler als im einfachen. Dies liegt an der gewählten Zusammenstellung der Fahrstuhl-Anfragen. In allen Anfrage-Konstellationen, die mit diesem Modell getestet wurden, hatten die bereits bekannten Anfragen eine kürzere Wartezeit. Die Arbeitsthese lässt sich daher bestätigen: Wenn das Fahrstuhl-System weiß, dass zum Zeitpunkt x in der Zukunft, eine Anfrage auf dem Stockwerk a gestellt wird, dann muss der Anfragesteller kürzer warten.

5.2 Auswirkung auf die Realität

In der Realität können solche Fahrstuhl-Anfragen allerdings folgende Auswirkung haben: Angenommen es ist bekannt, dass es zum Zeitpunkt x in der Zukunft eine Anfrage auf Stockwerk a gestellt wird, dann wird der Fahrstuhl im Idealfall zum Zeitpunkt x im Stockwerk a sein. Dann ist allerdings nicht garantiert, dass zu dem Zeitpunkt auch Personen auf dem Stockwerk bereit stehen zum Einsteigen. Es kann passieren, dass der Fahrstuhl in dem Stockwerk anhält, ohne dass jemand einsteigt. Hier entsteht eine unnötige Verzögerung der Fahrstuhlfahrt. Möglicherweise rufen die Personen, die bei dieser Anfrage hätten einsteigen sollen, erneut einen Fahrstuhl, sodass für diese Personen mehrere Fahrstuhl-Anfragen erzeugt wurden.

Eine bekannte zukünftige Anfrage garantiert nicht, dass auch jemand in den Fahrstuhl einsteigt. Dieses Problem könnte gelöst werden, indem durch beispielsweise Sensoren geprüft wird, ob sich zum Zeitpunkt x in Stockwerk a jemand vor den Fahrstuhltüren befindet, der einsteigen könnte.

Anders ist dies bei einem einfachen Fahrstuhl-Ruf. Um diesen auszulösen muss sich eine Person vor den Fahrstuhltüren befinden, um den Ruf-Knopf zu drücken. Es ist also davon auszugehen, dass bei einem einfachen Fahrstuhl-Ruf eine Person bereit steht zum Einsteigen. Es gibt den Spezialfall, bei denen Personen den Ruf-Knopf drücken und dann den Ort verlassen, um beispielsweise doch die Treppen zu nutzen. In diesem Fall könnte ebenfalls durch Sensoren verifiziert werden, dass jemand vor den Fahrstuhltüren steht, bevor dort ein Fahrstuhl anhält und dadurch seine Fahrt unterbricht.

5.3 Ausblick

Wie bereits in Kapitel 5.2 erwähnt, könnte man durch Sensoren prüfen, ob sich Personen vor Fahrstuhltüren befinden, die in den Fahrstuhl einsteigen könnten. Dadurch könnte geprüft werden, ob ein Fahrstuhl in einem Stockwerk halten soll, um dort Personen aufzunehmen. Falls niemand in dem Stockwerk wartet, braucht der Fahrstuhl seine Fahrt nicht unterbrechen und kann andere Passagiere früher an ihr Ziel bringen. Dazu ist es notwendig, dass der Fahrstuhl unterscheiden kann, ob er zu einem Stockwerk fahren soll, weil dort jemand eine Anfrage gestellt hat oder weil das Stockwerk das Ziel eines Passagiers ist. Dies sollte allerdings keine Hürde sein, denn diese Fälle lassen sich unterscheiden, durch die Unterscheidung der Knöpfe die zum Anfahren der Stockwerke gedrückt werden. Soll ein Fahrstuhl ein Stockwerk anfahren, um dort Passagiere aufzunehmen, dann wurde der Ruf-Knopf *vor* den Fahrstuhltüren gedrückt. Soll ein Fahrstuhl ein Stockwerk anfahren, um dort Passagiere abzuliefern, dann wurde der Stockwerk-Knopf *innerhalb* des Fahrstuhls gedrückt. Es ist zu prüfen, ob sich die Fahrtzeit für Passagiere verringert, falls nicht in Stockwerken gehalten wird, für das es zwar einen Fahrstuhl-Ruf gibt, aber niemand dort wartet zum Einsteigen.

Literaturverzeichnis

[Manasse u. a. 1988] Manasse, Mark ; McGeoch, Lyle ; Sleator, Daniel: Competitive algorithms for on-line problems. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing* ACM (Veranst.), 1988, S. 322–333

[Pinedo 2012] Pinedo, Michael L.: *Scheduling: theory, algorithms, and systems.* Springer Science & Business Media, 2012