Albert-Ludwigs-Universität Freiburg
Fakultät für Angewandte Wissenschaften
Institut für Informatik

Wintersemester 2006/2007

Diplomarbeit

Identifikation von Phasenübergängen in Handlungsplanungsbenchmarks

vorgelegt von
Paul Stefan Dischinger

am 29.05.2007

Zusammenfassung

Bei vielen **NP**-vollständigen Entscheidungsproblemen sind sogenannte Phasenübergänge zwischen einem Bereich mit nur schwach eingeschränkten positiven und einem Bereich mit stark eingeschränkten negativen Instanzen zu beobachten. Diese Übergänge lassen sich meist durch einen problemabhängigen Ordnungsparameter charakterisieren. Von Instanzen, die weit von der Phasengrenze entfernt liegen, kann häufig leicht gezeigt werden, dass es sich um positive bzw. negative Instanzen handelt. Viele der wirklich schweren Instanzen liegen in der Nähe der Phasengrenze.

Im Rahmen dieser Arbeit werden Phasenübergänge in Handlungsplanungs-Benchmarkproblemen empirisch untersucht und beschrieben, da die Kenntnis der Phasenübergänge potentiell die Erzeugung besonders schwerer Benchmark-Instanzen erlaubt.

Danksagungen

An dieser Stelle möchte ich mich bei allen bedanken, die mich unterstützt und somit zum Gelingen dieser Arbeit mit beigetragen haben. Mein besonderer Dank gilt Robert Mattmüller für die ausgezeichnete Betreuung dieser Arbeit und die Beisteuerung vieler guter Ideen und Vorschläge. Bei Malte Helmert möchte ich mich für die Bereitstellung des Quellcodes des domänenspezifischen MICONIC-Planers bedanken. Herrn Prof. Dr. Bernhard Nebel danke ich für die Möglichkeit, diese Arbeit an seinem Lehrstuhl anzufertigen.

Schließlich möchte ich besonders meinen Eltern danken, die mir das Studium der Informatik erst ermöglicht haben.

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Freiburg, den 29. Mai 2007. (Paul Stefan Dischinger)

Inhaltsverzeichnis

Kapitel 1.

Einleitung

In der Handlungsplanung geht es um die Erforschung von zielgerichtetem Denken und Verhalten von künstlichen Agenten. Ein Agent ist dabei eine Einheit, die bestimmte Reize aus der Umgebung wahrnimmt, diese verarbeitet und dann aufgrund eines Weltbilds (Wissen um die Umwelt und eigenes Können) handelt. Das zielgerichtete Verhalten eines planenden Agenten zeichnet sich dadurch aus, dass der Agent für einen gegebenen Anfangszustand, Mengen von Aktionen und Zielzuständen eine Folge von Aktionen findet, deren Ausführung in einen Zielzustand führt. Beim klassischen Planen ist die Umgebung des Agenten vollständig beobachtbar (vollständiger Zustand der Umgebung ist dem Agenten bekannt), deterministisch (der nächste Zustand der Umgebung wird durch den aktuellen Zustand und die durch den Agenten durchgeführte Aktion festgelegt), statisch (Umgebung kann nur durch den Agenten selbst verändert werden) und diskret (es gibt nur eine endliche Menge von Zuständen).

Ein wichtiger Aspekt der Handlungsplanung ist das Erforschen von Formalismen, mit deren Hilfe man die Aktionen, die Ziele und die Umwelt eines Agenten beschreiben kann, sowie die Erforschung von Algorithmen, die Probleme in der gewählten Repräsentation lösen. Da für zahlreiche Planungsdomänen das Problem, für gegebene Planungsaufgaben (Tasks) kürzeste Pläne zu finden, **NP**-schwer oder schwerer ist, gilt der Untersuchung von Heuristiken bei der Erforschung der Algorithmen ein großes Interesse. Unter Heuristiken versteht man Problemlösungsstrategien, die sehr effizient, dafür aber suboptimal sind. Deshalb wendet man Heuristiken oft dann an, wenn man Lösungen zu Problemen finden möchte, für die es wahrscheinlich keinen schnellen, optimalen Algorithmus gibt. Ein Beispiel für den Einsatz einer Heuristik ist ein Backtracking-Algorithmus zur Lösung eines CSP. Hier besteht die Heuristik aus festen Regeln, die zum Beispiel bestimmen, welche Variable als nächste instanziiert werden soll, und bei der Wertzuweisung an eine Variable bestimmen, welcher Wert als nächster zugewiesen wird.

Für Planungsalgorithmen gibt es unterschiedliche Strategien. Dazu zählen deduktives Planen (Planen durch Theorembeweisen) [Nau, Ghallab und Traverso, 2004], Planen mit Planungsgraphen [Blum und Furst, 1995], Planen als heuristische Suche im Zustandsraum [Bonet und Geffner, 2001] und Planen durch Erfüllbarkeitstests [Kautz und Selman, 1992]. Einer der Gründe für die unterschiedliche Leistungsfähigkeit von Planern ist also die Strategie, auf der der Planer aufbaut.

Instanzen von Planungsdomänen dienen Planern als Eingabe, wobei es die Aufgabe eines Planers ist, für die gegebene Instanz einen Plan zu finden. Domäne und Instanz werden in der Regel in der Sprache PDDL (Planning Domain Definition Language) kodiert.

Als geeignetes Performancemaß von Planern haben sich Benchmarks (Vergleichstests) etabliert. So findet seit der Artificial-Intelligence-Planning-Systems-Konferenz (AIPS) 1998 zweijährlich die International Planning Competition (IPC) statt. Hier werden aktuelle Planungssysteme miteinander anhand von Benchmarks verglichen. Alle teilnehmenden Planungsprogramme bekommen dabei dieselben Aufgaben (Instanzen ausgewählter Planungsdomänen) gestellt. Bewertet wird dann die benötigte Zeit zum Auffinden einer Lösung und die Anzahl der Probleme, die ein System lösen kann.

Als äußerst kritisch hat sich jedoch die Wahl der Planungsdomänen und deren Instanzen herausgestellt, anhand derer die Planungssysteme verglichen werden sollen. Laut Rintanen [2004] sind SAT-Planer vor allem bei kleinen, aber schweren Probleminstanzen erfolgreich, während Planungssysteme, die auf einer heuristischen Suche aufbauen, bei großen, aber dafür leichten Probleminstanzen effizienter sind. Die meisten Planungsdomänen der Wettbewerbe vor 2004 waren extrem leicht, und Hoffmann [2002] zeigt, dass ihre Planexistenzprobleme mit einem domänenunabhängigen Algorithmus in polynomieller Zeit entschieden werden können. Da in den Planungsdomänen der Realweltanwendungen die Instanzen fast immer schwer sind [Kautz, 2006], ist man daher eher an Planern interessiert, die bei den schweren Instanzen die beste Performance liefern.

Um nun gezielt schwer zu lösende Instanzen generieren zu können, wird in dieser Diplomarbeit der Vermutung nachgegangen, dass die Schwere der Instanzen ausgewählter Planungsdomänen (Miconic, Logistics, Grid) von domänenspezifischen Parameterkonstellationen abhängt. Huberman und Hogg [1987] argumentieren, dass komplexe Systeme durch wenige kritische Parameter verstanden werden können, die für die Komplexität des Systems charakteristisch sind. Dabei diskutieren sie auch die Bedeutung von Phasenübergängen für die Künstliche Intelligenz.

In dieser Diplomarbeit betrachten wir Phasenübergänge bei Entscheidungsproblemen. Phasenübergänge bei Entscheidungsproblemen verlaufen nach einem Einfach-Schwer-Einfach-Muster. Auch bei Optimierungsproblemen gibt es Phasenübergänge, wobei diese nach einem Einfach-Schwer-Muster verlaufen, was andeutet, dass Optimierungsprobleme schwieriger sind. Es ist sinnvoll, sich zunächst auf Entscheidungsprobleme zu beschränken, da der somit gewonnene Schätzer für die Planlänge zumindest bei langen Plänen ein Indikator für nicht einfache Instanzen darstellt.

Ein Phasenübergang wird durch einen Ordnungsparameter bestimmt, der jeder Probleminstanz einen reellen Wert zuordnet. Jedem Wert des Ordnungsparameters wird die Wahrscheinlichkeit zugeordnet, dass eine Instanz mit diesem Ordnungsparameterwert eine positive Instanz ist. Für den Ordnungsparameter gibt es dabei einen kritischen Wert, bei dem es zum Phasensprung kommt.

Phasenübergänge zeichnen sich somit immer durch abrupte Änderungen aus. In den in dieser Diplomarbeit untersuchten domänenspezifischen beschränkten Planexistenzproblemen ändert sich die Lösbarkeitswahrscheinlichkeit für eine Instanz abrupt, wobei

diese Änderung durch eine geringe Änderung der die Struktur der Probleminstanz charakterisierenden Parameter ausgelöst wird. Dabei zeigt sich eine extreme Laufzeitspitze bei Lösungsalgorithmen.

Ein Phasenübergang unterteilt immer zwei Gebiete. Im ersten Gebiet liegen überwiegend Instanzen, von denen mit geringem Berechnungsaufwand nachgewiesen werden kann, dass sie nicht lösbar sind. Das erste Gebiet wird durch eine Phasengrenze vom zweiten Gebiet getrennt. Der Bereich um diese Grenze enthält viele der für unsere Untersuchung interessanten intrinsisch schweren Instanzen, für die der Berechungsaufwand eines Planers drastisch steigt. Es ist nämlich für einen Planer nicht sofort offensichtlich, ob es sich bei einer Instanz aus diesem Bereich um eine positive oder negative Instanz handelt. Im zweiten Gebiet liegen alle Instanzen, für die ein Planer schnell eine Lösung findet, da viele Lösungsmöglichkeiten zur Verfügung stehen und somit der Berechungsaufwand wiederum gering ausfällt.

Die vorliegende Arbeit ist wie folgt gegliedert:

Zuerst führen wir in Kapitel 2.1 die Notation ein. In Kapitel 2.2 beschreiben wir die Grundlagen der Handlungsplanung und führen in Kapitel 2.3 die Definitionen der von uns betrachteten Planungsdomänen ein. In Kapitel 2.4 führen wir den Begriff des Domänenparameters ein. In Kapitel 2.5 betrachten wir bisherigen Untersuchungen von Phasenübergängen in klassischen **NP**-schweren Entscheidungsproblemen. In Kapitel 3.1 führen wir Definitionen ein, die wir für unsere Versuche benötigen. In Kapitel 3.2 wird die Versuchsdurchführung beschrieben. In Kapitel 4 sind alle Ergebnisse zusammengefasst und in Kapitel 5 wird ein Ausblick auf weitere Untersuchungen gegeben.

Kapitel 2.

Grundlagen und Definitionen

Es werden die wichtigsten Begriffe im Rahmen der Handlungsplanung kurz erläutert und wichtige Definitionen eingeführt. Außerdem werden wir eine Phasenübergangsvermutung aufstellen. Bevor die eigentlichen Untersuchungen beginnen, werden bisherige Untersuchungen in anderen Planungsdomänen sowie deren Ergebnisse diskutiert.

2.1. Notation

Die Notation dieser Arbeit entspricht weitestgehend der üblichen Schreibweise. Um jedoch Missverständnisse zu vermeiden, werden die wichtigsten Symbole noch einmal kurz erklärt.

- $\mathbb{N}$ sei die Menge der natürlichen Zahlen *einschließlich* der Null.
- Ist X eine Menge, so ist 2^X die Potenzmenge und $|X|$ ist die Mächtigkeit von X. Mit $\binom{X}{2} = \{\{x, y\} \mid x, y \in X, x \neq y\}$ bezeichnen wir die Menge aller 2-elementigen Teilmengen von X.

  X^* bezeichnet die Menge aller endlichen Folgen von Elementen aus X. Für $w \in X^*$ bezeichnet $\|w\|$ die Länge von w.
- $\mathrm{dom}(f) = \{a \mid \exists b : (a, b) \in f\}$ ist der Definitionsbereich der Funktion f.

  $\mathrm{ran}(f) = \{b \mid \exists a : (a, b) \in f\}$ ist der Bildbereich der Funktion f.
- Ist V eine Menge von Zustandsvariablen, dann ist ein Zustand eine Funktion $s : V \to \{0, 1\}$. Wir identifizieren die Menge aller Zustände über V mit der Menge $S := 2^V$.
- Σ bezeichnet immer ein endliches Alphabet.

2.2. Handlungsplanung

Die folgenden Definitionen orientieren sich an Helmert [2001] und Rintanen [2004].

Zuerst werden wir definieren, was wir unter einer Planungsinstanz verstehen, was ein Zustand ist und wie Zustandsübergänge definiert sind. Die Sichtweise von Zuständen und deren Übergängen ähnelt sehr der von endlichen Automaten.

Definition 1 (STRIPS-Planungsinstanz). Eine *STRIPS-Planungsinstanz*, kurz *Planungsinstanz*, ist ein Tupel $I = \langle V, s_0, O, g \rangle$ so dass:

- V eine endliche Menge von Zustandsvariablen,
- s_0 der Startzustand,
- O eine endliche Menge von Operatoren $o = (pre(o), add(o), del(o))$ mit $add(o), del(o), pre(o) \subseteq V$, $add(o) \cap del(o) = \varnothing$ und
- g eine aussagenlogische Formel, die Zielformel, ist. $S_G := \{s \in S \mid s \models g\}$ ist die Menge der Zielzustände.

Die Operatoren induzieren eine partielle Zustandsübergangsfunktion $\delta : S \times O \nrightarrow S$ mit $\delta(s, o) = (s \setminus del(o)) \cup add(o)$, falls $pre(o) \subseteq s$.

Die *erweiterte Übergangsfunktion* $\hat{\delta} : S \times O^* \nrightarrow S$ ist diejenige partielle Funktion mit minimalem Definitionsbereich mit:

- $\hat{\delta}(s, \varepsilon) = s$ und
- $\hat{\delta}(s, ow) = \hat{\delta}(\delta(s, o), w)$ für alle $o \in O$ und $w \in O^*$, wenn $\delta(s, o)$ und $\hat{\delta}(\delta(s, o), w)$ definiert sind.

$\mathcal{M}$ sei die Menge aller Planungsinstanzen.

Mit der Definition von Operatoren, die eine Zustandsübergangsfunktion induzieren, können wir nun definieren, was wir unter einem Plan verstehen. Statt von einer Folge von Operatoren sprechen wir beim Planen auch von einer Folge von Aktionen.

Definition 2 (Plan). Eine Folge von Aktionen $p \in O^*$ heißt *Plan für* I genau dann, wenn $\hat{\delta}(s_0, p)$ definiert ist und in S_G liegt. Ein Plan p heißt *optimal*, wenn seine Länge unter den Längen aller Pläne für I minimal ist. I heißt *lösbar*, wenn ein Plan für I existiert.

Intuitiv verstehen wir unter einer Domäne eine Menge von Instanzen mit gleichartiger zugrundeliegender Struktur. Formal definieren wir eine Planungsdomäne wie folgt.

Definition 3 (Planungsdomäne). Eine *Planungsdomäne* ist eine Funktion $\mathcal{D} : L_\mathcal{D} \to \mathcal{M}$ für eine feste Sprache $L_\mathcal{D}$ über einem festen endlichen Alphabet Σ. $L_\mathcal{D}$ heißt die *Kodierungssprache* dieser Domäne. Jedes $w \in L_\mathcal{D}$ kodiert eine *Planungsinstanz* $\mathcal{D}(w)$, und $\|w\|$ heißt die *Kodierungslänge*. Die Kodierungslänge $\|t\|$ einer *Planungsinstanz* t ist die minimale Kodierungslänge eines Wortes $w \in L_\mathcal{D}$ mit $\mathcal{D}(w) = t$.

Probleme bei Planungsdomänen hängen mit Problemen im komplexitätstheoretischen Sinne zusammen.

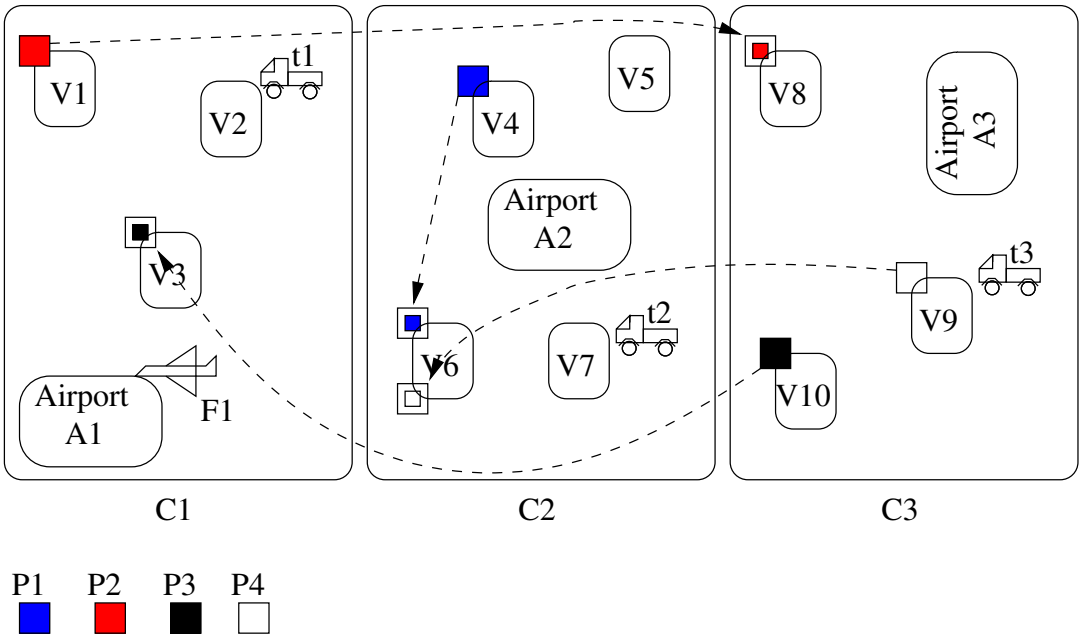

Abb. 2.1.: Beispielinstanz LOGISTICS. Der Übersichtlichkeit halber wurden die befahrbaren Kanten weggelassen.

Definition 4 (Entscheidungs- und Suchprobleme bei Planungsdomänen). Sei $\mathcal{D}$ eine Planungsdomäne. Dann sei PLANEX-$\mathcal{D}$ die Sprache aller lösbaren Instanzen von $\mathcal{D}$. PLANLEN-$\mathcal{D}$ sei die Sprache aller Paare (t, l), die aus einer Instanz t von $\mathcal{D}$ und einer natürliche Zahl l bestehen, sodass es für t einen Plan der Länge höchstens l gibt. Das Suchproblem PLANGEN-$\mathcal{D}$ besteht daraus, für eine Instanz t von $\mathcal{D}$ einen Plan p beliebiger Länge zu finden, falls diese Instanz lösbar ist. Ansonsten soll „unlösbar“ ausgegeben werden. Das Suchproblem PLANOPT-$\mathcal{D}$ besteht darin, für eine Instanz t von $\mathcal{D}$ einen Plan p minimaler Länge zu finden, falls diese Instanz lösbar ist, und ansonsten „unlösbar“ auszugeben.

Die Kodierungslänge $\|I\|$ einer PLANLEN-$\mathcal{D}$-Instanz I ergibt sich aus der Kodierungslänge der Planungsinstanz t, der Kodierungslänge der natürlichen Zahl l und einem Trennzeichen zu $\|I\| = \|t\| + 1 + \lceil \log(l) \rceil$.

2.3. Planungsdomänen

2.3.1. Logistics

Abbildung 2.1 zeigt eine LOGISTICS-Instanz. Jeder Ort innerhalb einer Stadt ist mit allen anderen Orten der jeweiligen Stadt verbunden. Die Städte sind nur über die Flugplätze paarweise verbunden. Der Übersichtlichkeit wegen wurden die Verbindungen nicht

eingezeichnet. Die Pakete können mit Hilfe der Fahrzeuge (LKW, Flugzeug) bewegt und somit von ihrem Anfangsort zu dem entsprechenden Zielort gebracht werden. Ein LKW darf immer nur die Orte der jeweiligen Stadt und das Flugzeug nur die Flugplätze der einzelnen Städte ansteuern.

Definition 5 (Logistics-Instanz). Eine *Logistics-Instanz* ist ein 7-Tupel

$I = (V, E, M, P, loc_0, loc_G, road)$, wobei

- (V, E) ein ungerichteter Graph mit endlicher Knotenmenge V (Orte) ist. Die Orte sind unterteilt in paarweise disjunkte, jeweils vollständig verbundene Teilmengen (Städte). Jede Stadt hat genau einen als Flugplatz ausgezeichneten Ort. Die Flugplätze aller Städte sind paarweise verbunden, sonst bestehen keine weiteren Verbindungen.
- M ist eine endliche Menge von Fahrzeugen. Die Menge der Fahrzeuge ist unterteilt in LKWs und Flugzeuge, wobei sich jedes Flugzeug im Startzustand an einem Flugplatz befinden muss und jeder LKW eine unbeschränkte Kapazität hat.
- P ist eine endliche Menge von Paketen,
- $loc_0 : (M \cup P) \to V$ ist die Funktion, die die Anfangsorte von Fahrzeugen und Paketen bestimmt,
- $loc_G : P \to V$ ist die Funktion, die die Zielzustände zuweist,
- $road : M \to 2^E$ ist die Funktion, die jedem Fahrzeug m die Menge aller von m befahrbaren Kanten zuordnet.

Die Anfangsorte aller Pakete müssen sich von ihren Zielorten unterscheiden, d. h. $loc_0(p) \neq loc_G(p)$ für alle $p \in P$. V, M und P müssen disjunkt sein.

Zur Definition der LOGISTICS-Planungsdomäne fehlt noch die Angabe der *Aktionen*:

- *Bewegung* eines Fahrzeugs zwischen zwei Orten,
- *Aufladen* eines Pakets, das sich am gleichen Ort wie das Fahrzeug befindet,
- *Abladen* eines Pakets an dem Ort, an dem sich das Fahrzeug mit dem Paket befindet.

Für nähere Details siehe Anhang oder [Helmert, 2001].

Bei der Beispielinstanz aus Abbildung 2.1 ist

- $C_1 = \{v_1, v_2, v_3, A_1\}$, $C_2 = \{v_4, v_5, v_6, v_7, A_2\}$, $C_3 = \{v_8, v_9, v_{10}, A_3\}$, $A = \{A_1, A_2, A_3\}$
- $Tr = \{t_1, t_2, t_3\}$, $Fl = \{F_1\}$
- $P = \{P_1, P_2, P_3\}$
- $R = \bigcup_{i=1}^{3} \binom{C_i}{2}$, $L = \binom{A}{2}$

Damit ist

- $V = C_1 \cup C_2 \cup C_3$, $E = R \cup L$
- $M = Tr \cup Fl$
- $road(m) = R$ für alle $m \in Tr$
- $road(F_l) = L$

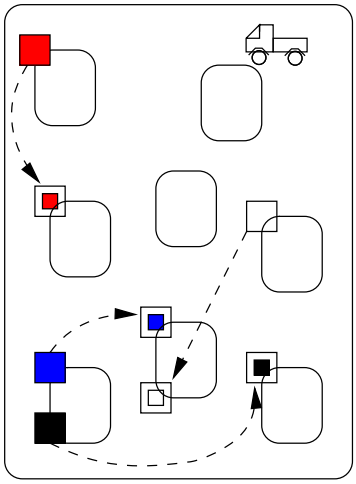

Abb. 2.2.: Beispielinstanz MICONIC-10

- $loc_0(t_1) = v_2$, $loc_0(t_2) = v_7$, $loc_0(t_3) = v_9$,
  $loc_0(F_1) = A_1$, $loc_0(P_1) = v_4$, $loc_0(P_2) = v_1$,
  $loc_0(P_3) = v_{10}$, $loc_0(P_4) = v_9$
- $loc_G(P_1) = v_6$, $loc_G(P_2) = v_8$,
  $loc_G(P_3) = v_3$, $loc_G(P_4) = v_6$

2.3.2. Miconic-10-STRIPS

Bei der STRIPS-Variante von MICONIC-10 hat man einen Aufzug und Passagiere, die damit zu ihren Zielstockwerken transportiert werden sollen. Eine MICONIC-10-Instanz ist isomorph zu einer LOGISTICS-Instanz unter der Einschränkung, dass es nur eine Stadt mit einem LKW gibt und es zudem keinen Flugplatz und kein Flugzeug gibt. Außerdem gilt die Konvention, dass der LKW hier Aufzug, die Pakete Passagiere und die Orte Stockwerke heißen.

Da wir in unseren späteren Untersuchungen nur an optimalen Plänen interessiert sind, sind auch folgende Einschränkungen für die MICONIC-10-STRIPS-Instanzen zulässig:

Es ist nicht möglich, dass ein Passagier den Aufzug an jedem Ort verlässt. Passagiere dürfen den Aufzug nur an den Zielorten verlassen. Hat ein Passagier einmal den Aufzug verlassen, darf er ihn nicht wieder betreten. Start- und Zielort müssen immer unterschiedlich sein. Für jeden Passagier ist ein Zielort spezifiziert.

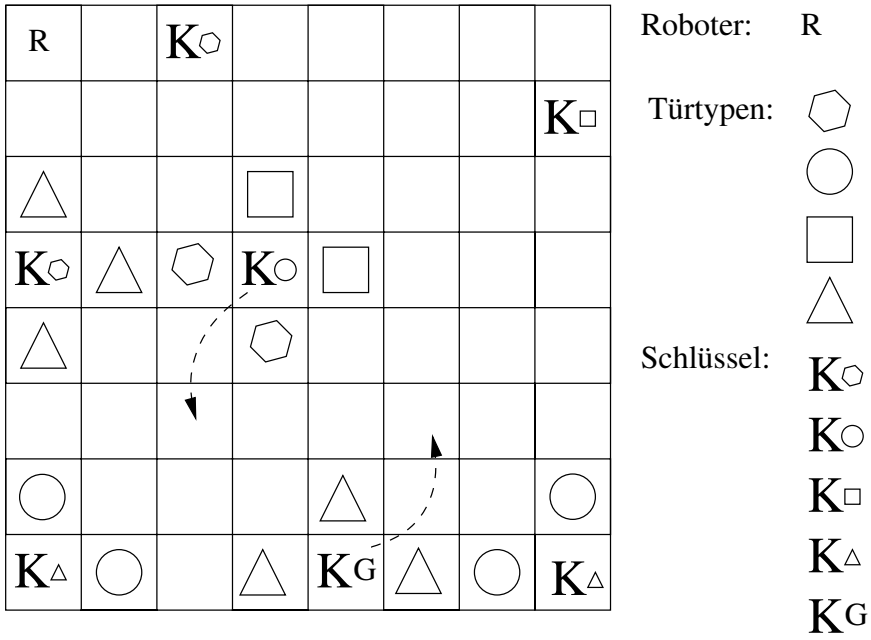

Abb. 2.3.: Beispielinstanz GRID

2.3.3. Grid

Abbildung 2.3 zeigt eine Beispielinstanz der GRID-Planungsdomäne. Es gibt hier einen Roboter, der sich an einem Ort befindet. Die Orte sind in Form eines Gittergraphen angeordnet. Statt Paketen gibt es hier Schlüssel, die zu einem Zielort gebracht werden müssen. Manche Orte sind durch Türen blockiert und somit von dem Roboter erst betretbar, nachdem er die Tür durch einen passenden Schlüssel von einer benachbarten Position aus geöffnet hat.

Definition 6 (Gittergraph). Für $w, h \in \mathbb{N}_0$ ist das *Standardgitter* Grid[w,h] mit Breite $w+1$ und Höhe $h+1$ definiert als der Graph mit der Knotenmenge $V = \{0, \ldots, w\} \times \{0, \ldots, h\}$ und der Kantenmenge $E = \{\{(a,b),(a',b')\} \subseteq V \mid |a-a'| + |b-b'| = 1\}$. Ein Graph ist ein *Gittergraph*, wenn er isomorph zu Grid[w,h] für $w, h \in \mathbb{N}$ ist.

Definition 7 (Grid-Instanz). Eine GRID-*Instanz* ist ein 9-Tupel mit $I = (V, E, l_0, \mathcal{K}, loc_0, loc_G, S, door, type)$, wobei

- (V, E) ein Gittergraph ist,
- $l_0 \in V$ die Anfangsposition des Roboters ist,
- $\mathcal{K}$ eine endliche Menge von Schlüsseln ist,
- $\mathcal{K}_G \subseteq \mathcal{K}$ die Menge der Zielschlüssel ist,
- $loc_0 : \mathcal{K} \to V$ die Funktion, die die Anfangsorte der Schlüssel bestimmt, ist,
- $loc_G : \mathcal{K}_G \to V$ die Funktion, die die Zielorte der Schlüssel bestimmt, ist,
- S die Menge der Türtypen ist,
- $door : V \setminus \{l_0\} \nrightarrow S$ ist die Funktion, die die Typen der Türen bestimmt,
- $type : \mathcal{K} \to S$ ist die partielle Funktion, die die Typen der Schlüssel bestimmt.

Jede Position kann höchstens eine Tür darstellen, wobei die Türen durch ihre Typen unterschieden werden. Ist eine Position eine Türe, spezifiziert die *door*-Funktion ihren Typ. Auf die gleiche Art und Weise spezifiziert die Funktion *type* den Typ der Schlüssel. Mit einem Schlüssel kann eine Tür genau dann geöffnet werden, wenn Schlüssel und Türe denselben Typ haben. Außerdem wird vorausgesetzt, dass der Roboter an keiner versperrten Position starten darf.

Die Aktionen, die zur Definition der GRID-Planungsdomäne noch fehlen, sind die folgenden: *Bewegung* des Roboters zu einem benachbarten, freien Ort (keine Diagonalbewegungen). *Aufnehmen* eines Schlüssels an der aktuellen Position des Roboters (falls der Roboter momentan keinen anderen Schlüssel trägt). *Ablegen* eines Schlüssels, den der Roboter momentan trägt. *Vertauschen* des aktuell getragenen Schlüssel mit dem Schlüssel, der sich an der aktuellen Position des Roboters befindet. *Öffnen* einer Türe, die sich an einer benachbarten Position des Roboters befindet und welche denselben Typ hat wie der vom Roboter getragene Schlüssel.

Für nähere Details siehe auch hier Anhang oder Helmert [2001].

2.4. Domänenparameter

Im folgenden wollen wir die Instanzen einer Planungsdomäne durch Domänenparameter charakterisieren.

Für eine feste Domäne gilt folgende Notation:

- $\mathcal{X} = \{X_1, \ldots, X_m\}$ ist die Menge aller Domänenparameter.
- $v : \mathcal{X} \to \mathbb{N}$ ist eine Belegung der Domänenparameter. Wir haben $\mathbb{N}$ als Wertebereich gewählt, da die im folgenden von uns betrachteten Domänen nur natürlichzahlige Domänenparamterbelegungen haben.
- $\mathcal{V} := \{v \mid v : \mathcal{X} \to \mathbb{N}\}$ ist die Menge aller Domänenparameterbelegungen.
- $\mathcal{M}_\mathcal{D} = \text{ran}(\mathcal{D})$ ist die Menge aller Planungsinstanzen der Planungsdomäne $\mathcal{D}$.
- Ist $t \in \mathcal{M}_\mathcal{D}$ eine Planungsinstanz, so ist $v_t : \mathcal{X} \to \mathbb{N}$ die Belegung der Domänenparameter, die in t realisiert ist.
- $\mathcal{M}_\mathcal{D}^v := \{t \in \mathcal{M}_\mathcal{D} \mid v_t = v\}$ ist die Menge aller Planungsinstanzen aus $\mathcal{D}$ mit der vorgegebenen Belegung v.

Die Planungsinstanzen t unter einer Belegung v aus $\mathcal{M}_\mathcal{D}^v$ müssen nur in der Belegung der Domänenparameter übereinstimmen. In allen Elementen, die nicht durch die Domänenparameter beschrieben sind, können sich die Instanzen mit gleicher Domänenparameterbelegung aber unterscheiden.

Ist die Domänenparameterbelegung v aus dem Kontext ersichtlich, so identifizieren wir gelegentlich einen Domänenparameter X mit seinem Wert $v(X)$.

Für das LOGISTICS-Domänen-Beispiel auf Abbildung 2.1 gilt etwa:

- $\mathcal{X} = \{\mathit{Flugzeuge}, \mathit{Städte}, \mathit{Orte}, \mathit{Pakete}\}$
- t wie im Beispiel
- $v_t = \{\mathit{Flugzeuge} \mapsto 1, \mathit{Städte} \mapsto 3, \mathit{Orte} \mapsto 13, \mathit{Pakete} \mapsto 4\}$
- $\mathcal{M}_\mathcal{D}$:= Menge aller MICONIC-Planungsinstanzen
- $\mathcal{M}_\mathcal{D}^v := \{t \mid v_t = (1, 3, 13, 4)\}$

Da einige Bestandteile der weiter oben definierten LOGISTICS-Instanz, wie zum Beispiel die LKWs nicht durch Domänenparameter spezifiziert werden, würden sich die einzelnen Task unter der Domänenparameterbelegung v_t in diesem Punkt unterscheiden. Ein Problemgenerator könnte alle nicht beschriebenen Elemente zufällig bestimmen.

2.5. Phasenübergänge

Da das Phänomen von Phasenübergängen bei **NP**-vollständigen Problemen auftritt, werden wir auf die Grundzüge der Komplexitätstheorie eingehen, bevor wir zur eigentlichen Definition der Phasenübergangsvermutung kommen.

2.5.1. NP-Schwere

Ein Ziel der Komplexitätstheorie besteht darin, Probleme gemäß ihres Bedarfs an Berechnungsressourcen (Rechenzeit, Speicherplatz) zu klassifizieren und die handhabbaren von den nicht-handhabbaren Problemen abzugrenzen. Nicht-handhabbare Probleme sind Probleme, deren benötigte Zeit zum Lösen von Instanzen so mit der Größe der Instanzen wächst, dass selbst moderat große Instanzen solcher Probleme nicht mehr exakt in annehmbarer Zeit gelöst werden können.

Die Komplexitätsklasse **P** enthält alle Probleme, welche man mit Hilfe von deterministischen Algorithmen mit polynomiellem Zeitaufwand lösen kann [Papadimitriou, 1994].

Die Komplexitätsklasse **NP** enthält die Probleme, welche man mit Hilfe von nichtdeterministischen Algorithmen mit polynomiellem Zeitaufwand lösen kann. **NP** kann äquivalent dadurch charakterisiert werden, dass ein Problem genau dann in **NP** liegt, wenn eine gegebene Lösung in polynomieller Zeit von einem deterministischen Algorithmus verifiziert werden kann.

Eine der wichtigsten Fragen der Komplexitätstheorie ist, ob $\mathbf{P} = \mathbf{NP}$ oder $\mathbf{P} \neq \mathbf{NP}$. Dass $\mathbf{P} \subseteq \mathbf{NP}$ gilt, ist klar, aber ob die beiden Klassen nun tatsächlich gleich sind oder **P** echt in **NP** liegt, ist unbekannt [Papadimitriou, 1994]. Von vielen wichtigen Problemen (TSP, SAT etc.) weiß man, dass sie **NP**-vollständig sind. Man kennt dafür aber bisher keinen deterministischen in Polynomialzeit durchführbaren Algorithmus, weshalb man vermutet, dass es sich hierbei tatsächlich um Probleme in $\mathbf{NP} \setminus \mathbf{P}$ handelt.

Ein Problem P ist **NP**-schwer, wenn jedes Problem Q aus **NP** in polynomieller Zeit auf P reduziert[1] werden kann. Ist ein Problem **NP**-schwer und liegt es in **NP**, so wird es als **NP**-vollständiges Problem bezeichnet. Es liegen entweder alle **NP**-vollständigen Probleme in **P** oder keine [Papadimitriou, 1994]. Da es trotz intensiven Forschens bisher noch niemandem gelungen ist, für eines der **NP**-vollständigen Probleme einen deterministischen Algorithmus zu finden, der dieses in polynomieller Zeit löst, sieht man dies als ein starkes Indiz, dass höchstwahrscheinlich $\mathbf{P} \neq \mathbf{NP}$ gilt.

2.5.2. Phasenübergangsvermutung

Im Folgenden wollen wir beschreiben, was wir unter einem Phasenübergang verstehen. Wir haben ein *Problem* gegeben, bestehend aus einer Menge von *Probleminstanzen* $\mathcal{I} \subseteq \Sigma^*$ und einer Menge $\mathcal{P} \subseteq \mathcal{I}$ von *positiven Instanzen*. Häufig identifizieren wir ein Problem mit der Menge $\mathcal{P}$ seiner positiven Instanzen. Die *Kodierungslänge* einer Probleminstanz $I \in \mathcal{I}$ bezeichnen wir mit $\|I\|$.

Bei der Phasengrenze findet ein plötzlicher Sprung der Funktion statt, die den Probleminstanzen die Wahrscheinlichkeit, eine positive Instanz zu sein, zuordnet. Der Sprung an der Phasengrenze ist um so steiler, je größer die Probleminstanzen sind.

Es wird vermutet, dass für jedes **NP**-schwere Problem eine schnell berechenbare, durch einen einfachen Term darstellbare Funktion $r : \mathcal{I} \to \mathbb{R}$ und ein $r^* \in \mathbb{R}$ existiert, so dass für $P(n, r) = \Pr(I \in \mathcal{P} \mid I \in \mathcal{I}, \|I\| = n, r(I) = r)$ gilt:

$$\text{für alle } r < r^* : \lim_{n\to\infty} P(n, r) = 1$$
$$\text{für alle } r > r^* : \lim_{n\to\infty} P(n, r) = 0$$

2.5.3. Bekannte Phasenübergänge in NP-vollständigen Problemen

Im Folgenden wollen wir einige ausgewählte Untersuchungen betrachten, die sich mit dem Thema Phasenübergänge beschäftigt haben.

3-SAT

Das *Erfüllbarkeitsproblem der Aussagenlogik*, kurz SAT, ist das folgende:

- *Gegeben:* Eine aussagenlogische Formel F.
- *Gefragt:* Ist F erfüllbar, d.h. gibt es eine erfüllende Variablenbelegung für F?

3-SAT ist die Unterklasse von SAT, bei der alle Formeln in konjunktiver Normalform sind und die Klauseln aus genau drei unterschiedlichen Literalen bestehen.

[1] Zum Begriff der polynomiellen Reduzierbarkeit siehe [Papadimitriou, 1994].

Selman, Mitchell und Levesque [1996] untersuchen Phasenübergänge bei 3-SAT. Sie können sowohl empirisch als auch analytisch nachweisen, dass die Wahrscheinlichkeit, dass eine Instanz erfüllbar ist, mit einem größer werdenden Verhältnis von Klauseln zu Variablen von 1 auf 0 fällt. Nähert sich dabei das Verhältnis dem kritischen Wert 4,27 von unten, fällt die Wahrscheinlichkeit, dass eine Formel erfüllbar ist. Nähert sich dieses Verhältnis von oben der 4,27, steigt die Wahrscheinlichkeit, ein Modell für die Formel zu finden. Um den Wert 4,27 beträgt die Wahrscheinlichkeit für eine positive Instanz $\frac{1}{2}$. Der Phasenübergang an dieser Stelle korreliert mit der Schwierigkeit, eine Formel auf Erfüllbarkeit zu testen. Sobald das Variablen-Klauseln-Verhältnis den Wert 4,27 annimmt, benötigen alle bekannten Algorithmen exponentielle Laufzeit in der Größe der Formelgröße [Rintanen, 2004]. Die Laufzeit verbessert sich wiederum drastisch, sobald das Verhältnis wächst oder fällt [Rintanen, 2004].

Hamiltonkreis

Sei $G = (V, E)$ ein ungerichteter Graph mit $V = \{v_0, \ldots, v_{n-1}\}$. Ein *Hamiltonkreis* in G ist eine Permutation π der Knoten, so dass $v_{\pi(0)}, \ldots, v_{\pi(n-1)}$ ein Kreis in G ist, d.h. dass $\{v_{\pi(i)}, v_{\pi(i+1 \bmod n)}\} \in E$ für $i = 0, 1, \ldots, n-1$.

Das *Hamiltonkreisproblem* ist das folgende:

- *Gegeben:* Ein Graph $G = (V, E)$.
- *Gefragt:* Enthält G einen Hamiltonkreis?

Cheeseman, Kanefsky und Taylor [1991] vermuten, dass die Wahrscheinlichkeit für die Existenz eines Hamiltonkreises mit dem mittleren Grad des Graphen korreliert. In ihrer empirischen Untersuchung generieren sie jeweils 20 zufällige Graphen mit unterschiedlicher Anzahl von Knoten und unterschiedlicher Anzahl von Verbindungen und bestimmen davon den Anteil der Graphen, die einen Hamiltonkreis enthalten.

Dabei kommen sie zu folgendem Ergebnis: Am einen Extrem befinden sich die Graphen, deren mittlerer Grad kaum größer als 2 ist, die mit sehr geringer Wahrscheinlichkeit überhaupt zusammenhängend sind und die mit äußerst geringer Wahrscheinlichkeit einen Hamiltonkreis enthalten. Am anderen Extrem sind die vollständig zusammenhängenden Graphen, für die es immer einen Hamiltonkreis gibt, sowie die fast vollständig zusammenhängenden Graphen, die mit einer sehr hoher Wahrscheinlichkeit einen Hamiltonkreis enthalten. Zwischen diesen beiden Extremen steigt ab einem durchschnittlichen Grad von $\ln N + \ln \ln N$ die Wahrscheinlichkeit, einen Hamiltonkreis zu enthalten, steil von 0 auf 1 [Bollobás, 1985].

Um zu sehen, ob auch die Berechnungskosten für das Auffinden eines Hamiltonkreises mit dem mittleren Grad variieren, generieren Cheeseman u. a. [1991] zufällige Graphen festen Grades. Anschließend wird auf jeden dieser Graphen eine Backtrackingsuche angewendet. Das Kostenmaß zum Auffinden eines Hamiltonkreises basiert auf der Anzahl der Backtrackingschritte. Es stellt sich heraus, dass eine Steigerung der Anzahl von Backtrackingschritten an der Stelle stattfindet, an der auch die Wahrscheinlichkeit, einen Hamiltonkreis zu finden, von 0 auf 1 steigt.

Die Stelle, an der der Ordnungsparameter (mittlerer Grad) den kritischen Wert von $\ln N + \ln \ln N$ annimmt, ist sowohl für den Phasenübergang der Wahrscheinlichkeit der Existenz eines HamiltonKreises als auch für den Phasenübergang der Berechnungszeit, also dem sprunghaften Anstieg benötigter Backtrackingschritte, verantwortlich. Cheeseman u. a. [1991] sehen bei ihrer Untersuchung folgenden Zusammenhang:

Im Gebiet links dieser Sprungstelle liegen die Graphen mit wenigen Verbindungen, für die es mit nur sehr geringer Wahrscheinlichkeit einen Hamiltonkreis gibt. Die verwendete Backtrackingsuche terminiert früh, da alle potentiellen Hamiltonkreise früh genug aus der Suche ausscheiden. Im Gebiet rechts des kritischen Wertes des Ordnungsparameters liegen die Graphen mit vielen Verbindungen, weshalb das verwendete Backtrackingverfahren schnell eine Lösung findet und die Anzahl der Backtrackingschritte somit wiederum sehr gering ist. Im Gebiet um den kritischen Wert des Ordnungsparameters befinden sich die schweren Probleme, da es hier viele Beinahe-Hamiltonkreise gibt. Diese große Anzahl an lokalen Minima macht es dann sehr schwierig, einen Hamiltonkreis zu finden, sofern es überhaupt einen gibt, was sich durch die steigende Anzahl von Backtrackingschritten in der Untersuchung von Cheeseman u. a. [1991] widerspiegelt.

Graphfärbung

Sei $G = (V, E)$ ein ungerichteter Graph und $K \in \mathbb{N} \setminus \{0\}$. Eine K-Färbung von G ist eine Abbildung $c : V \to \{1, \ldots, K\}$ mit der Eigenschaft, dass $c(v) \neq c(w)$ für alle $\{v, w\} \in E$.

Das *Graphfärbungs-Entscheidungsproblem* ist das folgende:

- *Gegeben:* Ein ungerichteter Graph $G = (V, E)$ und $K \in \mathbb{N} \setminus \{0\}$.
- *Gefragt:* Gibt es für G eine K-Färbung?

In den meisten Fällen ist das Graphfärbungs-Entscheidungsproblem leicht zu lösen, weshalb Cheeseman u. a. [1991] das Entscheidungsproblem der K-Färbbarkeit untersuchen, bei dem keine der folgenden Vereinfachungsoperationen mehr angewendet werden können:

- Entfernung von Knoten, die weniger als K Nachbarn haben (ein solcher Knoten kann immer eingefärbt werden).
- Entfernung von Knoten N, wenn es einen zweiten Knoten M gibt, der mit jedem Knoten verbunden ist, mit dem auch N verbunden ist, nicht aber mit N selbst (jede Farbe, die man M zuweisen kann, kann man auch N zuweisen).
- Verschmelzung von Knoten, die dieselbe Farbe haben müssen (alle Knoten, die jeweils komplett mit der gleichen Clique der Größe $K - 1$ verbunden sind, müssen dieselbe Farbe haben und können zusammengefasst werden).

Diese Vereinfachungs-Operatoren garantieren, dass der vereinfachte Graph genau dann K-färbbar ist, wenn der ursprüngliche Graph K-färbbar ist.

Cheeseman u. a. [1991] untersuchen empirisch die Wahrscheinlichkeit für die Existenz einer Lösung des K-Färbbarkeitsproblems für verschiedene K und $\mathcal{N}$ (Anzahl der Knoten). Anhand ihrer gesammelten Daten sind zwei Dinge beobachtbar. Zum einen kommt

es zu einer plötzlichen Änderung der Lösbarkeitswahrscheinlichkeit bei größerem K und größerem durchschnittlichen Grad. Zweitens kommt es zu einem steileren Übergang bei größerem $\mathcal{N}$.

Sie zeigen auch, dass der Berechnungsaufwand vom durchschnittlichen Grad des Graphen und der Größe von K abhängig ist. Bei ihren Untersuchungen stellen sie sowohl für die 3-Färbbarkeit als auch für die 4-Färbbarkeit einen Phasenübergang der Kosten fest, der bei beiden sehr ähnlich verläuft, allerdings ist bei der 4-Färbbarkeit ein steilerer Übergang auffällig.

Cheeseman u. a. [1991] deuten das Auftreten des Phasenübergangs der Kosten anhand ihres eingesetzten Backtrackings-Algorithmus. Bei schweren Instanzen muss der Algorithmus häufig bis zum Anfang zurücklaufen, da es hier sehr viele Beinahe-Lösungen gibt. Der Algorithmus findet dabei fast immer eine vollständige Zuweisung und merkt deshalb erst sehr spät, dass er wieder zurücklaufen muss. Diese lokalen Minima sind folglich der Grund, warum der Algorithmus teilweise sehr lange braucht, um eine Lösung zu finden.

Traveling-Salesman-Problem

Das *Traveling-Salesman-Entscheidungsproblem*, kurz TSP, ist das folgende:

- *Gegeben:* Ein vollständiger, ungerichteter, gewichteter Graph $G = (V, E, w)$ mit nicht-negativen Kantengewichten $w : V \to \mathbb{N}$ und $V = \{v_0, \ldots, v_{n-1}\}$ sowie $K \in \mathbb{N}$.
- *Gefragt:* Existiert eine Permutation π von V, sodass

$$\sum_{i=0}^{n-1} w(\{v_{\pi(i)}, v_{\pi(i+1 \mod n)}\}) \leq K$$

Cheeseman u. a. [1991] begnügen sich bei ihren Untersuchungen mit ganzzahligen Kostenmatrizen, die im Mittel Kosten von 10 für ihre Kanten haben, wobei die einzelnen Instanzen durch unterschiedliche Standardabweichungen der einzelnen Kosten generiert werden, sowie für eine unterschiedliche Anzahl von Knoten. Als der die Phasentransition charakterisierende Parameter stellt sich die Standardabweichung der Kostenmatrix heraus.

Um die Berechnungskosten zum Lösen des TSPs abzuschätzen, benutzen Cheeseman u. a. [1991] einen Backtracking-Algorithmus, der garantiert eine Lösung mit minimalen Kosten findet. Es wurde beobachtet, dass die Steilheit des Phasenübergangs der Kosten deutlich mit der Anzahl der Knoten steigt.

Im Gebiet, in dem die Standardabweichung hoch ist, werden nur die Pfade geringer Kosten durch den Backtracking-Algorithmus betrachtet, da jede Tour minimalen Gewichts nur diese Übergänge mit geringen Kosten nutzen wird. Im anderen Gebiet, in dem die Kosten immer gleich sind, gibt es viele alternative Touren, die alle dieselben minimalen Kosten haben, und es ist somit auch hier sehr einfach, eine minimale Tour zu finden. An der Phasengrenze gibt es relativ viele lokale Minima, also verhältnismäßig viele Touren,

die fast alle minimale Kosten haben, was das Backtracking in viele Sackgassen führt und somit für den Anstieg der Berechnungszeit verantwortlich ist.

Kapitel 3.

Phasenübergänge beim Planen

PlanLen-$\mathcal{D}$ ist das folgende Problem:

- *Gegeben:* Eine Planungsinstanz t aus $\mathcal{D}$ und ein $l \in \mathbb{N}$.
- *Gefragt:* Gibt es für t einen Plan der Länge $\leq l$?

Wir betrachten folgende Fragestellung: Gegeben eine Domänenparameterbelegung v und $l \in \mathbb{N}$. Wie hoch ist die Wahrscheinlichkeit, dass eine zufällige PlanLen-$\mathcal{D}$-Probleminstanz (t, l) mit $v_t = v$ eine positive Instanz ist? Wir vermuten, dass es bei diesen Wahrscheinlichkeiten zu einem Phasenübergang kommt, und wollen den Ordnungsparameter r identifizieren, an dessen kritischem Wert der Phasenübergang stattfindet. Besonderes Interesse gilt dabei der Abhängigkeit des Ordnungsparameters r von der Domänenparameterbelegung v_t und von l.

3.1. Versuchsaufbau

Wir wollen die Instanzen der Domänen Miconic, Logistics und Grid empirisch untersuchen und versuchen, anhand der gewonnen Daten Rückschlüsse auf die kritischen Domänenparameter einer Domäne zu machen. Da im Allgemeinen $|\mathcal{V}| = \infty$ ist und $|\mathcal{M}^v_{\mathcal{D}}|$ für die meisten Domänenparameterbelegungen v zu groß wird, um alle $t \in \mathcal{M}^v_{\mathcal{D}}$ zu betrachten, sind unseren Untersuchungen natürliche Grenzen gesetzt, weshalb wir mit Stichproben arbeiten.

Zu diesem Zweck führen wir folgende Notation ein:

- $\mathcal{L} := \{1, \ldots, max\}$ sei die Menge der gefragten Planlängen.
- $\mathcal{S} := \{v^1, \ldots, v^n\}$ ist eine Stichprobe der Größe n aus $\mathcal{V}$.
- $\mathcal{T}_v := \{t^1, \ldots, t^k\}$ mit $t^i \in \mathcal{M}^v_{\mathcal{D}}$ ist eine Stichprobe der Größe k aus $\mathcal{M}^v_{\mathcal{D}}$.
- $\mathcal{I} = \{(t^i, l) \mid \exists v \in \mathcal{S} : t^i \in \mathcal{T}_v, l \in \mathcal{L}\}$ ist die Menge von PlanLen-$\mathcal{D}$-Probleminstanzen, die man aus den Stichproben bilden kann.

Wir geben in unseren Stichproben für jeden Domänenparameter $X_j \in \mathcal{X}$ ein Intervall $\mathcal{J}_j \subset \mathbb{N}$ von Werten an, anhand derer die Domänenparameterbelegungen im Lauf der Untersuchung gebildet werden. Somit ist $|\mathcal{S}| = |\mathcal{J}_1 \times \cdots \times \mathcal{J}_m|$, wobei jedes v^i einem Wert dieses Kreuzproduktes entspricht. Für jedes dieser v^i betrachten wir k Planungsinstanzen, womit wir also insgesamt $k \times |\mathcal{S}|$ Planungsinstanzen betrachten. Für die Menge $\mathcal{I}$ von PlanLen-$\mathcal{D}$-Probleminstanzen (t, l), die aus unseren Stichproben gebildet werden, d.h. für $k \times |\mathcal{S}|$ Planungsinstanzen und für max betrachtete Planlängen, folgt somit $|\mathcal{I}| = max \times k \times |\mathcal{S}|$.

Um nun einen geeigneten Ordnungsparameter angeben zu können, benötigen wir folgende Notation:

- $g : \mathcal{M}_\mathcal{D} \to \mathbb{N} \cup \{\infty\}$ weist einer Planungsinstanz ihre optimale Planlänge zu. Sollte die Planungsinstanz nicht lösbar sein, wird ihr die Länge ∞ zugewiesen.
- $\overline{g} : \mathcal{V} \to \mathbb{R} \cup \{\infty\}$ ordnet jeder Belegung v der Domänenparameter die mittlere optimale Planlänge aller Instanzen $t \in \mathcal{M}^v_\mathcal{D}$ zu. [1]
- $\hat{g} : \mathcal{S} \to \mathbb{R} \cup \{\infty\}$ ist ein Schätzer für $\overline{g}$, der definiert ist durch $\hat{g}(v_t) = \frac{1}{k} \cdot \sum_{i=1}^{k} g(t^i)$ für die Stichprobe $t^1, \ldots, t^k$ aus $\mathcal{M}^v_\mathcal{D}$.

Wir werden nun den Ordnungsparameter definieren, mit dessen Hilfe wir die Domänenparameter identifizieren können, die für die intrinsisch schweren Instanzen verantwortlich sind.

Definition 8 (Ordnungsparameter). Sei $\mathcal{I}$ die Menge aller PlanLen-$\mathcal{D}$-Probleminstanzen und $(t, l) \in \mathcal{I}$. Dann ist der Ordnungsparameter $r : \mathcal{I} \to \mathbb{R}$ eine Funktion, so dass für alle $l \in \mathbb{N}$ gilt:

$$r(t, l) = \frac{l}{\overline{g}(v_t)}.$$

Der Ordnungsparameter ist eine einfache, schnell berechenbare Funktion, die jeder Probleminstanz einen reellen Wert zuordnet. Der kritische Wert des Ordnungsparameters ist 1. An dieser Stelle findet der Phasensprung statt. Viele der wirklich schweren Instanzen bekommen einen Wert in der Nähe der 1 zugeordnet. Die Instanzen, die weit von der Phasengrenze entfernt liegen, bekommen entsprechend einen Wert, der deutlich größer oder deutlich kleiner als 1 ist. Für diese Instanzen kann häufig leicht gezeigt werden, dass es sich um positive bzw. negative Instanzen handelt.

Sollten wir $\overline{g}$ nicht zur Verfügung haben, so benutzen wir den geschätzten Ordnungsparameter.

[1] $\overline{g}$ ist wohldefiniert, da es bei den von uns betrachteten Domänen zu jeder Belegung v immer nur endlich viele Planungsinstanzen gibt.

Definition 9 (Geschätzer Ordnungsparameter). Sei $\mathcal{I}$ die Menge der PlanLen-$\mathcal{D}$-Probleminstanzen mit $I = (t, l) \in \mathcal{I}$, wobei $t \in \mathcal{T}_v$. Dann ist der geschätzte Ordnungsparameter $\hat{r} : \mathcal{S} \times \mathcal{L} \to \mathbb{R}$ eine Funktion, so dass für alle $l \in \mathcal{L}$ gilt:

$$\hat{r}(t, l) = \frac{l}{\hat{g}(v_t)}$$

Da für alle $t^i \in \mathcal{T}_v$ für ein gegebenes $l \in \mathcal{L}$ gilt

$$\hat{r}(t^1, l) = \hat{r}(t^2, l) = \cdots = r(t^k, l),$$

können wir den geschätzten Ordnungsparameter auch als Funktion $\hat{r} : \mathcal{S} \times \mathcal{L} \to \mathbb{R}$ verstehen mit

$$\hat{r}(v, l) = \frac{l}{\hat{g}(v)}.$$

Definition 10 (Phasenübergangsfunktion). Sei $\mathcal{I}$ die Menge aller PlanLen-$\mathcal{D}$-Probleminstanzen und $(t, l) \in \mathcal{I}$, $\mathcal{P} \subseteq \mathcal{I}$ die Menge der positiven Instanzen und r der entsprechende Ordnungsparameter.

Dann gilt für die *Phasenübergangsfunktion* P: $\mathrm{ran}(r) \to [0, 1] \subset \mathbb{R}$

$$P(u) = \Pr(I \in \mathcal{P} \mid I \in \mathcal{I}, r(I) = u).$$

Wir wollen mit Hilfe unserer Experimente folgenden Vermutungen nachgehen:

- Falls $l < \overline{g}$, dann $\Pr(I \in \mathcal{P} \mid I \in \mathcal{I}, r(I) = u) \to 0$ für $\|I\| \to \infty$
- Falls $l > \overline{g}$, dann $\Pr(I \in \mathcal{P} \mid I \in \mathcal{I}, r(I) = u) \to 1$ für $\|I\| \to \infty$
- Falls $l = \overline{g}$, dann $\Pr(I \in \mathcal{P} \mid I \in \mathcal{I}, r(I) = u) \to 0{,}5$ für $\|I\| \to \infty$

Somit gilt: Wenn $u > 1$, dann handelt es sich vermutlich um eine positive Instanz des Entscheidungsproblems, falls $u < 1$, handelt es sich wahrscheinlich um eine negative Instanz und falls $u = 1$, ist dies die Sprungstelle des Phasenübergangs.

Definition 11 (Phasenübergangsrelation). Sei $\mathcal{I}$ eine Menge von PlanLen-$\mathcal{D}$-Probleminstanzen und $(t, l) \in \mathcal{I}$, $\mathcal{P} \subseteq \mathcal{I}$ die Menge der positiven Instanzen und $\hat{r}$ der entsprechende geschätzte Ordnungsparameter. Sei $\varepsilon > 0$.

Seien $a_l^v := \{t \in \mathcal{T}_v \mid g(t) \leq l\}$, $P_l^v := |a_l^v| / k$ für alle $v \in \mathcal{S}$ und $l \in \mathcal{L}$ sowie $R := \{(u, p) \in \mathbb{R}^2 \mid \exists v \in \mathcal{V}, \exists l \in \mathcal{L} : P_l^v = p, \hat{r}(v, l) = u\}$.

Dann definieren wir die *Phasenübergangsrelation* Φ_ε: $\mathrm{ran}(\hat{r}) \to [0, 1] \subset \mathbb{R}$:

$$\Phi_\varepsilon(u) = \frac{\sum_{(u', p) \in R_u} p}{|R_u|}, \text{ wobei } R_u := \{(u', p) \in R \mid u \leq u' < u + \varepsilon\}.$$

Beachte, dass Φ_ε nur für u aus $\mathrm{ran}(\hat{r})$ definiert ist und es somit keine Division durch Null geben kann.

Neben der Wahrscheinlichkeit, mit der eine PlanLen-$\mathcal{D}$-Instanz I lösbar ist, ist für uns auch der benötigte Berechnungsaufwand, um eine PlanLen-$\mathcal{D}$-Instanz I zu entscheiden, von Interesse und im Besonderen auch die Korrelation dieser beiden Größen.

Sei $e(t, l)$ die Laufzeit des PlanLen-$\mathcal{D}$-Entscheidungsalgorithmus 4 der in Teilabschnitt 3.2.1 beschriebenen Implementierung.

Definition 12 (Laufzeitkurve). Sei $\mathcal{I}$ eine Menge von PlanLen-$\mathcal{D}$-Probleminstanzen und $(t, l) \in \mathcal{I}$, $\mathcal{P} \subseteq \mathcal{I}$ die Menge der positiven Instanzen und $\hat{r}$ der entsprechende geschätzte Ordnungsparameter. Sei $\varepsilon > 0$.

Seien $z_l^v := \sum_{t \in \mathcal{T}_v} e(t, l)$, $T_l^v := z_l^v / k$ für alle $v \in \mathcal{S}$ und $l \in \mathcal{L}$ sowie $R := \{(u, \tau) \in \mathbb{R}^2 \mid \exists v \in \mathcal{V}, \exists l \in \mathcal{L} : T_l^v = \tau, \hat{r}(v, l) = u\}$.

Dann definieren wir die *Laufzeitkurve* Ψ_ε*:* $\mathrm{ran}(\hat{r}) \to [0, 1] \subset \mathbb{R}$:

$$\Psi_\varepsilon(u) = \frac{\sum_{(u', \tau) \in R_u} \tau}{|R_u|}, \text{ wobei } \; R_u := \{(u', \tau) \in R \mid u \leq u' < u + \varepsilon\}$$

Beachte, dass Ψ_ε nur für u aus $\mathrm{ran}(\hat{r})$ definiert ist und es somit keine Division durch Null geben kann.

3.2. Versuchsdurchführung

In einem ersten Schritt werden wir die Funktion der mittleren Planlänge $\overline{g}$ genauer untersuchen und veranschaulichen, in welcher Weise die mittlere Planlänge von der Belegung der einzelnen Domänenparameter X_j abhängig ist. Im darauf folgenden Schritt werden wir für jedes u die relative Häufigkeit $\Phi(u)$ der positiver Instanzen berechnen und die *Phasenübergangsfunktion* P entsprechend den Werten u und $\Phi(u)$ angeben. In einem letzten Schritt werden wir die Laufzeitkurve berechnen.

3.2.1. Verwendete Hilfsmittel

Im Folgenden wollen wir kurz die Hilfsmittel vorstellen, die bei unseren Experimenten zum Einsatz kommen. Es handelt sich dabei einerseits um Problemgeneratoren, mit deren Hilfe wir zufällige Instanzen einer Domäne erstellen können, und andererseits um Planer, mit deren Hilfe wir für eine Instanz einen optimalen Plan berechnen und dessen optimale Planlänge angeben können.

Problemgeneratoren

Die von uns verwendeten Problemgeneratoren, die für die automatische Generierung der Instanzen verantwortlich sind, nehmen als Eingabe eine Domänenparameterbelegung v und liefern als Ausgabe eine zufällig generierte Instanz t_v entsprechend diesen

Werten. Der LOGISTICS-*Problemgenerator* dient sowohl zur Erzeugung der Instanzen der LOGISTICS-Domäne als auch zur Erzeugung der Instanzen der MICONIC-Domäne (unter den entsprechenden Einschränkungen, siehe Kapitel 2). Sowohl der LOGISTICS-Problemgenerator als auch der GRID-Problemgenerator stammen von Hoffmann [2003].

Erzeugung der Miconic-Instanzen:

Der LOGISTICS-Problemgenerator, der zur Generierung unserer MICONIC-Instanzen verwendet wird, erhält als Eingabe folgende Domänenparameter:

- o für die Anzahl der Orte
- p für die Anzahl der Pakete

Bei den von dem LOGISTICS-Problemgenerator erzeugten Instanzen ist die Verteilung der Start- und Zielorte der Pakete zufällig, wobei im Unterschied zum Originalgenerator von Hoffmann [2003] die von uns leicht abgewandelte Version Instanzen liefert, bei denen jedes Paket unterschiedliche Anfangs- und Zielpositionen hat.

Erzeugung der Logistics-Instanzen:

Der LOGISTICS-Problemgenerator erhält zum Erzeugen unserer LOGISTICS-Instanzen folgende Domänenparameter als Eingabe:

- a für die Anzahl der Flugzeuge
- c für die Anzahl der Städte
- o für die Anzahl der Orte pro Stadt
- p für die Anzahl der Pakete

Der LOGISTICS-Problemgenerator erzeugt Instanzen, bei denen die Verteilung der Start- und Zielorte der Pakete zufällig sind. Auch hier haben wir den Generator so modifiziert, dass jedes Paket unterschiedliche Anfangs- und Zielpositionen hat.

Erzeugung der Grid-Instanzen:

Unser Problemgenerator erhält als Eingabe folgende Domänenparameter:

- x und y für die Ausdehnung des Gittergraphen
- t für die Anzahl der Tür- und Schlüsseltypen
- k für die Anzahl der Schlüssel
- l für die Anzahl der verschlossenen Türen
- p für den prozentualen Anteil (als natürliche Zahl zwischen 1 und 100) von Zielschlüsseln unter allen Schlüsseln

Die Menge der Schlüssel, die an eine Zielposition gebracht werden müssen, wird zufällig aus der Gesamtmenge der vorgegebenen Schlüssel bestimmt. Der prozentuale Anteil

davon kann mit dem Parameter p angeben werden. Alle Ausgangs- und Zielpositionen werden zufällig bestimmt.

Planer

Um ausschließen zu können, dass manche Phänomene spezifisch für den von uns verwendeten Planer sind, werden wir die Experimente mit zwei verschiedenen domänenunabhängigen Planern ausführen, die auf unterschiedlichen Konzepten beruhen.

Hsp0: Der Planer *Hsp0* von Patrik Haslum ist ein domänenunabhängiger Planer. Er führt eine Rückwärtssuche durch, die durch die zulässige Heuristik h^2 geleitet wird. Mit diesem Planer ist es auch möglich, optimal zu planen.

> „The planner performs a regression search, guided by an admissible heuristic called h^2. The h^2 heuristic is defined by approximating the cost of sets of (subgoal) atoms by the cost of the most costly subset of size at most 2. It is computed (by a dynamic programming method) and stored in a table before search begins.“

[Haslum und Geffner, 2004]

LPG: Der LPG-Planer ist ebenfalls ein domänenunabhängiger Planer. Dieser Planer basiert auf Planungsgraphen und lokaler Suche. Die von uns verwendete Version LPG-td wurde auf der vierten International Planning Competition 2004 (IPC4) als „Top performer at IPC4 in domains involving ‚Timed Initial Literals‘“ und als „Top performer at IPC4 in plan quality (satisfycing planning track)“ ausgezeichnet. Diesem Planer kann man eine Zeit vorgeben, in der er versucht, den bereits gefundenen Plan zu verbessern.

> „[. . .] LPG, a domain-independent planner that took part in the 3rd International Planning Competition (IPC). LPG is an incremental, any time system producing multi-criteria quality plans. The core of the system is based on a stochastic local search method and on a graph-based representation called „Temporal Action Graphs“ (TA-graphs). [. . .] Often LPG outperforms all other fully-automated planners of the 3rd IPC in terms of speed to derive a solution, or quality of the solutions that can be produced.“

[Gerevini, Saetti und Serina, 2003]

Rechner

Alle Experimente wurden auf einem Rechner mit 3-GHz-Intel-Xeon-Prozessor mit 6 GB RAM unter dem Betriebssystem SuSE Linux 10.0 durchgeführt.

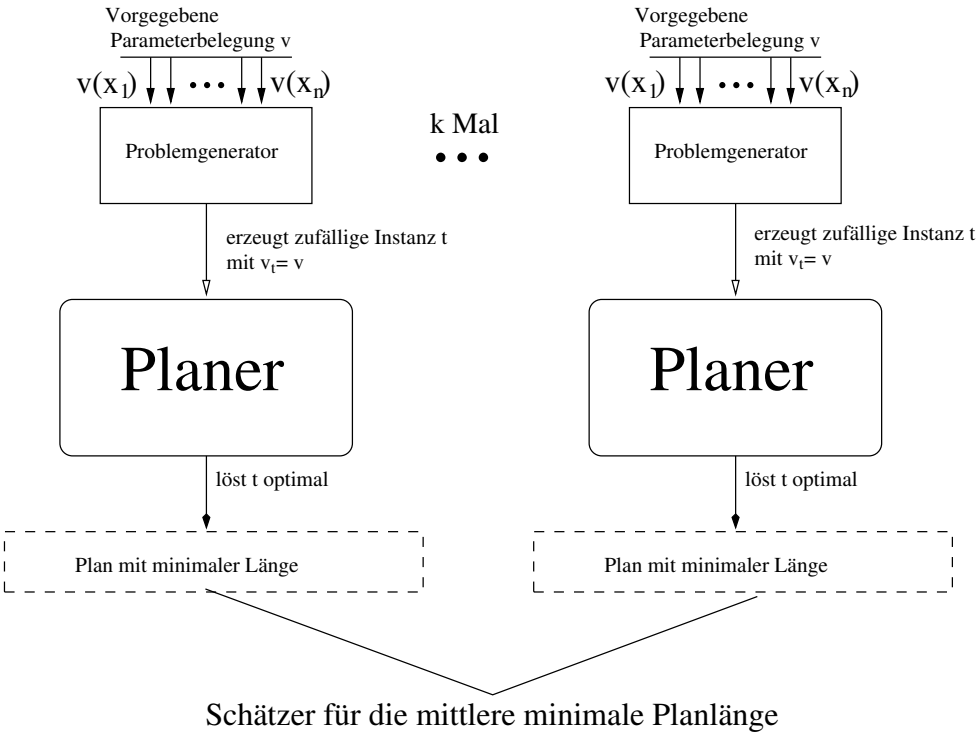

Abb. 3.1.: Versuchsaufbau zur Berechnung der mittleren Planlänge

3.2.2. Experimentelle Bestimmung der Funktion der optimalen mittleren Planlänge

Im Folgenden wollen wir für jede Domänenparameterbelegung v^i aus $\mathcal{S}$ jeweils den Schätzer $\hat{g}(v^i)$ für die mittlere Planlänge $\overline{g}(v^i)$ bestimmen.

Wir berechnen $\hat{g}(v^i)$ aus unseren Stichproben mit Algorithmus 1.

Algorithmus 1 Geschätzte mittlere Planlänge

1: **for all** $v \in \mathcal{S}$ **do**
2: **for** $i = 1, \ldots, k$ **do**
3: $t^i \leftarrow$ PROBLEMGENERATOR(v)
4: $g(t^i) \leftarrow$ OPTIMALPLANLENGTH(t^i)
5: **end for**
6: $\hat{g}(v) \leftarrow \frac{1}{k} \cdot \sum_{i=1}^{k} g(t^i)$
7: **end for**

Wie in Abbildung 3.1 zu sehen, bestimmen wir den Schätzer der mittleren optimalen Planlänge, indem wir für jedes $v \in \mathcal{S}$ mittels eines Problemgenerators k zufällige Planungsinstanzen $t^1, \ldots, t^n$ erstellen, für diese optimal planen und dann über die jeweiligen minimalen Planlängen mitteln.

3.2.3. Berechnung der Phasenübergangsfunktion

In Algorithmus 3 werden wir die Phasenübergangsfunktion berechnen. Wir berechnen dabei für jedes $v \in \mathcal{S}$ (a) die mittlere optimale Planlänge $\hat{g}(v)$ aller $t \in \mathcal{T}_v$ und daraus für alle $l \in \mathcal{L}$ den geschätzen Ordnungsparameter $\hat{r}(v, l)$. Ferner berechnen wir (b) für alle $l \in \mathcal{L}$ die relative Häufigkeit P_l^v der *positiven* Instanzen unter allen Instanzen (t, l).

Dazu benutzen wir folgende Datenstrukturen:

- Ein Array a der Länge max, sodass a_l die Anzahl der positiven Instanzen (t, l) mit $t \in \mathcal{T}_v$ angibt.
- Ein Array u, dessen $|S| \cdot max$ Einträge u_l^v die Werte $\hat{r}(v, l)$ für alle $l \in \mathcal{L}$ und $t \in \mathcal{T}$ sind.
- Ein Array P der Länge $|S| \cdot max$ für die Einträge P_l^v.

Algorithmus 2 Berechnung des Phasenübergangs

```
1: a ← 0  ∀l = 1, ..., max
2: for all v ∈ S do
3:     for i = 1, ..., k do
4:         t^i ← ProblemGenerator(v)
5:         g(t^i) ← optimalPlanLength(t^i)
6:         for l = g(t^i), ..., max do
7:             a_l ← a_l + 1
8:         end for
9:     end for
10:    ĝ(v) ← (1/k) · Σ_{i=1}^{k} g(t^i)
11:    for l = 1, ..., max do
12:        u_l^v ← l / ĝ(v)
13:        P_l^v ← a_l / k
14:        a_l ← 0
15:    end for
16: end for
```

Um die Daten auszugeben werden sie, wie in Algorithmus 3 beschrieben, geglättet. Wir wählen dazu als Definitionsbereich der Phasenübergangsfunktion disjunkte Intervalle $U_1, \ldots, U_{upper}$ der Größe ε, in die zum Teil mehrere Werte unserers Ordnungsparameters fallen. Dabei wird über die entsprechenden relativen Häufigkeiten aller Ordnungsparameterwerte, die in das gleiche Intervall fallen, gemittelt. Somit wird jedem Intervall eine gemittelte relative Häufigkeit zugeordnet.

3.2.4. Berechnung des Zeitverbrauchs

Wir wollen nun am Beispiel der Miconic-Domäne zeigen, wie wir den benötigten Zeitverbrauch zum Entscheiden unserer PlanLen-Miconic-Instanzen ermitteln. In einem

Algorithmus 3 Geglätteter Phasenübergang

1: $upper \leftarrow max(u)$
2: **for** $i = 0, \ldots, upper$ **do**
3: $\quad range_i \leftarrow 0$
4: $\quad counter_i \leftarrow 0$
5: **end for**
6: **for all** $l \in \mathcal{L}$ and $v \in \mathcal{S}$ **do**
7: $\quad idx \leftarrow \lfloor \frac{u_l^v}{\varepsilon} \rfloor$
8: $\quad range_{idx} \leftarrow range_{idx} + P_l^v$
9: $\quad counter_{idx} = counter_{idx} + 1$
10: **end for**
11: **for** $i = 0, \ldots, upper$ **do**
12: $\quad range_{idx} \leftarrow \frac{range_{idx}}{counter_{idx}}$
13: **end for**

ersten Schritt werden wir zeigen, dass die optimale Planlänge der MICONIC-Domäne für eine größer werdende Anzahl von Orten nach oben beschränkt ist und gegen die kleinste obere Schranke konvergiert. In einem zweiten Schritt bestimmen wir Heuristiken, mit denen wir die Mindest- und Höchstanzahl benötigter Planungsschritte abschätzen, die zum optimalen Lösen einer Planungsinstanz t mit Domänenparameterbelegung v_t benötigt werden. Dieses Wissen wollen wir nutzen, um in einem dritten Schritt einen effizienten Entscheidungsalgorithmus anzugeben, bei dem wir auch die Zeit zum Entscheiden von PLANLEN-MICONIC Instanzen messen wollen.

Konvergenzbeweis

Im Folgenden zeigen wir, dass die mittlere optimale Planlänge der MICONIC-Instanzen für eine steigende Anzahl von Orten gegen die kleinste obere Schranke $4p$ konvergiert.

Ist p die Anzahl der Pakete, so seien $s_1, \ldots s_p$ die Start- und $z_1, \ldots z_p$ die Zielorte der Pakete. Sei ferner s_0 der Startort des LKW. Wir nehmen im Folgenden an, dass $s_0 \neq s_i$ für $i = 1, \ldots, p$ gilt. Im Allgemeinen liegen die Plankosten einer MICONIC-Instanz zwischen 0 (da sich alle Pakete schon am Zielort befinden könnten) und $4p$ (hinfahren, aufladen, wegfahren, abladen). Sie betragen genau dann $4p$, wenn $s_1, \ldots, s_p, z_1, \ldots z_p$ paarweise verschieden sind.

Wir führen folgende Notation ein: Sei t eine MICONIC-Planungsinstanz und v_t die entsprechende Domänenparameterbelegung. Dann sei:

- O der Domänenparameter Orte und P der Domänenparameter Pakete
- $v_{n,p} = \{(O, n), (P, p)\}$ die Domänenparameterbelegung mit n Orten und p Paketen
- $N := \left|\mathcal{M}_{Miconic}^{v_{n,p}}\right| < \infty$
- $\mathcal{M}_{Miconic}^{v_{n,p,l}} := \{t \in \mathcal{M}_{Miconic}^{v_{n,p}} \mid g(t) = l\}$
- $N_l := \left|\mathcal{M}_{Miconic}^{v_{n,p,l}}\right|$

Damit gilt:

- $\Pr(g(t) = l \mid t \in \mathcal{M}^{v_{n,p}}_{Miconic}) = \frac{N_l}{N}$
- $\overline{g}_p(n) = \sum_{l=0}^{4p} \frac{N_l}{N} l$

Wir zeigen nun, dass für eine größer werdende Anzahl von Orten die Wahrscheinlichkeit, dass $s_1, \ldots, s_p, z_1, \ldots z_p$ paarweise verschieden sind, gegen eins geht. Damit können wir zeigen, dass für eine größer werdende Anzahl von Orten die mittlere optimale Planlänge gegen $4p$ geht, d.h.

$$\lim_{n\to\infty} \overline{g}_p(n) = 4p \tag{3.1}$$

Wegen $\lim_{n\to\infty} \overline{g}_p(n) = \lim_{n\to\infty} \sum_{l=0}^{4p} \frac{N_l}{N} \cdot l = \sum_{l=0}^{4p} l \cdot \lim_{n\to\infty} \frac{N_l}{N}$ reicht es zu zeigen, dass

$$\lim_{n\to\infty} \frac{N_{4p}}{N} = 1 \quad \text{und} \tag{3.2}$$

$$\lim_{n\to\infty} \frac{N_k}{N} = 0 \quad \text{für alle } k < 4p \tag{3.3}$$

gilt, da damit $\lim_{n\to\infty} \overline{g}_p(n) = \sum_{l=0}^{4p} l \cdot \lim_{n\to\infty} \frac{N_l}{N} = 4p$.

Gleichung (3.3) folgt aus Gleichung (3.2), denn

$$\lim_{n\to\infty} \frac{N_{4p}}{N} + \sum_{k=0}^{4p-1} \lim_{n\to\infty} \frac{N_k}{N} = \sum_{k=0}^{4p} \lim_{n\to\infty} \frac{N_k}{N} = \lim_{n\to\infty} \sum_{k=0}^{4p} \frac{N_k}{N} = \lim_{n\to\infty} 1 = 1 \quad \text{und} \tag{3.4}$$

$$\lim_{n\to\infty} \frac{N_k}{N} \geq 0 \quad \text{für alle } k \leq 4p \tag{3.5}$$

Die vorletzte Gleichheit in (3.4) sowie Gleichung (3.5) folgen unmittelbar aus der Definition der N_k und N.

Bleibt noch zu zeigen, dass Gleichung (3.2) gilt.

Sei $W_p(n)$ die Wahrscheinlichkeit, dass bei n Orten die Orte $s_1, \ldots, s_p, z_1, \ldots z_p$ paarweise verschieden sind. Dies entspricht der Fragestellung beim Geburtstagsparadoxon. Somit ist

$$W_p(n) = \frac{n \cdot (n-1) \cdot \ldots \cdot (n-2p+1)}{n \cdot n \cdot \ldots \cdot n}. \tag{3.6}$$

Es gilt $\lim_{n\to\infty} \frac{n-\ell}{n} = 1$ für alle $\ell \in \mathbb{N}$ [Forster, 1976]. Folglich ist

$$\lim_{n\to\infty} W_p(n) = \lim_{n\to\infty} \prod_{\ell=0}^{2p-1} \frac{n-\ell}{n} = \prod_{\ell=0}^{2p-1} \lim_{n\to\infty} \frac{n-\ell}{n} = \prod_{\ell=0}^{2p-1} 1 = 1 \tag{3.7}$$

Das Produkt $n \cdot (n-1) \cdot \ldots \cdot (n-2p+1)$ ist der Anzahl von Planungsinstanzen t, bei denen alle $s_1, \ldots, s_p, z_1, \ldots, z_p$ paarweise verschieden sind. Für jedes solche t wissen wir, dass $g(t) = 4p$ und dass t somit in $\mathcal{M}^{v_{n,p,4p}}_{Miconic}$ enthalten ist. Damit gilt

$$N_{4p} \geq n \cdot (n-1) \cdot \ldots \cdot (n-2p+1).$$

Division durch N ergibt

$$\frac{N_{4p}}{N} \geq \frac{n \cdot (n-1) \cdot \ldots \cdot (n-2p+1)}{N} = W_p(n),$$

weil $N = n^{2p}$ unter der Annahme gilt, dass der LKW anfangs an einem gesonderten Ort s_0 steht.

Grenzwertbildung ergibt zusammen mit Gleichung (3.6)

$$\lim_{n\to\infty} \frac{N_{4p}}{N} \geq \lim_{n\to\infty} W_p(n) = 1.$$

Da offensichtlich $\lim_{n\to\infty} \frac{N_{4p}}{N} \leq 1$, folgt daraus Gleichung (3.2). □

Miconic-Entscheider

Sei $S(t)$ eine untere Schranke für die Mindestanzahl von Planungsschritten, die man bei einer Planungsinstanz t benötigt. Entsprechend ist $G(t)$ eine obere Schranke. $J_{SG} = \{S(t), S(t)+1, \ldots, G(t)\}$ ist das Intervall zwischen diesen beiden Schranken. Ein Entscheidungsalgorithmus A kann für alle gefragten l, die außerhalb von J_{SG} liegen, sofort „ja“ bzw. „nein“ ausgeben. Ist $l < S(t)$, so liegt eine negative Instanz des Entscheidungsproblems vor, in dem Fall, dass $l \geq G(t)$ ist, eine positive Instanz. Im Folgenden nehmen wir an, dass $s_i \neq z_i$ für alle $i = 1, \ldots, k$. Eine naive Wahl von $S(t)$ und $G(t)$ bilden die Werte $2p$ und $4p$. Falls $l \in J_{SG}$, muss A im schlimmsten Fall die optimale Planlänge der Planungsinstanz t bestimmen, um (t, l) zu entscheiden. Aus diesem Grund sind wir daran interessiert, die Größe von J_{SG} zu minimieren.

Im folgenden geben wir zwei Heuristiken zur Berechnung von $S(t)$ und $G(t)$ an.

Untergrenze Es wird gezählt, wieviele Ein- und Ausladeaktionen noch benötigt werden und wieviele Orte mindestens noch angefahren werden müssen. $S(t)$ ist also $2p$ zuzüglich der Anzahl unterschiedlicher anzufahrender Orte, d.h.

$$S(t) = 2p + |\{s_1, \ldots, s_p, z_1 \ldots, z_p\}|.$$

Mit diesem Wert wird die Anzahl der tatsächlich benötigten Aktionen niemals überschätzt.

Obergrenze Alle Pakete, die den gleichen Start- und Zielort haben, werden einer Gruppe G_g von Paketen zugeordnet. Dann ist $2(|G_g|-1)$ die Anzahl von Aktionen, die für diese Gruppe durch das Zusammenfassen von gemeinsamen Transportaktionen gegenüber dem gesonderten Transport jedes einzelnen Pakets eingespart werden.

Ist K die Anzahl solcher Gruppen, dann gilt für die obere Schranke $G(t)$:

$$G(t) = 4p - 2\sum_{g=1}^{K}(|G_g| - 1)$$

Messung der Laufzeit des Entscheidungsalgorithmus

Wie in Algorithmus 4 zu sehen, berechnen wir die Laufzeitkurve, indem wir die Untergrenze und Obergrenze verwenden. Liegt das gefragte l außerhalb von J_{SG}, nehmen wir an, dass das Entscheiden keine Zeit verbraucht. Ist $l \in J_{SG}$, planen wir für die Planungsinstanz t optimal und messen die Laufzeit dieses Planungsvorgangs.

Algorithmus 4 Berechnung der Laufzeitkurve

$z \leftarrow 0 \quad \forall l = 1, \ldots, max$
for all $v \in \mathcal{S}$ **do**
  for $i = 1, \ldots, k$ **do**
    $t^i \leftarrow$ PROBLEMGENERATOR(v)
    $start \leftarrow Time.now$
    $g(t^i) \leftarrow$ OPTIMALPLANLENGTH(t^i)
    $stop \leftarrow Time.now$
    for $l = 1, \ldots, max$ **do**
      if $S(t^i) \leq l < G(t^i)$ **then**
        $z_l \leftarrow z_l + stop - start$
      end if
    end for
  end for
  for $l = 1, \ldots, max$ **do**
    $u_l^v \leftarrow \frac{l}{\tilde{g}(v)}; \quad T_l^v \leftarrow \frac{z_l}{k}; \quad z_l \leftarrow 0$
  end for
end for

Algorithmus 5 Geglättete Laufzeitkurve

$upper \leftarrow max(u)$
for $i = 0, \ldots, upper$ **do**
  $range_i \leftarrow 0$; $counter_i \leftarrow 0$
end for
for all $l \in \mathcal{L}$ and $v \in \mathcal{S}$ **do**
  $idx \leftarrow \lfloor \frac{u_l^v}{\varepsilon} \rfloor$
  $range_{idx} \leftarrow range_{idx} + T_l^v; \quad counter_{idx} = counter_{idx} + 1$
end for
for $i = 0, \ldots, upper$ **do**
  $range_{idx} \leftarrow \frac{range_{idx}}{counter_{idx}}$
end for

Kapitel 4.

Ergebnisse

4.1. Miconic

In Abbildung 4.1 sind innerhalb der Intervalle, die in unserem Experiment betrachtet wurden, für sämtliche Belegungen der Domänenparameter $Orte, Pakete$, die Werte der entsprechenden mittleren Planlängen aufgetragen. Die Punktmenge weist dabei entlang der Achse, auf der die Orte aufgetragen sind, wie nach unserem Konvergenzbeweis erwartet, ein beschränktes Wachstumsverhalten auf. Entlang der Achse der Pakete wächst die Punktmenge linear. Um dieses Verhalten zu verdeutlichen, projizieren wir in Abbildung 4.2 unsere Punktmenge auf die Achse, auf der die Anzahl der Orte läuft, und entsprechend in Abbildung 4.3 auf die Achse, auf der die Anzahl der Pakte läuft.

In Abbildung 4.2 sehen wir die Projektion unseres Datensatzes auf die Anzahl der Orte (x-Achse) denen jeweils auf der y-Achse die mittlere Planlänge zugeordnet wird. Es ist deutlich zu erkennen, dass sich die mittlere Planlänge immer der oberen Grenze $4p$ annähert.

In Abbildung 4.3 sehen wir die Projektion unseres Datensatzes auf die Anzahl der Pakete (x-Achse), denen jeweils auf der y-Achse die mittlere Planlänge zugeordnet wird. Es ist zu erkennen, dass die mittlere Planlänge linear bezüglich einer steigenden Anzahl von Paketen steigt. In Abbildung 5.2 ist dieses Verhalten noch deutlicher zu erkennen.

Die mittlere Planlänge steigt mit der Anzahl der Pakete, da jedes zusätzliche Paket für mindestens zwei zusätzliche Aktionen (auf- und abladen) und höchstens vier zusätzliche Aktionen (hinfahren, aufladen, wegfahren, abladen) sorgt.

In Abbildung 4.5 sehen wir die Phasenübergangsfunktion und die entsprechende Laufzeit zum Entscheiden einander gegenübergestellt. Es fällt auf, dass die Laufzeitspitze nach links verschoben ist. Vermutlich ist bei unserem Entscheidungsalgorithmus die obere Grenze exakter, d.h. für die positiven Instanzen im Bereich des Phasenübergangs ist das Intervall kleiner, für das optimal geplant werden müsste, da wir mit der Belegung von $G(t)$ die obere Intervallgrenze deutlich nach links verschieben konnten. Man kann also für eine Längenvorgabe, die sich knapp über der Intervallgrenze befindet, sofort angeben, dass es sich hierbei um eine positive Instanz handelt.

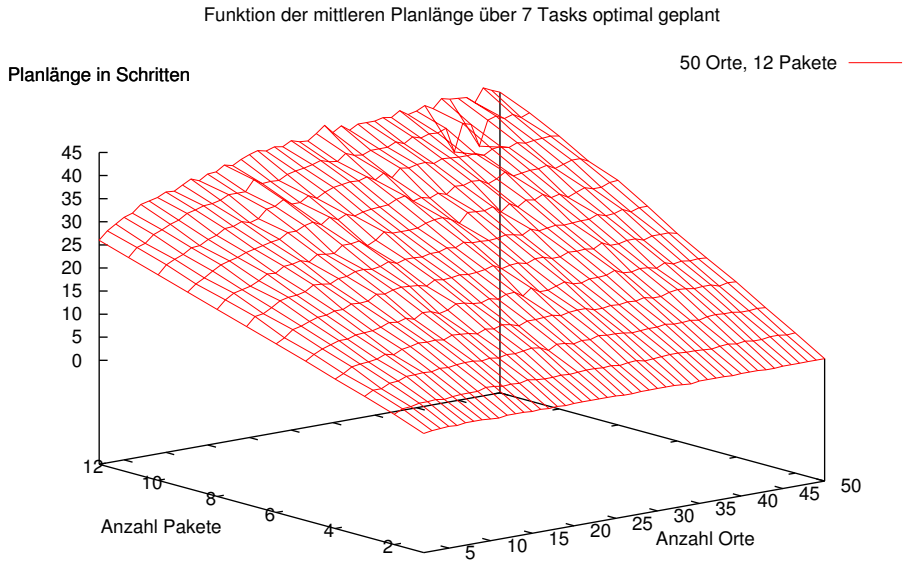

Abb. 4.1.: Miconic mittlere Planlänge – Gesamtansicht

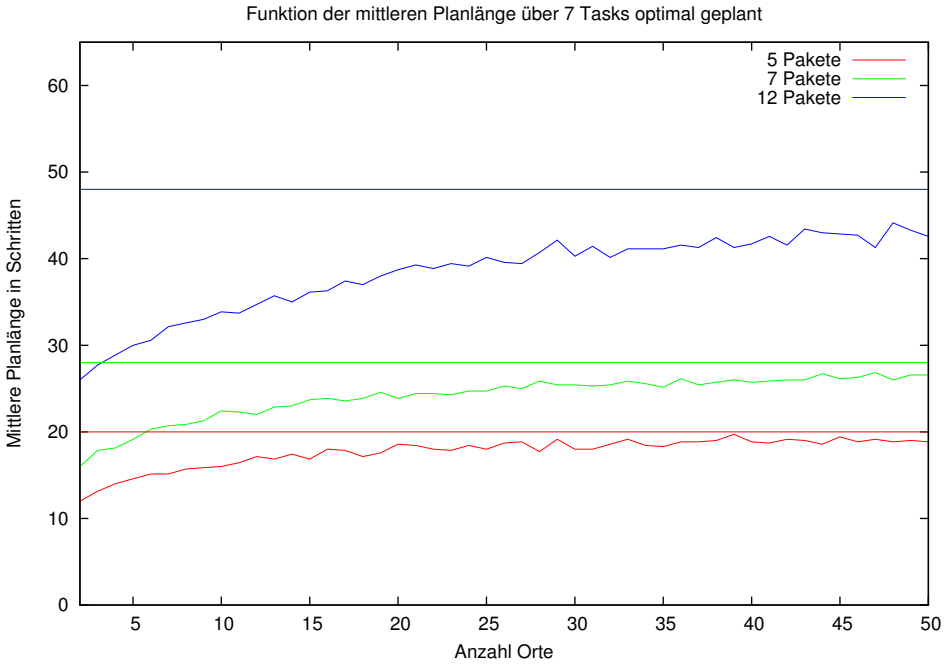

Abb. 4.2.: Miconic mittlere Planlänge – Projektion auf Orte

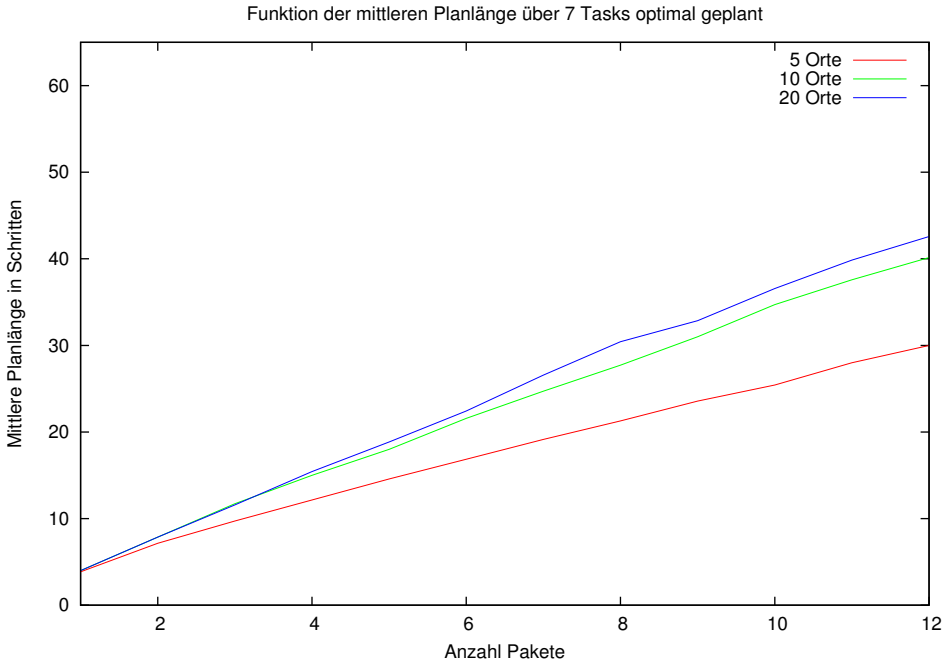

Abb. 4.3.: MICONIC mittlere Planlänge – Projektion auf Pakete

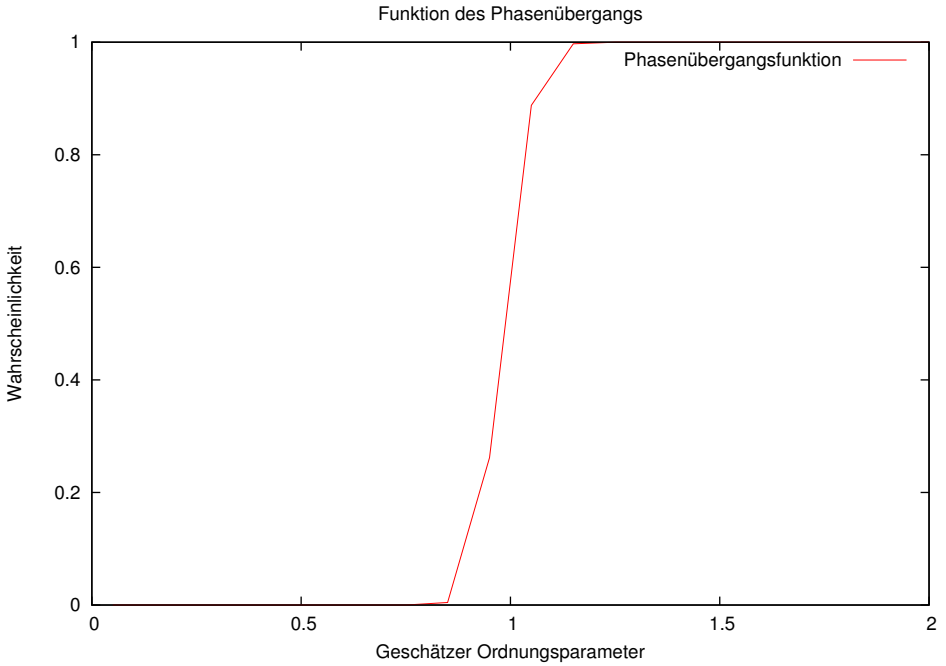

Abb. 4.4.: MICONIC Phasenübergangsfunktion

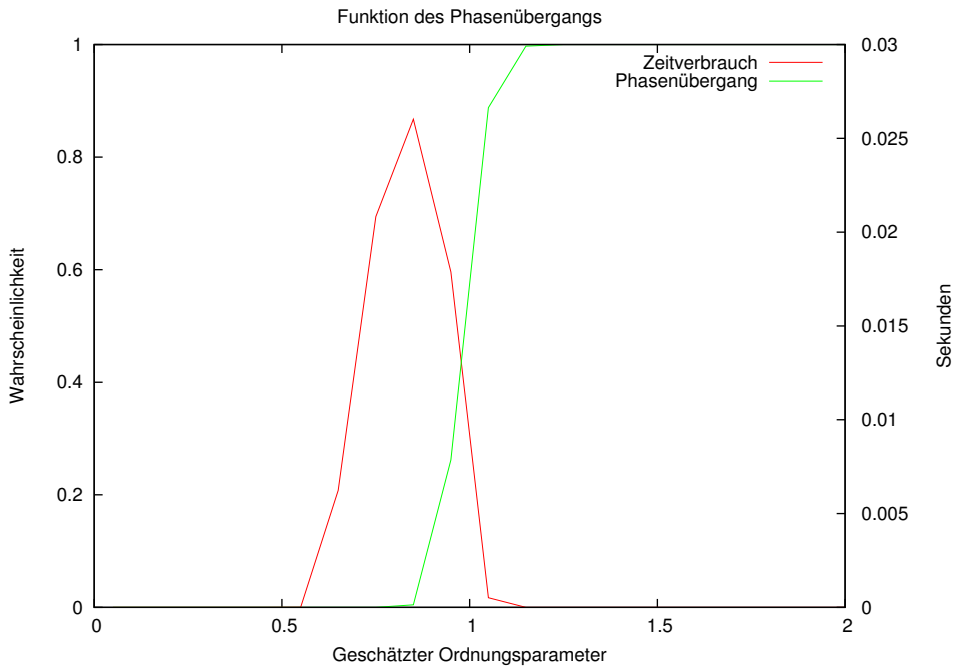

Abb. 4.5.: MICONIC Phasenübergangsfunktion und Laufzeit

4.2. Logistics

Bei der LOGISTICS-Domäne konnten wir nicht optimal planen, da die Laufzeiten aufgrund der vielen Domänenparameter beim optimalen Planen zu schnell anstiegen. Die Ergebnisse in den folgenden Schaubildern stehen somit alle nur für mittlere Planlängen.

In Abbildung 4.6 sehen wir die Projektion auf die Anzahl der Flugzeuge. Eigentlich würden wir hier erwarten, dass die mittlere Planlänge bei einer steigenden Anzahl von Flugzeugen konstant bleibt, oder sogar eher geringer wird.

In Abbildung 4.7 sehen wir die Projektion auf die Anzahl der Städte. Wir können hier ähnlich argumentieren wie schon beim beschränkten Wachstum der Orte bei MICONIC. Für eine steigende Anzahl von Städten nähert sich die Planlänge allerdings der oberen Grenze $12p$, bedingt durch folgende Aktionen:

$Hinfahren(LKW_{C1})$, $Aufladen(LKW_{C1})$, $FahrtFlugplatz(LKW_{C1})$,
$Abladen(LKW_{C1})$, $Hinfliegen(Flugzeug_{C1})$, $Aufladen(Flugzeug_{C1})$,
$Wegfliegen(Flugzeug_{C1})$, $Abladen(Flugzeug_{C1})$, $FahrtFlugplatz(LKW_{C2})$,
$Aufladen(LKW_{C2})$, $FahrtZielort(LKW_{C2})$, $Abladen(LKW_{C2})$.

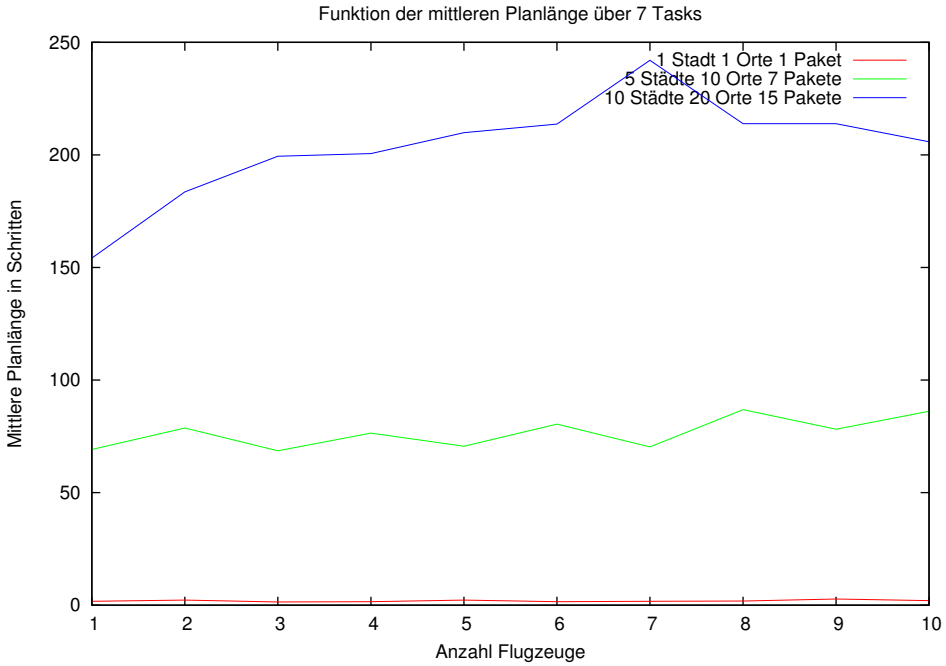

Abb. 4.6.: Logistics mittlere Planlänge – Projektion auf Anzahl Flugzeuge

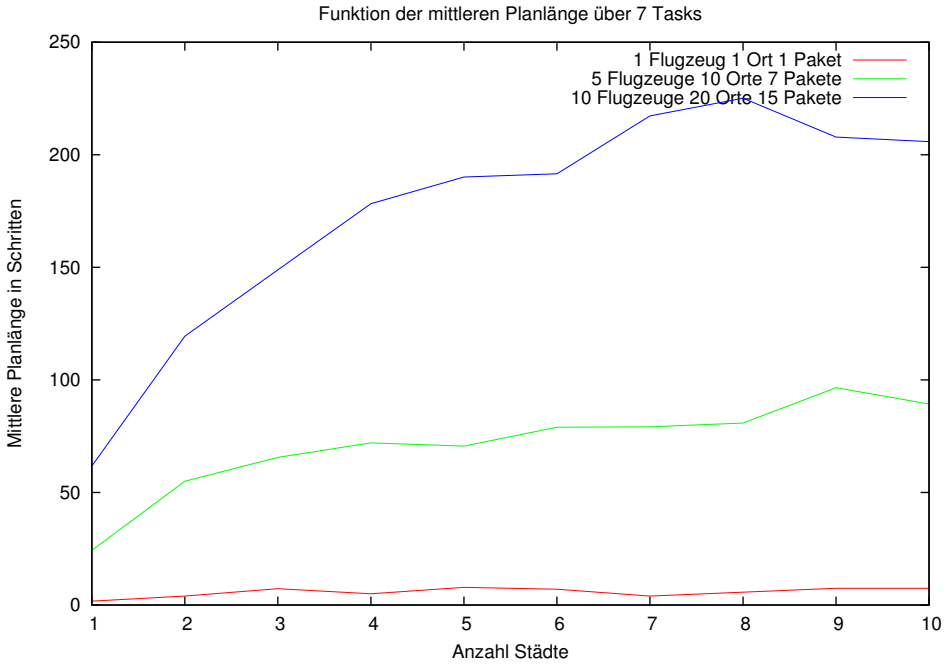

Abb. 4.7.: Logistics mittlere Planlänge – Projektion auf Städte

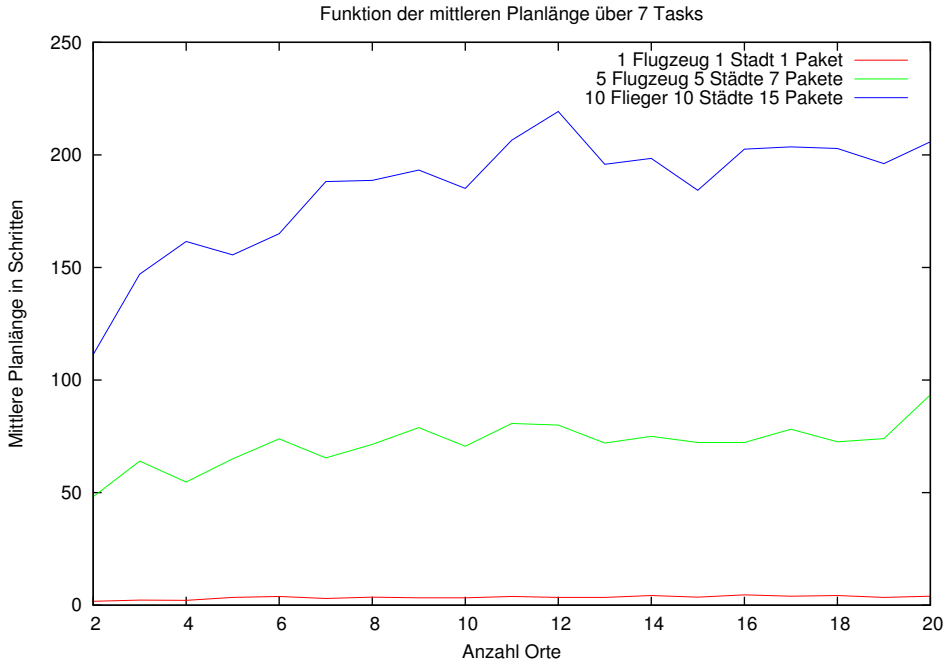

Abb. 4.8.: Logistics mittlere Planlänge – Projektion auf Orte

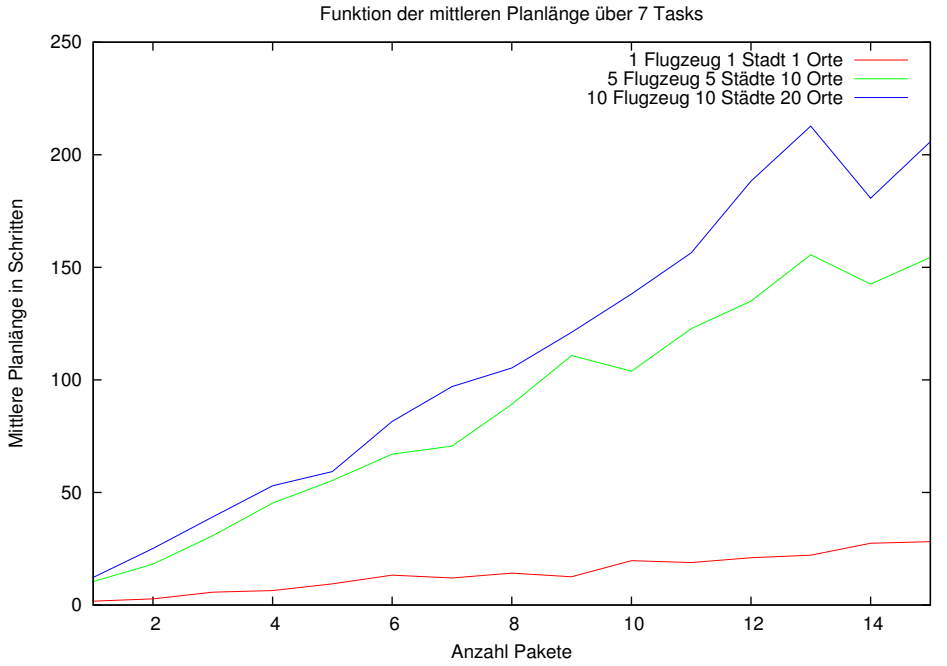

Abb. 4.9.: Logistics mittlere Planlänge – Projektion auf Pakete

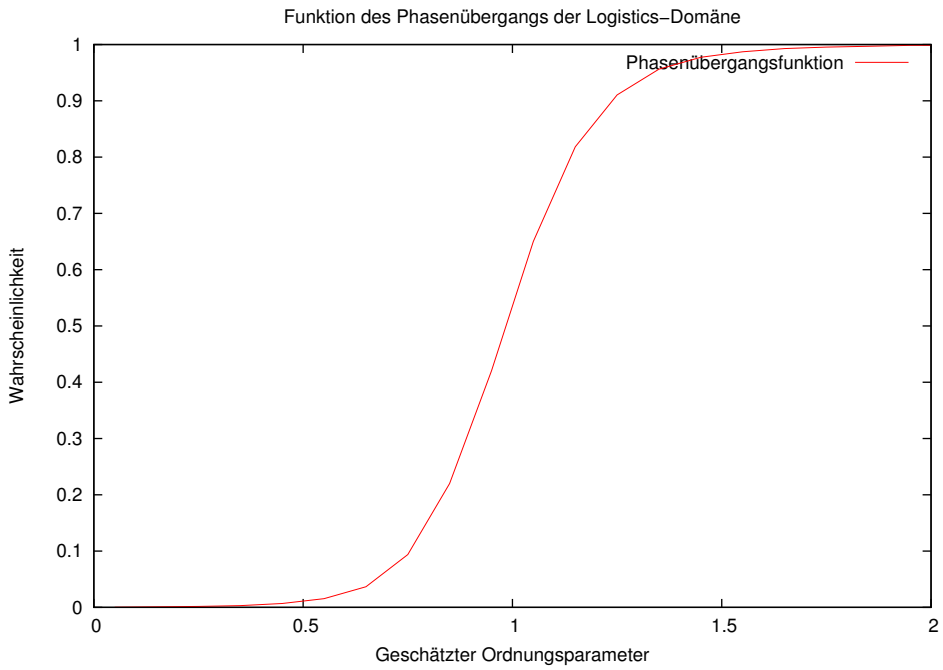

Abb. 4.10.: LOGISTICS-Phasenübergangsfunktion

4.3. Grid

Sei x der Domänenparameter für die Breite des Gittergraphen, und y der Domänenparameter für die Höhe des Gittergraphen, dann ist J_1 das Intervall, in dem sich die Werte für x bewegen und J_2 das Intervall, in dem sich die Werte für y bewegen. J_3 sei das Intervall, in dem die Werte für den Domänenparameter t, der für die Anzahl Tür- und Schlüsseltypen steht, bewegen. Im Intervall J_4 rangieren die Werte des Domänenparameters l, der für die Anzahl der Türen steht. J_5 ist das Intervall, in dem sich die Werte für den Domänenparameter k (Anzahl der Schlüssel) bewegen, und J_5 das Intervall für die Werte der prozentualen Anteile der Zielschlüssel.

Wir wählen die Intervalle, in denen sich die Domänenparameterbelegungen bewegen, auf folgende Weise:

- $x = y$ d.h. wir wollen nur quadratische Gittergraphen.
- $min(J_1) = \lceil \sqrt{i(J_4) + j(J_5)} \rceil$, wobei i der Laufindex von J_4 und j der Laufindex von J_5, da $min(J_1)$ dynamisch berechnet wird.
  Pro Feld darf maximal eine Tür plaziert werden. Schlüssel und Tür dürfen nicht zusammen auf einem Feld plaziert werden.
- In Intervall J_5 rangiert unsere Domänenparameterbelegung nicht über jeden Wert. Es wird nur jeder z-te Wert gewählt, wobei wir z vor dem Experiment mit angeben.

Bei der GRID-Domäne haben wir, wie schon bei der LOGISTICS-Domäne, auf optimales Planen verzichtet. In Abbildung 4.11 sind die einzigen Ergebnisse für optimal geplante

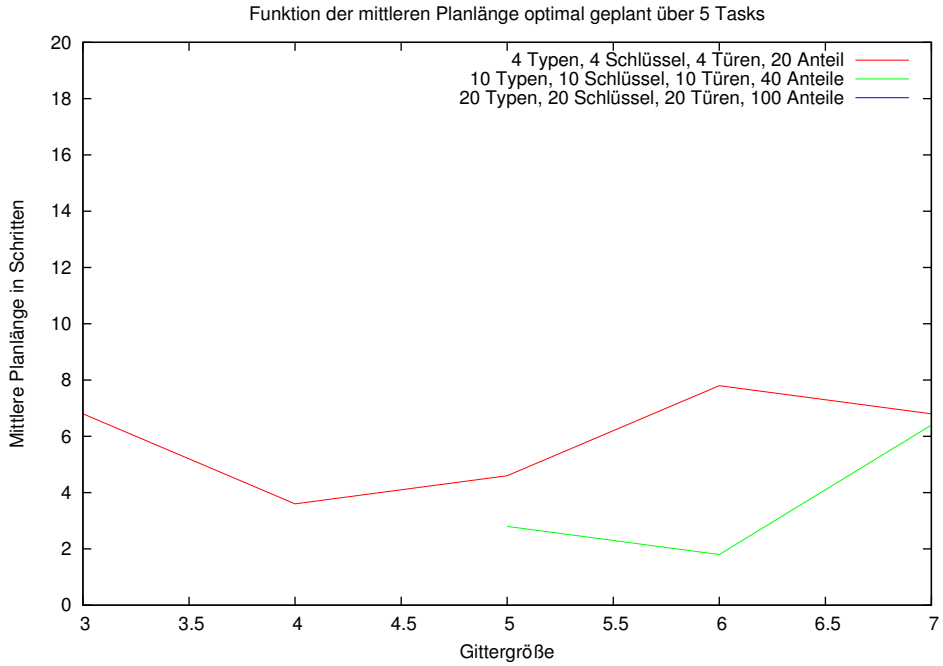

Abb. 4.11.: GRID mittlere Planlänge – Projektion auf die Gitterausmaße

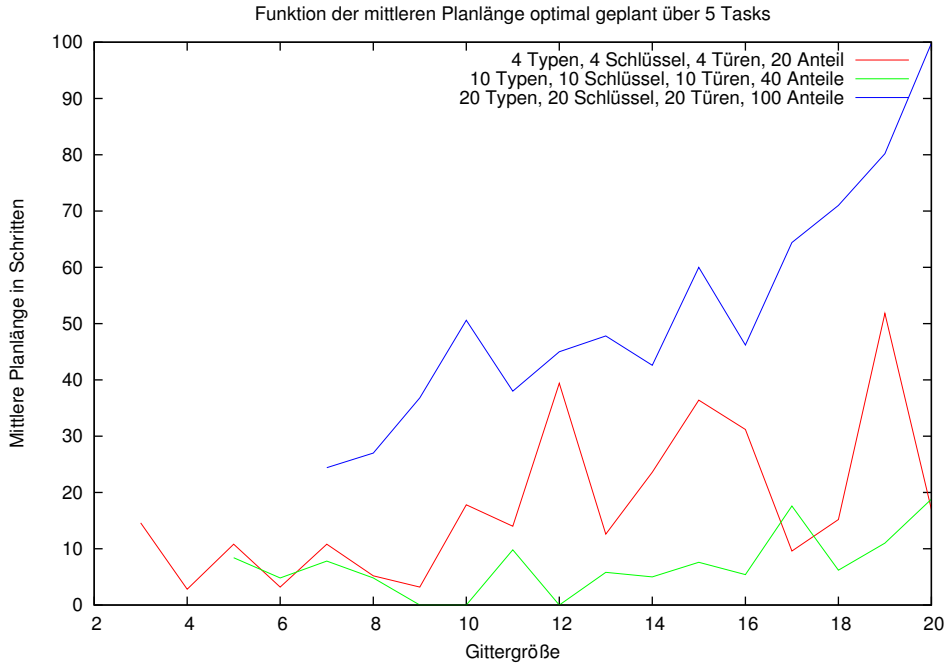

Abb. 4.12.: GRID mittlere Planlänge – Projektion auf die Gitterausmaße

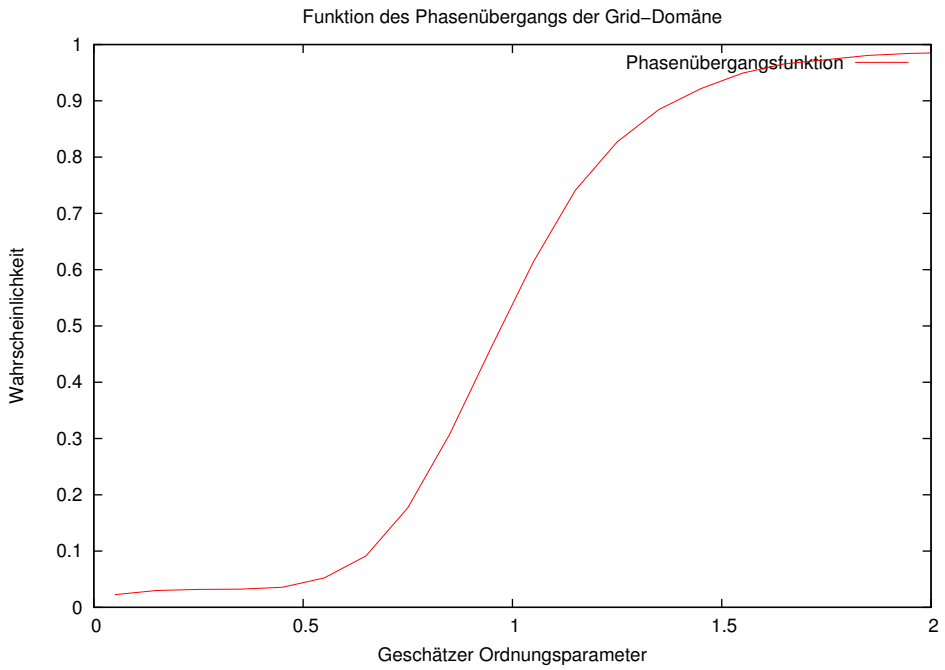

Abb. 4.13.: Grid mittlere Planlänge – Projektion auf die Gitterausmaße

mittlere Planlängen aufgetragen. Der Bereich, für den wir bei der Grid-Domäne optimal planen können, ist allerdings zu klein, um irgendwelche Aussagen treffen zu können.

Für die restlichen Schaubilder der Grid-Domäne wurde nicht optimal geplant. Die Daten in allen Schaubildern weisen ein sehr starkes Rauschen auf und sind daher ebenfalls ungeeignet, um Aussagen treffen zu können.

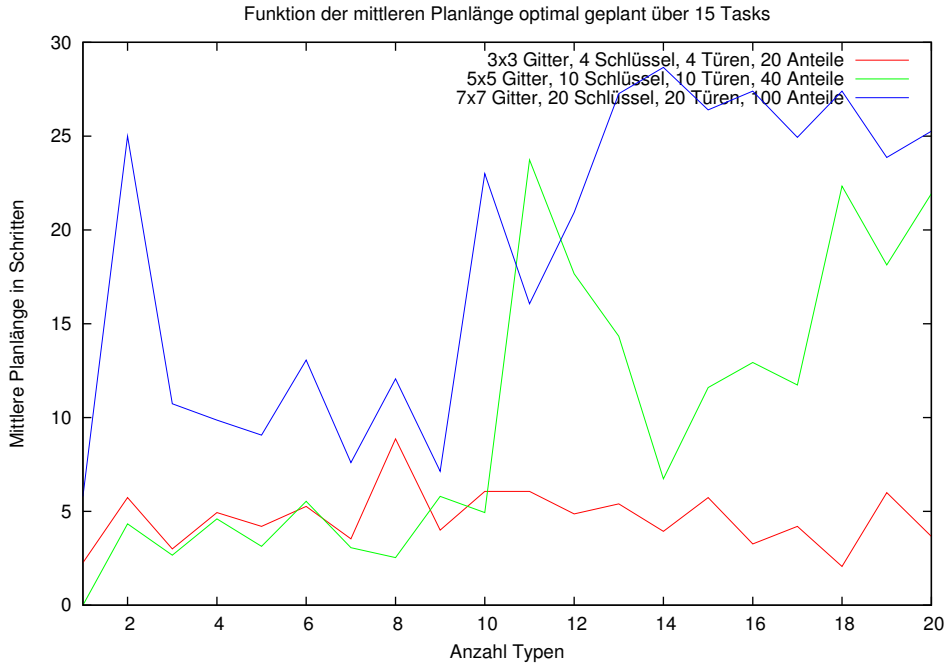

Abb. 4.14.: GRID mittlere Planlänge – Projektion auf die Anzahl der Typen

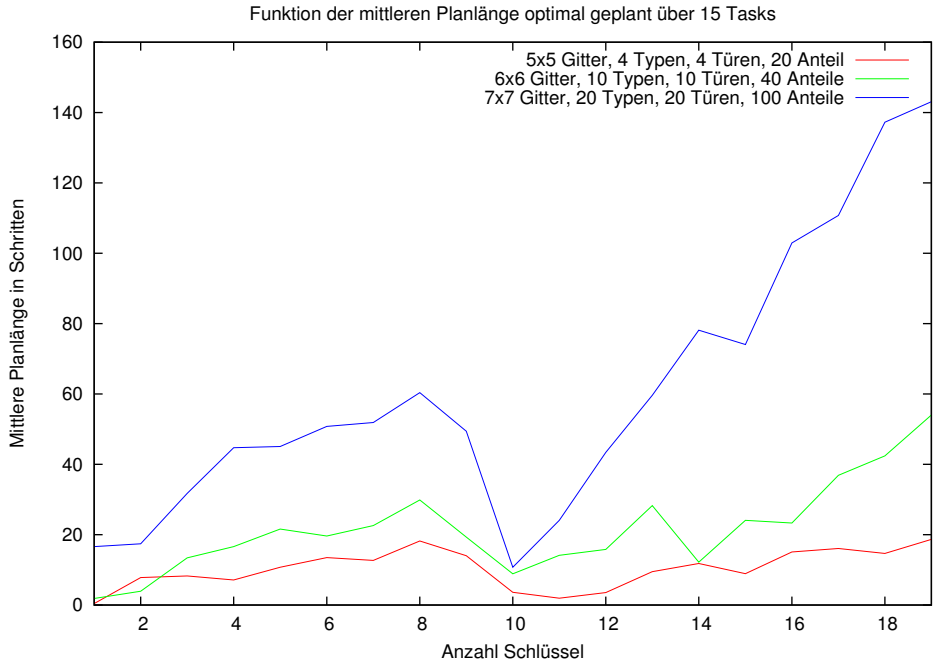

Abb. 4.15.: GRID mittlere Planlänge – Projektion auf die Anzahl der Schlüssel

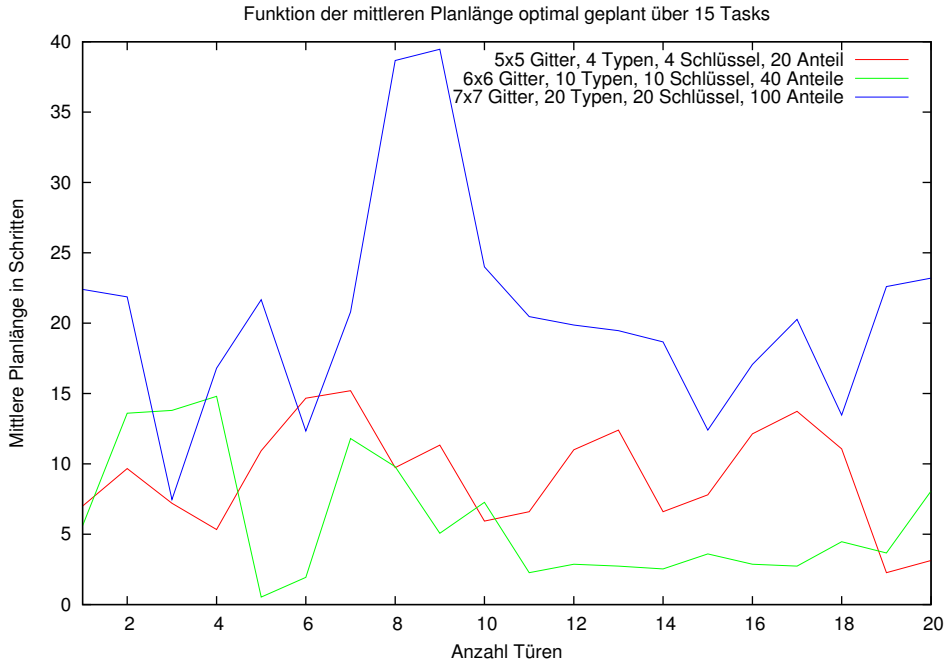

Abb. 4.16.: Grid mittlere Planlänge – Projektion auf die Anzahl der Türen

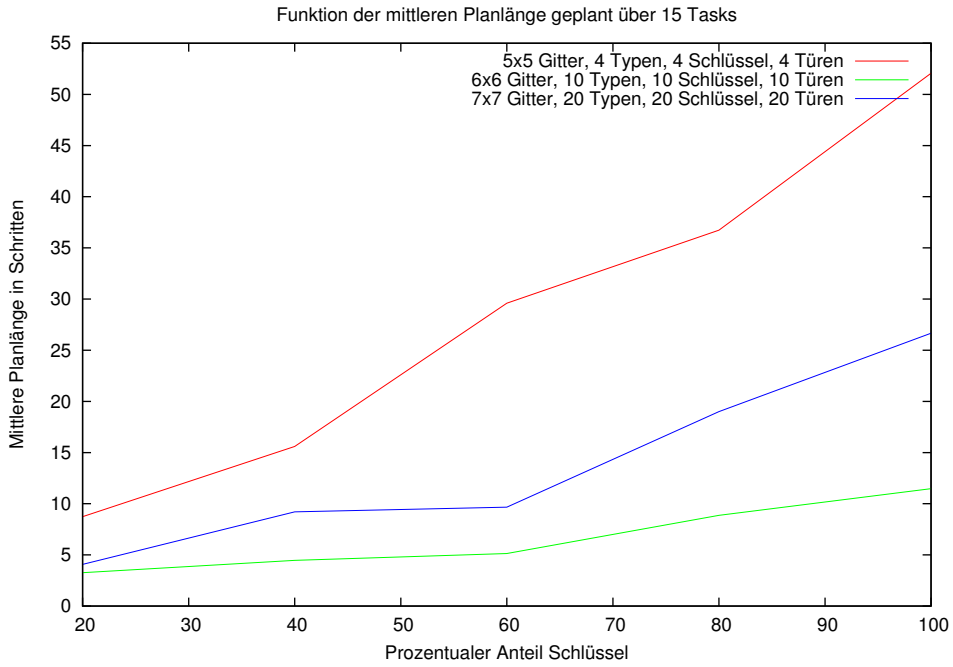

Abb. 4.17.: Grid mittlere Planlänge – Projektion auf die Anzahl der Zielschlüsselanteile

Kapitel 5.

Zusammenfassung

5.1. Ergebnisse

Ziel der vorliegenden Arbeit war die Bestimmung von Phasenübergängen in den Handlungsplanungsbenchmarkdomänen Logistics, Miconic und Grid. Bei der Miconic-Domäne konnten wir beweisen, dass die mittlere optimale Planlänge, die durch die kleinste obere Grenze $4p$ beschränkt ist, für eine wachsende Anzahl von Orten bei fester Paketzahl gegen $4p$ konvergiert. Der kritische Parameter ist bei der Miconic-Domäne die Anzahl der Pakete. Anhand der Domänen Miconic, Logistics und Grid wurde die Berechnung des Phasenübergangs durchgeführt. In der Miconic-Domäne wurde zusätzlich die Korrelation zwischen Phasenübergang und Berechnungszeit untersucht.

Der in dieser Arbeit verwendete Algorithmus zur Identifikation der Phasenübergänge lässt sich leicht an andere Domänen anpassen, solange die Anzahl der Domänenparameter nicht zu groß ist, da die Laufzeiten exponentiell in der Anzahl der Domänenparameter steigen. Um verwertbare Daten zu gewinnen, benötigt man domänenspezifische Planer und Heuristiken.

5.2. Ausblick

5.2.1. Ausgleichsrechnung

Statt nur mit einem geschätzten Ordnungsparameter $\hat{r}$ zu arbeiten, könnte man versuchen, den tatsächlichen Ordnungsparameter r zu bestimmen. Dazu müsste man aus dem Datensatz die Funktion der mittleren optimalen Planlänge $\overline{g}$ bestimmen. Dies könnte man durch Annäherung einer Funktion an die Datenmenge erreichen, welche die Datenmenge am besten approximiert. Wir wissen zum Beispiel, dass die mittlere optimale Planlänge der Miconic-Domäne für eine wachsende Anzahl von Orten gegen die kleinste obere Schranke $4p$ konvergiert. Es handelt sich hierbei somit um ein beschränktes Wachstum. Aus diesem Grund könnte man ausgehend von einer Funktion, die ein beschränktes Wachstum beschreibt, welche unserem Datensatz möglichst nahe kommt, ein Approximationsverfahren anwenden. Für die Projektion des Datensatzes auf die Anzahl der Orte

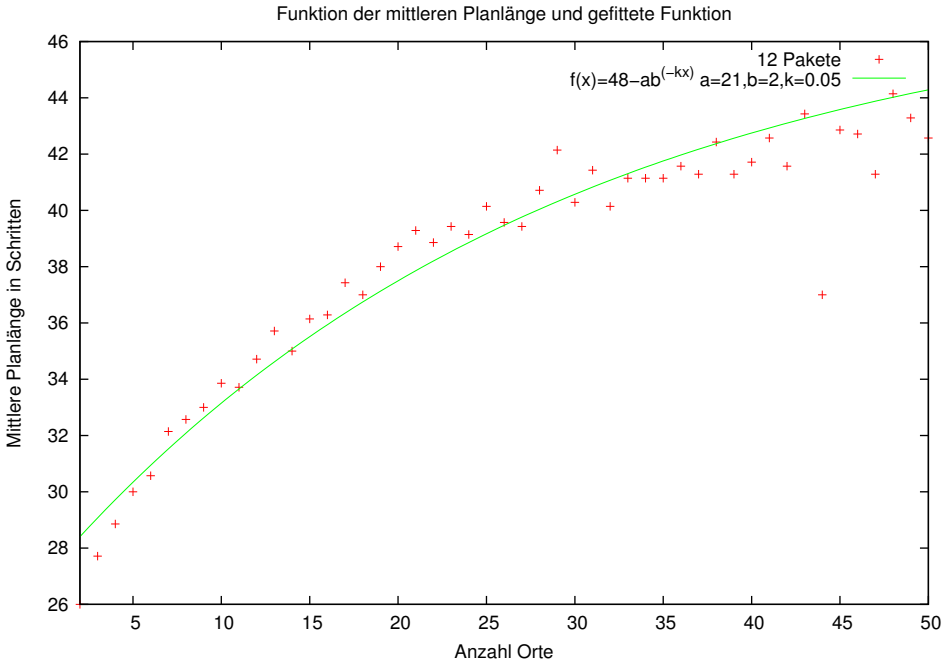

Abb. 5.1.: MICONIC Projektion auf Orte und approximierte Funktion

haben wir das Verfahren von Marquardt-Levenberg [Levenberg, 1944; Marquardt, 1963], wie in *gnuplot*[1] realisiert, angewandt. Abbildung 5.1 zeigt das Ergebnis.

Weil die mittlere optimale Planlänge immer zwischen $2p$ und $4p$ liegt, sind wir bei der MICONIC-Domäne von einer linearen Funktion ausgegangen. Abbildung 5.2 zeigt das Ergebnis.

Sollte es nun gelingen, mit diesen Informationen eine zweidimensionale Funktion an den Datensatz anzunähern, könnte man damit eine geschlossene Formel für $\hat{r}$ bestimmen. Davon ausgehend könnte man versuchen, diese auch analytisch herzuleiten.

5.2.2. Verbesserung des Entscheidungsalgorithmus

Um die untere Grenze $S(t)$ des Intervalls J_{SG} weiter nach rechts zu verschieben, könnte man eine weitere Heuristik direkt in einen domänenspezifischen Planungsalgorithmus, der auf einer A^*-Suche basiert, einbauen. Da die Knoten, die als nächstes expandiert werden, bei der A^*-Suche entsprechend ihrer f-Werte in einer Vorrangwarteschlange gespeichert werden, wird immer der Knoten mit minimalem f-Wert als nächster expandiert. Somit wächst der kleinste f-Wert in der Warteschlange monoton. Bei einer A^*-Suche im Zustandsraum sind die minimalen f-Werte in der Warteschlange untere Schranken für die tatsächliche optimale Planlänge. Wird also der kleinste f-Wert in der

[1] `www.gnuplot.info`

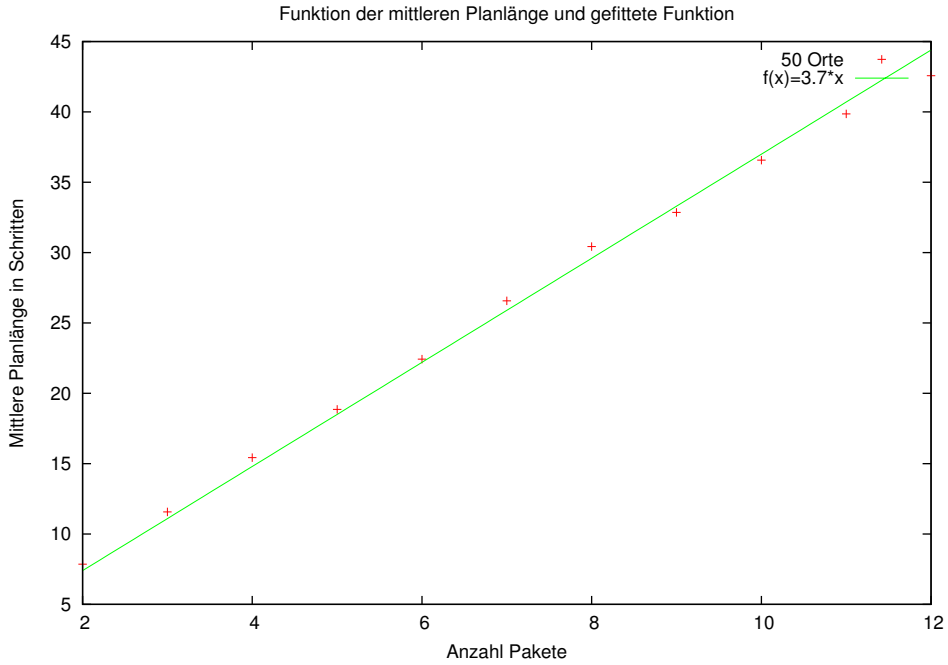

Abb. 5.2.: MICONIC Projektion auf Pakete und approximierte Funktion

Warteschlange größer als die gefragte Planlänge l, ist klar, dass es keinen Plan gibt, der höchstens Länge l hat, und wir können die Suche abbrechen.

5.2.3. Optimierungsprobleme und weitere Planungsdomänen

Neben der Untersuchung von Phasenübergängen bei Entscheidungsproblemen könnte man auch Optimierungsprobleme auf Phasenübergänge untersuchen und die Ergebnisse vergleichen. Außerdem könnte es interessant sein, die Untersuchungen auf weitere Planungsdomänen auszuweiten.

Anhang A.

PDDL-Kodierungen der betrachteten Domänen

A.1. Logistics

```
(define (domain logistics-strips)
  (:requirements :strips)
  (:predicates  (OBJ ?obj)
                (TRUCK ?truck)
                (LOCATION ?loc)
                (AIRPLANE ?airplane)
                (CITY ?city)
                (AIRPORT ?airport)
                (at ?obj ?loc)
                (in ?obj1 ?obj2)
                (in-city ?obj ?city))

(:action LOAD-TRUCK
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
   (at ?truck ?loc) (at ?obj ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?truck)))

(:action LOAD-AIRPLANE
  :parameters
   (?obj
    ?airplane
    ?loc)
```

```
  :precondition
   (and (OBJ ?obj) (AIRPLANE ?airplane) (LOCATION ?loc)
   (at ?obj ?loc) (at ?airplane ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?airplane)))

(:action UNLOAD-TRUCK
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
        (at ?truck ?loc) (in ?obj ?truck))
  :effect
   (and (not (in ?obj ?truck)) (at ?obj ?loc)))

(:action UNLOAD-AIRPLANE
  :parameters
   (?obj
    ?airplane
    ?loc)
  :precondition
   (and (OBJ ?obj) (AIRPLANE ?airplane) (LOCATION ?loc)
        (in ?obj ?airplane) (at ?airplane ?loc))
  :effect
   (and (not (in ?obj ?airplane)) (at ?obj ?loc)))

(:action DRIVE-TRUCK
  :parameters
   (?truck
    ?loc-from
    ?loc-to
    ?city)
  :precondition
   (and (TRUCK ?truck) (LOCATION ?loc-from)
   (LOCATION ?loc-to) (CITY ?city)
   (at ?truck ?loc-from)
   (in-city ?loc-from ?city)
   (in-city ?loc-to ?city))
  :effect
   (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))

(:action FLY-AIRPLANE
```

```
  :parameters
   (?airplane
    ?loc-from
    ?loc-to)
  :precondition
   (and (AIRPLANE ?airplane) (AIRPORT ?loc-from) (AIRPORT ?loc-to)
      (at ?airplane ?loc-from))
  :effect
   (and (not (at ?airplane ?loc-from)) (at ?airplane ?loc-to)))
)
```

A.2. Grid

```
(define (domain grid)
(:requirements :strips)
(:predicates (conn ?x ?y)
             (key-shape ?k ?s)
             (lock-shape ?x ?s)
             (at ?r ?x )
             (at-robot ?x)
             (place ?p)
             (key ?k)
             (shape ?s)
             (locked ?x)
             (holding ?k)
             (open ?x)
             (arm-empty))

(:action unlock
:parameters (?curpos ?lockpos ?key ?shape)
:precondition (and (place ?curpos) (place ?lockpos)
                   (key ?key) (shape ?shape)
                   (conn ?curpos ?lockpos) (key-shape ?key ?shape)
                   (lock-shape ?lockpos ?shape) (at-robot ?curpos)
                   (locked ?lockpos) (holding ?key))
:effect (and  (open ?lockpos) (not (locked ?lockpos))))

(:action move
:parameters (?curpos ?nextpos)
:precondition (and (place ?curpos) (place ?nextpos)
                   (at-robot ?curpos) (conn ?curpos ?nextpos)
                   (open ?nextpos))
```

```
:effect (and (at-robot ?nextpos) (not (at-robot ?curpos))))

(:action pickup
:parameters (?curpos ?key)
:precondition (and (place ?curpos) (key ?key)
                   (at-robot ?curpos) (at ?key ?curpos) (arm-empty))
:effect (and (holding ?key)
             (not (at ?key ?curpos)) (not (arm-empty))))

(:action pickup-and-loose
:parameters (?curpos ?newkey ?oldkey)
:precondition (and (place ?curpos) (key ?newkey) (key ?oldkey)
                   (at-robot ?curpos) (holding ?oldkey)
                   (at ?newkey ?curpos))
:effect (and (holding ?newkey) (at ?oldkey ?curpos)
             (not (holding ?oldkey)) (not (at ?newkey ?curpos))))

(:action putdown
:parameters (?curpos ?key)
:precondition (and (place ?curpos) (key ?key)
                   (at-robot ?curpos) (holding ?key))
:effect (and (arm-empty) (at ?key ?curpos) (not (holding ?key)))))
```

Abbildungsverzeichnis

Literaturverzeichnis

[Blum und Furst 1995] Blum, Avrim L. ; Furst, Merrick L.: Fast Planning Through Planning Graph Analysis. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, 1995, S. 1636–1642

[Bollobás 1985] Bollobás, Béla: *Random Graphs*. Academic Press London, 1985

[Bonet und Geffner 2001] Bonet, Blai ; Geffner, Héctor: Planning as Heuristic Search. In: *Artificial Intelligence* 129 (2001), Nr. 1–2, S. 5–33

[Cheeseman u. a. 1991] Cheeseman, Peter ; Kanefsky, Bob ; Taylor, William M.: Where the Really Hard Problems Are. In: *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, (IJCAI-91), Sidney, Australia*, 1991, S. 331–337

[Forster 1976] Forster, Otto: *Analysis 1. Differential- und Intergralrechnung einer Veränderlichen*. Vieweg Studium, 1976

[Gerevini u. a. 2003] Gerevini, Alfonso ; Saetti, Alessandro ; Serina, Ivan: Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. In: *Journal of Artificial Intelligence Research (JAIR)* Volume 20 (2003), S. 239–290

[Haslum und Geffner 2004] Haslum, Patrik ; Geffner, Héctor: *HSP* Homepage*. http://www.ida.liu.se/∼pahas/hsps/. 2004. – Zuletzt abgerufen 28. Mai 2007

[Helmert 2001] Helmert, Malte: *On the Complexity of Planning in Transportation and Manipulation Domains*, Albert-Ludwigs-Universität Freiburg, Diplomarbeit, 2001

[Hoffmann 2002] Hoffmann, Jörg: Local search topology in planning benchmarks: A theoretical analysis. In: *Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-02)*, 2002, S. 92–100

[Hoffmann 2003] Hoffmann, Jörg: *Lecture Notes in Computer Science*. Bd. 2854/2003: *Utilizing Problem Structure in Planning A Local Search Approach*. Springer-Verlag Berlin / Heidelberg, 2003

[Huberman und Hogg 1987] Huberman, B. A. ; Hogg, T.: Phase Transitions in Artificial Intelligence Systems. In: *Artificial Intelligence* 33 (1987), S. 155–171

[Kautz 2006] Kautz, Henry: Deconstructing Planning as Satisfiability. In: *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, 2006

[Kautz und Selman 1992] Kautz, Henry ; Selman, Bart: Planning as Satisfiability. In: *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, 1992, S. 359–363

[Levenberg 1944] Levenberg, K.: A method for the solution of certain non-linear problems in least squares. In: *Quarterly Journal of Applied Mathematics* 2 (1944), Nr. 2, S. 164–168

[Marquardt 1963] Marquardt, D.: An algorithm for least-squares estimation of non-linear parameters. In: *Journal of the Society for Industrial and Applied Mathematics* 11 (1963), Nr. 2, S. 431–441

[Nau u. a. 2004] Nau, Dana ; Ghallab, Malik ; Traverso, Paolo: *Automated Planning: Theory and Practice*. San Fransisco, CA : Morgan Kaufmann Publishers, 2004

[Papadimitriou 1994] Papadimitriou, Christos H.: *Computational Complexity*. Addison-Wesley, 1994

[Rintanen 2004] Rintanen, Jussi: *Phase transitions in classical planning: an experimental study*. 2004

[Selman u. a. 1996] Selman, Bart ; Mitchell, David ; Levesque, Hector: Generating hard satisfiability problems. In: *Artificial Intelligence 81(1-2)* (1996)