Experimentelle Untersuchung von HLA-Algorithmen

Benjamin Udiljak

15. September 2008

Seminar „Algorithm Engineering“ von Tobias Jacobs (SS 08)

Inhaltsverzeichnis

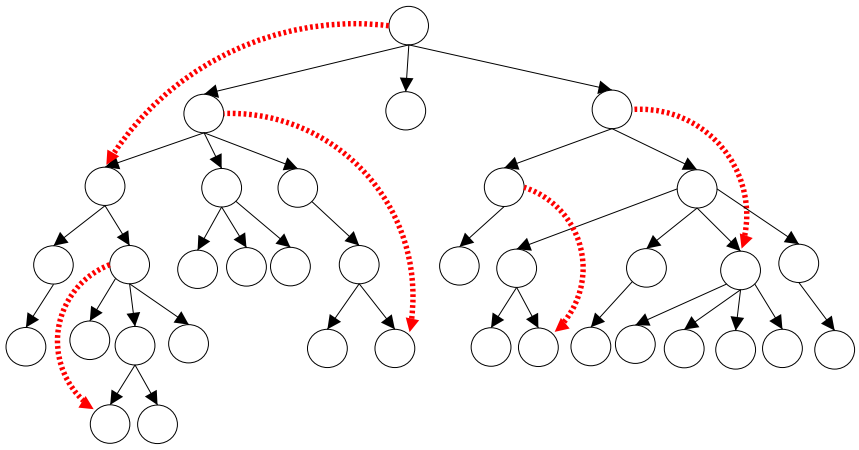
1 Problemstellung

1.1 Überblick

Um die durchschnittliche Pfadlänge einer Website zu minimieren werden sogenannte Hotlinks eingeführt. Ein Hotlink verbindet eine beliebige Seite in der Navigationsstruktur (Hotparent) mit einer Seite (Hotchild). Dabei wird die Website in eine Baumstruktur überführt, die folgende Bedinungen erfüllt:

- Es führt genau ein Pfad (der kürzeste) zu jedem Knoten.
- Die Blätter des Baumes werden als die Seiten mit den gesuchten Informationen angenommen. Daher werden nur diese mit Gewichten versehen, die den Zugriffshäufigkeiten entsprechen.
- Die Gewichte der anderen Knoten ergeben sich aus den Summen der Kinder des jeweiligen Knotens.

Es können dann Hotlinks eingefügt werden, so daß nur ein Hotlink von jedem Knoten ausgeht. Das Hotchild liegt dabei auf dem Pfad, der vom Hotparent ausgeht. Optimalerweise werden Hotlinks so gewählt, daß sie bevorzugt kurze Pfade für Blätter mit hohem Gewicht erzeugen, also wäre in einer Website eine häufig besuchte Seite über wenige Schritte erreichbar.



1.2 Formale Beschreibung

Folgende Größen sind zentral für die formale Beschreibung des Problems.

$dist(u, v)$

Die Distanz $dist(u, v)$ ist der Abstand zwischen 2 Knoten. Sie zählt die Kanten auf dem kürzesten Pfad zwischen 2 Knoten.

$\omega(u)$

Das Gewicht $\omega(u)$ eines Knotens entspricht bei Blättern der Zugriffswahrscheinlichkeit. Bei Navigationsknoten entspricht es der Summe der Gewichte der Kinder.

1.2.1 Ziel eines Hotlink Assignments

Ziel eines Hotlink Assignments A ist es die durchschnittliche Pfadlänge zu minimieren. Man summiert dabei für jedes Blatt l das Produkt aus Distanz $dist^A(r, l)$ (die neue Distanz, die sich durch das verwenden der Hotlinks ergibt) und dem Gewicht des Blattes $\omega(l)$:

$$p(A) = \sum_{l \in L} \omega(l) dist^A(r, l)$$

Daraus berechnet sich der Gain $g(A)$ eines Hotlink Assignments als die eingesparte Pfadlänge. $p(\emptyset)$ und $p(A)$ sind die durchschnittlichen Pfadlängen von ursprünglichem Baum und dem Baum mit Hotlinks:

$$g(A) = p(\emptyset) - p(A)$$

Als Gütemaße für Näherungsalgorithmen definieren wir im folgenden zwei Näherungsgrade:

Näherungsgrad Pfadlänge: $p(A)/p(OPT)$
Näherungsgrad Gain: $g(OPT)/g(A)$

Wobei $p(OPT)$ und $g(OPT)$ der durchschnittlichen Pfadlänge und dem Gain des *optimalen* Hotlink Assignments entsprechen. Es wird sich herausstellen, daß der Näherungsgrad für die Pfadlänge das schlechtere Gütekriterium ist,

da der Näherungsgrad für den Gain die Pfadlänge des Ursprungsbaumes mit in Betracht zieht.

1.2.2 Besondere Eigenschaften

Der Vollstädigkeit halber beschreibe ich hier noch formal einige weitere Eigenschaften der Hotlink Assignments:

1. Wie bereits erwähnt liegt ein Hotchild v immer auf einem Pfad $desc(u)$ der vom Hotparent u ausgeht:

$$v \in desc(u) \text{ für alle } (u, v) \in A$$

2. Da wir vom Greedy User Modell ausgehen, bentzt ein Besucher jeden Hotlink, der ihn näher an sein Ziel bringt. Auch wenn ein Hotlink dabei übersprungen wird, der mehr Schritte einsparen würde. Solche Hotlinks werden daher ausgeschlossen:

$$\text{Wenn } (u, v) \in A, \text{ dann gibt es kein } (u', v') \in A, \text{ so daß}$$
$$u' \in desc(u) \cap anc(v) \text{ und } v' \in desc(v)$$

3. Von dem Knoten geht nur ein Hotlink aus:

$$\text{Für jedes } u \in T \text{ gibt es höchstens ein } (u, v) \in A$$

1.2.3 Schwerste Kinder und „heavy path“

In einigen Algorithmen ist der „heavy path“ von Bedeutung. Er bezeichnet den Pfad der Kanten, die zum jeweils „schwersten Kind“ führen. Das schwerste Kind ist definiert über:
„$\succ$“ definiert eine Ordnung über den Kindern eines Knotens, so daß:

$$\omega(u) > \omega(u') \Rightarrow u \succ u'$$

u ist das schwerste Kind, wenn für alle Paare (u, u') von Geschwisterknoten gilt $u \succ u'$

2 Algorithmen

2.1 GREEDY

Der GREEDY-Algorithmus ist ein naives Verfahren mit erstaunlich guter Leistung.
Das Vorgehen ist denkbar einfach:

- Suche den optimalen Hotlink mit der Wurzel des Baumes als Hotparent
- Wiederhole diesen Schritt rekursiv für alle Teilbäume unter der Wurzel

Der GREEDY-Algorithmus ist eine sogenannte „top-down"-Methode, weil er vollständig über die Wahl eines Hotchilds charaktierisiert ist. Bei allen „top-down"-Algorithmen wird erst ein Hotlink für die Wurzel des Baumes gesucht und dann wird rekursiv die gleiche Methode für alle Teilbäume unter der Wurzel angewendet. Zu beachten ist dabei, daß von den Teilbäumen diejenigen Teilbäume entfernt werden müssen, deren Wurzel das Hotchild des gefundenen Hotlinks sind. Dadurch wird die zweite besondere Eigenschaft von Hotlink Assignments erfüllt. (vgl. Abschnitt 1.2.2)
Es lässt sich zeigen das der Näherungsgrad für den Gain im schlechtesten Fall bei 2 liegt. Das macht diesen Algorithmus trotz seines naiven Ansatzes sehr interessant, da er natürlich auch leicht zu implementieren ist.

2.2 PMIN

Der PMIN-Algorithmus ist ebenfalls eine „top-down"-Methode. Allerdings ist die Wahl des Hotchilds nicht so trivial wie beim GREEDY-Algorithmus. Zunächst definieren wir folgende Formel p_min:

$$p_{min}(T) = \sum_{u \in V \ (L \cup \{r\})} (\omega(u) - \max_{v \in ch(u)} \omega(v)) + \sum_{l \in L} \omega(l)$$

Diese Formel summiert alle Gewichte aller Knoten in T ausgenommen der Wurzel und aller „schwersten Kindern" jedes Knotens (außer der Wurzel). In Abbildung 1 ist die Auswahl der Knoten deren Gewichte summiert werden grafisch dargestellt. In diesem Graphen wird davon ausgegangen, daß der Baum so normiert wurde, daß die schwersten Kinder eines Knoten ganz links aufgezeichnet werden.

Ein Hotchild v von r wird dann so gewählt, daß folgende Formel minimiert wird:

$$p_{min}(T(v)) + \sum_{u \in ch(r)} p_{min}(T(u)(-\{v\}))$$

Wobei $T(u) - (-\{v\})$
Für diesen Algorithmus sind keine „worst-case“-Näherungsgrade bekannt.
Bei den Versuchen zeigt er aber sehr gute Werte für die maximalen Näherungsgrade.
Er ist also extrem stabil, trotz des relativ einfachen Ansatzes.

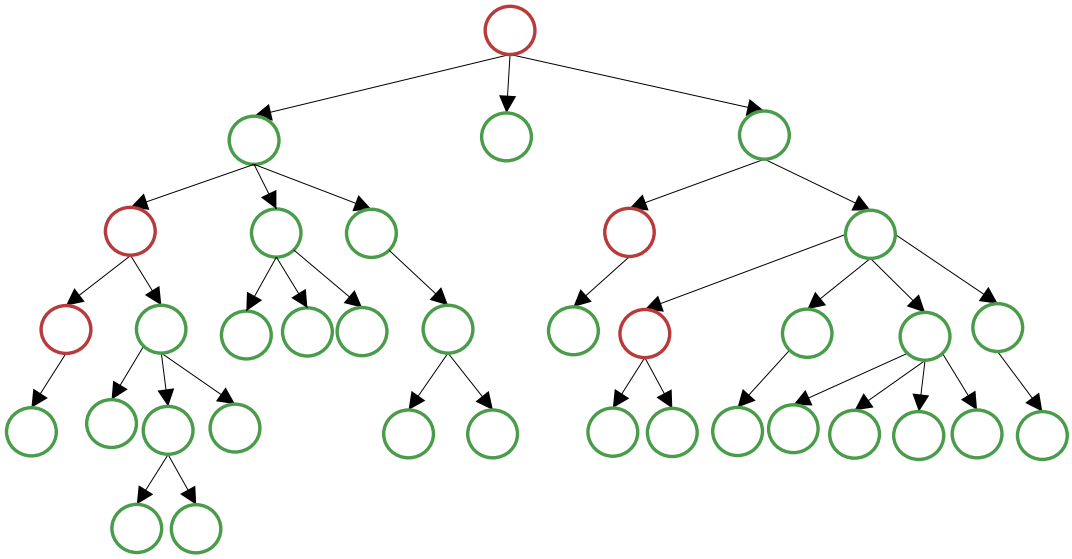


Abbildung 1: PMIN: Gewichte der grünen Knoten werden summiert, rote Kreise sind ausgeschlossen

2.3 H/PH

Der H/PH-Algorithmus ist der letzte „top-down“-Algorithmus, der hier vorgesellt wird. Die Grundidee der letzten Algorithmen ist komplexer und der Baum wird auf verschiedenen Weise vorverarbeitet.

H/PH verfährt wie folgt,um den Hotlink für die Wurzel zu finden:

- Es wird nach dem Knoten h gesucht, dessen Gewicht $\omega(h)$ möglichst nah bei der Hälfte des Gewichtes der Wurzel $\omega(r)/2$ liegt.
- h wird Hotchild der Wurzel r, wenn $\omega(h) > \alpha\omega(r)$. Sonst wird der Vater p_h von h zum Hotchild.

Der Faktor α berechnet sich dabei folgendermaßen:

$$\alpha = (\frac{\alpha}{1-\alpha})^{2(1-\alpha)} \approx 0,2965$$

Der Näherungsgrad für Pfadlänge ist maximal $1,141 \log(\Delta + 1)$. (Δ ist der Ausgangsgrad des Graphen) In der Praxis zeigen sich jedoch relativ schlechte Ergebnisse für die Näherungsgrade.

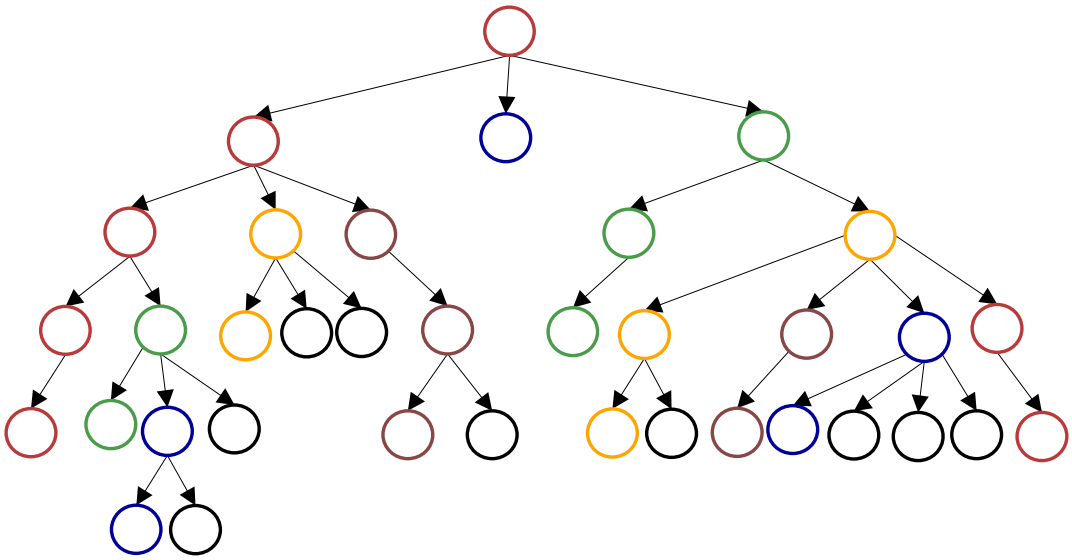
2.4 HEAVY PATH

Bei diesem Algorithmus wird der Baum in eine Menge von „heavy paths“ aufgeteilt. Das sind die Pfade entlang der schwersten Kinder (siehe Abschnitt 1.2.3). In der Abbildung 2 sind die „heavy paths“ gleichfarbige, miteinander verbundene Knoten.
In diesen „heavy paths“ werden den Knoten neue Gewichte $W(u)$ zugewiesen. Dieses Gewicht ergibt sich aus dem alten Gewicht abzüglich des Gewichtes des schwersten Kindes v_f:

$$W(u) = \omega(u) - \omega(v_f)$$

Dadurch entspricht das Gewicht eines Knotens praktisch dem abgeschnittenen Teilbäumen, deren Wurzel nicht das schwerste Kind ist.
Die Pfade werden als Listen notiert: $u_1, ..., u_n$. In jeder dieser Listen weißt man einen Hotlink (u_1, u_i) so zu, daß gilt:
$\sum_{1 \leq j < i} W(u_j) < \frac{1}{2} \sum_{1 \leq j < n} W(u_j)$ und $\sum_{i < j \leq n} W(u_j) < \frac{1}{2} \sum_{1 \leq j < n} W(u_j)$

Abbildung 2: HEAVY PATH: Gleichfarbige, verbundene Knoten sind ein „heavy path“



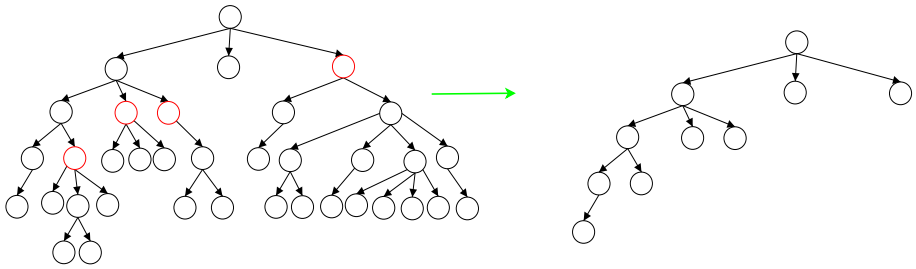
Weitere Hotlinks werden genauso rekursiv für die Unterlisten $u_2, ..., u_{i-1}$ und $u_i, ..., u_n$ erstellt.
Der HEAVY PATH-Algorithmus kann mit linearer Laufzeit implementiert werden und hat daher die beste Worst-Case-Laufzeit. Das macht ihn zum Favoriten bei zeitkritischen Anwendungen. Der Näherungsgrad für die Pfadlänge ist maximal auf $3\log(\Delta + 1)$ beschränkt.

2.5 CENTIPEDE

Beim CENTIPEDE-Algorithmus darf nur das jeweils schwerste Kind von einem Hotlink übersprungen werden. Dazu werden alle Knoten aus dem Baum entfernt, die nicht Kinder von der Wurzel sind und nicht in dem Teilbaum liegen, dessen Wurzel das schwerste Kind ist. (Siehe Abbildung 3)

Ein Hotlink Assignment wird jetzt sowohl für den neuen Baum als auch rekursiv für die Bäume der nicht-schwersten Kinder unabhängig berechnet.

Abbildung 3: CENTIPEDE



Durch diese Vorgehensweise läßt sich der maximale Näherungsgrad für die Pfadlänge auf 2 beschränken.

2.6 LPATH

Beim LPATH-Algorithmus sind die Hotlinks auf die Länge h beschränkt. Je größer man h wählt desto näher kommt das erzeugte Hotlink Assignment an die Optimallösung heran. Wählt man h größer oder gleich der Tiefe des Baumes ist das Ergebnis optimal. Bei den später folgenden Experimenten wurden die Gütemaße der Optimallösung mit diesem Verfahren bestimmt.
Es handelt sich um einen dynamischen Programmieralgorithmus in dem das Problem folgendermaßen aufgeteilt wird:
Teilprobleme sind Teilbäume T_u und ein Pfad mit Knoten, die bereits Hotlinks auf Elemente in T_u haben. Sie sind definiert über das Tripel (q, a, u):
q: Gerichteter Pfad an den T_u angehängt wird.
a: Vektor für Hotlinks aus q
u: Definiert T_u über die Formel:

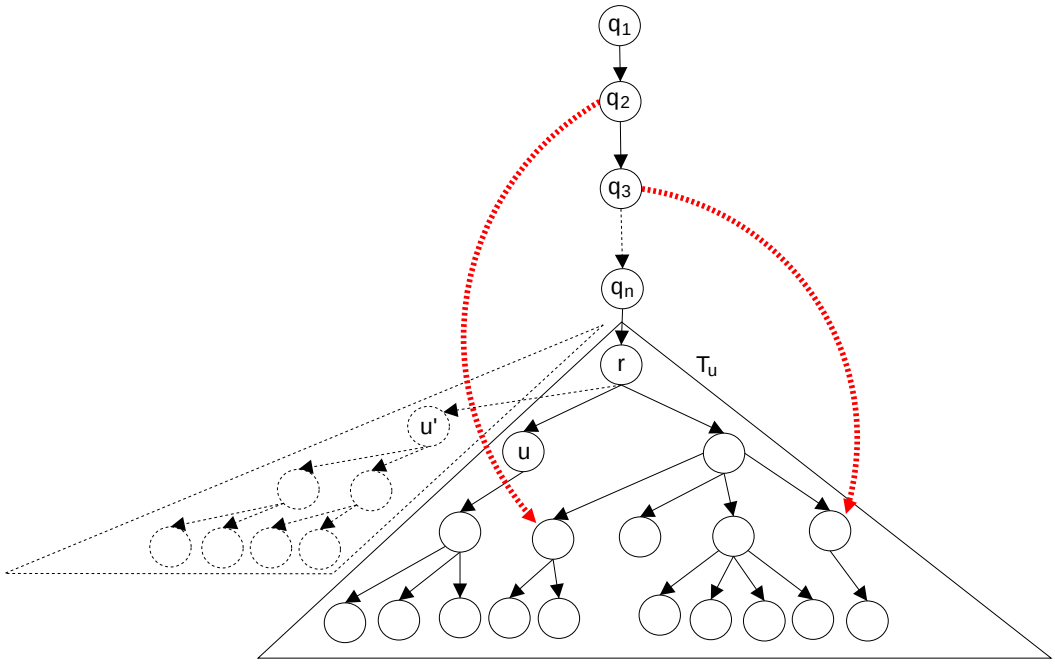
$$T_u = T(par(u))(- \{u' | u' \succ u\})$$

Die Abbildung 4 zeigt ein solches Teilproblem.

Der Algorithmus ist ein sogenannter PTAS (Laufzeit $O(|V|3^h)$), weil er das Ergebnis beliebg nah an die Optimallösung annähern kann.

Der Näherungsgrad für den Gain ist dann maximal $\frac{h}{h-1}$.

Abbildung 4: LPATH Teilproblem



3 Experimente

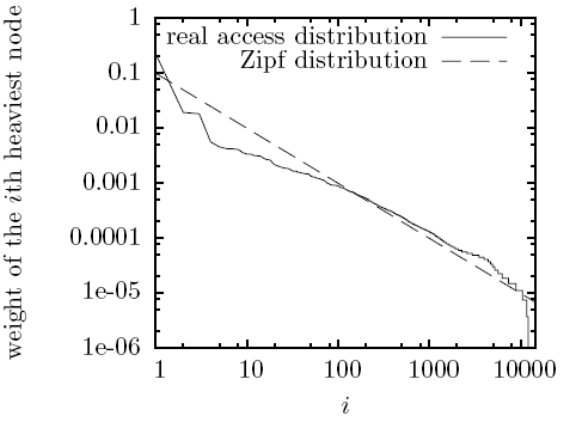
3.1 Versuchsumgebung

3.1.1 Testwebseiten

Es wurden 33 echte Testwebseiten untersucht. Die Seiten waren von brasilianischen und deutschen Universitäten. Sie hatten eine Größe von 48 bis 217.213 Knoten und eine Tiefe von 4 bis 88 Ebenen

Außerdem wurden 1000 künstliche Webseiten erzeugt. Das Verfahren war dabei wie folgt: Jeder neue Knoten wird einem anderen als Kind zugewiesen, wobei die Wahrscheinlichkeit für einen Knoten ausgewählt zu werden abhängig von Anzahl seiner bisheriger Kinder ist.
Die 1000 Seiten setzten sich aus jeweils 10 Instanzen der Größe 1000, 2000, ... , 100000 zusammen. Die Gewichte der Blätter wurden nach der Zipf-Verteilung zugeordnet. Das ist eine „typische Zugriffsverteilung“ aus der Literatur. Abbildung 5 zeigt diese Verteilung.

Abbildung 5: i-schwerstem Blatt wird das Gewicht $\frac{1}{iH_m}$ zugeordnet(H_m ist die m-te harmonische Zahl; m ist die Anzahl der Blätter des Baumes)



3.2 Messungen

Jeder Algorithmus wurde auf alle Testinstanzen angewendet. (LPATH mit den Parametern $h = 2, 3, ..., 15$) Gemessen wurden jeweils Laufzeit und Pfadlänge $p(A) = \sum_{l \in L} \omega(l) dist^A(r, l)$ des Hotlink Assignments. Aus der Pfadlänge wurden folgende weitere Werte berechnet:

- Gain: $g(A) = p(\emptyset) - p(A)$
- Relativer Gain: $g(A)/p(\emptyset)$
- Näherungsgrad Pfadlänge: $p(A)/p(OPT)$
- Näherungsgrad Gain: $g(OPT)/g(A)$

Gain

Tabelle 1 zeigt die prozentualen Ergebnisse für den Gain. Die ersten beiden Spalten sind Größe und Tiefe einer exemplarischen Testwebseite. Danach folgt der Gain des optimalen Ergebnisses (berechnet vom LPATH-Algorithmus).
Es zeigen sich sehr gute Werte für GREEDY und PMIN.

Näherungsgrad

Die Tabelle 2 zeigt die Näherungsgrade für Gain bzw. Pfadlänge verschiedener Algorithmen. Besonders auffallend sind hier die großen Werte für H/PH und HEAVY PATH beim Näherungsgrad des Gain. Obwohl die angegebenen obere Schranke für den Näherungsgrad der Pfadlänge eigentlich gute Ergebnisse erwarten lässt. GREEDY, PMIN und auch CENTIPEDE schneiden wesentlich besser ab. LPATH ist allerdings schon ab $h = 4$ absolut unschlagbar. Das gilt übrigens auch für die exemplarischen Gain-Werte der Tabelle 1 bei denen LPATH „außer Konkurenz“ in einer eigenen Tabelle aufgeführt wurde.

Laufzeit

Abbildungen 6 und 7 zeigen die Laufzeiten der Algorithmen. Besonders bemerkenswert ist die steile Kurve von CENTIPEDE. Man beachte auch die logarithmische Skalierung in der Grafik des LPATH-Algorithmus. Bei einem $h > 8$ ist die Laufzeit bereits länger als bei den anderen Algorithmen. Das andere Extrem ist der HEAVY PATH Algorithmus mit seiner linearen Laufzeit und entsprechend hohen Geschwindigkeiten.

Tabelle 1: Gain

$\|V\|$	d	OPT	GREEDY	PMIN	H/PH	HEAVY PATH	CENTI-PEDE
48	6	53.41	53.41	53.41	50.58	49.10	51.14
117	4	21.77	21.77	21.77	14.12	0.00	15.38
320	7	42.73	40.88	42.68	29.68	31.72	41.66
369	5	30.39	30.39	30.25	12.15	12.74	24.13
443	10	49.72	49.54	44.81	37.37	30.97	40.24
542	6	26.53	26.03	24.86	7.15	6.44	22.35
556	4	12.70	10.98	12.70	0.32	0.00	9.69
646	5	28.12	27.26	28.12	14.91	11.99	26.35
746	9	44.74	44.56	44.52	31.24	29.70	41.16
842	10	51.67	50.25	50.70	38.02	33.41	41.58
1016	4	18.35	18.35	18.35	5.76	5.12	17.93
1100	8	46.07	45.96	45.21	27.36	34.14	43.81
1151	7	35.51	35.03	33.63	19.91	19.59	33.35
1158	7	26.53	25.93	25.88	16.81	6.07	22.30
1743	3	13.15	13.15	13.15	6.15	6.12	8.63
1892	9	36.99	35.98	36.28	19.14	22.40	32.98
2275	6	30.37	30.29	30.29	22.18	21.32	27.99
2317	6	21.38	20.75	21.36	8.43	5.57	18.65
10484	16	55.21	54.22	54.41	43.29	41.23	50.24
24986	16	48.22	47.07	47.85	33.27	31.56	45.72
26194	5	23.43	22.77	22.92	11.90	11.64	20.59
30890	56	38.81	37.98	38.14	27.83	25.50	34.54
45439	23	35.89	35.33	35.32	20.95	19.48	32.24
57877	10	39.99	37.20	39.71	25.21	24.74	36.39
78472	45	?	54.80	56.92	47.01	45.95	54.15
89894	41	?	45.86	45.99	32.25	31.02	43.66
96288	26	?	42.61	43.04	29.02	30.47	39.52
107591	37	?	49.12	49.09	39.70	34.29	49.21
110892	58	?	38.87	39.43	26.40	27.04	36.61
115044	42	?	45.27	44.14	28.54	31.23	40.68
120330	88	?	59.23	58.40	45.97	48.45	56.95
163237	40	?	48.28	48.39	34.73	35.39	?
217213	15	43.76	42.32	43.36	31.91	32.03	41.02

Table 1: Relative gain (%) concerning the real instances.

	approximation ratio (gain)		
algorithm	worst case	min	max
GREEDY	2	1.000	1.156
PMIN	?	1.000	1.109
H/PH	∞	1.056	39.967
HEAVY PATH	∞	1.088	4.372
CENTIPEDE	∞	1.024	1.524
LPATH, $h = 2$	2	1.000	1.403
LPATH, $h = 3$	1.5	1.000	1.103
LPATH, $h = 4$	1.333	1.000	1.010
	approximation ratio (path length)		
algorithm	worst case	min	max
GREEDY	∞	1.000	1.146
PMIN	?	1.000	1.098
H/PH	$1.141 \log(\Delta - 1)$	1.061	1.347
HEAVY PATH	$3 \log(\Delta + 1)$	1.081	1.378
CENTIPEDE	2	1.005	1.209
LPATH, $h = 2$	∞	1.000	1.351
LPATH, $h = 3$	∞	1.000	1.092
LPATH, $h = 4$	∞	1.000	1.009

Tabelle 2: Näherungsgrad

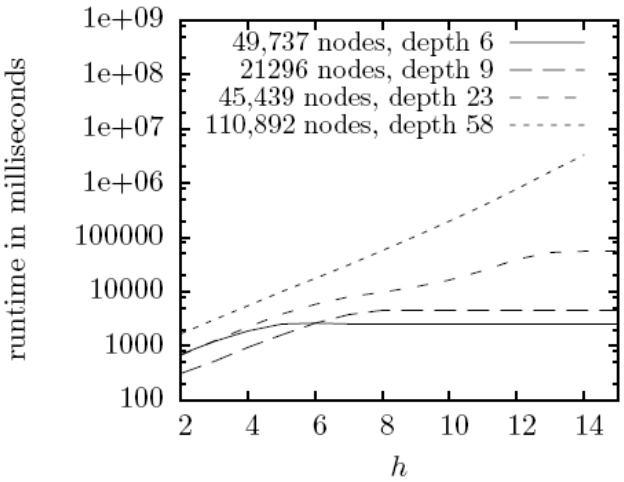


Figure 9: Runtime resulted from running LPATH on tree instances for different values of h .

Abbildung 6: Laufzeit LPATH

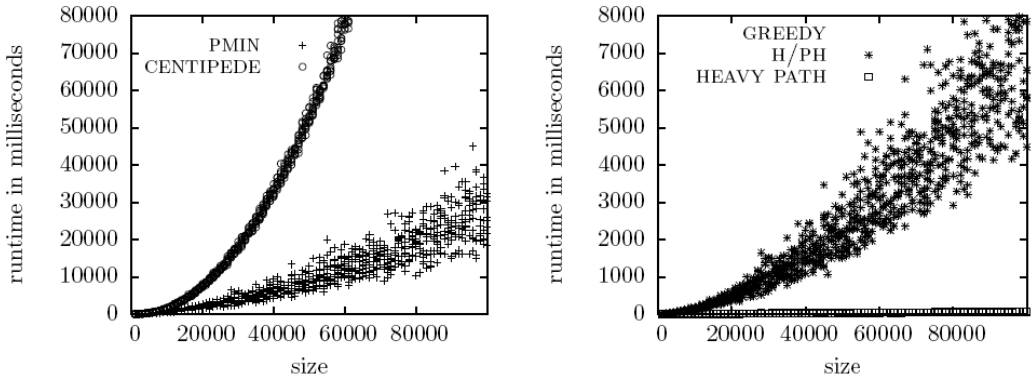


Abbildung 7: Andere Laufzeiten

4 Fazit

- GREEDY garantiert einen Näherungsgrad von 2 für den Gain und schneidet auch in der Praxis gut ab.
- PMIN schneidet sehr gut ab und ist ebenfalls leicht zu implementieren
- Auf Näherungsgrad der Pfadlänge zugeschnittene Algorithmen schneiden algemein schlecht ab.
  - HEAVY PATH glänzt allerdings durch seine schnelle Laufzeit
  - CENTIPEDE könnte interessant sein wenn ein Näherungsgrad von 2 für Pfadlänge garantiert sein soll
- LPATH räumt ab: Er liefert immer die beste Annäherung und auch noch gute Laufzeiten, wenn $h > 4$

5 Literatur

- T. Jacobs. An Experimental Study of Recent Hotlink Assignment Algorithms. In *Proceedings of the 9th Workshop on Algorithm Engineering & Experiments (ALENEX'08)*, 2008.

- T. Jacobs. Constant factor approximations for the hotlink assignment problem. In *Proceedings of the 10th Workshop on Algorithms and Data Structures (WADS 2007)*, Halifax, Canada, 2007.