ORCAN - Implementation von Methoden zur Compilierung von Anytime-Algorithmen

Fortgeschrittenenpraktikum Sommersemester 1997

Andreas Baus
anba@cs.uni-sb.de

Axel Beckert
abe@cs.uni-sb.de

27. Februar 1999

Inhaltsverzeichnis

1 Einführung

Die vorliegende Arbeit, das System ORCAN[1], entstand im Rahmen eines Fortgeschrittenenpraktikums am Lehrstuhl von Prof. Wahlster an der Universität des Saarlandes und ist Teil des Projektes REAL[2] (W. Wahlster und A. Blocher und J. Baus und E. Stopp und H. Speiser, 1998) des Sonderforschungsbereiches 378.

In REAL wird ein System zur Generierung von sprachlichen Raumbeschreibungen bzw. Beantwortung von Wo-Fragen mit beschränkten Ressourcen entwickelt und implementiert. Es basiert auf einer ressourcenadaptierenden Kombination von Anytime-Algorithmen (Siehe Abschnitt 2)

Das System ORCAN ist eine Implementation von Methoden zur Verteilung von beschränkten Ressourcen auf mehrere Anytime-Module, so daß die Qualität des Endresultats, das aus der Kombination der Einzelergebnisse der Module hervorgeht, möglichst groß wird. Bei der zu verteilenden Ressource kann es sich z. B. um Rechenzeit, Speicherplatz oder Prozessoren (bei Parallelrechnern) handeln; in dieser Arbeit wird exemplarisch Rechenzeit dafür angenommen, da in den meisten Fällen Zeit die beschränkbare Ressource der in REAL verwendeten Anytime-Algorithmen ist.

---

[1] ORCAN steht für „Operational Rationality through Compilation of Anytime Algorithms“
[2] REAL steht für „Ressourcenadaptierende Objektlokalisation“

2 Anytime-Algorithmen

Anytime-Algorithmen sind Algorithmen, die man zu jeder Zeit unterbrechen kann und die auf jeden Fall ein Ergebnis liefern. Je länger man einen solchen Algorithmus rechnen läßt, desto präziser wird sein Ergebnis. Die Genauigkeit der Näherung hängt von der zur Verfügung gestellten Zeit ab, und wird auch als *Qualität* des Ergebnisses bezeichnet. In (Dean, Thomas und Boddy, Mark, 1988) und (S. Zilberstein, 1993) werden u.a. zwei Arten von Anytime-Algorithmen vorgestellt:

- Unterbrechbare Algorithmen, die zu jedem Zeitpunkt ihrer Ausführung angehalten werden können, und dann das bisher berechnete Ergebnis zur Verfügung stellen.
- Contract-Algorithmen, denen die zur Verfügung stehende Laufzeit zu Beginn der Berechnung mitgeteilt werden muß und die innerhalb dieser Zeit das bestmögliche Ergebnis errechnen.

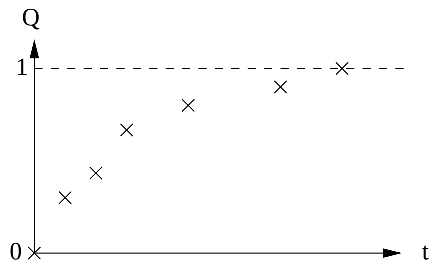

Abbildung 1: Ein Performanzprofil als Menge von Stützpunkten

Die Relation zwischen verfügbaren Ressourcen und der Qualität des Ergebnisses wird beschrieben durch ein sogenanntes *Performanzprofil*. Genauer stellt ein Performanzprofil eine Abbildung der Form

$$PP : \mathbb{R}^{+} \longrightarrow [0,1];\ t \longmapsto PP(t) \tag{1}$$

dar, die einer Zeit T eine Qualität $Q = PP(T) \in [0,1]$ zuordnet, die das Ergebnis, das der zugehörige Algorithmus nach einer Laufzeit von T liefert, erwartungsgemäß hat. (Russell, Stuart J. und Wefald, Eric, 1991)

Bei unserer Arbeit gehen wir davon aus, daß Performanzprofile als Menge von Stützpunkten der Form

$$PP = \{(t_i, q_i) \mid t_i \in \mathbb{R}_0^{+}; q_i \in [0,1]; i = 1, \ldots, n\} \tag{2}$$

vorliegen, die als empirische Informationen aus einem oder mehreren früheren Läufen des zugehörigen Algorithmus gewonnen wurde. (Siehe Abb. 1)

3 Kombination von zwei Performanzprofilen

Zilberstein (S. Zilberstein, 1993) beschreibt Methoden zur Kombination von jeweils zwei Performanzprofilen, die entweder als lineare oder exponentielle Funktionen gegeben sind. Wie diese Methoden auf Profile, die in Form von Stützpunktmengen vorliegen, angewendet werden können, wird im folgenden beschrieben:

3.1 Annäherung durch lineare Funktionen

Die erste Methode, die Zilberstein beschreibt, kombiniert Performanzprofile, die als Geradenfunktionen in der Form $PP(t) = q_0 + at$ vorliegen. Dabei ist a die Steigung und q_0 der q-Achsenabschnitt der Gerade. Um aus den gegebenen Profilen solche Geradenfunktionen zu gewinnen, bietet sich die Methode der linearen Regression an (siehe Abb. 2):

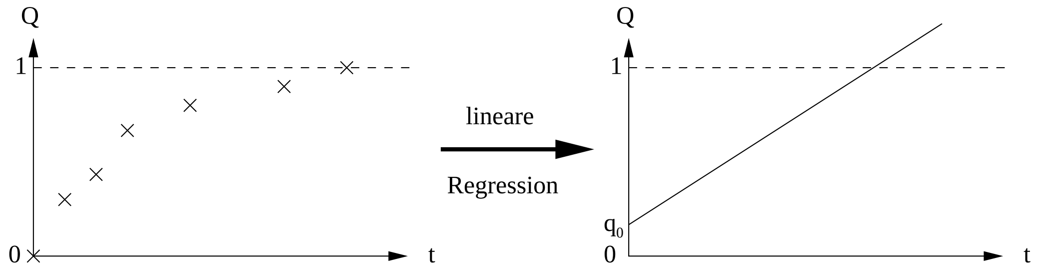

Abbildung 2: Annäherung durch lineare Regression

Sei PP ein Performanzprofil wie in Abschnitt 2. Um eine Geradenfunktion $PP(x) = q_0 + ax$, deren durchschnittlicher Abstand zu allen Stützpunkten minimal ist, zu erhalten, wählt man, wie man aus einer geeigneten mathematischen Formelsammlung (I. N. Bronstein und K. A. Semendjajew, 1967) ablesen kann, die Parameter

$$q_0 = \frac{\sum_i t_i^2 \sum_i q_i - \sum_i t_i q_i \sum_i t_i}{n \sum_i t_i^2 - \left(\sum_i t_i\right)^2} \tag{3}$$

$$a = \frac{n \sum_i t_i q_i - \sum_i t_i \sum_i q_i}{n \sum_i t_i^2 - \left(\sum_i t_i\right)^2} \tag{4}$$

Zur Kombination von zwei Performanzprofilen werden nun zunächst mit dieser Methode die zugehörigen Näherungsfunktionen

$$PP_1(x) = q_1 + a_1 x \quad \text{und} \quad PP_2(x) = q_2 + a_2 x \tag{5}$$

bestimmt. Wird nun eine Gesamtzeit T auf diese beiden Profile verteilt, so erhält das Profil PP_1 die Zeit t_1 und PP_2 die Restzeit $t_2 = T - t_1$. Die Qualität des Endergebnisses ist somit abhängig von T und der Verteilung (die bereits durch t_1 bestimmt ist, da sich t_2 unmittelbar als Differenz von t_1 und T ergibt):

$$\begin{aligned} Q_{ges}(T, t_1) &= PP_1(t_1) PP_2(T - t_1) \\ &= (q_1 + a_1 t_1)\,(q_2 + a_2(T - t_1)) \\ &= -a_1 a_2 t_1^2 + (a_1 a_2 T + a_1 q_2 - a_2 q_1) t_1 + q_1 a_2 T + q_1 q_2 \end{aligned} \tag{6}$$

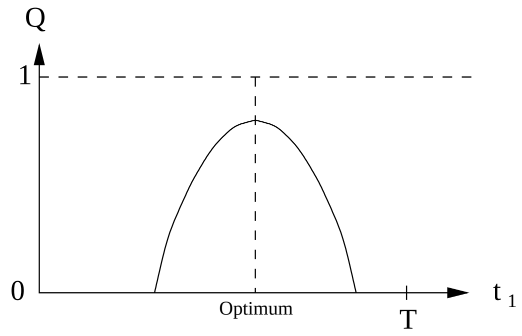

Abbildung 3: Gesamtqualität

Abbildung 3 zeigt die Funktion der Gesamtqualität $Q_{ges}(T, t_1)$ in Abhängigkeit von t_1 und für konstantes T. Gesucht ist die Verteilung, für die die Gesamtqualität maximal wird. Um nun ein Maximum zu finden, sucht man nach einem t_1, für das die Steigung 0, d. h. die Bedingung $\frac{\delta Q_{ges}}{\delta t_1} = 0$ erfüllt ist. Durch Lösen nach t_1 ergibt sich:

$$\frac{\delta Q_{ges}}{\delta t_1} = -2a_1a_2t_1 + a_1a_2T + a_1q_2 - a_2q_1 \tag{7}$$

$$\begin{aligned} -2a_1a_2x_1 + a_1a_2T + a_1q_2 - a_2q_1 &= 0 \\ \Leftrightarrow t_1 &= \frac{a_1q_2 - a_2q_1 + a_1a_2T}{2a_1a_2} \\ \Leftrightarrow t_1 &= \frac{T}{2} + \frac{a_1q_2 - a_2q_1}{2a_1a_2} \end{aligned} \tag{8}$$

Das bedeutet, daß das Gesamtresultat optimal wird, wenn

$$t_1 = \frac{T}{2} + \frac{a_1q_2 - a_2q_1}{2a_1a_2} \quad \text{und} \quad t_2 = T - t_1 = \frac{T}{2} + \frac{a_2q_1 - a_1q_2}{2a_1a_2} \tag{9}$$

gewählt wird.

3.2 Annäherung durch Exponentialfunktionen

Im allgemeinen lassen sich Performanzprofile mit Hilfe linearer Funktionen nicht sehr genau annähern, da sie selten über einen nahezu linearen Verlauf verfügen. Daher ist es interessant, auch andere Klassen von Funktionen zu betrachten, die den Verlauf von zumindest einigen Klassen von Profilen besser annähern können.

Eine solche Klasse ist die exponentieller Funktionen der Form $PP(t) = 1 - \eta e^{-\lambda t}$, wie sie von Zilberstein in der zweiten von ihm angeführten Methode verwendet werden.

Aus den als Stützpunktmengen vorliegenden Performanzprofilen lassen sich wieder durch die Methode der Regression Funktionen in dieser Form gewinnen: Wenn PP wieder eine Menge von Stützpunkten ist, dann wird eine Funktion

$$PP : \mathbb{R}_0^+ \longrightarrow [0, 1];\ t \longmapsto 1 - \eta e^{-\lambda t} \tag{10}$$

gesucht, deren durchschnittlicher Abstand zu allen Stützpunkten minimal ist (siehe Abb. 4). Dieses Problem läßt sich auf die lineare Regression zurückführen, wenn man folgende Äquivalenzumformung betrachtet:

$$
\begin{aligned}
& & q &= 1-\eta e^{-\lambda t} \\
&\Leftrightarrow & 1-q &= \eta e^{-\lambda t} \\
&\Leftrightarrow & \ln\left(\frac{1-q}{\eta}\right) &= -\lambda t \\
&\Leftrightarrow & ln(1-q) &= ln(\eta)-\lambda t.
\end{aligned} \tag{11}
$$

Durch Setzen von $\bar{q} := ln(1-q)$, $\bar{\eta} := ln(\eta)$ und $\bar{\lambda} := -\lambda$ ergibt sich eine lineare Funktion:

$$\bar{q} = \bar{\eta} + \bar{\lambda}t \tag{12}$$

Durch lineare Regression (wie in Abschnitt 3.1) auf $\overline{PP} = (t_i, \bar{q_i})$ mit $\bar{q_i} = ln(1-q_i)$, erhält man nun die Parameter $\bar{\eta}$ und $\bar{\lambda}$ für eine Geradenfunktion, die $\overline{PP}$ annähert und daraus die entsprechenden Parameter $\eta = e^{\bar{\eta}}$ und $\lambda = -\bar{\lambda}$ für die gesuchte exponentielle Funktion.

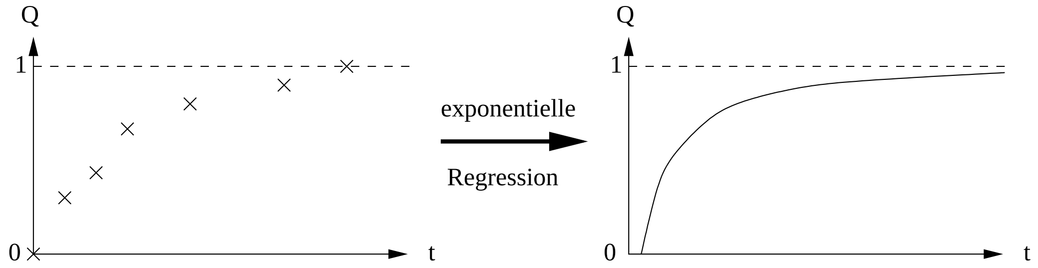

Abbildung 4: Annäherung durch exponentielle Regression

Dabei ist zu beachten, daß $\bar{q_i} = ln(1-q_i)$ undefiniert ist für $q_i = 1$, d.h. wenn im Profil ein Stützpunkt auftritt, dessen $q_i = 1$ ist. Exponentielle Funktionen von obiger Form erreichen nie die Gerade $q = 1$, sie nähern sich ihr nur asymptotisch für $t \to \infty$. Daher ist es nicht möglich, Profile, die diese Gerade erreichen, exakt nachzubilden. Um wenigstens das Problem der Undefiniertheit von $\bar{q_i}$ für $q_i = 1$ zu umgehen, kann man das Ausgangsprofil um einen Faktor $1-\epsilon$, für ein geeignetes, sehr kleines ϵ entlang der q-Achse stauchen, so daß im Profil keine Stützstelle mit $q_i = 1$ mehr vorkommt.

Mit den so erhaltenen Funktionen läßt sich dann eine Verteilung ähnlich wie im linearen Fall errechnen: Sei T die gesamte zu verteilende Zeit, $PP_1(t) = 1-\eta_1 e^{-\lambda_1 t}$ das Profil für den ersten Algorithmus und $PP_2(t) = 1-\eta_2 e^{-\lambda_2 t}$ das für den zweiten. Bei einer Zuteilung von t_1 auf PP_1 und $t_2 = T - x_1$ auf PP_2 ergibt sich eine zu erwartende Gesamtqualität von:

$$
\begin{aligned}
Q_{ges}(T, t_1) &= PP_1(t_1) + PP_2(T-t_1) \\
&= 1-\eta_1 e^{-\lambda_1 t_1} + 1 - \eta_2 e^{-\lambda_2 (T-t_1)} \\
&= 2-\eta_1 e^{-\lambda_1 t_1} - \eta_2 e^{-\lambda_2 T} e^{\lambda_2 t_1}
\end{aligned} \tag{13}
$$

Abbildung 5 zeigt wieder die Gesamtqualität für festes T in Abhängigkeit von t_1. Wiederum sucht man das Maximum, indem man nach Stellen sucht, an denen die Ableitung 0 ist, man löst also die Gleichung $\frac{\delta Q_{ges}}{\delta t_1}$ und erhält:

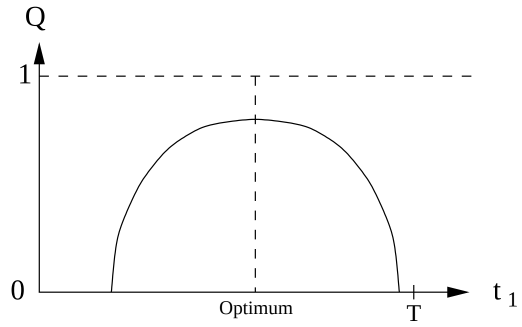

Abbildung 5: Gesamtqualität

$$
\begin{array}{rrcl}
 & \frac{\delta Q_{ges}}{\delta t_1} & = & 0 \\
\Leftrightarrow & \eta_1\lambda_1 e^{-\lambda_1 t_1} - \eta_2\lambda_2 e^{-\lambda_2 T} e^{\lambda_2 t_1} & = & 0 \\
\Leftrightarrow & \eta_1\lambda_1 e^{-\lambda_1 t_1} & = & \eta_2\lambda_2 e^{-\lambda_2 T} e^{\lambda_2 t_1} \\
\Leftrightarrow & -\lambda_1 t_1 + \ln(\eta_1\lambda 1) & = & \lambda_2 t_1 - \lambda_2 T + \ln(\eta_2\lambda_2) \\
\Leftrightarrow & t_1 & = & \frac{\ln(\eta_1\lambda_1) - \ln(\eta_2\lambda_2) + \lambda_2 T}{\lambda_1 + \lambda_2}
\end{array}
\tag{14}
$$

Die Verteilung wird also optimal, wenn man dem ersten Algorithmus die Zeit

$$t_1 = \frac{\ln(\eta_1\lambda_1) - \ln(\eta_2\lambda_2) + \lambda_2 T}{\lambda_1 + \lambda_2} \tag{15}$$

und dem zweiten die Zeit

$$t2 = T - t_1 = \frac{\ln(\eta_2\lambda_2) - \ln(\eta_1\lambda_1) + \lambda_1 T}{\lambda_1 + \lambda_2} \tag{16}$$

zuteilt.

3.3 Annäherung durch abschnittsweise lineare Funktionen

Eine weitere Methode zur Gewinnung von Funktionen aus Mengen von Stützpunkten, die sich anbietet, ist die lineare Interpolation zwischen den Punkten (siehe Abb. 6). Sei z. B. ein Performanzprofil gegeben als Stützpunktmenge (wie in Abschnitt 2) und es gelte $t_i < t_{i+1} \forall i = 1, \ldots, n$, dann ist die linear interpolierte Funktion

$$PP(t) := q_i + (t - t_i)\frac{q_{i+1} - q_i}{t_{i+1} - t_i} \quad \text{für} \quad i \quad \text{mit} \quad t_i \leq t < t_{i+1} \tag{17}$$

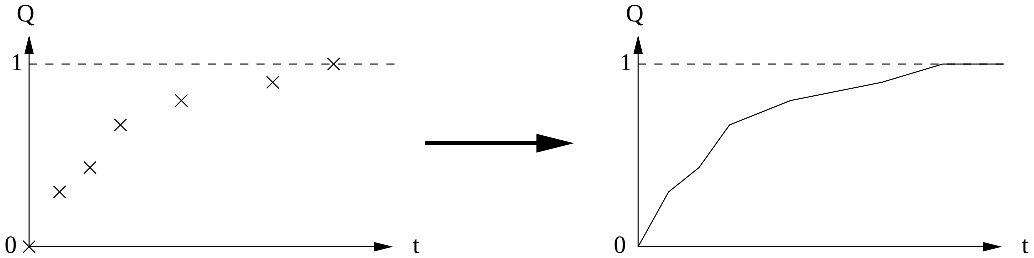

Abbildung 6: Annäherung durch lineare Interpolation

Für diese Klasse von Funktionen läßt sich keine geschlossene Lösung für die optimale Kombination angeben, sie muß also jedesmal explizit berechnet werden:

Bei der Kombination wird wieder eine Gesamtzeit T so verteilt, daß das Profil PP_1 die Zeit t und PP_2 die Zeit $T-t$ erhält. Dabei ist

$$
\begin{aligned}
PP_2(T-t) &= q_i + (T-t-t_i)\frac{q_{i+1}-q_i}{t_{i+1}-t_i} && \text{für } i \text{ mit } && t_i \leq T-t < t_{i+1} \\
&= q_i + (T-t_i-t)\frac{q_{i+1}-q_i}{T-t_{i+1}-T-t_i} && \text{für } i \text{ mit } && T-t_{i+1} < t \leq T-t_i \\
&= q_i + (t-(T-t_i))\frac{q_{i+1}-q_i}{(T-t_i)-(T-t_{i+1})} && \text{für } i \text{ mit } && T-t_{i+1} < t \leq T-t_i
\end{aligned}
\tag{18}
$$

Wenn also PP_2 gegeben ist durch die Punktmenge $\{(t'_i, q'_i)\}$, so ist $PP_2(T-t)$ darstellbar als linear interpoliertes Profil $\overline{PP}_2(t)$ über der Punktmenge $\{(T-t'_i, q'_i)\}$. Dies entspricht einer Spiegelung des Profils an der q-Achse bei gleichzeitiger Verschiebung um T in Richtung der positiven t-Achse (siehe Abb. 8 und 7).

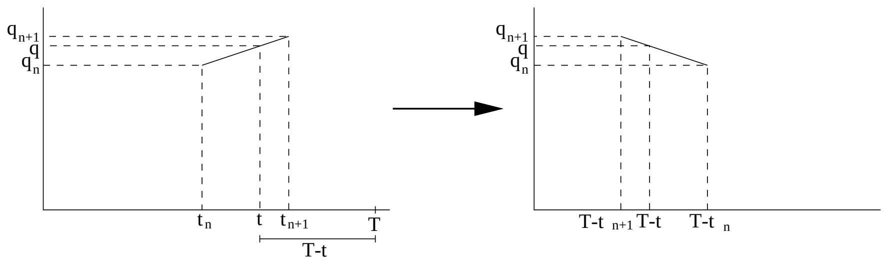

Abbildung 7: Spiegelung und Verschiebung

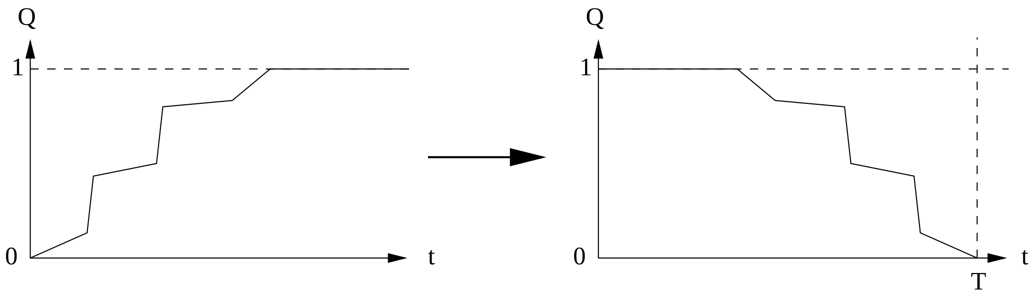

Abbildung 8: Das Ausgangsprofil und das gespiegelte Profil

Die Kombination der beiden Performanzprofile ist dann

$$
\begin{aligned}
Q_{ges}(T,t) &= PP_1(t) \circ PP_2(T-t) \\
&= PP_1(t) \circ \overline{PP}_2(t).
\end{aligned}
\tag{19}
$$

Dabei ist $\circ$ eine geeignete binäre Verknüpfung (z. B. Multiplikation oder das arithmetische Mittel zweier Zahlen). Um den optimalen Wert für t zu erhalten, berechnet man $Q_{ges}(T,t)$ für eine geeignete Anzahl von Stützstellen, z. B. eine Teilmenge der Vereinigung der Stützstellen von PP_1 und PP_2 und bestimmt davon das Maximum. Der zugehörige Wert von t ist dann die Zeitzuteilung für das erste Profil; die für das Zweite ist entsprechend $T-t$.

4 Kombination von n Profilen

Die in Abschnitt 3 erläuterten Methoden dienen dazu, zwei Performanzprofile miteinander zu kombinieren. Um mehrere Profile kombinieren zu können benötigt man das *Gesamtprofil* von mehreren Profilen.

4.1 Erweiterung der Näherungsverfahren auf n Profile

4.1.1 Gesamtprofile von 2 Performanzprofilen

Betrachtet man das Ergebnis der Kombination linear approximierter Performanzprofile aus Abschnitt 3.1 genauer, so fällt auf, daß die Zeitzuteilungen von der Gesamtzeit T abhängig sind; dies bedeutet insbesondere, daß t_1 eine Funktion $t_1(T)$ darstellt, die zu jeder Gesamtzeit T die Zeitzuteilung zum ersten Algorithmus liefert, für die die zu erwartende Qualität $Q_{ges}(T, t_1)$ des Gesamtergebnisses maximal wird. Daher läßt sich t_1 in Q_{ges} eliminieren, indem dafür $t_1(T)$ eingesetzt wird:

$$\begin{aligned} Q_{ges}(T, t_1(T)) &= PP_1(t_1(T))\ PP_2(T - t_1(T)) \\ &= \left(q_1 + a_1\left(\frac{T}{2} + \frac{a_1 q_2 - a_2 q_1}{2a_1 a_2}\right)\right)\left(q_2 + a_2\left(\frac{T}{2} + \frac{a_2 q_1 - a_1 q_2}{2a_1 a_2}\right)\right) \\ &= \frac{(a_1 a_2 T + a_1 q_2 + a_2 q_1)^2}{4a_1 a_2} \\ &=: Q^{lin}_{ges}(T) \end{aligned} \tag{20}$$

Das Ergebnis ist somit eine Funktion $Q^{lin}_{ges}(T)$, die nur noch von der Gesamtzeit T abhängig ist. (siehe Abb. 9)

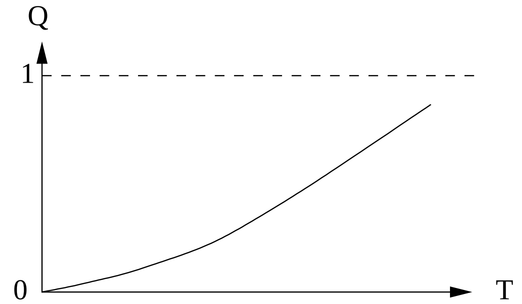

Abbildung 9: Gesamtprofil im linearen Fall

Diese Funktion liefert zu jeder Gesamtzeit T die zu erwartende Qualität, die das Ergebnis der Kombination der beiden Algorithmen haben wird, wenn T nach der Methode aus Abschnitt 3.1 verteilt wird. $Q^{lin}_{ges}(T)$ liefert also ein *Gesamtperformanzprofil*, das ein Performanzprofil darstellt, welches das Verhalten beschreibt, das beiden Algorithmen aufweisen, wenn man sie mit einer der in Abschnitt 3 beschriebenen Zeitverteilungsmethoden als in sich geschlossene Einheit betrachtet.

Ähnlich kann man bei den exponentiell approximierten Performanzprofilen aus Abschnitt 3.2 vorgehen: Setzt man wieder die von T abhängige Zeit $t_1(T)$ in Q_{ges} ein, erhält man für das Gesamtprofil:

$$Q^{exp}_{ges}(T) = 2 - \eta_1 e^{-\lambda_1 \frac{\ln(\eta_1\lambda_1) - \ln(\eta_2\lambda_2) + \lambda_2 T}{\lambda_1 + \lambda_2}} - \eta_2 e^{-\lambda_2 \frac{\ln(\eta_2\lambda_2) - \ln(\eta_1\lambda_1) + \lambda_1 T}{\lambda_1 + \lambda_2}} \tag{21}$$

Bei den linear interpolierten Performanzprofilen (3.3) läßt sich für das Gesamtprofil keine geschlossene Formel angeben; erhalten kann man es durch explizite Berechnung von kompletten Zeitverteilungen für eine geeignete Menge von Gesamtzeiten T_i. Aus den resultierenden Verteilungen $(t_1)_i$ und $(t_2)_i$ lassen sich die Gesamtqualitäten $Q_i = Q_{ges}(T_i)$ errechnen. Das Gesamtprofil ist dann das linear interpolierte Performanzprofil über (T_i, Q_i).

4.1.2 Gesamtprofile von mehr als 2 Performanzprofilen

Seien $PP_1, \ldots, PP_n$ die zu kombinierenden Profile. Ein Algorithmus, der das Gesamtprofil von n Profilen liefert, ist:

Gesamtprofil $(PP_1, \ldots, PP_n)$:

- falls $n = 1$, gib PP_1 als Ergebnis zurück.
- falls $n = 2$, berechne Gesamtprofil von PP_1 und PP_2 mit den Methoden aus Abschnitt 4.1.1 und gib es als Ergebnis zurück.
- sonst teile $PP_1, \ldots, PP_n$ in zwei Hälften und berechne rekursiv $G_1 =$ **Gesamtprofil**$(PP_1, \ldots, PP_{\lfloor \frac{n}{2} \rfloor})$ und $G_2 =$ **Gesamtprofil**$(PP_{\lceil \frac{n}{2} \rceil}, \ldots, PP_n)$.
- Berechne das Gesamtprofil der Gesamtprofile G_1 und G_2 mit den Methoden aus Abschnitt 4.1.1 und gib es als Ergebnis zurück.

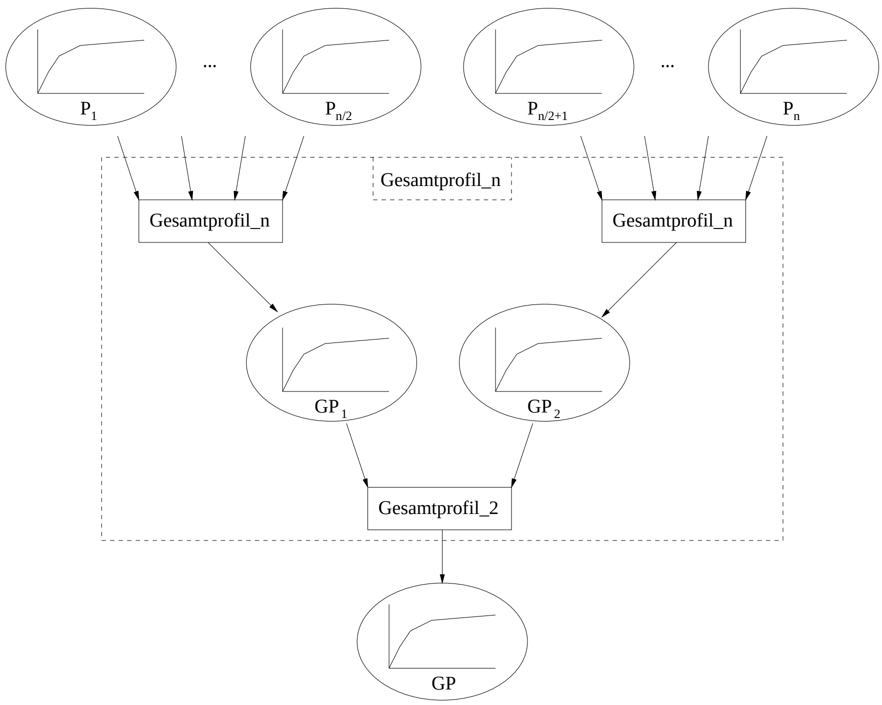

Abbildung 10: Schematische Darstellung des Algorithmus zur Berechnung des Gesamtprofils von n Performanzprofilen

4.1.3 Zeitverteilung für n Profile

Mit Hilfe der Funktion aus 4.1.2 kann man nun rekursiv die Verteilung von Zeit T für n Profile bestimmen:

Zeitverteilung $(T, PP_1, \ldots, PP_n)$:

- falls $n = 1$, gib T als Ergebnis zurück.
- falls $n = 2$, berechne Verteilung (t_1, t_2) auf PP_1 und PP_2 nach den Methoden aus Abschnitt 3 und gib sie als Ergebnis zurück.
- sonst teile $PP_1, \ldots, PP_n$ in zwei Hälften und berechne die Gesamtprofile
  $G_1 = \textbf{Gesamtprofil}(PP_1, \ldots, PP_{\lfloor \frac{n}{2} \rfloor})$ und
  $G_2 = \textbf{Gesamtprofil}(PP_{\lceil \frac{n}{2} \rceil}, \ldots, PP_n)$.
- Verteile (nach Abschnitt 3) die Zeit T auf die Profile G_1 und G_2. (Sei das Ergebnis die Verteilung (T_1, T_2).)
- Verteile T_1 und T_2 rekursiv auf die beiden Hälften; Dabei sei V_1 die Verteilung auf die erste Hälfte, V_2 die auf die zweite, also:
  $V_1 = \textbf{Zeitverteilung}(T_1, PP_1, \ldots, PP_{\lfloor \frac{n}{2} \rfloor})$ und
  $V_2 = \textbf{Zeitverteilung}(T_2, PP_{\lceil \frac{n}{2} \rceil}, \ldots, PP_n)$.
- Die resultierende Verteilung auf alle Profile ist dann die Konkatenation der partiellen Verteilungen V_1 und V_2.

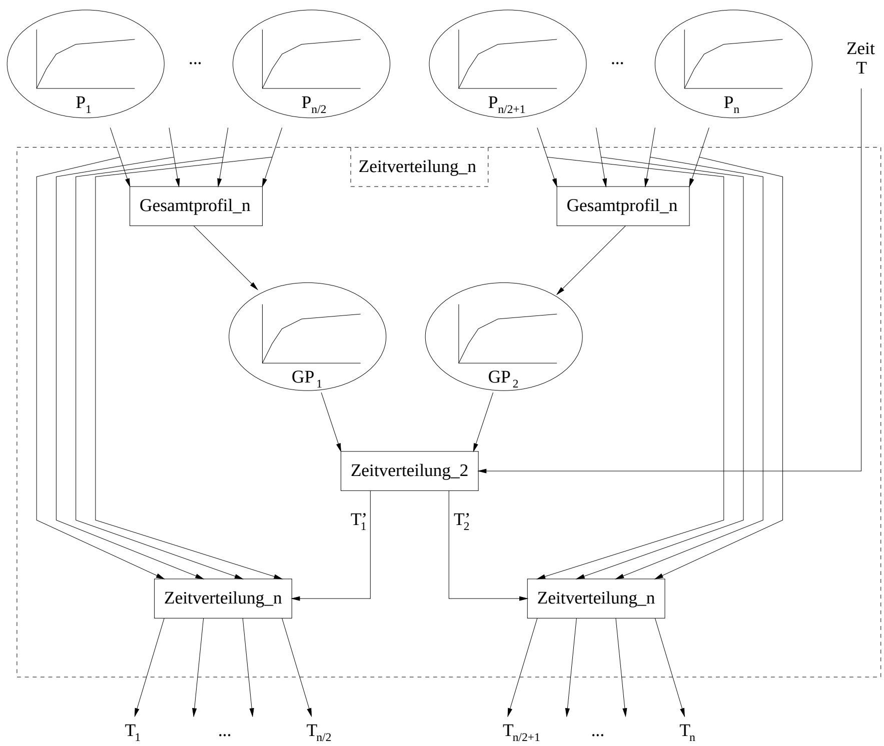

Abbildung 11: Schematische Darstellung des Algorithmus zur Berechnung der Zeitverteilung auf n Performanzprofilen

4.2 Kombination von n Profilen mittels Gradientenverfahren

4.2.1 Gradientenverfahren

Bei einem Gradientenverfahren, auch Hillclimbing-Algorithmus genannt, wird eine Funktion nach einem Extremum abgesucht, indem von einem Startpunkt aus immer eine bestimmte Schrittweite in die Richtung weitergegangen wird, in der ein besserer (z. B. bei der Suche nach einem Maximum ein höherer) Funktionswert zu finden ist. Eine einfachere Variante ändert die Richtung erst dann, wenn der Funktionswert wieder schlechter wird. Ist irgendwann kein besserer Wert mehr zu finden, wird die Schrittweite herabgesetzt und weitergesucht.

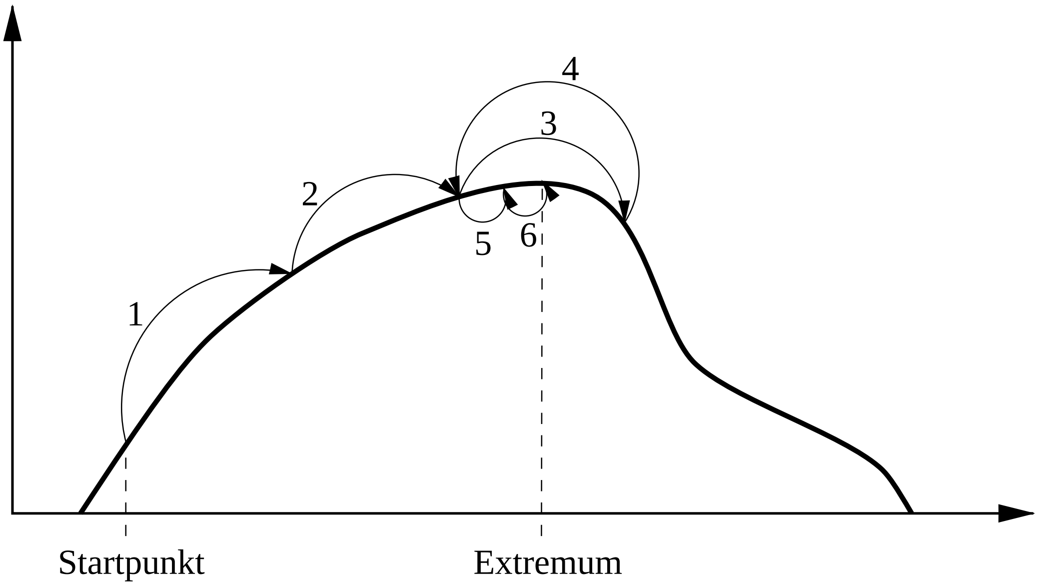

Abbildung 12: Skizze des Ablaufes eines Gradientenverfahren

Abbruchbedingungen sind meist untere Toleranzgrenzen für Schrittweite und bzw. oder Funktionswertänderung. Abbildung 12 zeigt den Ablauf eines Gradientenverfahren zur Bestimmung eines Maximums auf einer einstelligen Funktion.

4.2.2 Gradientenverfahren zur Ressourcenverteilung

Die bisher vorgestellten Berechnungsmethoden zur Ressourcenverteilungen können nur Ressourcen zwischen zwei Anytime-Modulen verteilen (siehe Abschnitt 3). Wenn die Verteilung auf mehr als zwei Module erfolgen soll muß auf binäre Baumstrukturen zurückgegriffen werden (siehe Abschnitt 4.1.3).

Mit einem Gradientenverfahren kann man Ressourcen auch gleichzeitig auf mehr als zwei Module verteilen. Dazu wird ein Hillclimbing-Algorithmus auf eine Verknüpfung[3] der aus den Performanzprofilen der Anytime-Algorithmen erstellten Funktionen angesetzt.

Im folgenden Abschnitt wird näher auf die Hillclimbing-Methode eingegangen.

[3] Diese Verknüpfung sollte den Sinn der Performanzprofile respektieren. Beispiele wären + oder *.

4.3 Eine Hillclimbing-Methode zur Kombinierung von Performanzprofilen

4.3.1 Eingabeformat für den Hillclimbing-Algorithmus

Als Eingabeformat für den Hillclimbing-Algorithmus wird die abschnittsweise lineare Darstellung (Abschnitt 3.3) verwendet, da diese für als Punktemengen gegebene Performanzprofile genauer repräsentiert, als Funktionen, welche durch lineare oder exponentielle Regression approximiert werden. Letztere sind, wie in den Abschnitten 3.1 und 3.2 beschrieben, nur eine Annäherung an die verwendeten Performanzprofile.

Intern werden die Punktemengen in zwischen den Stützpunkten lineare Funktionen umgewandelt, welche dann zum Vergleich der Qualitätsänderung aufsummiert oder multipliziert (siehe Abschnitt 5.1, Parameter `:combine`) werden. Generell ist der verwendete Hillclimbing-Algorithmus so gestaltet, daß er jede Funktion in Form eines unären Lambda-Ausdruckes[4] als Performanzprofil verwenden kann.

4.3.2 Skizzierung des Hillclimbing-Algorithmus

Der hier verwendete Hillclimbing-Algorithmus soll eine optimale Zeitverteilung finden. Dazu fängt er mit einer (standardmäßig gleichverteilten) Aufteilung[5] an und verteilt die zu vergebende Gesamtzeit solange um, bis ein Optimum erreicht wird, d.h. bis sich die Qualität und bzw. oder (siehe Abschnitt 5.1, Parameter `:logical-operator`) die Schrittweite nur noch um weniger als einen im voraus festgelegten Toleranzwert ändert. Die Schrittweite entspricht hier der umzuverteilenden Menge an Ressourcen.

Zur Veranschaulichung des Hillclimbing-Algorithmus eignet sich sehr gut die Variante mit zwei Performanzprofilen (siehe Abb. 13). In der dritten Dimension wird dann die zu erwartende Qualität

$$Q\left(t_1, t_2\right) \quad = \quad PP_1\left(t_1\right) \circ PP_2\left(t_2\right) \tag{22}$$

aufgetragen. (Abb. 13b)

Da sich der Algorithmus dabei ausschließlich auf der Strecke $\overline{(0|T)\,(T|0)}$ bewegt (Abb. 13a), gilt $t_2 = T - t_1$ mit $t_1, t_2 > 0$ und somit ist die Formel (22) auf

$$Q\left(t\right) \quad = \quad PP_1\left(t_1\right) \circ PP_2\left(t - t_1\right) \tag{23}$$

reduzierbar.

4.3.3 Mehrdimensionaler Hillclimbing-Algorithmus

Aus dem zweidimensionalen Modell aus Formel (23) kann das n-dimensionale Modell $Q = PP_1\left(t_1\right) \circ PP_2\left(t_2\right) \circ \cdots \circ PP_n\left(t_n\right)$ abgeleitet werden, wobei immer $T = \sum_{i=1}^{n} t_i$ gilt.

Bei Darstellung des n-dimensionalen Modells mit $n = 3$ bietet es sich an, sich die Qualität als Dichte im Raum interpretieren (siehe Abb. 14b, Darstellung der Qualität in Form von Isographen). Der Hillclimbing-Algorithmus selbst läuft nun nicht entlang einer Geraden, sondern auf einer Ebene. Allerdings ist der mögliche Raum

[4] Die Lambda-Ausdrücke werden weder auf Monotonie oder Normierung noch auf Korrektheit überprüft.

[5] Optional können auch andere Startpunkte angegeben werden. (Siehe Abschnitt 5.1, Parameter `:starting-point`)

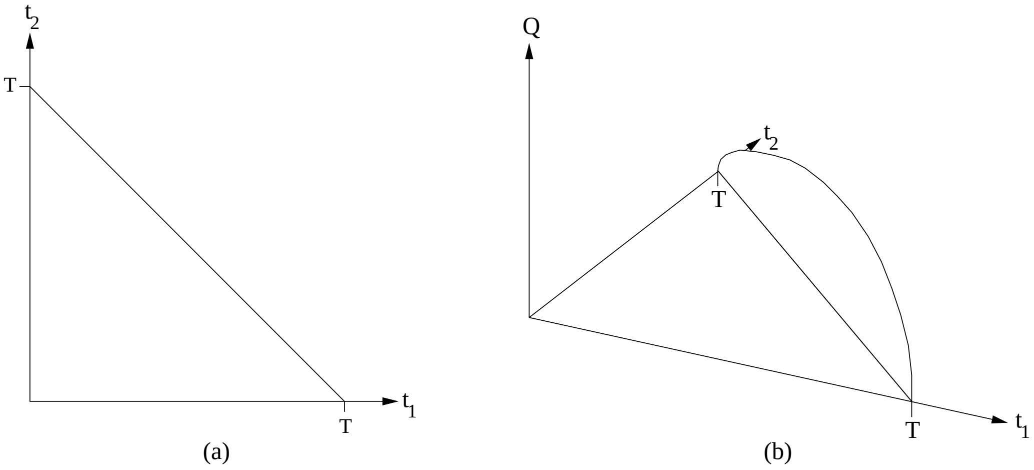

Abbildung 13: Zweidimensionaler Hillclimbing-Algorithmus

für Lösungen durch die Gleichung $T = \sum_{i=1}^{n} t_i$ auf eine bestimmte Fläche beschränkt (siehe Abb. 14a), so wie der Lösungsraum beim zweidimensionalen Hillclimbing-Algorithmus auf eine Strecke begrenzt wurde (siehe Abb. 13a).

Als Standardwerte für den Start des Hillclimbing-Algorithmus wird in ORCAN die Gleichverteilung der Ressourcen gewählt: $t_i = \frac{T}{n}\ \forall i \in \{1, \cdots, n\}$. Da $\sum_{i=1}^{n} \frac{T}{n} = T$, liegt der Punkt $\left(\frac{T}{n}, \ldots, \frac{T}{n}\right)$ im Ergebnisraum.

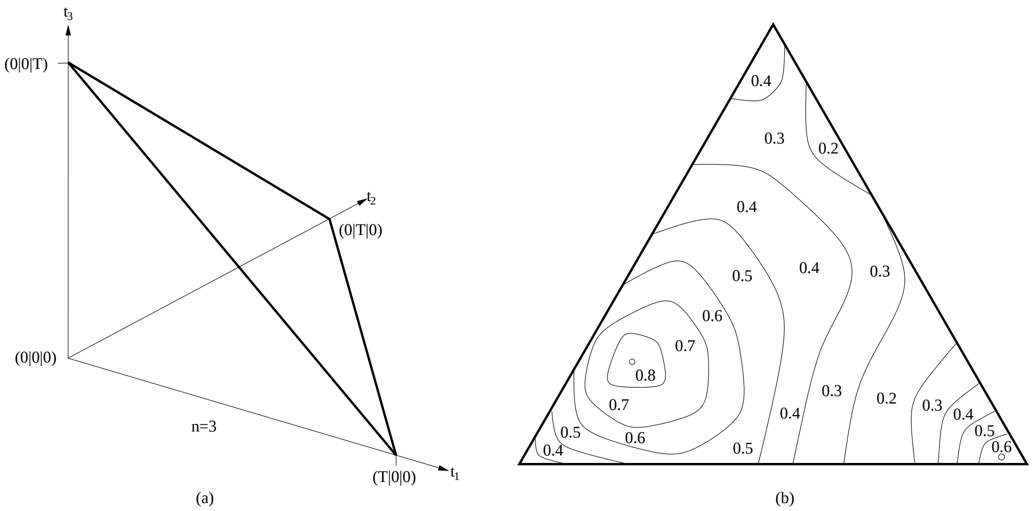

Abbildung 14: Dreidimensionaler Hillclimbing-Algorithmus

4.3.4 Verfahren zur Umverteilung der Gesamtzeit

Der Hillclimbing-Algorithmus rechnet zuerst für jedes Anytime-Modul[6] aus, was für eine Qualitätsänderung sich bei einer Ressourcenumverteilung zugunsten des entsprechenden Moduls ergeben würde und wählt daraus die günstigste Richtung aus,

[6]Die Nummer des Moduls wird im folgenden als Richtung bezeichnet. Dies läßt sich in Abb. 14 leicht veranschaulichen: Jede Umverteilung zugunsten eines Moduls bedeutet eine Bewegung innerhalb des Dreiecks in Richtung der entsprechenden Ecke.

also jene mit der größten Qualitätssteigerung.

Es wird Zeit zugunsten dieses Moduls solange umverteilt, wie dadurch noch eine Verbesserung der Qualität erreicht werden kann. Der Rechenaufwand wird stark verringert – vor allem bei sehr vielen Performanzprofilen – wenn die optimale Richtung nicht bei jeder Umverteilung neu berechnet wird, da sich dann der Verteilungsvektor weniger oft ändert und entsprechend seltener berechnet werden muß.

Ist der Algorithmus an einer Stelle angelangt, an der die aktuell eingeschlagene Richtung keine Verbesserung mit sich bringt, so werden von diesem Punkt aus alle anderen Richtungen ausgetestet und die beste davon ausgesucht. Findet der Algorithmus keine Verbesserung in irgendeine Richtung, so wird die Schrittweite heruntergesetzt und erneut alle Richtungen nach der besten Möglichkeit abgesucht.

Abbruchbedingungen für die Suche sind:

- Eines der Performanzprofile hat die gesamte Zeit zugewiesen bekommen.
- Die Schrittweite und bzw.[7] oder Qualitätsänderung haben ihre Toleranzgrenze erreicht.

Bei der Umverteilung der zu vergebenden Zeit ist wichtig, daß keines der Anytime-Module weniger als Null Zeiteinheiten – also eine negative Laufzeit – zugewiesen bekommt[8].

Dies wird dadurch erreicht, daß man die prozentuale Zeitverteilung aller Anytime-Module, die *keine* Zeit hinzu addiert bekommen sollen, berechnet und dementsprechend dann die Zeiteinheiten abzieht. Kurz: Ein Anytime-Modul, dem bisher wenig Zeit zugeteilt wurde, bekommt bei einer Umverteilung weniger Zeit abgezogen als ein Modul welches bisher viel Zeit zugeteilt bekam. Die Umverteilung von *step* Zeiteinheiten auf das Modul *dir* sieht dann wie folgt aus:

$$step_{aktuell} = min\left(step, \sum_{j \neq dir} t_{alt_j}\right) \tag{24}$$

$$s_i = \left\{ \begin{array}{ll} -\frac{t_{alt_i}}{\sum_{i \neq dir} t_{alt_i}} \cdot step_{aktuell} & \text{falls } i \neq dir \\ step_{aktuell} & \text{sonst} \end{array} \right\} \forall i \in \{1, \ldots, n\} \tag{25}$$

$$t_{neu_i} = t_{alt_i} + s_i \quad \forall i \in \{1, \ldots, n\} \tag{26}$$

4.3.5 Vergleich mit dem Hillclimbing-Algorithmus von Zilberstein

Zilberstein hat in (S. Zilberstein, 1993, Seite 79f.) zwar auch einen Hillclimbing-Algorithmus aufgeführt; dieser hat aber immer nur die Zeit zwischen zwei Modulen ausgetauscht, egal auf wieviele Module die Zeit zu verteilen war. Dazu hat er die beiden Module herausgesucht, bei denen die Ressourcenumverteilung von dem ersten zu dem zweiten Modul die größte Qualitätssteigerung mit sich brachte.

Wie in den folgenden Absätzen gezeigt wird, hat dieser n-dimensionale Hillclimbing-Algorithmus aber einen Rechenaufwand von $2\left((n-1)!\right)$ probeweisen Umverteilungen pro Schritt, während der in Abschnitt 4.3.4 beschriebene Hillclimbing-Algorithmus nur n probeweise Umverteilungen pro Schritt braucht und somit vor allem bei hohem n weniger rechenintensiv ist.

[7] Siehe Abschnitt 5.1, Parameter `:logical-operator`

[8] In Bezug auf die Abbildungen 13 und 14 bedeutet dies, daß der aktuelle Standpunkt innerhalbder Strecke (Abb. 13) bzw. des Dreiecks (Abb. 14) bleibt

Zilbersteins Methode berechnet für jedes mögliche Paar von Modulen, wie groß der Qualitätsgewinn bei einer Umverteilung in eine der beiden möglichen Richtung ist. Dazu werden $2\,((n-1)!)$ Umverteilungen durchgeführt. Dies kann man durch Induktion über die Anzahl der Module zeigen: Die zu erwartende Qualität einer Umverteilung des n-ten Moduls mit jedem einzelnen der restlichen, also $n-1$ Modulen wird berechnet. Die Umverteilung des $n-1$-te Modul mit einem anderen Modul ist nur noch für $n-2$ Module notwendig, da die Umverteilung mit dem Modul n ja bereits im vorherigen Schritt berechnet wurde. Sind nur (noch) zwei Module zu vergleichen, so ist nur noch eine Umverteilung zu berechnen. Jede der möglichen Umverteilungen muß in zwei Richtungen berechnet werden, also muß man $2\,((n-1)!)$ Umverteilungen berechnen. Danach muß natürlich noch das Maximum der $2\,((n-1)!)$ Werte gefunden werden.

Die in Abschnitt 4.3.4 beschriebene Methode braucht dagegen nur n Umverteilungen, was man ebenfalls leicht zeigen kann: Für jedes der n Module wird die Umverteilung zugunsten dieses Moduls berechnet. Also braucht man n Umverteilungen und muß auch nur von n Werten das Maximum finden.

5 Realisierung und Einsatz

Die Implementation der verschiedenen Berechnungsmethoden zur Compilierung von Anytime-Algorithmen – das System ORCAN – wurde in der Sprache Common Lisp realisiert. Die zur Entwicklung verwendete Implementation von Common Lisp war Liquid Common Lisp.

Das System ORCAN wird innerhalb der Optimierung der Ressourcenverteilung auf verschiedene Anytime-Algorithmen in dem System JAMES[9] (Wittig, 1998), verwendet, welches z. B. Anwendung in BOLA[10] (Blocher, Anselm, 1999) findet.

5.1 Interface

---

distribute *Funktion*

Syntax: distribute *time profiles [key keyvalue ...]*

Aufgabe: Die Funktion `distribute` fungiert als Interface zu ORCAN und wird verwendet zur Berechnung einer Ressourcenverteilung auf mehrere, durch Performanzprofile repräsentierte Anytime-Module

Parameter:

time: Anzahl der zur Verfügung stehenden Ressourceneinheiten

profiles: Liste der zu verwendenden Performanzprofile

Optionale Keyword-Parameter:

`:method` bestimmt die verwendete Berechnungsmethode:

- **`:segment`** Abschnittsweise lineare Funktionen (Defaulteinstellung)
- **`:linear`** Lineare Regression
- **`:exponetial`** Exponentielle Regression
- **`:hill`** Gradientenverfahren / Hillclimbing-Methode

`:output` legt fest, ob die Zeitangaben des Resultats als Absolutwerte oder der prozentualer Anteil an der insgesamt verteilten Zeit sein sollen:

- **`:absolute`** Absolutwerte (Defaulteinstellung)
- **`:percentage`** prozentuale Anteile

Optionale Keyword-Parameter für das Gradientenverfahren:

Wird das Gradientenverfahren verwendet, also die Funktion `distribute` mit `(distribute` *`time profiles`* `:method :hill)` aufgerufen, sind weiterhin folgende Keyword-Parameter möglich[11]:

`:combine` bestimmt die Funktion, mit der die zu erwartenden Qualitäten der einzelnen Performanzprofile zusammengerechnet werden. Neben

---

[9] JAMES steht für Java Anytime Management & Editor System.

[10] BOLA steht für Beschränkt-optimaler Lokalisationsagent.

[11] Bei anderen Methoden haben sie keinerlei Wirkung.

#'+ Summierung aller Qualitäten (Defaulteinstellung)

#'* Multiplizierung aller Qualitäten (Diese Einstellung ist nicht zu empfehlen, sobald ein Performanzprofil für ein $x > 0$ eine Qualität von 0 zurückliefert.)

#'+* hat die gleiche Funktionsweise wie #'+, es sei denn, daß eine der Qualitäten für ein $x > 0$ Null ist. Dann verhält es sich wie #'*. Die Ergebnisse sind bei dieser Einstellung allerdings nicht mehr auf 1 normiert.

kann auch jede andere Funktion, die eine beliebige Anzahl von Zahlen akkumuliert, angegeben werden. In diesem Fall sind die Ergebnisse ebenfalls nicht mehr normiert!

:step ist die anfängliche Schrittweite. Werte kleiner als 10^{-5} sind aufgrund der Rechengenauigkeit von Liquid Common Lisp nicht zu empfehlen. Ebenfalls sollte der Werte einen bestimmten Bruchteil der zu vergebenden Zeit nicht überschreiten. 10^{-1} ist die Defaulteinstellung.

:tolerance-diff legt den Grenzwert für die Änderung der Qualität fest. Werte kleiner als 10^{-5} sind aufgrund der Rechengenauigkeit von Liquid Common Lisp nicht zu empfehlen. Ebenfalls sollte der Wert eindeutig kleiner als 1 sein, da der Algorithmus sonst im Zweifelsfall sofort und ohne irgendwelche sinnvollen Berechnungen terminiert. 10^{-4} ist die Defaulteinstellung.

:tolerance-step legt den Grenzwert für die Änderung der Schrittweite fest. Werte kleiner als 10^{-5} sind aufgrund der Rechengenauigkeit von Liquid Common Lisp nicht zu empfehlen. Ebenfalls sollte der Werte einen bestimmten Bruchteil der anfänglichen Schrittweite nicht überschreiten. 10^{-4} ist die Defaulteinstellung.

:logical-operator gibt die Verknüpfungsfunktion an, mit der die beiden Abbruchbedingungen verknüpft werden. Neben

#'or Oder-Verknüpfung (Defaulteinstellung)

#'and Und-Verknüpfung

kann auch jede andere zweistellige logische Funktion angewendet werden.

:tolerance kann genutzt werden, um die beiden obigen Grenzwerte gleichzeitig auf einen gemeinsamen Grenzwert zu setzen. Standardmäßig werden die beiden Grenzwerte getrennt angegeben.

:step-reduction ist der Faktor durch den die Schrittweite dividiert wird, falls sie verringert werden muß bzw. mit dem sie multipliziert wird, falls die Schrittweite vergrößert werden muß. Er darf auf keine Fall auf einen Wert kleiner oder gleich 1 gesetzt werden. Die Defaulteinstellung ist 2.

:starting-point gibt die initiale Zeitverteilung (den „Startpunkt") an. Die Defaulteinstellung ist $\frac{time}{Anzahl\ der\ Performanzprofile}$, also eine Gleichverteilung der Ressource auf alle Module.

Rückgabewert: Das Resultat der Funktion sind drei Werte[12]. Dabei ist

1. **Wert:** Eine Liste, in der für jedes Anytime-Modul die Anzahl der Ressourceneinheiten welche ihm zugeteilt wurden, aufgeführt sind. Die Reihenfolge entspricht der der übergebenen Performanzprofilliste.

[12]in Form von sog. Values

2. Wert: Die approximierte Qualität des Ergebnisses, die bei dieser Verteilung zu erwarten ist.

3. Wert: Die nicht verbrauchte Zeit, falls mehr Zeit verteilt werden sollte, als die Module benötigen um ein optimales Ergebnis zu liefern, ansonsten 0.

Lisp-Package: orcan

5.2 Beispiele

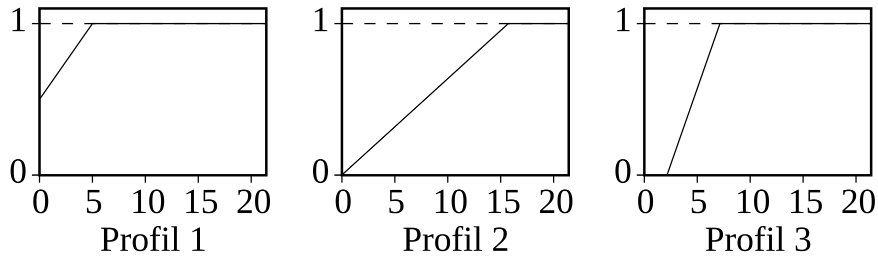

Abbildung 15: Die für die Beispiele verwendeten Profile

Profil 1:	$\{(0\|0.5),(1\|0.6),(2\|0.7),(3\|0.8),(4\|0.9),(5\|1),(6\|1),(7\|1))\}$
Profil 2:	$\{(0\|0),(16\|1))\}$
Profil 3:	$\{(0\|0),(1\|0),(2\|0),(3\|0.2),(4\|0.4),(5\|0.6),(6\|0.8),(7\|1),(8\|1)\}$

Tabelle 1: Stützstellen der in den Beispielen verwendeten Performanzprofile

Beispiel 1 (Lineare Regression)

In diesem Beispiel werden 28 Zeiteinheiten auf die drei Anytime-Module – welche durch die drei Performanzprofile in Abbildung 15 bzw. Tabelle 1 repräsentiert werden – durch die lineare Regressionsmethode verteilt und die Verteilung in absoluten Werten – also in Zeiteinheiten – ausgegeben.

Das Ergebnis bedeutet folgendes: Das erste Anytime-Modul sollte fünf Zeiteinheiten bekommen, das zweite sechzehn und das dritte sieben. Die zu erwartende Qualität des Ergebnisses ist fast Eins und es bleiben bei der Berechnung zwei Zeiteinheiten übrig.

Eingabe

```
(distribute 28 (list profil1 profil2 profil3)
           :method :linear
           :output :absolute)
```

Ausgabe

```
(5 16 7)
0.9999999999999999
2.0
```

□

Beispiel 2 (Gradientenverfahren)

In diesem Beispiel werden dieselben Profile und dieselbe Gesamtzeit verwendet, diesesmal wird das Ergebnis jedoch nach dem Gradientenverfahren errechnet.

Durch optionale Parameter werden für den Hillclimbing-Algorithmus noch folgende Einstellungen festgelegt: Die Qualitäten der einzelnen Module sollen aufsummiert werden, die anfängliche Schrittweite soll ein Zehntel Zeiteinheit betragen, die Toleranzgrenze für Schrittweite als auch Qualitätsänderung soll ein Tausendstel Zeiteinheiten betragen, es muß nur eine der beiden Toleranzgrenzen überschritten werden, um den Algorithmus zu stoppen und falls keine Verbesserung auftritt soll die Schrittweite halbiert werden.

Das Ergebnis ist – bis auf minimale, durch Rundungsfehler entstandene Differenzen – das gleiche wie bei dem vorangegangenen Beispiel.

Eingabe

```
(distribute 28 (list profil1 profil2 profil3
               :method :hill
               :combine #'+
               :step 0.1
               :tolerance 0.001
               :logical-operator #'or
               :step-reduction 2)
```

Ausgabe

```
(5 16 7)
1.0
1.99999999999999990
```

□

Beispiel 3 (Exponentielle Regression)

In diesem Beispiel werden nur zehn Zeiteinheiten auf die bereits bekannten drei Anytime-Module verteilt; diesesmal durch die exponentielle Regressionsmethode.

Das Ergebnis ist wie folgt zu deuten: Das erste Anytime-Modul sollte etwa 2.3 Zeiteinheiten bekommen, das zweite etwa 2.8 und das dritte 4.8. Die zu erwartende Qualität des Ergebnisses liegt bei ca. 86% und es bleibt bei der Berechnung keine Zeit übrig.

Eingabe

```
(distribute 10 (list profil1 profil2 profil3)
               :method :exponential)
```

Ausgabe

```
(2.3215458319752576 2.837863998770923 4.84059016925382)
0.8613160703139167
0
```

□

6 Vergleich der verschiedenen Methoden

Die in diesem Abschnitt vorgestellten Vor- und Nachteile der verschiedenen Methoden zur Ressourcenverteilung werden in Tabelle 2 kurz zusammengefaßt.

Methode	Vorteile	Nachteile
Linear und Exponentiell	schnell für wenig Module; genau für bestimmte Profilklassen	Näherungsmethode; aufwendig für viele Profile
Abschnittsweise linear	schnell bei wenigen Stützstellen; genauste Berechnungsmethode	langsam bei vielen Stützstellen; aufwendig für viele Profile
Hillclimbing	für viele Profile geeignet	bei wenigen Profilen höhere Laufzeit; evtl. Probleme bei mehreren lokalen Extrema

Tabelle 2: Übersicht über die verschiedenen Methoden

Die Vorteile der linearen und der exponentiellen Regressionsmethode liegt in der Geschwindigkeit bei wenigen Modulen und in der Genauigkeit bei bestimmten Profilklassen.

Bei der linearen Regressionsmethode ist dies die Klasse der Profile, welche eine konstante Steigung aufweisen. Mit dieser Bedingung können sie durch lineare Regression in äquivalente Funktionen umgewandelt werden. Siehe auch Abschnitt 3.1.

Im Falle der exponentiellen Regressionsmethode ist dies die Klasse der Profile, welche der Form $PP(t) = 1 - \eta e^{-\lambda t}$ genügen. Dadurch können sie durch exponentielle Regression in äquivalente Funktionen umgewandelt werden. Siehe auch Abschnitt 3.2.

Ein Nachteil der Regressionsmethoden liegt darin, daß durch die lineare bzw. exponentielle Regression in anderen Fällen als den beiden obengenannten der Fehler gegenüber dem eigentlichen Verlauf des Profils noch größer werden kann als er durch die Generierung der Performanzprofile aus empirischen Daten bereits ist. Ein weiterer Nachteil liegt darin, daß der Rechenaufwand mit der Anzahl der Profile exponentiell steigt.

Die abschnittsweise lineare Methode approximiert die Performanzprofile gegenüber den beiden vorgenannten *nicht* weiter und liefert daher weniger fehlerbehaftete Ergebnisse. Dafür steigt der Rechenaufwand mit der Anzahl der Profile und Stützstellen stark an. Sie ist also nur für Fälle mit wenigen Profilen und wenigen Stützstellen geeignet.

Der Vorteil der Hillclimbing-Methode liegt in der im Vergleich zu den anderen

Methoden niedrigen Laufzeit bei vielen zu kombinierenden Anytime-Modulen, weil der Rechenaufwand mit der Anzahl der Module nur linear steigt. Da aber in den meisten Fällen nicht nach jedem Schritt die Richtung neu berechnet werden muß, ist dies eine Obergrenze. Auch bei sehr vielen ($n > 100$) Anytime-Modulen funktioniert er noch in akzeptabler Rechenzeit.

Folgendes sind die Nachteile der Hillclimbing-Methode: Bei wenigen Profilen wird im Vergleich zu den anderen Methoden verhältnismäßig viel Rechenzeit verbraucht. Desweiteren besteht die Möglichkeit, daß nur ein lokales und kein globales Extremum gefunden wird. Außerdem kann evtl. kein verwertbares Ergebnis gefunden werden, falls alle Performanzprofile zu Beginn konstant bleiben. In solchen Fällen sind andere Startparameter oder eine andere anfängliche Schrittweite zu wählen.

7 Aussichten und Erweiterungsmöglichkeiten

Eine Erweiterungsmöglichkeit, welche vor allem auch das Einsatzspektrum von ORCAN vergrößert, ist die Verwendung von Funktionen bzw. deren Parameter (z. B. Exponentialfunktionen oder Lambda-Ausdrücke) als zusätzliches Eingabeformat. Dies hat beim Hillclimbing-Algorithmus außerdem im Normalfall eine Geschwindigkeitserhöhung zur Folge, da die Umrechnung der Stützpunktmengen in abschnittsweise lineare Funktionen (bzw. Lambda-Ausdrücke mit vielen Fallunterscheidungen und deren Auswertung) entfällt.

Anstelle der Umwandlung von Stützpunktmengen in abschnittsweise lineare Funktionen können alternativ auch die Regressionsmethoden auf diese Mengen angewendet und die dadurch gewonnenen Funktionen als Eingabe für den Hillclimbing-Algorithmus genutzt werden.

Der Hillclimbing-Algorithmus kann auch noch durch verschiedene Verfahren zum Umgehen von lokalen Extrema erweitert werden. Dadurch braucht der Hillclimbing-Algorithmus zwar länger zum Berechnen, aber liefert genauere Ergebnisse. Solche Verfahren würden sich vor allem bei der Implementation des Hillclimbing-Algorithmus in Form eines Anytime-Algorithmus lohnen, da mit der zusätzlichen Zeit ein bereits gefundenes Ergebnis z. B. daraufhin überprüft werden kann, ob es sich um ein lokales Extremum oder eine Ebene handelt. Anschließend kann gegebenenfalls versucht werden das globale Maximum zu finden.

Eine ganz anderer Art der Erweiterung wäre die Interpretation der Stützpunkte als treppenstufenförmige Performanzprofile, da die Performanzprofile von Anytime--Algorithmenoft treppenstufenförmig sind[13]. Dann gäbe es nur maximal S mögliche (unterschiedliche) Zeitverteilungen, wobei

$$S = \prod_{i=1}^{n} S_i \text{ mit } S_i = \text{Anzahl der Stützpunkte des } i\text{-ten Performanzprofil} \quad (27)$$

Bei einer Implementation eines solchen Algorithmus als Anytime-Algorithmus kann z. B. bei für eine vollständige Berechnung nicht ausreichenden Ressourcen auf die Einbeziehung einiger Stützpunkte verzichtet werden.

[13] Solange ein neuer, genauerer Wert berechnet wird, wird der zuletzt errechnete Wert zurückgegeben und die Qualität bleibt konstant. Ist der neue Wert errechnet springt die Qualität sprunghaft an und eine Treppenstufe entsteht.

Literatur

Blocher, Anselm. (1999). *Ressourcenadaptierende Raumbeschreibung: Ein beschränkt-optimaler Lokalisationsagent.* Doktorarbeit, Fachbereich 14 - Informatik, Univ. des Saarlandes, Saarbrücken.

Dean, Thomas und Boddy, Mark. (1988). An Analysis of Time-Dependent Planning. In *Proc. of the Seventh National Conf. on Artificial Intelligence* (S. 49-54). Minneapolis, Minnesota.

Görz, Günther und Kesseler, Marcus. (1994). An Anytime Protocol for Speech Parsing. In Niemann, Heinrich und de Mori, Renato und Hanrieder, Gerhard (Hrsg.), *Progress and Prospects of Speech Research and Technology* (S. 257-261).

I. N. Bronstein und K. A. Semendjajew. (1967). *Taschenbuch der mathematik (7. auflage).* Zürich, Frankfurt am Main: Harri Deutsch Verlag.

Russell, Stuart J. und Wefald, Eric. (1991). *Do the right thing: studies in limited rationality.* Cambridge, MA: MIT Press.

S. Zilberstein. (1993). *Operational Rationality through Compilation of Anytime Algorithms.* Doktorarbeit, Department of Computer Science, Univ. of California, Berkeley, CA.

W. Wahlster und A. Blocher und J. Baus und E. Stopp und H. Speiser. (1998). Ressourcenadaptierende Objektlokalisation: Sprachliche Raumbeschreibung unter Zeitdruck. *Kognitionswissenschaft, Sonderheft zum Sonderforschungsbereich 378*, *7*(3).

W. Wahlster und W. Tack. (1997). SFB 378: Ressourcenadaptive Kognitive Prozesse. In M. Jarke und K. Pasedach und K. Pohl (Hrsg.), *Informatik'97 - Informatik als Innovationsmotor, 27. Jahresta gung der Gesellschaft für Informatik, Aachen, 24.-26. September 1997* (S. 51-57). Berlin, Heidelberg: Springer.

Wittig, F. (1998). *JAMES: Java Anytime Management and Editor System.* Diplomarbeit, Universität des Saarlandes, Technische Fakultät, Saarbrücken.