Seminar „Ausgewählte Themen des Operations Research“
bei Prof. R. Schrader

# Heuristiken

Greedy-Algorithmen
Lokale Suche
Tabu Suche

Wintersemester 2007/2008

Eingereicht von:

Florian Schuster
Datum: 12.12.2007

# 1 Allgemeines

„Heuristik“(altgriech. $\epsilon\upsilon\rho\iota\sigma\kappa\omega$ *heurisko* „ich finde, entdecke“): bezeichnet die Kunst, wahre Aussagen zu finden, im Unterschied zur Logik, die lehrt, wahre Aussagen zu begründen.
„Als Heuristik bezeichnet man eine Methode, komplexe Probleme, die sich nicht vollständig lösen lassen, mit Hilfe einfacher Regeln und unter Zuhilfenahme nur weniger Information zu entwirren.“
(Definition von Prof. G. Gigerenzer, Max-Planck-Institut für Bildungsforschung)

Der Unterschied zwischen einer Approximation und einer Heuristik besteht darin, dass man eine Approximation hinsichtlich ihrer „Güte“, d.h. ihrer Fähigkeit, die realen Verhältnisse anzunähern, quantitativ bewerten kann. Im Gegensatz dazu ist eine Heuristik ein eigenständiges Modell, für das kein solches Maß zur Bewertung existiert. Trotzdem hat sich auf einigen wissenschaftlichen Gebieten die Anwendung bestimmter Heuristiken bewährt, da sie oft bessere Ergebnisse erzielten.
Häufige Anwendungsgebiete für Heuristiken sind Kognitionswissenschaften, speziell Philosophie (Ähnlichkeit, z.B. Platons „Politeia“) und Psychologie (einfache, effiziente Regeln, die durch evolutionäre Prozesse erlernt oder gefestigt wurden), sowie Wirtschaftswissenschaften, speziell Verhaltensökonomik und Operations Research. In der Mathematik werden Heuristiken seit den späten 50er Jahren angewandt. Zunächst als Methoden zur Identifikation von Problemlösungstechniken (insbesondere für mathematische Beweise), später als „Regeln“, die verwendet wurden um gute Lösungen in Problemlöse-Prozessen zu finden. Heute versteht man unter Heuristik Techniken, die die durchschnittliche Performance von Problemlöseverfahren verbessern. Zu beachten ist hierbei, dass nicht notwendigerweise auch die Worst-Case-Performance verbessert wird!

Im Gegensatz zu Heuristiken, die ausschließlich problemspezifisch, also auf ein bestimmtes Optimierungsproblem beschränkt eingesetzt werden, definieren Meta-Heuristiken eine Abfolge von Schritten, die theoretisch auf beliebige Problemstellungen angewandt werden können. Die einzelnen Schritte jedoch müssen problemspezifisch implementiert werden (häufig wieder in Form von Heuristiken). Auch hier ist nicht garantiert, dass die optimale Lösung gefunden wird. Erfolg und Laufzeit meta-heuristischer Verfahren hängen entscheidend von der Implementierung der einzelnen Schritte ab.
Auf dem Gebiet der Frequenzzuweisungsprobleme (FAP's) handelt auf Grund der Komplexität der zu lösenden Probleme ein Großteil der veröffentlichten wissenschaftlichen Schriften von (Meta-)Heuristiken.

Folgende Heuristiken zur Lösung von FAP's sollen in diesem Vortrag vorgestellt werden:

- Greedy-Algorithmen (gefräßige Algorithmen)
- Lokale Suche (Umgebungs-basiert)
- Tabu Suche (Erweiterung von LS)

## 2 Greedy-Algorithmus

Der Greedy(gefräßige, gierige)-Algorithmus arbeitet konstruktiv. Er wählt iterativ Knoten (also in unserem Falle Antennen) aus, denen er jeweils eine zulässige Frequenz zuordnet. Knoten-Wahl und Frequenz-Zuweisung folgen bestimmten vorgegebenen Regeln, die von lokalen Charakteristika abhängen und die Optimierung der globalen Zielfunktion zum Ziel haben. Dabei ist die Wahl in jedem Iterationsschritt unwiderruflich.
Der Greedy-Algorithmus hat anders als dynamische Programme kein Gedächtnis (also eine Tabelle, in der die Güte untersuchter Teillösungen gespeichert wird; nötig ist dafür eine Bewertungsfunktion, die die Güte einer Lösung festlegt), sondern trifft Entscheidungen ausschließlich auf Grund der lokal verfügbaren Information. Dabei wählt er immer das „beste verfügbare Stück“, was die Bezeichnung als „gefräßiger“ Algorithmus motiviert.

Ein Greedy-Algorithmus zur Lösung des MS-FAP nach Zoellner/Ball (1977) geht folgendermaßen vor:
Zunächst werden die Knoten im zum gewählten FAP gehörenden Interferenz-Graphen geordnet. Dabei unterscheidet man folgende Anordnungs-Methoden:

- *highest degree first:* Anordnung in nicht-ansteigendem Interferenz-Grad
- *smallest degree last:* Anordnung so, dass der Grad des letzten Knotens der kleinste ist
- *random order:* zufällige Anordnung

Dann wird in der nun bestehenden Reihenfolge jedem Knoten eine Frequenz zugewiesen. Hier unterscheidet man zwei Zuweisungs-Methoden:

- *frequency exhaustive:* kanonische Zuweisung
- *uniform:* dem aktuellen Knoten wird die bislang am wenigsten genutzte zulässige Frequenz zugewiesen

### 2.1 Beispiel: Generalisierter Sättigungs-Grad

Bei dieser Methodik wird die bekannte DSATUR-Prozedur für Graphen-Färbung (vg. Brélaz 1979) auf FAP's übertragen. Der Sättigungsgrad eines Knotens $v$ entspricht dabei einfach der Anzahl der geblockten Frequenzen, also derer, die dem Knoten $v$ auf Grund harter Einschränkungen nicht zugewiesen werden können. In jedem Iterationsschritt wählt der Greedy-Algorithmus nun den Knoten mit höchstem Sättigungsgrad und weist ihm die kleinste zulässige, nicht-geblockte Frequenz zu. Der Generalisierte Sättigungs-Grad (GSD) (Valenzuela et al. 1998) ist eine Erweiterung dieser Heuristik: Ist die Frequenz $f$ für den Knoten $v$ geblockt, so wird das Gewicht von $f$ definiert als die höchste Strafe $p_{vw}(f, g)$ für alle $w \in V$, wenn Frequenz $g$ Knoten $w$ zugewiesen wurde. Der Generalisierte Sättigungs-Grad von $v$ ist nun die Summe der Gewichte aller geblockten Frequenzen.

# 3 Lokale Suche

Lokale Suchmethoden starten mit einer gegebenen Lösung und verändern diese iterativ durch kleine sog. „moves“, die zur Verbesserung der Lösung führen sollen. Standard-Methoden erlauben dabei nur solche „moves“, die die Lösung sofort verbessern („downhill“-Methoden). Manchmal findet man jedoch bessere Lösungen, indem man moves erlaubt, die die Lösung zunächst schlechter erscheinen lassen, um von dieser aus zu einer viel besseren Lösung zu gelangen („uphill“-Methode). Um solche Lösungen zu finden, muss jedoch eine größere Umgebung der Startlösung untersucht werden, was zu höherer Rechenzeit führt.
Das klassische Standard-Verfahren der lokalen Suche funktioniert folgendermaßen:

1. Starte mit gegebener Lösung
2. Durchsuche den Suchraum nach einer besseren Lösung („improving solution“)
3. Falls keine besser Lösung vorhanden, breche ab
4. Sonst setze bessere Lösung als Startlösung und beginne bei 1.

Der eingeschränkte Suchraum ist dabei definiert als die Menge der Lösungen in der Umgebung („neighborhood“) der aktuellen Lösung, die durch erlaubte moves erreicht werden können. Wir betrachten hier lediglich lokale Suchmethoden, die ausschließlich verbessernde moves zulassen (vgl. Hill-Climbing Algorithmus).
Entscheidend bei solchen Umgebungs-basierten Methoden ist die Fähigkeit, effizient über die Umgebung der aktuellen Lösung zu optimieren. Einerseits versucht man einen möglichst großen Raum zu durchsuchen, um die Wahrscheinlichkeit für eine verbessernde Lösung zu erhöhen, andererseits gehen große Suchräume mit exponentiell wachsender Such-Dauer einher.
Eine Frequenz-Zuweisung ist eine Partition eines Knoten-Sets, wobei jede Klasse zu einer Frequenz im möglichen Spektrum korrespondiert. Die sog. *1-exchange-neighborhood* enthält alle Lösungen, die aus der aktuellen dadurch hervorgehen, dass ein Knoten ausgewählt wird und in eine andere Klasse bewegt wird. Eine solche Operation heißt *1-exchange-move*. Bezeichnet $n$ die Anzahl der Knoten und $f_{max}$ die Anzahl der Frequenzen, so enthält die *1-exchange-neighborhood* genau $n(f_{max} - 1)$ Lösungen. Entsprechend enthält die *2-exchange-neighborhood* alle Lösungen, die möglich sind, wenn man die Frequenzen von 2 Knoten ändert.
„Echte“ lokale Suchmethoden findet man auf Grund der lokale-Maxima-Problematik nur selten in der Praxis. Jedoch werden sie teilweise als Konstruktions-Block verwendet wie im folgenden Beispiel beschrieben.

## 3.1 Beispiel: MI-FAP nach Castelino (1996)

- Erzeuge Startlösung zufällig
- Durchlaufe Sequenzen von moves, jeweils $|V|$ Iterationsschritte.
  Durchsuche in jeder Iteration $i$ die *1-exchange-neighborhood* des Knoten $v_i$ nach einer besseren Lösung
- Breche ab bei 0-Kosten-Lösung oder nach vorgegebener Anzahl von Schritten

# 4 Tabu Suche

Tabu Suche (zurückgehend auf Glover, 1986) ist eine lokale Suchmethode, die im Gegensatz zur Standard-LS nicht-verbessernde moves zulässt. In jedem Iterationsschritt wird die beste benachbarte Lösung ausgewählt, die auch schlechter als die aktuelle Lösung sein kann. So wird verhindert, dass die Suche in einem lokalen Optimum anhält. Um Schleifen zu vermeiden, werden die Lösungen der letzten $k$ Iterationsschritte in einer Tabu-Liste gespeichert und können nicht wieder ausgewählt werden. Im Gegensatz zur LS verfügt dieser Algorithmus also über ein Gedächtnis. Problematisch kann hierbei sein, dass für die Tabu-Listen viel Speicherplatz verwendet werden muss und dass die Überprüfung der Liste je nach Länge viel Rechenzeit erfordern kann. Daher hat es sich als effizienter erwiesen, nicht den Status einer Lösung (tabu oder nicht tabu) zu untersuchen, sondern Lösungs-Attribute zu speichern, sodass jede Lösung, die eines dieser Attribute enthält, „Tabu“ und damit ausgeschlossen wird.
Für FAP's wird dieses Verfahren realisiert, indem die letzten $k$ moves, also Bewegungen von Knoten zwischen Klassen, gespeichert werden und verhindert wird, dass einer dieser moves invertiert wird. Der Parameter $k$ bezeichnet die Länge der Tabu-Liste. Typischerweise bricht die Suche nach einer vorgegebenen Anzahl nicht-verbessernder moves ab, andere Abbruch-Bedingungen können z.B. eine leere Umgebung (das heißt keine mögliche neue Lösung) oder das Auffinden einer ausreichend guten Lösung sein. Wie alle betrachteten Heuristiken gibt es zahlreiche unterschiedliche Anwendungen des Tabu-Suche-Prinzips. Im Folgenden sollen die Grund-Zutaten für eine Tabu-Suche beschrieben werden:

1. Generierung der Start-Lösung
2. Fitness-Funktion. D.h. die Funktion, die es zu minimieren (bzw. maximieren) gilt. Generell ist dies die ursprüngliche Zielfunktion, die jedoch teilweise angepasst wird
3. Definition der Umgebung. Meist wird die *1-exchange-neighborhood* gewählt
4. Einschränkung der Umgebung. Zusätzliche Mechanismen zur Verkleinerung des Raumes, der in jedem Schritt durchsucht wird

## 4.1 Beispiel: TS für Graphen-Färbeproblem

- *Startlösung:* zufällig
- *Evaluierungskriterium:* Minimiere Anzahl der „verletzten“ Kanten
- *Umgebung:* Färbung, die sich genau in der Farbe eines Knotens unterscheidet
- *Tabu-Attribute:* Paare $(V_i; j)$, d.h. bestimmte Farbzuweisungen $(j)$ zu bestimmten Knoten $(V_i)$
- *Tabu-Kriterium:* Wird Move $(V_i; j) \rightarrow (V_i; j')$ durchgeführt, wird Attribut $(V_i; j)$ verboten für $k$ Iterationen
- *Einschränkung der Umgebung:* Betrachte Knoten an „verletzten“ Kanten