Technische Universität München
Institut für Informatik
Prof. Dr. Angelika Steger

SS 1999
Merkblatt 4
18. Mai 1999

Praktikum Diskrete Optimierung
Branch-and-Bound

1 Idee

Die grundsätzliche Idee von Branch-and-Bound besteht darin, daß die Lösungen des Optimierungsproblems schrittweise rekursiv aufgebaut werden. Dabei wird eine aktuelle Teillösung bei jedem Schritt erweitert, bis eine vollständige Lösung erreicht ist. Wenn die Lösung eines Optimierungsproblems beispielsweise durch eine Belegung der Variablen $a_1 \dots a_n$ mit 0 oder 1 beschrieben wird, wird bei jedem Schritt (angefangen bei a_1) eine weitere Variable belegt.
Ein Schritt besteht aus zwei Teilen:

1. **Bound:** Anfangs wird überprüft, ob die aktuelle Teillösung bereits teurer ist, als die bisher beste gefundene vollständige Lösung, falls ja kann abgebrochen werden. Dazu muß es natürlich möglich sein, daß bereits die Kosten einer noch unvollständigen Lösung ermittelt werden können.

2. **Branch:** Ist man bei einer vollständigen Lösung angelangt, wird diese Lösung mit der bisher besten verglichen und evtl. übernommen. Sonst wird über alle möglichen Erweiterungen der aktuellen Teillösung iteriert und jeweils zum nächsten Schritt übergegangen.

Man hat also beispielsweise eine Belegung der Variablen $a_1 \dots a_i$ gegeben. Zuerst wird geprüft, ob die Kosten $cost(a_1, \dots, a_i)$ höher sind als die Kosten der bisher besten vollständigen Belegung. Falls ja, kann abgebrochen werden. Ansonsten wird rekursiv jeweils für $a_{i+1} = 0$ und $a_{i+1} = 1$ entsprechend fortgefahren.
Bei Branch-and-Bound wird ein implizit gegebener Suchbaum mit der sogenannten Tiefensuche durchlaufen. Die Wurzel des Baumes entspricht einer „Teillösung“, bei der noch nichts festgelegt ist. Ein innerer Knoten ist eine Teillösung, der als Kinder alle möglichen Erweiterungen dieser Teillösung um einen Schritt hat. Die Blätter des Suchbaums sind alle gültigen Lösungen. Die Tiefensuche wird bereits auf einem höherem Level des Baumes abgebrochen, wenn man frühzeitig erkennt, daß ein Ast des Baumes hoffnungslos ist (weil ein Knoten = Teillösung bereits teurer ist, als das bisher beste gefundene Blatt = Lösung). Beim obigen Beispiel wäre der Suchbaum also ein binärer Baum der Tiefe n. Da der gesamte Suchbaum durchlaufen wird (bzw. alle relevanten Teile, bei korrekter Heuristik; siehe unten) findet Branch-and-Bound immer die optimale Lösung.

```
proc branchAndBound(PartialSol p)

    if p ist vollständige Lösung then
        if Wert von p < bisheriges Minimum then
            Speichere p als neues Minimum
        fi
    else
        if Wert von p + Heuristik ≥ bisheriges Minimum then
            // Das kann kein neues Minimum mehr werden → Abbruch
            return
        fi
        // Eine vollständige Lösung bestehe aus den Variablen α_1, ..., α_n.
        // In p seien davon α_1, ..., α_{i-1} belegt
        for all Belegungen b von α_i do
            Erzeuge p' durch Belegen von α_i := b in p
            rekursiver Aufruf von branchAndBound(p')
        od
    fi
```

Abbildung 1: Branch-and-Bound-Algorithmus

Noch ein Hinweis zu Implementierung: Da die Datenstrukturen für partielle Lösungen meist recht aufwendig sind, vermeidet man wenn möglich, diese bei den rekursiven Aufrufen des Algorithmus zu kopieren. Man versucht stattdessen, die aktuelle Teillösung zu erweitern, darauf den rekursiven anzuwenden und nach dessen Rückkehr die Änderungen an der Datenstruktur rückgängig zu machen.

2 Zusätzliche Heuristik

Damit im Suchbaum einzelne Äste noch früher abgeschnitten werden, kann man sich zu einer gegebenen Teillösungen noch überlegen wie teuer diese bei Vervollständigung bestensfalls wird. Diese *unterschätzten* Kosten werden dann mit den Kosten der bisher besten Lösung verglichen.
Der Gedanke dahinter ist, daß oft bereits an „billigen“ Teillösungen zu erkennen ist, ob sie sich überhaupt noch zu einer guten Lösung vervollständigen lassen. Bei der Entwicklung einer guten Heuristik ist Ideenreichtum gefragt. Die Berechnung der Heuristik darf natürlich auch nicht zu lange dauern, da es sonst evtl. doch wieder schneller ist alle möglichen Vervollständigungen zu betrachten. Wichtig ist vor allem, daß die Kosten unterschätzt werden, da sonst unter Umständen gerade der Ast abgeschnitten wird, in dem die optimale Lösung liegt.