Intervall-Partitionierung

Intervall-Partitionierung ist eine weitere Methode, um die Mengen S^i klein zu halten.

Sei P_i das Maximum von allen $\sum_{j=1}^{i} p_j x_j$, unter allen zulässigen Einschränkungen in S^i.

- Wir unterteilen das Intervall $[0, P_i]$ in Unterintervalle der Größe $\epsilon P_i/(n-1)$ (wobei das letzte kleiner sein kann).
- Von allen zulässigen Einschränkungen, deren $\sum_{j=1}^{i} p_j x_j$ in das gleiche Unterintervall fallen, eliminieren wir alle bis auf eine mit Dominanzregeln. Dabei tun wir so, als wären ihre Zielfunktionen gleich.
- Das S^{i+1} wird dann aus dem reduzierten S^i berechnet.

Intervall-Partitionierung

- Der Fehler, der beim Identifizieren aller zulässigen Einschränkungen in einem Unterintervall gemacht werden kann, ist höchstens $\epsilon P_i/(n-1)$.
- $P_i \leq F^*(I)$, denn P_i ist kleiner oder gleich der Zielfunktion einer zulässigen Lösung.
- Fehler können sich fortpflanzen. Bei jedem Übergang $S^i \rightarrow S^{i+1}$ kann ein neuer Fehler hinzukommen.
- Der Gesamtfehler ist höchstens

$$F^*(I) - F(I) = \sum_{i=1}^{n-1} \frac{\epsilon P_i}{n-1} \leq \epsilon F^*(I).$$

Intervall-Partitionierung

Die Anzahl der Unterintervalle ist höchstens $\lceil n/\epsilon \rceil + 1$.

Also ist $|S_i| = O(n/\epsilon)$ und

$$\sum_{i=1}^{n} |S_i| = O(n^2/\epsilon).$$

Unter guten Voraussetzungen ergibt dies einen $O(n^2/\epsilon)$-Zeit-Algorithmus.

Ist eine bessere untere Schranke für $F^*(I)$ bekannt, so kann diese jederzeit statt P_i verwendet werden.

Beispiel

Wir haben wieder einen Rucksack der Kapazität 1112 und Gegenstände deren Größe und Wert $1, 2, 10, 100, 1000$ ist. Wir wählen $\epsilon = 0.1$.

$P_1 = 1$ und die Unterintervalle haben Länge $0.1/4 = 0.025$. Das ergibt $S^1 = \{(0), (1)\}$.

Daraus wird zunächst $S^2 = \{(0,0), (0,1), (1,0), (1,1)\}$ berechnet.

Die zugehörigen Werte der Zielfunktion sind $0, 2, 1, 3$. Das ergibt $P_2 = 3$ und Unterintervalle der Länge $0.1 \cdot 3/4 = 0.075$.

Die Intervalle sind $[0, 0.075), [0.075, 0.15), [0.225, 0.3), \ldots$. Es findet keine Reduzierung statt, da nur ein Wert in jedes Unterintervall fällt.

Als nächstes wird $S^3 = \{(0,0,0),(0,0,1),\ldots,(1,1,1)\}$ bestimmt. Die Werte sind jetzt $0, 10, 2, 12, 1, 11, 3, 13$.

Die Unterintervalle haben Länge $0.1 \cdot 13/4 = 0.325$. Wieder findet keine Reduktion statt.

Beim nächsten Schritt haben wir eine Länge von $0.1 \cdot 113/4 = 2.825$. Jetzt fallen beispielsweise $0, 1, 2$ in das gleiche Intervall.

Hätten wir die Gegenstände in umgekehrter Reihenfolge betrachtet, wäre es einfacher gewesen.

Wir können aber auch eine eigene untere Schranke L statt der P_i verwenden.

$L = \max\{p_i\} = 1000$. Die erste Intervalllänge ist daher bereits $0.1 \cdot 1000/4 = 25$.

Jetzt sind die Werte der Zielfunktion für S^1, S^2 und S^3 nur 0, für S^4 sind es 0 und 100 und für S^5 schließlich 0, 100, 1000 und 1100.

Als beste Lösung wird $(0, 0, 0, 1, 1)$ bestimmt.

Separation

Die Intervall-Partitionierung hat $[0, P_i]$ stur in Unterintervalle geteilt. Nur in gleich Unterintervalle fallende zulässige Einschränkungen werden zusammengefasst.

Separation geht ähnlich vor, teilt aber nicht in Unterintervalle ein.

Stattdessen werden zulässige Einschränkungen als gleichwertig angesehen, wenn sich ihre Werte um höchstens $\epsilon P_i/(n-1)$ unterscheiden.

Auch bei diesem Vorgehen kann der hinzugefügte Fehler nur $\epsilon P_i/(n-1)$ sein.

Separation

Sind die Werte $3.9, 4.1, 7.9, 8.1, 11.9, 12.1$ dann führt eine Intervalllänge von 2 zu keiner Reduktion.

Separation halbiert allerdings die Anzahl der Werte.

Intervall-Partitionierung kann aber ebenfalls besser sein:

Wir betrachten hierzu eine Knapsack-Instanz mit $(p_1, \ldots, p_5) = (w_1, \ldots, w_5) = (3, 1, 5.1, 5.1, 5.1)$. Es sei eine untere Schranke bekannt, so daß

$$\epsilon L/(n-1) = 2.$$

Was ergibt sich nun bei Intervall-Partitionierung und was bei Separation?

Nichtapproximierbare Probleme

Wir betrachten das folgende Optimierungsproblem:

$$\text{Maximiere } \sum_{i=1}^{n} a_i x_i$$

$$\text{unter } \sum_{i=1}^{n} a_i x_i \leq m$$

$$x_1, \ldots, x_n \in \{0, 1\}$$

$$\text{mit } m = \frac{1}{2} \sum_{i=1}^{n} a_i$$

Die optimale Lösung ist m, falls sich $\{a_1, \ldots, a_n\}$ in zwei Mengen mit gleicher Summe partitionieren läßt.

Dies ist ein *NP*-hartes Problem.

Nichtapproximierbare Probleme

Betrachte nun folgende Variante:

$$\text{Minimiere } 1 + k\left(m - \sum_{i=1}^{n} a_i x_i\right) \text{ unter } \sum_{i=1}^{n} a_i x_i \leq m$$

$$x_1, \ldots, x_n \in \{0, 1\}$$

Die optimale Lösung ist jetzt 1, falls sich $\{a_1, \ldots, a_n\}$ in zwei Mengen mit gleicher Summe partitionieren läßt.

Andernfalls ist sie mindestens $1 + k$.

Angenommen es gäbe einen ϵ-Approximationsalgorithmus.

Wähle $k > \epsilon$. Dann löst der Approximationsalgorithmus das Problem exakt, falls die optimale Lösung m ist.

Dieses Problem ist für kein $\epsilon > 0$ approximierbar, solange $P \neq NP$.