

### Schnittebenenverfahren

Da es oft unmöglich ist eine komplette beschreibung eines ILPs durch Ungleichungen zu erhalten, nimmt man nur einen kleinen Teil der Ungleichungen. Danach sucht man nach einer Lösung und kontrolliert ob die Lösung alle Restriktionen erfüllt, wenn ja ist die Lösung optimal, wenn nein wird die Restriktion hinzugefügt und erneut gelöst...

### Spaltengenerierungsverfahren

Basis ist der Simplexalgorithmus. Es werden Variablen hinzugefügt, die die Kosten maximal reduzieren. Dies wird solange wiederholt, bis keine Variablen mehr mit positiven reduzierenden Kosten existieren und somit mittels Simplex eine optimale Lösung gefunden wurde.

Beschreiben Sie in einfachen Worten, was das *Separationsproblem* in einem *Schnittebenenverfahren* ist.

Finden einer weiteren Restriktion die für alle Punkte im Polytop gültig ist, jedoch für den gegebenen Punkt  $x_*$  verletzt ist.

Beschreiben Sie in einfachen Worten, was das *Pricing-Problem* in einem *Spaltengenerierungsverfahren* ist.

Lösen des Subproblems das jene Variable mit maximalen(minimalen) reduzierenden Kosten findet.

Beschreiben Sie in einfachen Worten, was *Branch-and-Price* von einfachem LPbasierten Branch-and-Bound unterscheidet.

Beim Branch-and-Price werden die Teilprobleme mittels Spaltengenerierung gelöst. Sollte dies keine Lösung bringen, wird das Problem in weitere Teilprobleme aufgespalten(branching)

Beschreiben Sie in einfachen Worten, was *Branch-and-Cut* von einfachem LPbasierten Branch-and-Bound unterscheidet.

Dieses verfahren verbindet das branch-and-bound verfahren mit dem schnittebenenverfahren. beim branch-and-cut verfahren wird das schnittebenenverfahren zum lösen der teilprobleme verwendet. Sollte dies keine lösung bringen, wird weiter aufgeteilt(branching)

Was ist der Unterschied zwischen den *Las-Vegas-Verfahren* und den *Monte-Carlo-Verfahren*?

Las-Vegas-Verfahren liefern stets korrekte Ergebnisse während Monte-Carlo-Verfahren eine Lösung liefern, die mit einer gewissen Fehlerwahrscheinlichkeit behaftet ist.

Was ist der Unterschied zwischen einer *perfekten Skipliste* und einer *randomisierten Skipliste*? Welche Vorteile besitzt die bessere der beiden Datenstrukturen?

Bei einer perfekten skipliste ist die höhe der container fix.  
Bei der randomisierten werden die höhen durch zufall ermittelt.  
Der vorteil der randomisierten skipliste ist, dass beim einfügen und löschen nie reorganisiert werden muss sondern nur zeiger verbogen werden müssen(weil es keine vorgabe für die höhe der container an bestimmten plätzen gibt)

### Allgemeine Lage

1. Kein Segment ist senkrecht
2. Der Schnitt zweier Segmente ist stets leer oder genau ein Punkt
3. Es schneiden sich nie mehr als zwei Segmente in einem Punkt
4. Alle Endpunkte und Schnittpunkte haben paarweise verschiedene x-Koordinaten.

Skizzieren Sie das hierarchische Speichermodell typischer moderner Computer. Beschriften und beschreiben Sie es in kurzen Worten.

CPU – Cache – RAM – Disk(secondary memory)  
Ram Zugriff: um Faktor 100 langsamer  
Disk Zugriff: um Faktor 100 000 langsamer

Die Funktionsfähigkeit eines Cache-Speichers hängt wesentlich von der Lokalität der eingesetzten Programme ab. Welche beiden Formen der Lokalität gibt es, und wie können diese jeweils vom Cache ausgenutzt werden?

-) **Zeitliche Lokalität:** gleiche Speicherbereiche werden innerhalb einer kurzen Zeitspanne angesprochen. Caches nutzen dies aus indem sie versuchen, die angesprochenen Speicherbereiche möglichst lange im Cache zu lassen.

-) **Örtliche Lokalität:** es werden öfter Speicherbereiche angesprochen, die nahe beieinander liegen. Caches nutzen dies aus, indem sie bei jedem Miss jeweils ganze Speicherblöcke in den Cache holen.

Erläutern Sie in kurzen Worten den Unterschied zwischen Algorithmen, die sich *cache-aware* bzw. *cache-oblivious* verhalten. Welches Verhalten ist im allgemeinen Fall vorzuziehen, und warum?

**Cache-aware:** der Algorithmus enthält parameter mit denen die Cache-Komplexität für die gegebene Cachegröße  $Z$  und Zeilenlänge  $L$  optimiert werden kann. Sonst heißt er **cache-oblivious**.  
Im Allgemeinen Fall ist cache-oblivious vorzuziehen, da dieser Algorithmus für unbekanntes  $Z$  und  $L$  optimiert wurde und dadurch automatisch Zugriffe in jeder Ebene optimiert.

Erklären Sie in wenigen Worten die Verbesserung nach Edmonds-Karp.

es werden augmentierende pfade gesucht, die möglichst wenig kanten haben.  
Laufzeit:  $= (n \cdot m^2)$

Scan-Line Algorithmus:

Was ist eine Ereignisdatenstruktur bzw. was wird darin gespeichert? Womit wird sie initialisiert?

Es werden alle Ereignisse(Startpunkt, Endpunkt, Schnittpunkt) aufsteigend nach deren X-Koordinate gespeichert.  
Initialisiert wird sie mit den Start- und Endpunkten, die Schnittpunkte werden während der Laufzeit hinzugefügt.

Randomisierter Primzahltest von Miller-Rabin:

Allgemein

Monte-Carlo Verfahren, also mit Fehlerwahrscheinlichkeit behaftet  
Mehrmaliges Suchen von zeugen der nicht-primheit mittels satz von fermat  
→ wenn kein zeuge gefunden wird, annahme es ist primzahl

• Welche Aussage hat das Ergebnis, wenn  $a$  ein Zeuge für die Nicht-Primheit von  $n$  ist?

$N$  ist sicher keine Primzahl

• Welche Aussage hat das Ergebnis, wenn  $a$  kein Zeuge für die Nicht-Primheit von  $n$  ist?

$N$  könnte eine Primzahl sein

Definition Fluss

1) $f(u,v) = -f(v,u), u,v \in V$	Schiefsymmetrie
2) $f(u,v) \leq c(u,v), u,v \in V$	Kapazitätsbeschränkung
3) $\sum_{v \in V} f(u,v) = 0, u \in V \setminus \{s,t\}$	Flusserhaltung

Max-Flow-Min-Cut-Theorem

- 1) Es gibt einen Schnitt  $S$  in  $N$ , der von  $f$  saturiert wird
- 2)  $F$  ist max. Fluss in  $N$
- 3) Es existiert kein augmentierender Pfad im Restgraph  $G_f$

Preflow

1) $f(u,v) = -f(v,u), u,v \in V$	Schiefsymmetrie
2) $f(u,v) \leq c(u,v), u,v \in V$	Kapazitätsbeschränkung
3) $e_f(v) = \sum_{u \in V} f(u,v) \geq 0, v \in V \setminus \{s\}$	Überschussbedingung