

---

**Aufgabe 1**

---

QuickPick erzeugt zufällig Join-Bäume ohne Kreuzprodukte. Welche Art von Bäumen werden erzeugt und wie beurteilen Sie dieses Verfahren?

**Lösung**

- + erzeugt randomisierte bushy Join-Bäume
- + alle Bäume können erzeugt werden
- keine Gleichverteilung wenn der Joingraph keine Clique ist

---

**Aufgabe 2**

---

In Abbildung 1 ist ein Kostengebirge abgebildet. Beschreiben Sie für jeden der folgenden Algorithmen, welche Punkte im Kostengebirge der Algorithmus erreichen kann, wenn der initiale Plan die Kosten in Punkt A hat.

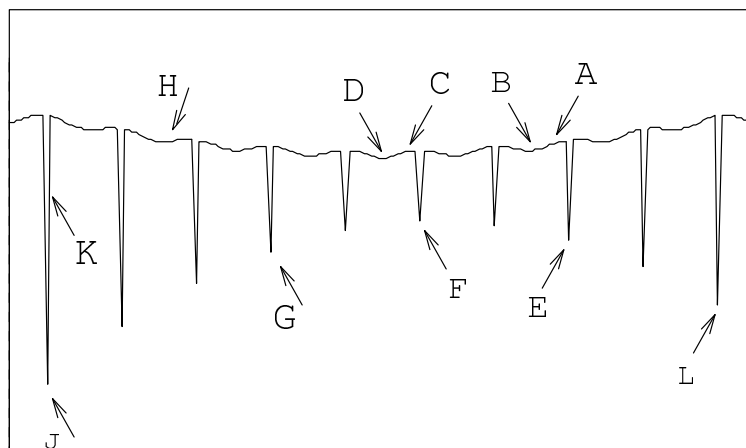


Abbildung 1: Eine Kostenfunktion (aus Vance, 1998)

## Aufgabe 2 a)

1. Iterative Improvement
2. Simulated Annealing
3. TabuSearch

## Lösung

1. Iterative Improvement erreicht nur ein lokales Minimum, d.h. es wird Punkt B erreichen.
2. Simulated Annealing kann den Punkt E oder den nächsten “Graben” nach links erreichen. Vermutlich wird es aber diesem tiefen Minimum nicht entkommen können.
3. TabuSearch kann prinzipiell jeden Punkt erreichen. Das hängt von der Größe des TabuSets ab: Ist es zu klein, wird u.U. nur das lokale Minimum B erreicht. Mit einem größeren TabuSet können auch tiefe lokale Minima überwunden werden. Dann kann es aber lange dauern, bevor der Algorithmus terminiert bzw. die Abbruchbedingung erreicht ist.

## Aufgabe 2 b)

Welche Auswirkung haben verschiedene zufällige Startpunkte? Schätzen Sie die Chance ein, daß das globale Minimum (J) gefunden wird? Wie sieht es mit dynamischen Programmieren aus?

## Lösung

- Viele Startpunkte werden bessere Startpunkte erzeugen, z.B. C. Dadurch wird insgesamt ein besserer Plan gefunden. Die prinzipiellen Einschränkungen der Algorithmen werden dadurch nicht aufgehoben.
- Die Wahrscheinlichkeit, daß ein zufälliger Plan in einem “Kostengraben” liegt (z.B. bei K) sind bei dieser Kostenverteilung gering.
- Dynamisches Programmieren wird das globale Minimum mit Sicherheit erreichen.
- Wenn es nur wenige Kostentäler gibt, ist die Chance größer, daß das globale Minimum gefunden wird.

---

### Aufgabe 3

---

Aktualisieren Sie die Klassifikation der Joinalgorithmen, d.h. tragen Sie die folgenden Algorithmen ein:

- Transformationsbasierter Ansatz
- Erzeugen von Permutationen
- Quick Pick
- Iterative Improvement
- Simulated Annealing
- TabuSearch

### Lösung

Name	Query Graph	Join-Tree	Kreuzprodukte	Kostenfunktion	Komplexität	optimal	Anmerkungen
Permutations	beliebig	left-deep	ja	beliebig	$O(2^n)$	ja	
Memoization	beliebig	u.a. bushy	optional	beliebig	$O(2^n) \dots O(3^n)$	ja	Komplexität abhängig von query graph
Transformation	abhängig von RuleSet			beliebig	$O(2^n) \dots O(3^n)$	ja	Komplexität abhängig von RuleSet
QuickPick	connected	bushy	nein	beliebig	$O(n)$	nein	Komplexität je Iteration
II	abhängig von RuleSet			beliebig	$O(n)$	nein	Komplexität je Iteration
SA	abhängig von RuleSet			beliebig	$O(n)$	nein	Komplexität je Iteration
TabuSearch	abhängig von RuleSet			beliebig	$O(n)$	nein	Komplexität je Iteration

---

### Aufgabe 4

---

Schreiben Sie ein Programm, das zuerst einen zufälligen links-tiefen Plan erzeugt und diesen dann mittels *Iterative Improvement* verbessert.

### Lösung

siehe IterativeImprovement.java