Stefan Theussl
Gerold Köb
Goran Lovric
Martin Gartner

GRASP = Greedy randomized adaptive search process

Inhalt

- Einleitung
- Algorithmus
- Simulation, Parameterwahl
- Recheneffizienz
- Unser Benchmark
- Schlussfolgerungen und Ausblick
- Live-Demonstration in R

Einleitung I

- Grundprinzip:
  - Erzeugung einer „greedy“ Startlösung mit der dann Local Search gestartet wird
- Construction Phase:
  - Kandidatenliste der Städte, die von Startpunkt erreichbar sind
  - Auswahl eines Kandidaten nach Bewertung mit „greedy-function“ bis eine komplette Lösung vorliegt
- …

Einleitung II

- …
- Local Search:
  - Permutation der „greedy“-Startlösung und Berechnung des Zielwerts
  - wenn Zielwert besser als jener der „greedy“-Startlösung à Zielwert als neuer „benchmark“
- Wiederholung des Prozesses bis Abbruchbedingung erfüllt wird (zB Anzahl der Iterationen)

Ablauf des Algorithmus

- „grasp.solve“ wird aufgerufen
- optimaler Zielwert = unendlich
- In „grasp.solve“ wird mit „.construct_greedy“ eine Startlösung erzeugt
- Startlösung in Funktion „.local_search“
- wenn Zielwertverbesserung durch „.local_search“ à neuer Zielwert ist neues Optimum
- Wiederholung bis Abbruchbedingung erfüllt (bei uns: Anzahl der Wiederholungen von „grasp.solve“)

Algorithmus I

- Funktion: „grasp.solve“:
  - Inputs: Distanzmatrix, Startpunkt
  - Parameter: alpha, k, iterations, n
  - n wird „local_search“ übergeben
  - k entspricht Anzahl der vertauschten Städte beim k-opt-move

Algorithmus II

- Funktion: „.construct_greedy“:
  - min <- minimale Distanz
  - max <- maximale Distanz
  - alpha <- gleichverteilte ZZ zwischen 0 und 1
    Verteilung für alpha kann bei Aufruf von „.grasp.solve“ übergeben werden
  - treshold <- min + alpha * (max – min)
    à Gewichtung zwischen der am kürzesten und der am weitesten entfernten Stadt
  - zufällige Auswahl eines Kandidaten, wo Distanz kleiner „treshold“

Algorithmus III

- Funktion: „local_search“:
  - Inputs: Distanzmatrix, greedy Startlösung
  - Parameter: n, k
  - k gibt Anzahl für k-opt-move an
  - n ist Abbruchbedingung falls keine Zielwertverbesserung gefunden wird (n ist Anzahl der maximalen Durchläufe der Local Search)

Simulation, Parameterwahl I

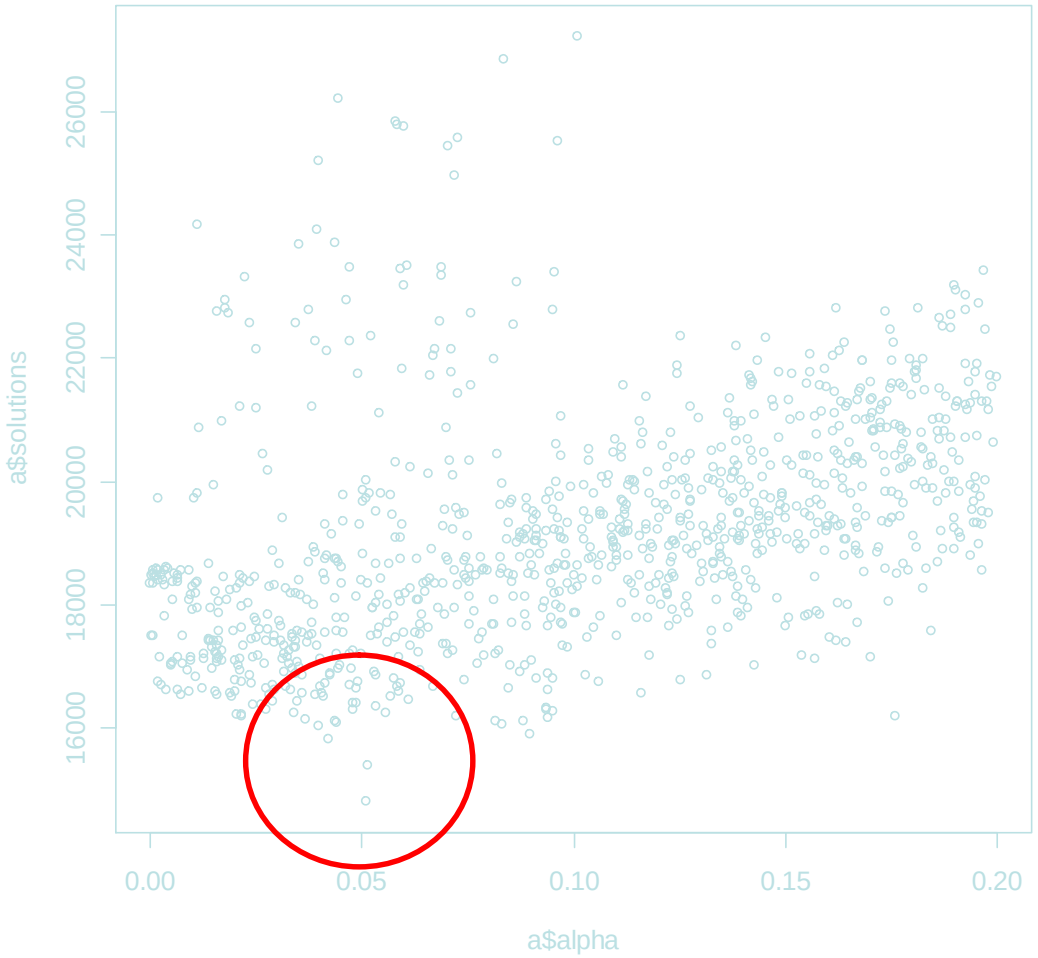
- Zufallsvariable: alpha
- Verwendung bei Erzeugung einer „greedy“ Startlösung
- treshold <- min + alpha * (max – min)

- Simulation in R: mit festem seed (dh: gleichen Pseudozufallszahlen werden verwendet!)
- folgende Beispiele mit 3-opt-moves!

alpha ~ U(0,1)

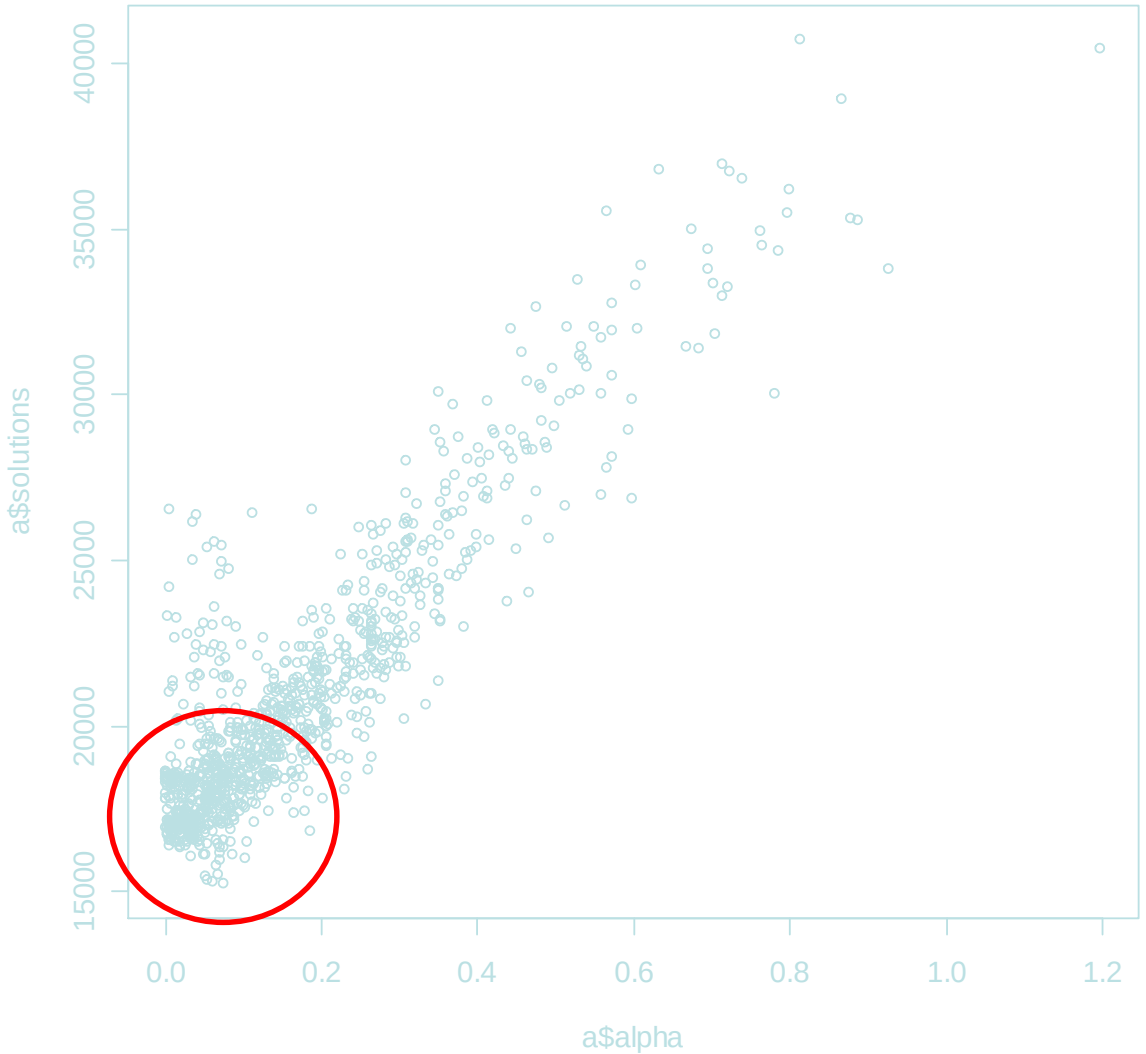
- kleineres alpha à bessere Zielwerte
- Zielwerte zwischen ca. 14.000 km und 45.000 km

alpha ~ U(0, 0.2)



- Zielwerte zwischen ca. 14.000 km und 27.000 km
- unser Optimum bei alpha von „0.054966“

alpha ~ Γ(1, 6)



- Zielwerte zwischen ca. 40.000 km und 15.000 km aber Häufung bei Zielwerten zwischen 15.000 km und 20.000 km

Simulation, Parameterwahl II

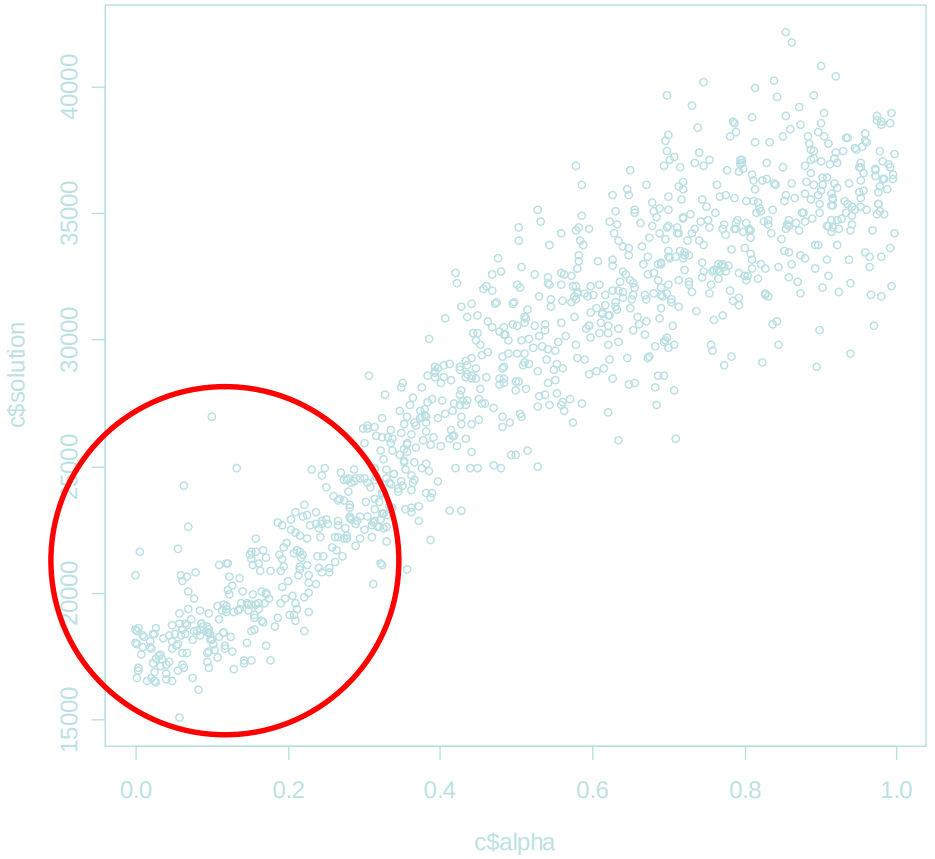
- Parameter k:
- k-opt-move: Positionen von k Städten wird zufällig vertauscht
- zB A-B-C-D-E
- 2-opt-move zB: A-D-C-B-E
- 3-opt-move zB: C-B-E-D-A

- Simulation in R: mit festem seed (dh: gleichen Pseudozufallszahlen werden verwendet!)

Vergleich bei Verwendung verschiedener k

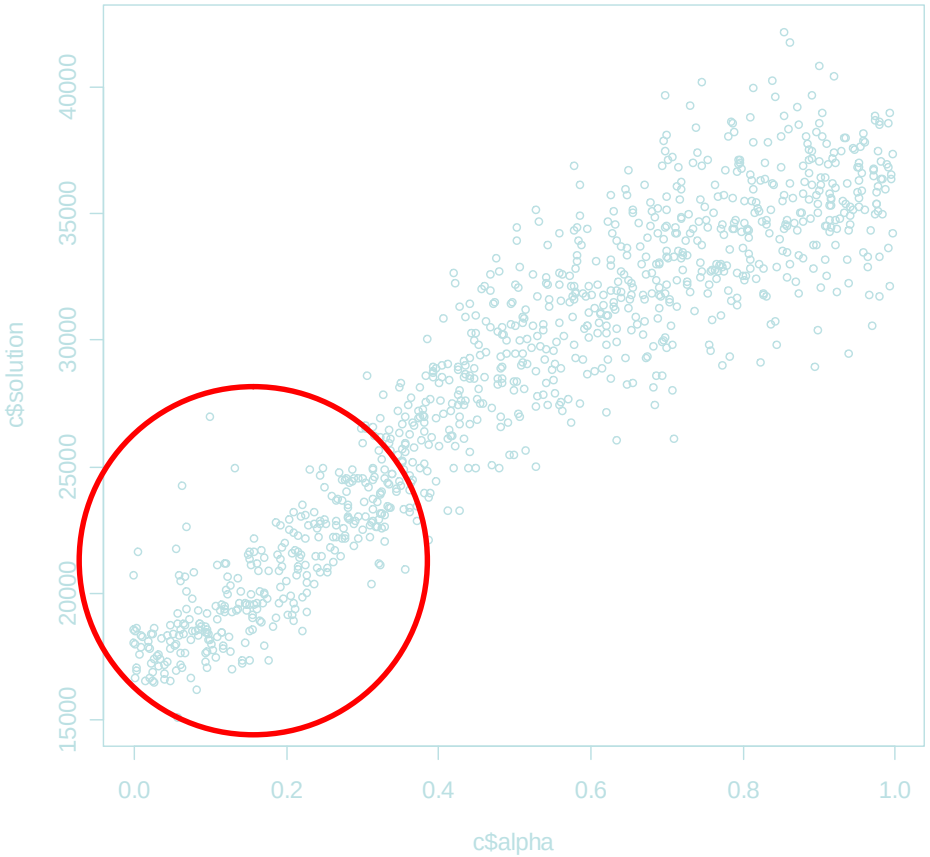
2-opt-move

3-opt-move

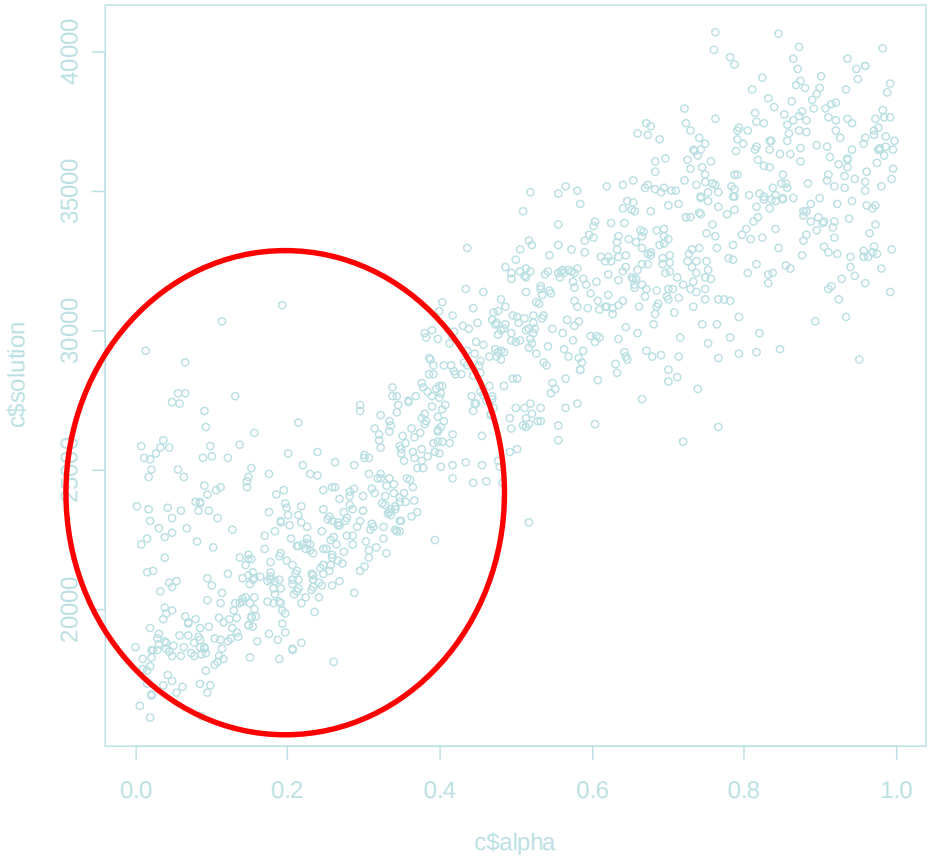


Vergleich bei Verwendung verschiedener k

3-opt-move



4-opt-move



Recheneffizient I

- Test der durchschnittlichen Rechenzeiten bei unterschiedlicher Wahl der Parameter
- auf Banias-Centrino™ mit 1.4 Ghz, 1 MB L2 Cache, 512 MB
- alpha keine Auswirkungen auf Rechenzeit
  à hier alpha ~ U(0, 0.2)
- Angaben in Sekunden!!!

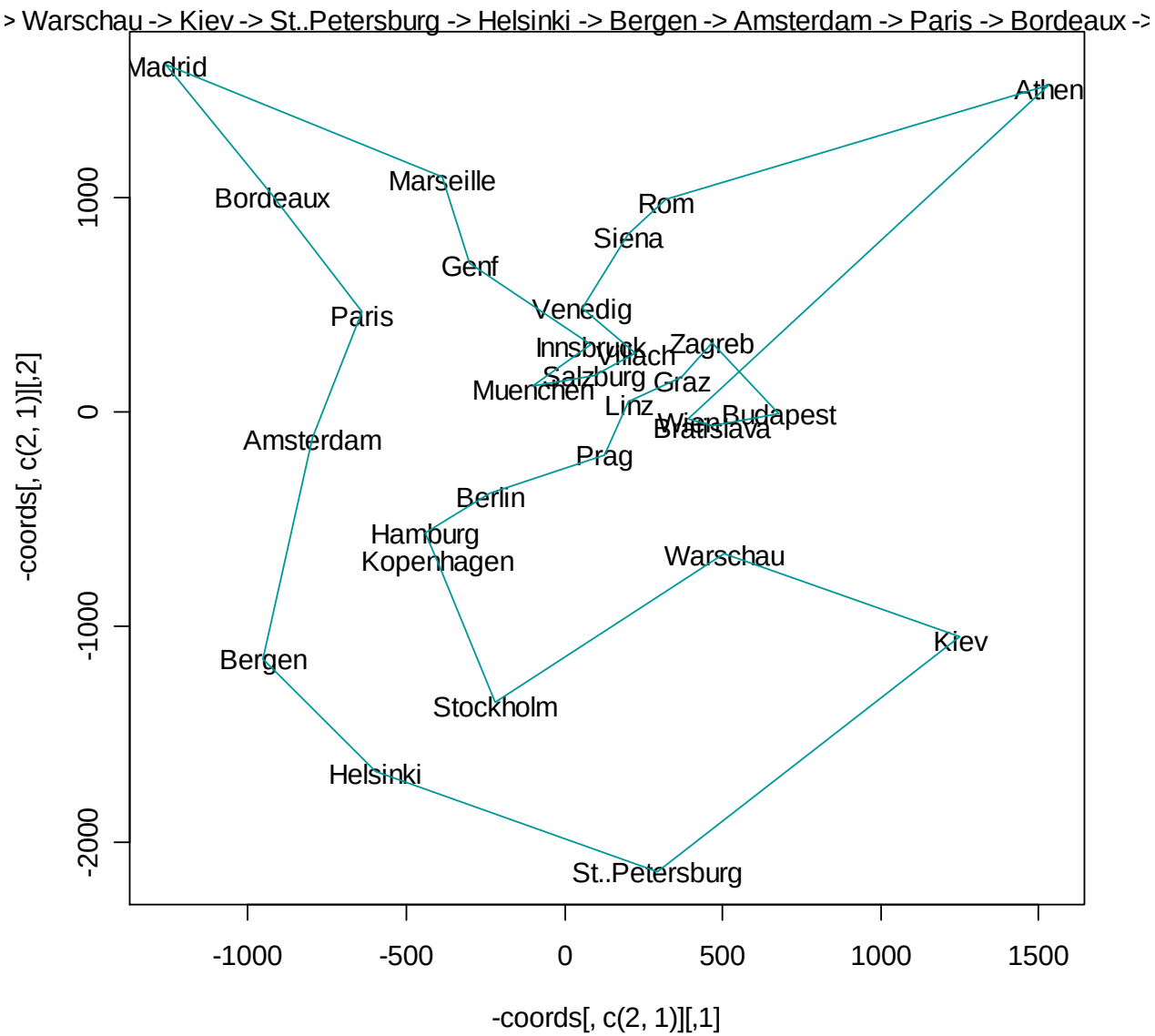
Recheneffizienz II

Sekunden	Iterations Grasp (iterations = …)	Iterations Local Search (n = …)	k-opt-move (k = …)
19.466	1,000	10,000	2
122.965	1,000	10,000	3
423.745	1,000	10,000	4
2.442	100	10,000	2
5.138	250	10,000	2
9.316	500	10,000	2
19.929	1,000	10,000	2
14.706	1,000	1,000	2
15.224	1,000	2,500	2
16.747	1,000	5,000	2
18.381	1,000	7,500	2
19.311	1,000	10,000	2

Unser Benchmark:

- Start in Wien (und Rückkehr nach Wien)
- 14.422 km
- Route:
  - Wien -> Bratislava -> Budapest -> Zagreb -> Graz -> Linz -> Prag -> Berlin -> Hamburg -> Kopenhagen -> Stockholm -> Warschau -> Kiev -> St..Petersburg -> Helsinki -> Bergen -> Amsterdam -> Paris -> Bordeaux -> Madrid -> Marseille -> Genf -> Innsbruck -> Muenchen -> Salzburg -> Villach -> Venedig -> Siena -> Rom -> Athen -> Wien

Unser Benchmark - Route:



Ausblick

- Einbinden einer Hash-Tabelle
- Multi-Tasking
- Effizienzsteigerung
- Einbinden einer Möglichkeit, dass Wahl für Verteilung von alpha automatisch an Ergebnisverteilung angepasst wird
- Package-Dokumentation

Zum Abschluss …

- … Live-Demonstration in R