

Simulated Annealing Resultate

Franziska Becker

28.11.2017

1 Parameter-Konfigurationen

Zunächst werden alle Parameter-Konfigurationen aufgezählt und erläutert:

1. Abkühlungsfunktion (i = Iteration)

(a) **square:** $t_i = i^2$

(b) **log:** $\frac{c}{\log(i+1)}$, $c = \frac{\max \text{ iterations}}{2}$

2. Startlösung

(a) **cheapest:** Sortiere Elemente nach nicht-aufsteigendem Nutzen $\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \dots \geq \frac{c_n}{w_n}$ und packe davon ausgehend solange Elemente nacheinander ein, bis der Rucksack voll ist.

(b) **lightest:** Sortiere Elemente nach nicht-absteigendem Gewicht $w_1 \leq w_2 \leq \dots \leq w_n$ und packe davon ausgehend solange Elemente nacheinander ein, bis der Rucksack voll ist.

(c) **random:** Gehe Elemente in originaler Reihenfolge durch und packe zufällig ein, bis der Rucksack voll ist.

Zusätzlich wurden verschiedene Abbruchskriterien ausprobiert:

1. **iterimprove:** Die maximale Anzahl an Iteration wurde erreicht oder die letzte Verbesserung war vor $\frac{\max \text{ iterations}}{5}$ Iterationen.

2. **time:** Das Zeitlimit wurde erreicht. (2 Sekunden)

2 Resultate

Für alle Konfigurationen wurden 100 Durchläufe gemacht und die Anzahl der maximalen Iterationen betrug immer 500.000.

Bei den Ergebnissen in Tabelle 1 wurden einige Zellen aus Gründen der Übersicht freigelassen. Dies bedeutet, dass sie den selben Inhalt wie ihr letzter Vorgänger, der eine Beschriftung hat, haben. Dies kann beispielsweise in Spalte 1 und Zeile 3 beobachtet werden, diese hat denselben Wert (rucksack0010) wie in Spalte 1 und Zeile 1.

| Instanz | Abkühlung | Startlösung | Abbruchkriterium | $\frac{1}{100} \sum_{i=1}^{100} c_i^*$ | Worst | Best |
|---------------|-----------|-------------|------------------|--|-------|------|
| rucksack0020a | square | cheapest | iterimprove | 133 | 133 | 133 |
| | | lightest | | 133 | 133 | 133 |
| | | random | | 133 | 133 | 133 |
| | log | cheapest | time | 133 | 133 | 133 |
| | | | iterimprove | 133 | 133 | 133 |
| | | lightest | | 133 | 133 | 133 |
| | | random | | 133 | 133 | 133 |
| | | | | | | |
| rucksack0030 | square | cheapest | iterimprove | 298.52 | 287 | 307 |
| | | lightest | | 297.96 | 286 | 307 |
| | | random | | 297.86 | 285 | 307 |
| | log | cheapest | time | 306.94 | 304 | 307 |
| | | | iterimprove | 298.42 | 287 | 307 |
| | | lightest | | 298.46 | 284 | 307 |
| | | random | | 299.08 | 285 | 307 |
| | | | | | | |
| rucksack0050 | square | cheapest | iterimprove | 446.43 | 419 | 476 |
| | | lightest | | 445.28 | 423 | 480 |
| | | random | | 446.57 | 414 | 482 |
| | log | cheapest | time | 466.51 | 452 | 483 |
| | | | iterimprove | 446.19 | 421 | 484 |
| | | lightest | | 445.54 | 423 | 484 |
| | | random | | 443.7 | 422 | 478 |
| | | | | | | |
| rucksack0100a | square | cheapest | iterimprove | 609.07 | 581 | 659 |
| | | lightest | | 607.25 | 570 | 668 |
| | | random | | 607 | 568 | 668 |
| | log | cheapest | time | 632.93 | 605 | 665 |
| | | | iterimprove | 608.76 | 578 | 662 |
| | | lightest | | 608.49 | 572 | 658 |
| | | random | | 605.35 | 568 | 666 |
| | | | | | | |
| rucksack0500 | square | cheapest | iterimprove | 6000 | 6000 | 6000 |
| | | lightest | | 6000 | 6000 | 6000 |
| | | random | | 6000 | 6000 | 6000 |
| | log | cheapest | time | 6000 | 6000 | 6000 |
| | | | iterimprove | 6000 | 6000 | 6000 |
| | | lightest | | 6000 | 6000 | 6000 |
| | | random | | 6000 | 6000 | 6000 |
| | | | | | | |
| rucksack1000 | square | cheapest | iterimprove | 868.42 | 794 | 999 |
| | | lightest | | 864.46 | 808 | 964 |
| | | random | | 858 | 780 | 977 |
| | log | cheapest | time | 867.67 | 773 | 980 |
| | | | iterimprove | 876.3 | 779 | 1007 |
| | | lightest | | 864.18 | 800 | 977 |
| | | random | | 858.75 | 789 | 950 |
| | | | | | | |

Table 1: Einsatz von Simulated Annealing für Binary Knapsack Instanzen

3 Auswertung

Die Resultate zeigen, dass sich die Ergebnisse der verschiedenen Strategien nicht signifikant unterscheiden. Bei den kleinen Instanzen sind die Lösungswerte sogar identisch. Es ist zu erkennen, dass die logarithmische Abkühlfunktion teilweise bessere Werte erzielt, der Differenz ist jedoch gering.

Bei den verschiedenen Abbruchkriterien zeigt sich, dass ein zeitbasierter Abbruch bei kleinen bis mittelgroßen Instanzen bessere Durchschnittswerte erzeugt, bei großen Instanzen ist kein besonderer Unterschied zu erkennen.

Für die Instanz *rucksack0500* ist zu sehen, dass alle Ergebnisse identisch sind. Dies lässt sich eventuell auf die Struktur der Instanz-Daten zurückführen, da jedes Item einen Nutzen von 1 hat ($c_i = w_i$) und somit die Sortierungen weniger Unterschiede erzeugen.