Simulated Annealing Resultate

Franziska Becker

26.11.2017

1 Parameter-Konfigurationen

Zunächst werden alle Parameter-Konfigurationen aufgezählt und erläutert:

1. Abkühlungsfunktion
   (a) Bezeichner: **square** $t_i = i^2$
   (b) Bezeichner: **recursive** $t_0 = \frac{max\ iterations}{2},\ t_i = t_{i-1}\alpha$
2. Startlösung
   (a) Bezeichner: **cheapest** Sortiere Elemente nach nicht-aufsteigendem Nutzen $\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \cdots \geq \frac{c_n}{w_n}$ und packe davon ausgehend solange Elemente nacheinander ein, bis der Rucksack voll ist.
   (b) Bezeichner: **lightest** Sortiere Elemente nach nicht-absteigendem Gewicht $w_1 \leq w_2 \leq \cdots \leq w_n$ und packe davon ausgehend solange Elemente nacheinander ein, bis der Rucksack voll ist.
   (c) Bezeichner: **random** Gehe Elemente in originaler Reihenfolge durch und packe zufällig ein, bis der Rucksack voll ist.

Zusätzlich wurden verschiedene Abbruchskriterien ausprobiert:

1. Bezeichner: **iterimprove** Maximale Iteration erreicht $\vee$ letzte Verbesserung $= \frac{max\ iterations}{5}$
2. Bezeichner: **time** Zeitlimit erreicht (2 Sekunden)

2 Resultate

Für alle Konfigurationen wurden 100 Durchläufe gemacht und die Anzahl der maximalen Iterationen betrug immer 500.000. Für den Einsatz der Abkühlung *recursive* galt immer, dass $\alpha = 0.9$.

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

Instanz	**Abkühling**	**Startlösung**	**Abbruchkriterium**	$\frac{1}{100}\sum_{i=1}^{100} c_i^*$
rucksack0010	square	cheapest	iterimprove	128.15
		lightest		119.68
		random		148.09
		cheapsest	time	134.43
	recursive	cheapest		130.77
		lightest		123.45
		random		141.31
rucksack0020a	square	cheapest	iterimprove	88.12
		lightest		76.81
		random		68.84
		cheapsest	time	86.52
	recursive	cheapest		86.71
		lightest		79.05
		random		68.54
rucksack0030	square	cheapest	iterimprove	164.74
		lightest		182.14
		random		148.43
		cheapsest	time	173.45
	recursive	cheapest		166.16
		lightest		174.93
		random		147.21
rucksack0050	square	cheapest	iterimprove	269.43
		lightest		277.48
		random		212.23
		cheapsest	time	271.39
	recursive	cheapest		281.88
		lightest		274.86
		random		195.65
rucksack0100a	square	cheapest	iterimprove	409.93
		lightest		376.28
		random		194.85
		cheapsest	time	400.62
	recursive	cheapest		405.91
		lightest		376.85
		random		191.59

Table 1: Einsatz von Simulated Annealing für Binary Knapsack Instanzen

3 Auswertung

TODO