Alternative Fitness Funktion

$$f(x) = -(s + m * v)$$

s = kumulierte Kosten für aktive Erweiterungen
v = Anzahl der Spannungsverletzungen
$m = Bestrafungsfaktor\ für\ Voltage\ Violations$

Idee: Ungültige Lösungen mit v>0 werden im Optimierungsverfahren zugelassen, erhalten jedoch für jede Spannungsverletzung einen Strafzuschlag in Höhe von m.

Getestete Algorithmen

1. Bergsteiger Algorithmus (bereits vorgegeben)
2. Genetischer Algorithmus
    a. Tournament Selection
    b. Crossover Operator: 1 Point Crossover
    c. Mutations Operator: Flip Operator
3. Simulated Annealing
4. Metropolis Algorithmus

Vorgehensweise

Testbedingungen:

- 10 Durchläufe
- Zeitlimit von 15 und 30 Sekunden
- Vorgegebener (=basic) und alternativer (=custom) Fitnessfunktion
- Bestrafung von Violations mit m = 100

Deskriptive Statistiken[1]

Tabelle 1 - Fitness Auswertung nach Algorithmen und Laufzeit

Laufzeit	Algorithm	Hillclimber		Evolutionary Algorithm		Simulated Annealing		Metropolis	
	Fitnessfunktion	basic	custom	basic	custom	basic	custom	basic	custom
15 Sekunden	Min	0,6786	-100,8000	0,6136	-3,4000	0,6923	-1,6000	0,6786	-1,8000
	Max	0,7273	-1,2000	0,6667	-2,0000	0,7273	-1,2000	0,7273	-1,2000
	Std.Abweichung	0,0152	29,7940	0,0140	0,3666	0,0078	0,0894	0,0145	0,1833
	Arithm. Mittel	0,7027	-11,4200	0,6315	-2,8400	0,7086	-1,4000	0,6995	-1,5200
30 Sekunden	Min	0,6471	-1,8000	0,6471	-2,4000	0,6923	-1,6000	0,6923	-1,6000
	Max	0,7083	-1,2000	0,6923	-1,6000	0,7273	-1,2000	0,7273	-1,2000
	Std.Abweichung	0,0189	0,2040	0,0148	0,2530	0,0122	0,1400	0,0108	0,1200
	Arithm. Mittel	0,6901	-1,4800	0,6799	-1,8000	0,7073	-1,4200	0,7124	-1,3600

[1] Detaillierte Auflistung der Ergebnisse in der Datei <Algorithmen_Vergleich.xlxs>

Tabelle 2 - Lösungsgüte nach Algorithmen und Laufzeit

		Hill Climber				Evolutionary Algorithm				Simulated Annealing				Metropolis			
	Fitnessfunktion	Basic		Custom		Basic		Custom		Basic		Custom		Basic		Custom	
Laufzeit		s	v	s	v	s	v	s	v	s	v	s	v	s	v	s	v
	Arithm. Mittel	1,48	0	1,42	0,10	2,84	0	2,82	0,00	1,40	0	1,46	0,00	1,52	0	1,52	0,00
15 Sekunden	Max	1,80	0	2,00	1,00	3,40	0	3,20	0,00	1,60	0	1,60	0,00	1,80	0	1,80	0,00
	Arithm. Mittel	1,66	0	1,48	0,00	1,80	0	1,84	0,00	1,42	0	1,42	0,00	1,36	0	1,32	0,00
30 Sekunden	Max	2,40	0	1,80	0,00	2,40	0	2,20	0,00	1,60	0	1,60	0,00	1,60	0	1,40	0,00

Auswertung

Gültigkeit der Lösungen

Die Kombination des Hill Climber Algorithmus mit meiner selbst entworfenen Fitness Funktion lieferte als einziger Algorithmus eine ungültige Lösung in einem einzelnen Durchlauf, alle anderen Konstellationen haben in meinem Test ausschließlich gültige Lösungen mit v=0 produziert. Verantwortlich dafür ist meiner Einschätzung nach, dass Hill Climber Algorithmen stark dazu neigen schnell in einem lokalen Optimum zu konvergieren, die Strategie der Bestrafung von ungültigen Lösungen ist daher wahrscheinlich nicht optimal für Hill Climber und eignet sich besser für Algorithmen wie Simulated Annealing, da hier eine höhere Wahrscheinlichkeit dafür haben, ein solches lokales Optimum mit v>0 wieder zu verlassen.

Lösungsqualität

Für eine Laufzeit von 30 Sekunden hat der Metropolis Algorithmus in meinem Test die beste Lösungsqualität im Mittelwert (s: 1,36 / 1,32) [basic/custom] erbracht und die Performance des Hill Climber (s: 1,66 / 1,48) übertroffen. Bei einer kürzeren Laufzeit von 15 Sekunden wurden die beste Lösungsqualitäten von den Algorithmen Simulated Annealing (s: 1,4 / 1,46) und Hill Climber (s: 1,48 / 1,42) erbracht. Ob diese Unterschiede in der Performance statistisch signifikant sind, soll im Anschluss getestet werden.

Der Evolutionäre Algorithmus liefert bei einer kurzen Laufzeit die niedrigste durchschnittliche Lösungsqualität (s: 2,84 / 2,83), bei einer Erhöhung der Laufzeit zeigt sich eine Verbesserung (s: 1,8 / 1,84), trotzdem bleibt die Lösungsqualität niedriger als bei den anderen getesteten Algorithmen.

Einfluss der Fitness Funktion

Die Unterschied der Lösungsqualität ist in fast allen Konstellationen sehr gering, ich bezweifle daher, dass meine entworfende Fitness Funktion eine signifikante Verbesserung der Performance bringt. Eine Ursache dafür könnte die Höhe des Faktors m sein, durch eine feinere Justierung könnte es eventuell möglich sein, die Funktion besser an das Ausgangsproblem anzupassen.

Statistischer Test

Testbedingungen

1. Teste die vielversprechensten Algorithmen: Hill Climber, Simulated Annealing und Metropolis
2. Abbruchbedingungen: 10.000 Fitness Evaluationen
3. Verwendete Fitnessfunktion: evaluate_objective_function(case)
4. Durchläufe pro Algorithmus: 10

Ergebnisse

Detaillierte Auflistung der Testergebnisse in der Datei <statistical_test.ipynb> (oder alternativ <statistical_test.html>)