Heuristische Optimierungsverfahren

1.Programmierübung

Johannes Reiter, Christian Gruber

01.Dezember 2009

1 Tool Switching Problem

Gegeben ist eine Menge von Jobs, die zu ihrer Abarbeitung jeweils einen gewissen Satz von Tools benötigt. Da nicht alle Tools gemeinsam in das Magazin der Maschine passen (max. Anzahl entspricht der Magazingröße), ist nun eine möglichst günstige Reihenfolge der Jobs gesucht, in der die Anzahl der Tool Switches minimiert wurde. Die Lösung zu einer Test-Instanz eines Tool Switching Problems ist nun eine Permutation in der alle Jobs abgearbeitet werden.

1.1 Tool-Konfiguration für eine vorgegebene Reihenfolge

Gesucht ist hier eine möglichst optimale Magazin-Konfiguration für eine vorgegebene Reihenfolge für die Abarbeitung der Jobs. In diesem Spezialfall kann man nun ausnutzen, dass der Algorithmus genau weiß wann ein Tool das nächste Mal benötigt wird und somit zu der vorgegebenen Reihenfolge die optimale Tool-Konfiguration erzeugen kann.

1.2 Konstruktionsheuristik

Unsere Konstruktionsheuristik wählt am Beginn zufällig den ersten Job aus. Die weiteren Jobs werden so gewählt, dass möglichst wenig Kosten entstehen.

1.3 Nachbarschaftsstrukturen

- **move:** In dieser Nachbarschaft wird ein Job ausgewält und dieser zufällig an eine andere Position in der Sequenz verschoben.
- **2-exchange (switch):** Beim 2-exchange tauschen 2 Jobs einfach ihre Position in der Sequenz aus.

Für die vier verschiedenen Nachbarschaftsstrukturen haben wir jeweils auch die drei Schrittfunktionen random neighbor, next improvement und best improvement implementiert. Eine inkrementelle Bestimmung der Zielfunktionswerte von Nachbarlöungen wäre zwar möglich gewesen, jedoch nicht besonders sinnvoll, da man mit dem im 1.Punkt implementierten Algorithmus die jeweils optimale Tool-Konfiguration für eine vorgegebene Reihenfolge von Jobs bekommt und das Ergebnis der Kostenfunktion eigentlich erst dann wirklich aussagekräftig ist.

1.4 Vergleich der verschiedenen Varianten für die lokale Suche

Testinstanz	Nachbarschaft	Schrittfunktion	Mittelwert
$4\zeta_{10}^{10}$	move	random	12.9
	move	next	11.2
	move	best	11.2
	2-exchange	random	13.2
	2-exchange	next	11.1
	2-exchange	best	11.6
$15\zeta_{40}^{30}$	move	random	157.6
	move	next	119.6
	move	best	119.5
	2-exchange	random	158
	2-exchange	next	122.7
	2-exchange	best	124
$20\zeta_{60}^{40}$	move	random	283.2
	move	next	196.2
	move	best	196.5
	2-exchange	random	283.8
	2-exchange	next	203.8
	2-exchange	best	203.5

1.5 Variable Neighborhood Descent (VND)

Wir haben mit der VND jeweils 30 runs gemacht und als Schrittfunktion ist best verwendet worden. Sehr schön kann man hier erkennen, wie man aus einem lokalen Optimum für eine Nachbarschaft durch einen Wechsel der Nachbarschaftsstruktur wieder heraus finden kann.

Testinstanz	Mittelwert	Bestes Ergebnis	Standardabweichung
$4\zeta_{10}^{10}$	11	10	0.8
$15\zeta_{40}^{30}$	117.2	114	2.3
$20\zeta_{60}^{40}$	193.9	188	4

1.6 Generalized Variable Neighborhood Search (GVNS)

Wir haben mit der GVNS jeweils 30 runs gemacht und als Schrittfunktion haben wir best verwendet. Man kann aus den erzielten Ergebnissen relativ eindeutig die Verbesserung durch die GVNS erkennen. In den log-Dateien sieht man auch wie oft dieses „shaking“ wirklich hilft.

Testinstanz	Mittelwert	Bestes Ergebnis	Standardabweichung
$4\zeta_{10}^{10}$	10.5	10	0.5
$15\zeta_{40}^{30}$	113.7	111	2.0
$20\zeta_{60}^{40}$	185.2	181	3.0