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 und eine dazu passende Sequenz von Magazin-Konfigurationen.

1.1 Tool-Konfiguration für eine vorgegebene Reihenfolge

Gesucht ist hier eine mglichst 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 mglichst wenig Kosten entstehen. D.h. der Algorithmus sieht sich alle noch offenen Jobs an und vergleicht die benötigten Tools mit der aktuellen Magazin-Konfiguration.

Results

test instance	k	objective function value	running time	branch-and-bound nodes	optimum node
fad-01.dat	6	629	0.02s		
	3	113	0.04s		
fad-02.dat	11	1575	0.88s	500	406
	6	716	0.44s	300	285
fad-03.dat	26				
	11	903	63.56s	12300	7124
fad-04.dat	36				
	15	972	6330s	530900	
fad-05.dat	51				
	21				
fad-06.dat	101				
	41				

2 Cycle Elimination Formulation

Der Vorteil der Cycle Elimination Formulierung ist, dass wir nicht exponentiell viele Ungleichungen benoetigen. Diese werden quasi on-demand, also falls ein Zyklus gefunden wird, ueber eine Callbackfunktion hinzugefuegt, die in jedem Branch-Node aufgerufen wird. Um Zyklen zu finden werden bei jeder Kante $e = u, v$ einmal die Kosten auf unendlich gesetzt und dann versucht den kuerzesten Pfad zwischen den Knoten u und v zu finden. Wenn jetzt $\sum_{e \in \delta(C)}(1-x_e) < 1$ (wobei C der gefundene Pfad plus die Kante e ist), dann wird ein Cut generiert und die Nebenbedingung (12) hinzugefuegt. Dieses Verfahren wird fuer alle markierten Kanten in der Loesung wiederholt.

$$min \sum_{e \in E} w_e * x_e \tag{1}$$

$$s.t. \sum_{e \in E} x_e = k - 1 \tag{2}$$

$$\sum_{e \in \delta(C)} (1 - x_e) \geq 1 \tag{3}$$

Mit der folgenden Gleichung wird ueber z_V sichergestellt, dass genau k Knoten ausgewaehlt werden muessen.

$$\sum_{v \in V} z_v = k \tag{4}$$

Falls eine Kante e ausgewaehlt wurde, muss auch der jeweilige Anfangs- bzw. Endknoten durch die Entscheidungsvariable z_v markiert sein.

$$\begin{aligned} x_{(u,v)} &\leq z_u \qquad \forall (u,v) \in E \\ x_{(u,v)} &\leq z_v \qquad \forall (u,v) \in E \end{aligned} \tag{5}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{6}$$

$$z_v \in \{0,1\} \qquad \forall v \in V \tag{7}$$

Results

test instance	k	objective function value	running time	branch-and-bound nodes	optimum node	separated cuts
fad-01.dat	6	629	0.02s			
	3	113	0.03s			
fad-02.dat	11	1575	0.06s			14
	6	716	0.1s	2	2	
fad-03.dat	26	3182	1.85s	34	34	30
	11	903	0.09s			15
fad-04.dat	36	3469	0.84s	3	3	23
	15	972	0.43s			44
fad-05.dat	51	5000	26.84s	100	97	37
	21	1322	0.42s			2
fad-06.dat	101	6790	628s	505	505	109
	41	1997	3.31s			8