Aufgabe 1 – Scheduling als Suchproblem

a) Suchproblemformulierung
Zustand: U/M

- U: Stack, der die noch nicht verteilten Jobs enthält[1]. Jeder Job ist ein Integer, der dessen Länge darstellt. (eine Unterscheidung von Jobs ist nicht notwendig, und die Reihenfolge, in der die Jobs gescheduled werden, ist bei dieser Problemrepräsentation irrelevant).
- M: m-elementiges Array von Integern, dargestellt als $[t_1,\ldots,t_m]$, wobei $t_1,\ldots,t_m$ die aktuelle Produktionszeit (das ist der letzte Auslastungszeitpunkt) der jeweiligen Maschine kennzeichnet.

  Bemerkungen:
  - in M sind die Namen der Maschinen irrelevant. Durch die gewählten Aktionen muss auch kein wirklicher Schedule verwaltet werden, weil jeder Job immer so eingescheduled wird, dass er direkt an den letzten anschließt.
  - Man könnte auch eine CSP-ähnliche Repräsentation wählen, in der im Zustand nur die Jobs verwaltet werden. Jedem Job wird dann in einer Aktion eine Maschine und eine Startzeit zugewiesen, zu Beginn sind alle Jobs noch „unassigned“. Die Bewertung der Qualität einer Lösung wäre dann gleichbedeutend mit dem letzten Job (größter Wert für Startzeit + Dauer).
  - Noch effizienter wäre es, die Maschinen in einer Menge (und nicht in einem Array) zu verwalten. Das würden den Verzweigungsfaktor des Baums noch verkleinern (denn die Zustände [x,y,z], [z,x,y], [y,z,x], [y,x,z] usw. wären dann dieselben: {x,y,z}. Zwar kann man im Zielzustand die Maschinen nicht mehr unterscheiden, aber das ist für die gegebene Aufgabenstellung eine zulässige Abstraktion.

Startzustand: $[J_1,\ldots,J_n]/[0,\ldots,0]$; wobei $J_1,\ldots,J_n$ die n Jobs darstellen. Alle Maschinen sind noch frei.

Ziel: $[]/[T_1,\ldots,T_m]$ (alle Jobs sind auf die Maschinen verteilt). Die optimale Lösung minimiert $\max([T_1,\ldots,T_m])$.

Aktion: schedule(x): $s_i \rightarrow s_{i+1}$, wobei $s_i=u/z$, $j=top(u)$, $s_{i+1}=pop(u)/z'$, $z'=z$ bis auf $z'[x]=z[x]+j$, , $x\in\{1,\ldots,m\}$.
Wähle eine Maschine x und plane den obersten Job j des Stacks U zum frühestmöglichen Zeitpunkt ein.

Aktionskosten: Differenz aus $\max(z_{i+1})$ - $\max(z_i)$; wobei $s_i=u/z_i$ und $s_{i+1}=u'/z_{i+1}$
Das Einschedulen eines Jobs ist genauso teuer, wie sich der Produktionszeitpunkt verschiebt. Eine Aktion kann also, da immer zum frühestmöglichen Zeitpunkt eingeplant wird, minimal 0 Kosten verursachen und maximal so teuer sein wie der längste Job lang ist.

Heuristik: Bei dieser Repräsentation existiert keine gut Heuristik, weil der Wert der optimalen Lösung nicht bekannt ist und die Zahl der Aktionen bis zur Lösung fix ist (n, bei n Jobs).

[1] Eine Menge wäre auch möglich, würde aber zu stärkerem Verzweigungsgrad führen, wenn die Wahl des nächsten Jobs beliebig ist.

Bemerkungen:

Ohne Backtracking geht es nicht, selbst eine „intelligente“ Aktion, die von sortierten Jobs ausgeht und immer nur die am geringsten ausgelastete Maschine belegt, führt nicht zur optimalen Lösung: z.B. bei 2 Maschinen und 5 Jobs der Längen {9,8,7,7,3} (die optimale Lösung ordnet die beiden längsten Jobs auf M1 und die übrigen auf M2 zu (M1([9,8]);M2([7,7,3])) – pz=17). Hier existiert ein lokales Optimum (M1([9,7]), M2([8,7,3])) mit pz=18, sodass ein Vertauschen oder Verlegen nicht zu einer Verbesserung führt: M2 benötigt im beschriebenen lokalen Optimum 18 Zeiteinheiten; M1 mit zusätzlich dem 3er Job hingegen 19.

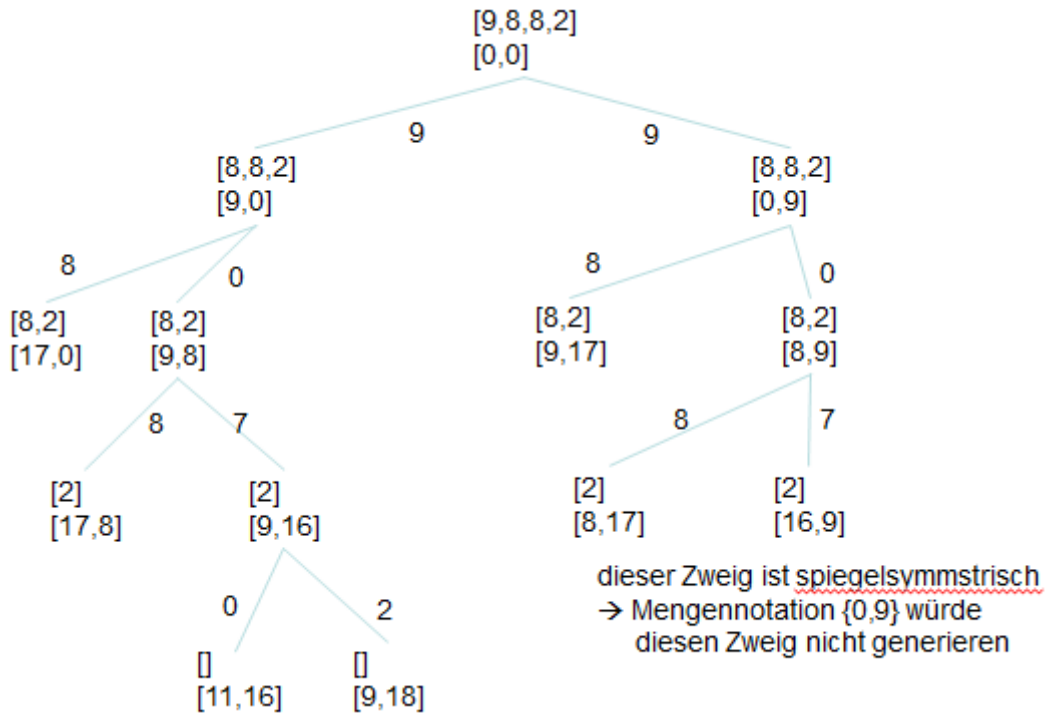
Mit einer konstruktiven, heuristisch gesteuerten Suche (Bestensuche) (weise Job auf Maschine zu, wähle diejenige Aktion, die die Gesamtzeit minimal hält, Ziel erreicht, wenn alle Jobs verteilt) kommt man auch zur optimalen Lösung.

Mit lokaler Suche (beginne mit zulässiger Lösung, vertausche 2 Jobs, Ziel erreicht, wenn keine Verbesserung mehr möglich ist) erreicht man ein lokales, nicht aber notwendigerweise globales Optimum.

Dieselbe Aktion, in Verbindung mit vollständiger Suche (Tiefen- oder Breitensuche, die den gesamten Baum aufspannt), liefert dann eine optimale Lösung, wenn man den besten Blattknoten sucht – diese Vorgehensweise entspricht aber nicht dem üblichen Verfahren bei Suchproblemen.

Zieltest: „Differenz zwischen schnellster und langsamster Maschine < kürzester Job“ funktioniert nicht für {9,1} und 2 Maschinen (Differenz ist mindestens 8).

b) Suchbaumcharakteristik: Der Verzweigungsgrad ist m, die Tiefe n.
c) Das ideale Suchverfahren wäre A* mit h(i)=0 für alle Zustände (maximal uninformiert, also Branch & Bound). Uninformierte Suchverfahren führen zwar (Tiefensuche) zu einer schnellen Lösung, können aber Optimalität nicht sicherstellen, und sind daher nicht geeignet.
d) Simulation. Hier steht der vollständige Suchbaum:



Die Lösung liegt im untersten linken Knoten. Maschine 1 bekommt die Jobs 9 und 2; Maschine 2 die beiden 8er-Jobs. Diese Zuordnung ist im Zustand nicht direkt ablesbar, kann aber aus dem Suchbaum (den zum Ziel führenden Zuständen) rekonstruiert werden. Der Zweig ganz links wird nicht expandiert, weil der Lösungspfad eine bessere Bewertung trägt. Dasselbe gilt für die nicht bis zur vollständigen Baumtiefe expandierten Pfade im rechten Teilbaum, hier endet die Suche auch jeweils mindestens dann, wenn die Pfadkosten der besten Lösung erreicht sind.
Simulation von A* mit f=g-Werten in der sortierten Liste (die ersten 4 Expandierungen):

0. <[9,8,8,2]/[0,0]>(0)
1. <[9,8,8,2]/[0,0], [8,8,2]/[9,0]>(9), <[9,8,8,2]/[0,0] , [8,8,2]/[0,9]>(9)
2. <[9,8,8,2]/[0,0], [8,8,2]/[9,0], [8,2]/[9,8]>(9), <[9,8,8,2]/[0,0], [8,8,2]/[0,9]>(9), <[9,8,8,2]/[0,0], [8,8,2]/[9,0], [8,2]/[17,0]>(17)
3. <[9,8,8,2]/[0,0] , [8,8,2]/[0,9]>(9), <[9,8,8,2]/[0,0], [8,8,2]/[9,0], [8,2]/[9,8], [2]/[9,16]>(16), <[9,8,8,2]/[0,0], [8,8,2]/[9,0], [8,2]/[9,8], [2]/[17,8]>(17), <[9,8,8,2]/[0,0], [8,8,2]/[9,0], [8,2]/[17,0]>(17)
4. <[9,8,8,2]/[0,0] , [8,8,2]/[0,9], [8,2]/[8,9]>(9) <[9,8,8,2]/[0,0], [8,8,2]/[9,0], [8,2]/[9,8], [2]/[9,16]>(16), <[9,8,8,2]/[0,0], [8,8,2]/[9,0], [8,2]/[9,8], [2]/[17,8]>(17), <[9,8,8,2]/[0,0], [8,8,2]/[9,0], [8,2]/[17,0]>(17), <[9,8,8,2]/[0,0] , [8,8,2]/[0,9], [8,2]/[9,17]>(17)

Aufgabe 2 – Constraints

a) Chronologisches Backtracking

a=1 b=1 c=1
c=2
c=4
b=2 c=1
c=2
c=4
b=4 c=1
c=2 d=1

Lösung gefunden, nach 7 Backtrackingschritten (nach jeder Zeile ein Backtrackschritt).

b) Backtracking mit Forward Checking. Wertebereichseinschränkungen hinter dem →

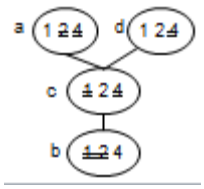
a=1 → c={2,4}
b=1
b=2
b=4 → c={2}
c=2 → d={1,2}
d=1

Lösung gefunden, nach 2 Backtrackingschritten (nach b=1 und b=2)

c) Minimum Remaining Values Heuristik (MRV)
Alle Variablen haben gleich große Wertebereiche. Die MRV-Heuristik liefert keine eindeutige Entscheidung.

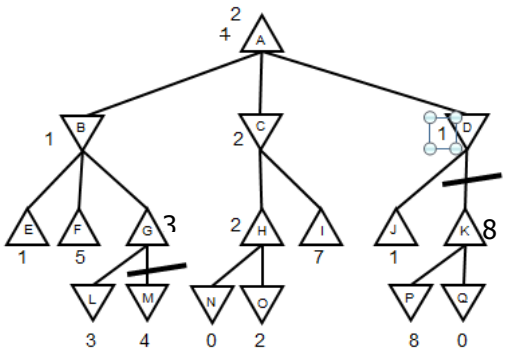
d) Most Constrained Variable Heuristik (Grad-Heuristik, MCV)
Variable c hat 3 Constraints und ist die MCV. Die restlichen Variablen sind jeweils nur in einem Constraint enthalten.

e) 2-Konsistenz



Die Variablen a, b und c sind instanziiert, d hat noch 2 Werte in seiner Domain.

Aufgabe 3 – Minimax mit α-β-Cutoff



Aufgabe 4 – Maschinelles Beweisen

(1) { P(1) }
(2) { P(2) }
(3) { P(3) }
(4) { P(4) }
(5) { Q(1,2) }
(6) { Q(2,3) }
(7) { Q(3,4) }
(8) { Q(4,3) }
(9) $\{ P(a),\ Q(a,b),\ \neg R(a,b)\}$
(10) $\{ \neg R(c,d),\ \neg S(d,c) \}$
(11) { S(3,4), }
(12) { S(e,e) }

zu zeigen: $\exists x, y\ P(x) \wedge Q(x,y) \wedge \neg R(x,y)$

negieren:
$$\neg\exists x, y\ P(x) \wedge Q(x,y) \wedge \neg R(x,y)$$
$$\Leftrightarrow\ \forall x, y\ \neg\big(P(x) \wedge Q(x,y) \wedge R(x,y)\big)$$
$$\Leftrightarrow\ \forall x, y\ \neg P(x) \vee\ \neg Q(x,y)\ \vee R(x,y)$$
(13) $$\Leftrightarrow \neg P(x) \vee\ \neg Q(x,y)\ \vee R(x,y)$$

Resolutionsbeweis führen

$(13 \wedge 10, \{c\backslash x,\ d\backslash y\})$ $\{\neg P(x), \neg Q(x,y), \neg S(y,x)\}$

$(14 \wedge 11, \{y\backslash 3,\ x\backslash 4\})$ $\{\neg P(4), \neg Q(4,3)\}$

$(15 \wedge 4, \{\})$ $\{\neg Q(4,3)\}$

$(16 \wedge 8, \{\})$ $\{\}$

Aufgabe 5 – Planung

a) Fehler im move/3-Operator: clear(T) sowohl in ADD als auch in DEL, falls {z/T} substituiert wird. Der Nachfolgezustand für move(A,B,T) ist dann nicht wohldefiniert, korrekt nur, falls die DEL-Liste vor der ADD-Liste ausgewertet wird.
Das Bewegen des Tisches ist in der durch S0 dargestellten Welt nicht möglich, das Fehlen von Typprädikaten ist daher kein Fehler!
Dass bei jeder Aktion ein clear(T) hinzukommt, ist ungewöhnlich, aber kein Problem!

b) Rückwärtsplanung mit S_Z= on(A,T)
   1. move(B,y,T) hat on(B,T) in ADD. Die PRE (und damit die neuen Subziele) wäre dann: {clear(B), clear(T), on(B,y)}. Durch Substitution von y/C wäre on(B,C) bereits erfüllt, also ist die letzte Aktion move(B,C,T), S_{Z-1}={ clear(B), on(B,C), clear(T)}
      Ebenfalls ins Ziel führt die Aktion move(B,A,T), dieses ist die 2. mögliche Aktion.

   2. Das erste Subziel clear(B) wird erfüllt, wenn move(x,B,z). Neue Subziele sind {on(x,B), clear(x), clear(z)}. Mit x/A sind die beiden ersten Subziele bereits erfüllt. mit z/T ist auch das 3. S_Z erfüllt. Also move(A,B,T), S_{Z-2}={on(B,C), on(A,B), clear(A)} oder {on(B,C), on(A,B), clear(A), clear(T)}, je nachdem, ob die PRE oder die ADD zuerst „verarbeitet“ wird.
      Hier können x und z noch wechselseitig mit {A,C,T} belegt werden. Macht insgesamt 6 alternative rückwärts anwendbare Aktionen.

   Das Subziel clear(T) ist ja bereits im Startzustand, und das Problem ist gelöst. S_{Z-2} enthält keine unerfüllten Subziele mehr, durch Anwendung von 2 Aktionen ist das Problem gelöst. Nochmal in Kürze:

   - move(B,C,T)
   - S_{Z-1}={ clear(B), on(B,C), clear(T)}
   - move(A,B,T)
   - S_{Z-2}={on(B,C), on(A,B), clear(A)} oder {on(B,C), on(A,B), clear(A), clear(T)}

1a) (15 Punkte)

-1 wenn zu wenig Information im Zustand (z.B. keine Jobnamen)

-1 bis -2 wenn Zieltest falsch

-3 wenn Maß für Qualität / Optimalität im Zustand fehlt

-3 wenn Optimalität nicht durch Heuristik oder Kosten erzielbar; -1 bis -2 wenn Optimalität diskutiert wird

-5 (jeweils) wenn Start-, Ziel, oder Aktionen fehlen

jedoch:
5 Punkte, wenn Suchproblem verstanden aber Beispiel nicht modelliert

6 Punkte, wenn Zustand beschrieben, Aktion und Optimalität fehlen

maximal 4 Punkte für Start- und Zielzustand

-1 bis -2,5 bei ungeschickter oder wenig formaler Modellierung

-1,5 bei jedem sonstigen Fehler oder Auslassung

-0,5 / -1 bei Fehler im Anfangs- oder Zielzustand

-2 wenn Zielzustand fehlt

-1 bei nicht formaler oder fehlerhafter Zustandsüberführung in Aktion

Keine Abzüge bei unnötig vielen oder umständlichen Aktionen

1b) (2 Punkte)

-0,5 wenn Inkonsistenz zu 1a), aber konsistent zu 1d)

1c) (3 Punkte)

3 Punkte, wenn optimale Lösung gefunden werden kann (A*, Bestensuche, B&B > vollständigen Baum mit Tiefensuche > vollst. Baum m. Breitensuche > lokale Suche, Tiefensuche, Breitensuche)

2,5 wenn Antwort unpräzise formuliert

2 wenn optimale Lösung nicht gefunden werden kann

1 wenn Breitensuche unnötigerweise vor Tiefensuche präferiert wird

1d) (10 Punkte)

5 maximal, wenn Baum klar falsch

8 maximal, wenn f-Werte bei informierter Suche fehlen

8 maximal, wenn nur 3 Knoten expandiert

6 maximal wenn weniger als 3 Knoten expandiert

-3 wenn Knotenexpandierung unvollständig

-1 bis -2 wenn Zustand falsch / unvollständig

2a) (8 Punkte)

kein Abzug bei 9 statt 7 Backtrackpunkten (wenn die Zuweisungen an b mitgezählt werden)

5-6 Punkte, wenn Baum korrekt, aber Backtrackingschrittzahl falsch

<= 3 Punkte, wenn zu frühes Backtracking (Testen vor dem Belegen) oder nicht systematisches Belegen (z.B. a=1,b=1,c=2) oder zu spätes Backtracking (naive Suche)

-0,5 bei Flüchtigkeitsfehler

+1 wenn Backtrackschritte korrekt gezählt

+5 wenn Baum weitgehend korrekt und Lösung erreicht

+2 wenn Baum teilweise korrekt

2b) (8 Punkte)

5 Punkte: Baum korrekt, Wertebereichseinschränkungen unzureichend notiert

4,5 Punkte: Baum korrekt, aber Wertebereichseinschränkungen nicht notiert

3 Punkte: Baum weitestgehend korrekt, aber keine Wertebereiche notiert

-0,5 bei Flüchtigkeitsfehler

-2 bei Fehler

5 Punkte, wenn 2 Fehler

2e) (5 Punkte)

4 Punkte bei 1 kleinem Fehler

3 Punkte bei frühem Fehler und Rest folgerichtig

2 Punkte bei 2 Fehlern

3) 10 Punkte

7 Punkte: 1 Cutoff zu viel, zu früh, an falscher Stelle oder zu wenig

≤5 Punkte wenn 1 Cutoff falsch und 1 Cutoff vergessen

2 Punkte wenn Werte folgerichtig nach oben propagiert, aber Cutoffs falsch

-2 wenn Wert falsch propagiert

4) (20 Punkte)

10 wenn noch ausreichend

-4 bei Fehler in KNF

-2 bei 2. Fehler in KNF

-5 bis -10 wenn KNF nicht verstanden

-8 wenn Widerspruchsbeweis nicht verstanden

-2 wenn Substitution falsch

-5 bis -10 wenn Resolution nicht verstanden

ansonsten:
-1 bis -2 pro Fehler

3 wenn Resolutionsregel korrekt angewendet

10 wenn Resolutionsbeweis folgerichtig

-1 wenn Resolutionsbeweis fehlgeschlagen und eine Interpretation fehlt.

5a) (6 Punkte)

2 wenn clear(T) als problematisch charakterisiert wurde und die Erklärung nachvollziehbar ist.

2 wenn fehlende Typprädikate benannt wurden

3 wenn beide Fälle benannt wurden

5b) (9 Punkte)

5 wenn ein korrekter Rückwärtsplanungsschritt dargestellt wurde

3 wenn Vorwärtsplanung korrekt durchgeführt wurde

2-3 wenn Prinzip der Rückwärtsplanung korrekt dargestellt wurde

-4,5 wenn move(B,y,T) nicht als erste Aktion gefunden

-1 wenn Vorgängerzustände unvollständig (kein Abzug, wenn erfüllte Ziele durchgängig nicht enthalten)

-1 wenn Zahl der anwendbaren Aktionen falsch

2 Wenn Plan korrekt angegeben wird

1 für Nennung der Zahl der konsistenten und relevanten Aktionen

-2 wenn eine Aktion fehlt

kein Abzug, wenn Variablen unnötig spät instantiiert wie beim Partial Order Planning

kein Abzug, wenn clear(T) im Zustand Sz-1 steht, obwohl in ADD enthalten.