# Verslag Project Gegevensstructuren & Algoritmen

Jaron Maene

17 december 2017

## Inleiding

In dit project werd het tabu search algoritme voor het job shop scheduling probleem geïmplementeerd zoals beschreven in het artikel van Dell'Amico en Trubian[1]. We kijken naar de moeilijkheden en details die hiermee gepaard gingen, evenals de resultaten. Ook analyseren we kort enkele parameters.

## 1 Implementatie

### 1.1 Representatie

We beginnen met de representatie/datastructuren voor de algoritmes, aangezien deze cruciaal zijn voor het begrip van de volgende secties.

Zowel het bidirectionele list scheduling algoritme als de tabu search opereren op de gewogen digraph $G = (V, E \cup A)$. Hierbij is $V$ de verzameling operaties met een toegevoegde start en eind node, $A$ de bogen die de volgorde van de operaties bepaald, en $E$ de bogen die de volgorde op de machines bepaald. Merk dus op dat $A$ altijd directioneel is, terwijl $E$ directioneel wordt gemaakt bij het oplossen van het probleem. Het gewicht van een boog tussen $v$ en $w$ is gelijk aan de processing time gegeven voor $v$ (notatie: $D_v$).

De jobshop klasse voorziet functies om voor een gegeven node/operatie de opvolger/voorgangers te vinden in zowel $A$ als $E$. Verder is op elke node ook de functies $r : V \to \mathbb{N}$ en $t : V \to \mathbb{N}$ gedefiniëerd. Deze drukken respectievelijk de lengte van het langste pad van start naar $v$ en het langste pad van $v$ naar het einde uit, min hun eigen gewicht.

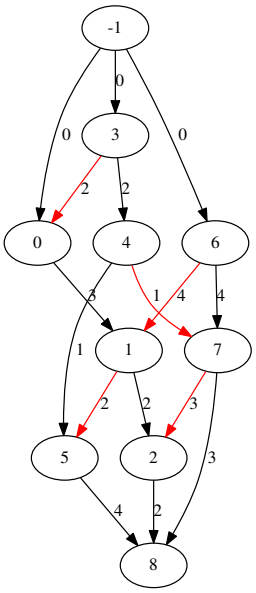
De enige keuze die gemaakt is omtrent de implementatie is de vorm van $E$ voor een gegeven configuratie[1]. We gaan hier dieper op in. We weten dat de initiële $E$ gegeven is door:

$$E = \{(v, w) \mid \mu_v = \mu_w\}$$

[1] Zie ook artikel [2] voor een uitgebreidere beschrijving.

De tussentijdse configuratie in het algoritme zal dus een deelverzameling hier van zijn. Bidir zal door middel van bogen te verwijderen $E$ directioneel maken en hierdoor dus de graaf "oplossen". Er is echter ook een tweede mogelijkheid, waarbij de restrictie wordt toegevoegd dat als $(i, j) \in E$, $j$ dan ook de directe machine opvolger van $i$ moet zijn. Deze methode is moeilijker om te handhaven, maar wel efficiënter (aangezien er op minder bogen moet gewerkt worden tijdens de estimation).

In het tabu search algoritme wordt altijd de laatste methode gebruikt. Echter voor bidir is dit minder vanzelfsprekend. Immers aangezien op een gegeven moment de oplossing slechts deels compleet is, is het moeilijk te weten welke opvolger het kan zijn. Hierdoor is een hybride methode gebruikt. Het algoritme legt wel de restrictie op, maar enkel binnen de richting (zie de methode exCleaner in de implementatie).




Figuur 1: Een voorbeeld van de representatie van een opgelost probleem. $E$ is in het rood afgebeeld, $A$ in het zwart.

## 1.2 Bidir

Bidir is helemaal geïmplementeerd zoals beschreven in het artikel van Dell'Amico en Trubian. In de pseudocode waren enkele stappen slechts vaag of helemaal niet beschreven, hierbij een uitleg hoe deze aangpakt zijn.

Ten eerste wordt een operatie $s$ gekozen met de priority rule. Dit wordt gedaan als volgt: elke node die mag gescheduled worden (en dus in $S$ zit) krijgt een estimated cost, berekend met:

$$est(i) = r_i + d_i + max(d_{SJ[i]} + t_{SJ[i]}, \max_{j \in \bar{V}}(d_j + t_j))$$

waarbij $\bar{V} = \{j \in V \setminus (L \cup \{i\}) : \mu_i = \mu_j\}$, of in woorden: j zijn de operaties die op dezelfde machine als i draaien en nog niet gescheduled zijn. Gegeven de kosten van de kandidaten, wordt de er gekozen via een cardinality-based semi-greedy heuristic met parameter $c$. Dit komt erop neer de $c$ nodes met laagste kost te kiezen en hieruit een willekeurige node terug te geven.

Vervolgens wordt de gekozen operatie $s$ gescheduled op zijn machine. In de representatie is dit heel makkelijk: de machine-bogen uit $s$ moeten gewoon directioneel gemaakt worden. Verder worden ook extra bogen verwijderd zoals besproken in de representatie, wat de efficientie verhoogd.

Ten slotte moeten de waarden voor $r_i$ upgedated worden. In eerste instantie zijn dit dus de nodes uit $SM[s]$ waarnaar een nieuwe boog getrokken is, maar ook hun opvolgers moeten mogelijks upgedated worden. Het algoritme gaat als volgt: eerst worden de opvolgers van $s$ bepaald (dit zijn alle nodes $w$ waarvoor er een pad van $i$ naar $w$ bestaat). Het is makkelijk in te zien dat als node $w$ geen opvolger is van $i$ dat deze geen nieuwe waarde $r_w$ kan krijgen door een verandering van $r_{SM[s]}$. Herinner immers dat $r_w$ het langste pad uit van de start node naar $w$ uitdrukt, en geen enkele van die paden kan $s$ bevatten. Vervolgens wordt een topologische ordening van de opvolgers bepaald. Hierbij worden de nodes geordend zodat een opvolger van een node altijd later voorkomt dan de node zelf. Deze topologische sortering wordt gerealiseerd door een depth-first search. Nu kunnen we $r_i$ updaten door deze lijst af te lopen en voor elke node het maximum te nemen van $r_{PJ[i]} + D_{PJ[i]}$ en $r_{PM[i]} + D_{PM[i]}$. Merk op dat we enkel kunnen garanderen dat een node upgedated mag worden aagezien al zijn voorgangers zijn upgedated, wat gegarandeerd is door de topologische sortering.

Men kan nagaan dat deze manier van werken consistent is met de manier waarop de estimations worden berekend.

Hier werd enkel het voorwaartse deel van het bidirectioneel algoritme beschreven, maar het achterwaartse deel werkt compleet analoog.

### 1.3 Neighbourhoods

Voor NA te bepalen moet het kritische pad worden afgegaan, waarna de estim_NA uit het artikel kan geïmplementeerd worden. Om het kritieke pad

te achterhalen. Word een er over de graaf gelopen, en steeds de node gekozen waarvoor $r_i + D_i + t_i$ maximaal is (want deze moeten natuurlijk per defenitie op het langste pad liggen). Als er meerdere nodes maximaal zijn, wordt de node met de kleinste $r_i$. Hierdoor kunnen geen shortcuts genomen worden.

In het speciale geval dat een graaf meerdere kritieke paden heeft, zal dit algoritme ook functioneren. De neighborhood zal wel beperkt worden tot een enkel kritiek pad. Zelfs bij grote instanties, blijft dit effect marginaal.[2] We kiezen er dus voor dit te verwaarlozen in het voordeel van de efficiëntie.

RNA berekenen uit NA is triviaal. NB implementeren is vrij analoog aan NA. Door de gekozen representatie van $E$ is het niet moeilijk alle blocks te vinden door het kritieke pad af te gaan en te kijken welke nodes elkaars directe machine opvolgers/voorgangers zijn. Nu kan afgegaan worden welke moves feasible zijn met de test uit het artikel: de move van $(b', x, b'')$ naar $(x, b', b'')$ is feasible als:

$$\forall k \in b' : C_{SJ(k)} \leq r_{PJ(x)}$$

Voor de move $(b', x, b'')$ naar $(b', b'', x)$ is de voorwoorde analoog:

$$\forall k \in b'' : r_{PJ(k)} \geq C_{SJ(x)}$$

### 1.4 Tabu Strategies

De tabu strategies werden geïmplementeerd zoals vermeld in het artikel. De enige onduidelijkheid bevond zich bij de keuze van de witness arc. Hierbij koos ik de eerste boog van een move. Echter bleek de impact van deze keuze tegenover bv. de laatste boog nihil.

De parameters voor het tabu search algoritme zijn genomen zoals in het artikel, buiten $MaxIter$ die het aantal restarts begrensd. Deze is lager gehouden dan in het artikel voor praktische redenen.

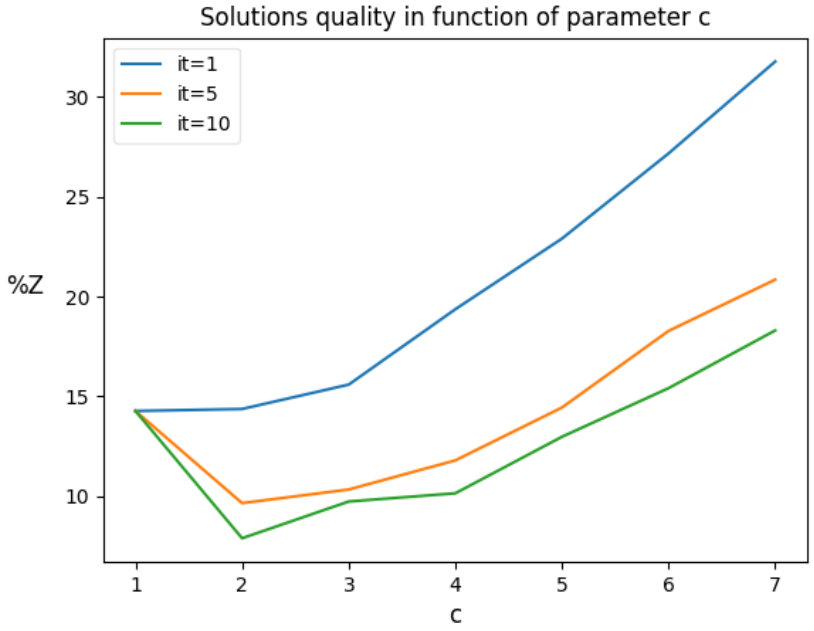
## 2 Resultaten

### 2.1 Bidir

Hierop volgen de resultaten voor de 40 lawrence instanties, waarop 10 maal het bidir algoritme is gedraaid. Het resultaat is weergegeven in tabel 1. Voor $c$ is de waarde 2 gekozen. De gemiddelde $\Delta$Z% voor alle instanties bedraagt 8,328%. Ter vergelijking: dit is 10,333% in het artikel. We kunnen dus besluiten dat Bidir goed presteert.

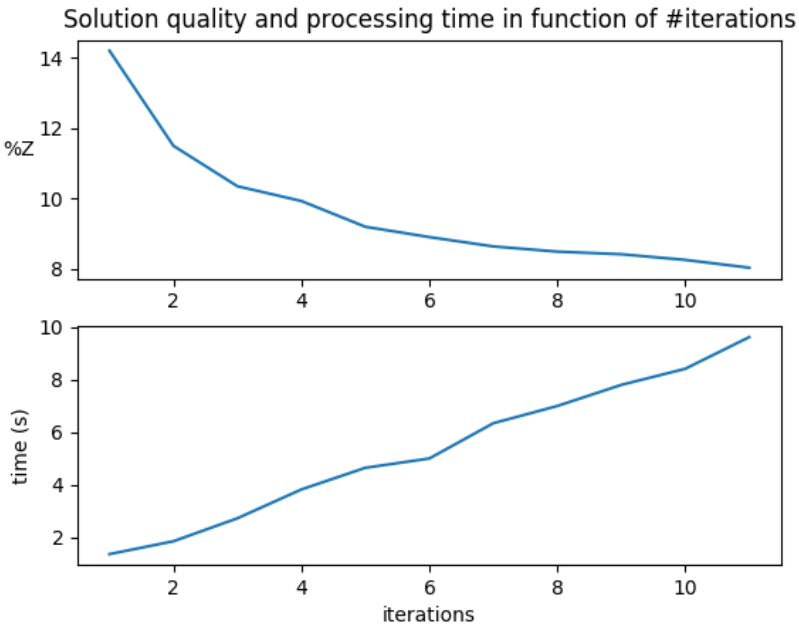
[2] Een testgeval op LA36 (standaard parameters): gemiddeld waren er 33,6484 nodes op alle kritieke paden. Als we altijd juist een enkelvoudig kritiek pad in beschouwing namen, waren dat er 32,9046.

Verder nog een korte analyse van de keuze van $c$ en de hoeveelheid iteraties. In figuur 2.1 werd de kwaliteit van de oplossing[3] geplot voor verschillende waarden van $c$. Hieruit kunnen we concluderen dat voor eender welke hoeveelheid iteraties (hoger dan n), $c = 2$ een goede keuze is.




We kijken ook naar de kwaliteit van de oplossingen in functie van de hoeveelheid iteraties (met $c = 2$) in figuur 2.1. Zoals verwacht verbetert de oplossingen naarmate er meer iteraties gedaan worden, en verzwakt dit effect voor grotere hoeveelheden. Hier moet men de afweging doen tegenover de hoeveelheid rekentijd, die natuurlijk lineair stijgt met het aantal iteraties.

[3]Deze werd berekend als het gemiddelde van de percentages dat de oplossing hoger lag dan het optimum, over alle lawrence instanties.


Solution quality and processing time in function of #iterations
%Z
14
12
10
8
2
4
6
8
10
time (s)
10
8
6
4
2
2
4
6
8
10
iterations

Tabel 1: Resultaten van Bidir

| Problem | n | m | Opt | $Z_{best}$ | $\Delta$%Z | $T_{avg}$ |
|---|---|---|---|---|---|---|
| LA01 | 10 | 5 | 666 | 690 | 3.60 | 0.0273 |
| LA02 | 10 | 5 | 655 | 713 | 8.85 | 0.0162 |
| LA03 | 10 | 5 | 597 | 662 | 10.89 | 0.0126 |
| LA04 | 10 | 5 | 590 | 657 | 11.36 | 0.0083 |
| LA05 | 10 | 5 | 593 | 593 | 0.00 | 0.0081 |
| LA06 | 15 | 5 | 926 | 926 | 0.00 | 0.0102 |
| LA07 | 15 | 5 | 890 | 920 | 3.37 | 0.0131 |
| LA08 | 15 | 5 | 863 | 929 | 7.65 | 0.0096 |
| LA09 | 15 | 5 | 951 | 959 | 0.84 | 0.0096 |
| LA10 | 15 | 5 | 958 | 958 | 0.00 | 0.0088 |
| LA11 | 20 | 5 | 1222 | 1228 | 0.49 | 0.0169 |
| LA12 | 20 | 5 | 1039 | 1051 | 1.15 | 0.0118 |
| LA13 | 20 | 5 | 1150 | 1163 | 1.13 | 0.0100 |
| LA14 | 20 | 5 | 1292 | 1292 | 0.00 | 0.0091 |
| LA15 | 20 | 5 | 1207 | 1243 | 2.98 | 0.0088 |
| LA16 | 10 | 10 | 945 | 1057 | 11.85 | 0.0203 |
| LA17 | 10 | 10 | 784 | 824 | 5.10 | 0.0136 |
| LA18 | 10 | 10 | 848 | 936 | 10.38 | 0.0141 |
| LA19 | 10 | 10 | 842 | 941 | 11.76 | 0.0122 |
| LA20 | 10 | 10 | 902 | 1020 | 13.08 | 0.0144 |
| LA21 | 15 | 10 | 1046 | 1214 | 16.06 | 0.0189 |
| LA22 | 15 | 10 | 927 | 1055 | 13.81 | 0.0203 |
| LA23 | 15 | 10 | 1032 | 1063 | 3.00 | 0.0161 |
| LA24 | 15 | 10 | 935 | 1065 | 13.90 | 0.0189 |
| LA25 | 15 | 10 | 977 | 1142 | 16.89 | 0.0178 |
| LA26 | 20 | 10 | 1218 | 1377 | 13.05 | 0.0286 |
| LA27 | 20 | 10 | 1235 | 1412 | 14.33 | 0.0276 |
| LA28 | 20 | 10 | 1216 | 1415 | 16.37 | 0.0289 |
| LA29 | 20 | 10 | 1152 | 1387 | 20.40 | 0.0273 |
| LA30 | 20 | 10 | 1355 | 1491 | 10.04 | 0.0268 |
| LA31 | 30 | 10 | 1784 | 1847 | 3.53 | 0.0679 |
| LA32 | 30 | 10 | 1850 | 1882 | 1.73 | 0.0664 |
| LA33 | 30 | 10 | 1719 | 1746 | 1.57 | 0.0680 |
| LA34 | 30 | 10 | 1721 | 1837 | 6.74 | 0.0651 |
| LA35 | 30 | 10 | 1888 | 1982 | 4.98 | 0.0707 |
| LA36 | 15 | 15 | 1268 | 1440 | 13.56 | 0.0471 |
| LA37 | 15 | 15 | 1397 | 1608 | 15.10 | 0.0479 |
| LA38 | 15 | 15 | 1196 | 1401 | 17.14 | 0.0418 |
| LA39 | 15 | 15 | 1233 | 1427 | 15.73 | 0.0419 |
| LA40 | 15 | 15 | 1233 | 1365 | 10.71 | 0.0358 |

## 2.2 Tabu Search

Hierbij volgen de resultaten van het tabu search algoritme op de 40 lawrence instanties. De resultaten zien we in tabel 2. De gemiddelde $\Delta$Z% voor alle instanties bedraagt 1,563%. Ter vergelijking: dit is 0,335% in het artikel. Hier zien we dus dat de performantie in de buurt komt van die in het artikel, maar hem niet evenaart.

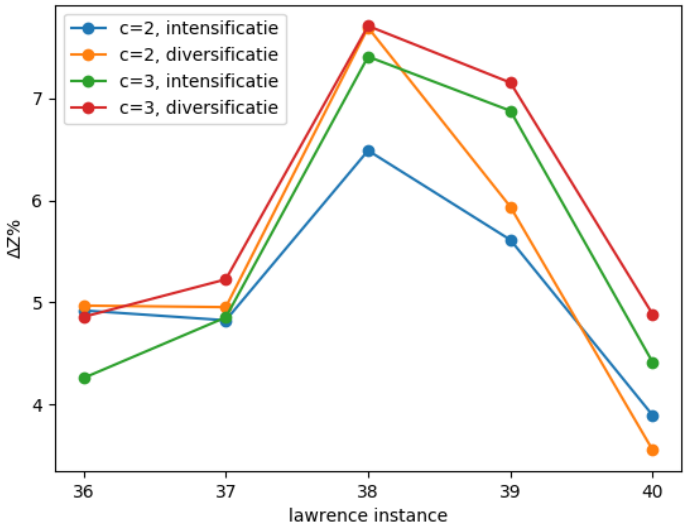
Een eerste keuze betreft het kiezen van de startwaarde. Immers kan men de beste resultaat van bidir steeds opnieuw gebruiken in de verschillende tabu search iteraties. Ofwel kan men verschillende startoplossingen gebruiken. Merk op dat dit een intensificatie (steeds oplossingruimte bij de beste oplossing doorzoeken) versus diversificatie (de oplossingsruimte vanaf verschillende oplossingen doorzoeken) probleem is.

Om dit te onderbouwen kijken we naar figuur 2. Hierin zien we de kwaliteit van de laatste vijf lawrence instanties, bekeken voor de twee strategiën alsook voor twee waarden van $c$. Hieruit zien we dat op het eerste zicht de intensificatie strategie beste resultaten levert. De conclusie dat $c = 2$ beter is dan de $c = 3$ gebruikt in het artikel, blijft dus overeind.

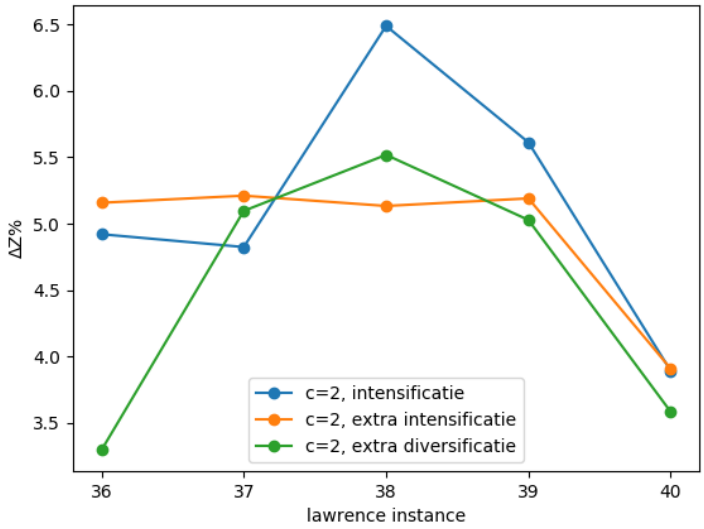
De intensificatie heeft het verdere voordeel dat men de hoeveelheden iteraties op bidir hoger kan kiezen dan die op tabu search. Dit is nuttig omdat uit experimenten blijkt dat bidir redelijk lang sneller verbeterd dan tabu search. Tenslotte bekijken we ook een hybride aanpak: ook hier draaien we bidir meer dan tabu search. Maar we gebruiken niet altijd het optimum zoals hiervoor. De beste resultaten iteraties worden gebruikt zodat er ook enige diversificatie is. We zien het resultaat van deze twee strategiën in figuur 3. In deze figuur zien we het resultaat van de laatste vijf lawrence instanties met $c = 2$, 5 tabu iteraties en 25 bidir iteraties.

We kunnen besluiten dat de tweede methode beter presteert, als we in overweging nemen dat voor hogere waarden voor $MaxIter$ die we in de praktijk gebruiken de betere gediversifieerde methode hiervan meer voordeel heeft. We gebruiken dus in het vervolg de hybride methode.

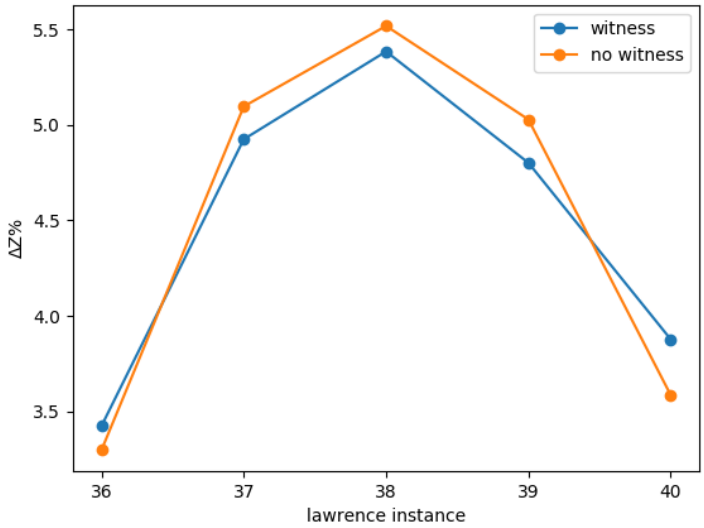
Verder analyseren we de invloed van het witness mechanisme. We gebruiken dezelfde methodiek om tot figuur 4 te bekomen. Hieruit zien we dat het gebruik van witness arcs wel nuttig blijkt, maar de invloed niet groot is.




Figuur 2: De oplossingkwaliteit van de laatste lawrence instanties, bekeken voor verschillende $c$'s en verschillende initiële strategiën. Alle parameterse zijn de standaardwaarden, buiten dat er geen witness gebruikt wordt en dat $MaxIter = 2$ is gehouden (dit uit praktische redenen).




Figuur 3: De oplossingkwaliteit van de laatste lawrence instanties, bekeken voor de twee nieuwe aanpakken. Ook de beste methode figuur 2 is opgenomen ter vergelijking. Zelfde parameters.




Figuur 4: De oplossingkwaliteit van de laatste lawrence instanties, bekeken met en zonder witness. Standaard parameters (behalve $MaxIter = 2$), hybride methode als startwaarde.

Tabel 2: Resultaten van Tabu Search

| Problem | n | m | Opt | $Z_{best}$ | %Z | $Z_{bidir}$ | $T_{avg}$ |
|---|---|---|---|---|---|---|---|
| LA01 | 10 | 5 | 666 | 666 | 0.00 | 701 | 13.1927 |
| LA02 | 10 | 5 | 655 | 655 | 0.00 | 724 | 11.4779 |
| LA03 | 10 | 5 | 597 | 606 | 1.51 | 672 | 11.9658 |
| LA04 | 10 | 5 | 590 | 590 | 0.00 | 691 | 11.3321 |
| LA05 | 10 | 5 | 593 | 593 | 0.00 | 593 | 10.4022 |
| LA06 | 15 | 5 | 926 | 926 | 0.00 | 976 | 19.6450 |
| LA07 | 15 | 5 | 890 | 890 | 0.00 | 950 | 19.9791 |
| LA08 | 15 | 5 | 863 | 863 | 0.00 | 926 | 19.4114 |
| LA09 | 15 | 5 | 951 | 951 | 0.00 | 975 | 19.2857 |
| LA10 | 15 | 5 | 958 | 958 | 0.00 | 985 | 18.9162 |
| LA11 | 20 | 5 | 1222 | 1222 | 0.00 | 1251 | 33.9669 |
| LA12 | 20 | 5 | 1039 | 1039 | 0.00 | 1058 | 29.6480 |
| LA13 | 20 | 5 | 1150 | 1150 | 0.00 | 1173 | 30.9583 |
| LA14 | 20 | 5 | 1292 | 1292 | 0.00 | 1296 | 30.8437 |
| LA15 | 20 | 5 | 1207 | 1207 | 0.00 | 1297 | 23.5149 |
| LA16 | 10 | 10 | 945 | 979 | 3.60 | 1051 | 17.3924 |
| LA17 | 10 | 10 | 784 | 785 | 0.13 | 855 | 15.9623 |
| LA18 | 10 | 10 | 848 | 862 | 1.65 | 969 | 16.5919 |
| LA19 | 10 | 10 | 842 | 852 | 1.19 | 958 | 15.4639 |
| LA20 | 10 | 10 | 902 | 924 | 2.44 | 1022 | 16.3647 |
| LA21 | 15 | 10 | 1046 | 1078 | 3.06 | 1186 | 27.6819 |
| LA22 | 15 | 10 | 927 | 963 | 3.88 | 1096 | 28.3073 |
| LA23 | 15 | 10 | 1032 | 1032 | 0.00 | 1096 | 28.1947 |
| LA24 | 15 | 10 | 935 | 974 | 4.17 | 1077 | 26.5499 |
| LA25 | 15 | 10 | 977 | 1009 | 3.28 | 1175 | 28.0792 |
| LA26 | 20 | 10 | 1218 | 1234 | 1.31 | 1397 | 41.9800 |
| LA27 | 20 | 10 | 1235 | 1292 | 4.62 | 1446 | 43.0297 |
| LA28 | 20 | 10 | 1216 | 1265 | 4.03 | 1406 | 41.9135 |
| LA29 | 20 | 10 | 1152 | 1253 | 8.77 | 1413 | 52.5106 |
| LA30 | 20 | 10 | 1355 | 1355 | 0.00 | 1514 | 54.3250 |
| LA31 | 30 | 10 | 1784 | 1784 | 0.00 | 1872 | 105.8964 |
| LA32 | 30 | 10 | 1850 | 1850 | 0.00 | 1923 | 110.5611 |
| LA33 | 30 | 10 | 1719 | 1719 | 0.00 | 1793 | 107.3949 |
| LA34 | 30 | 10 | 1721 | 1721 | 0.00 | 1841 | 78.5605 |
| LA35 | 30 | 10 | 1888 | 1888 | 0.00 | 1985 | 106.2617 |
| LA36 | 15 | 15 | 1268 | 1297 | 2.29 | 1504 | 54.0756 |
| LA37 | 15 | 15 | 1397 | 1438 | 2.93 | 1655 | 50.5369 |
| LA38 | 15 | 15 | 1196 | 1266 | 5.85 | 1381 | 51.9108 |
| LA39 | 15 | 15 | 1233 | 1304 | 5.76 | 1431 | 51.6280 |
| LA40 | 15 | 15 | 1233 | 1258 | 2.03 | 1371 | 54.8906 |

## Referenties

[1] Mauro Dell'Amico and Marco Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations research*, 41(3):231–252, 1993.

[2] Peter JM Van Laarhoven, Emile HL Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125, 1992.