Effective Neighborhood Functions for the Flexible Job Shop Problem: Appendix*

Monaldo Mastrolilli, Luca Maria Gambardella
IDSIA - Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
C.so Elvezia 36, 6900 Lugano, Switzerland
{monaldo, luca}@idsia.ch, http://www.idsia.ch

February 14, 2000

1 Moves Evaluation

We call the k-insertion of v for which the estimated longest path is minimized, approximate optimal k-insertion. Over 1.7×10^6 trials and 172 different problems, we experimentally found out that an approximate optimal k-insertion is also an optimal k-insertion of v in 99% of the cases with a maximal deviation of 1%. Table 1 shows the average error for data set sdata, edata, rdata and vdata.

Data	Av(error)
sdata	0,03%
rdata	0,07%
edata	0,10%
vdata	0,08%

Table 1: Mean relative optimal insertion error on different data sets.

2 Computational results

The Nopt1-version of the search procedure described in Section 7 was implemented in C++ on a 266 MHz Pentium. The non-deterministic nature of the algorithm makes it necessary to carry out multiple runs on the same problem instance in order to obtain meaningful results. We ran our algorithm five times from different starting solutions and tested it on a large number of test problems

*Supported by Swiss National Science Foundation project 21-55778.98, "Resource Allocation and Scheduling in Flexible Manufacturing Systems".

from the literature. The results obtained with our procedure were then compared with results obtained with all procedures for which we could find results (in terms of makespan and CPU time) in the literature.

Five different sets of experiments were conducted. The data are taken from Brandimarte [4] Dauzère-Pérès and Paulli [6] Chambers and Barnes [3] and from Hurink et al. [10] respectively. The procedure was also tested on a number of classical job shop scheduling problems [1, 9].

The following notations are used.

- (LB,UB) denotes the optimum value if known, otherwise, the best lower and upper bound found to date;
- Av(C_0) stands for the average starting solution value out of five runs;
- C_{best} is the value of the best solution found using our procedure out of 5 runs;
- Av(C) is the average solution value over 5 runs;
- DevStd(C) is the standard deviation of the 5 solution values;
- Av(CPU) is the average computing time in seconds.

When a solution is equal to the related lower bound, it is optimal and is marked with an asterisk. CPU is the amount of CPU seconds used by the procedure and the acronym CI-CPU stands for computer-independent CPU times. These values were computed using the normalization coefficients of Dongarra [8] as interpreted by Vaessens et al. [14] and used also in Balas and Vazacopoulos [2] and must be interpreted with care.

In the remainder, the acronym TSopt1 denotes our procedure.

2.1 Test sample 1

The first test sample comes from Brandimarte [4]. The data were randomly generated using a uniform distribution between given limits. They consist of ten problems mk1-10 where the number of jobs ranges from 10 to 20, the number of machines ranges from 6 to 15, operations for each job range from 5 to 15 and the maximum number of equivalent machines per operation ranges from 3 to 6. We limited the number of iterations to 10^5. Table 3 compares the performance of TSopt1 with the best tabu search procedure of Brandimarte. Our procedure improved upon the best known results. A comparison of the computational effort for these runs is not possible since Brandimarte does not report computing times [4].

The results seem to support the hypothesis that simultaneous approaches, like ours, perform better than hierarchical ones.

2.2 Test sample 2

The second test sample comes from Dauzère-Pérès and Paulli [6]. The data consist of five sets of 18 test problems where the number of jobs ranges from 10 to 20, machines range from 5 to 10, operations for each job range from 5 to 25. The set of machines capable of performing an operation was constructed by letting a machine be in that set with a probability that ranges from 0.1 to 0.5 [6]. Table 4 shows the computational results on the first set of 18 problems. The lower bound values were computed in a simple way and hence they are generally not very close to the optimum. The best known upper bounds found up to now are reported by [6]. We limited the number of iterations to 4×10^5. Table 4 shows that our procedure improves the best known upper bound in 18 cases. Although the quality of the starting solution is lower than the one reported by [6] our tabu search procedure leads to substantial improvements. Comparing our average solution values and our average CI-CPU with the best known solutions 17 test problems yield better results, while in 1 case our average value is worse. Figure 1 makes a comparison between the CI-CPU times required by procedure DP [6] and TS opt1 average CI-CPU. Note that the CI-CPU axis has a logarithmic scale. On average, TS opt1 is 5.6 times faster than DP.

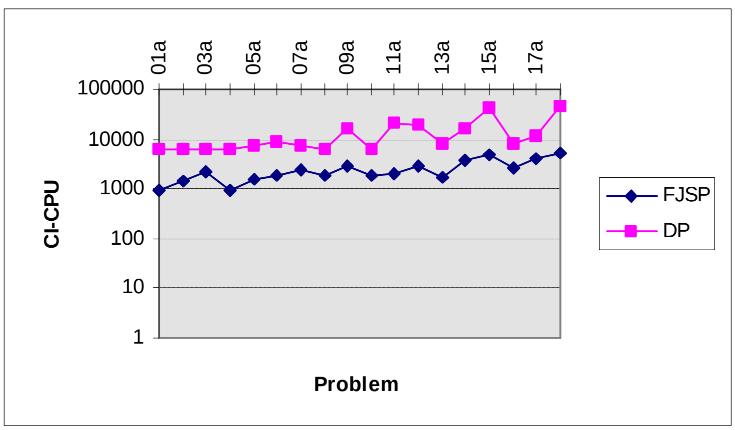

Figure 1: Comparison between FJSP and DP CI-CPU.

2.3 Test sample 3

The third test sample comes from Chambers and Barnes [3]. The data were constructed from three of the most challenging classical job shop problems (mt10, la24, la40) by replicating machines selected according to two simple criteria: the total processing time required by a machine and the cardinality of critical operations on a machine. The set consists of 21 test problems and the processing times for operations on replicated machines are assumed to be identical to the original. Table 5 shows the computational results. (Again, the lower bound values were computed in a very simple way and so they are generally not very close to the optimum).

We limited the number of iterations to 4×10^5. TSopt1 improves the best known upper bounds in 17 cases. Comparing our average solution values with the best known upper bounds we find that in 16 test problems we obtain better results, while in 4 cases our average solution values are worse. Figure 2 makes a comparison between the required CI-CPU times by procedure BC [3] and procedure TSopt1 average CI-CPU. On average TSopt1 is 21 times faster than BC.

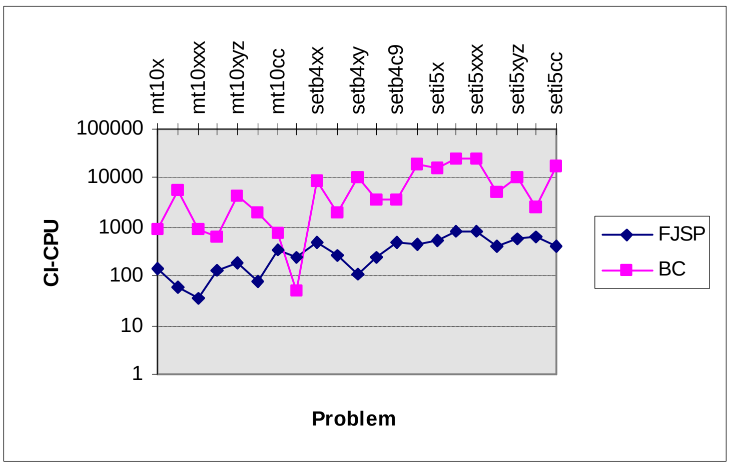

Figure 2: Comparison between FJSP and BC CI-CPU.

2.4 Test sample 4

The fourth test sample comes from Hurink et al. [10]. The problems are generated by modifying benchmark problems for the classical job shop problem. More specifically, they were obtained from three problems by Fisher and Thompson [9] (mt06, mt10, mt20) and 40 problems from Adams et al. [12] (la01–la40). Each set M_i is equal to the machine to which operation i is assigned in the original problem, plus any of the other machines with a given probability. Depending on this probability, Hurink et al. generated three sets of test problems: edata, rdata and vdata. The first set contains the problems with the least amount of flexibility, whereas the average size of M_i is equal to 2 in rdata and m=2 in vdata [10]. We limited the number of iterations to 10^5. Tables 6, 7 and 8 shows that TSopt1 outperforms the combined effort of all the other known procedures in 77 cases and it is never outperformed. Comparing our average solution values with the best known upper bounds, in 129 test problems we obtain 74 better results, while in 4 cases our average solution values are worse.

We compared TSopt1 with 3 approximation algorithms: MPMJSP [10] MMJSP [5] and DP [6]. A comparative overview of TSopt1 average solution and MPMJSP, MMJSP and DP, is given in Table 2 (B:E:W represents the number of examples for which TSopt1 average makespan is better (B), equal (E) or worse (W) than the best makespan found by MPMJSP, MMJSP and DP respectively; Total CI-CPU is the sum of the CI-CPU used to compute all the solutions except for procedure TSopt1 where it is the sum of the average CI-CPU). On average, our Av(CI-CPU) is 15, 32 and 25 times smaller than the corresponding CI-CPU of MPMJSP, MMJSP and DP respectively.

Figure 3 compares the required CI-CPU times to solve problem set vdata using the procedures MPMJSP, MMJSP and DP and the average CI-CPU for our procedure (a similar graph could be obtained for data set edata and rdata).

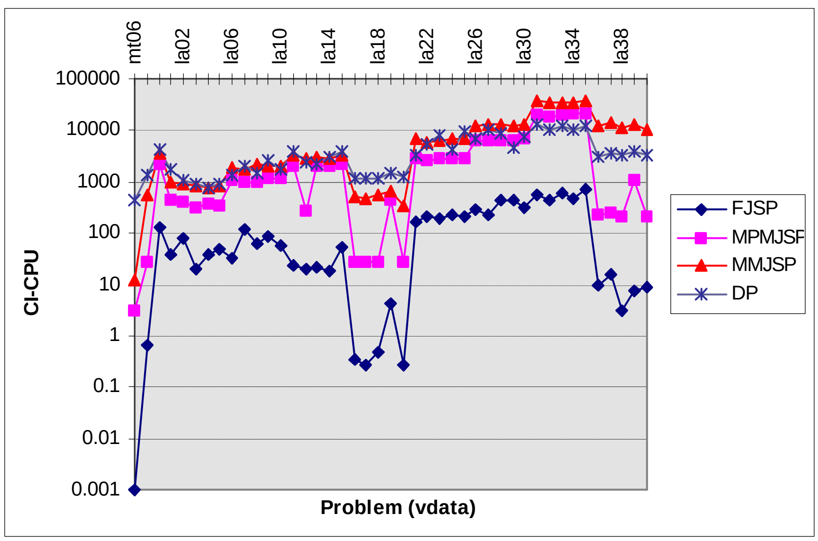

Figure 3: Comparisons between FJSP, MPMJSP, MMJSP and DP CI-CPU.

2.5 Test sample 5

The test problems in this sample are all classical job shop scheduling problems. Since the classical job-shop scheduling problem is a special case of the FJS problem, our tabu search procedure can also be used to solve these problems. We tested our procedure on the original 43 test problems (sdata) [?, 9] used to generate test sample 3 and 4. For this set of problem we ran TSopt1 10 times from different starting solutions and limited the number of iterations to 10^6. Table 9 shows that our tabu-search algorithm finds an optimal solution for 38 of 43 problems. With regard to the 5 remaining problems, the distance from the best lower bound (or optimum, if known) is smaller than 0.1%. It is worth noting that our procedure is able to find the optimal solution to the notorious 10×10 problem (mt10) by Fisher and Thompson in just 0.01 seconds starting from a solution with makespan equal to 1686. The search procedure for the classical job shop scheduling problem described by Balas and Vazacopoulos [2] is comparable to (but considerably faster than) TSopt1. However, our tabu search procedure is quite simple and it is an obvious topic for further computational experiments to analyze how more sophisticated search strategies could improve the solution quality (and perhaps also the computational time).

In order to make a more detailed comparison on problems for which it is meaningful, we selected the 13 most difficult instances, according to Balas and Vazacopoulos [2] among the 43 problems. Table 10 contains comparison related to the best makespans found by TSopt1, against those from DT (the tabu search procedure of Dell'Amico and Trubian [7]), NS (the tabu search procedure of Nowicki and Smutnicki [13]) and BV (the procedure SB-GLS2 of Balas and Vazacopoulos [2]). Our results are close to the ones obtained by the best search procedures to solve the job shop problems, although the computational time used by TSopt1 is considerably greater. Figure 4 shows a comparison between the CI-CPU of TSopt1, DT and BV. In this case the CI-CPU for TSopt1 is the total amount of CI-CPU to compute 10 runs (for NS and BC the total times are not reported).

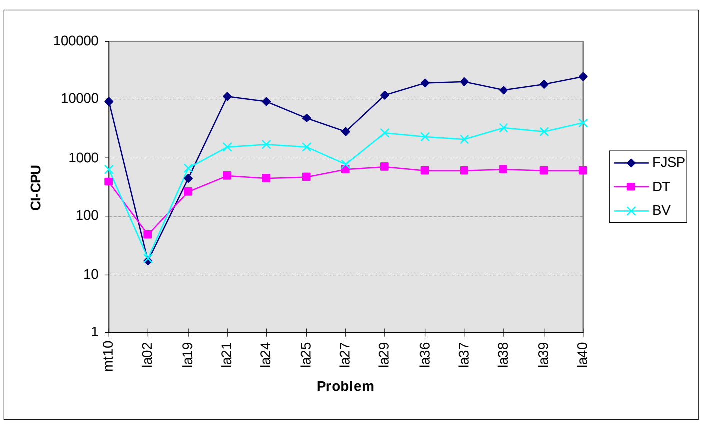

Figure 4: CI-CPU time for 13 hard problems.

References

[1] Applegate, D., and Cook, W. (1991) "A computational study of the job shop scheduling problem", ORSA Journal on Computing, 3, No. 2, Spring 1991.

[2] Balas E., Vazacopoulos A. (1998), "Guided Local Search with Shifting Bottleneck for Job Shop Scheduling", Management Science 44, 262-275.

[3] Barnes, J. W. and Chambers J. B. (1996), "Flexible Job Shop Scheduling by Tabu Search", Graduate Program in Operations Research and Industrial Engineering, The University of Texas at Austin, Technical Report Series, ORP96-09, http://www.cs.utexas.edu/users/jbc/.

[4] Brandimarte, P., (1993), "Routing and scheduling in a flexible job shop by tabu search", Annals of Operations Research 22, pp 158-183.

[5] Brucker, P., Neyer, J. (1998): "Tabu-search for the multi-mode job-shop problem" Osnabruecker Schriften zur Mathematik P 185; University of Osnabrück, OR Spektrum 20, 21-28.

[6] Dauzère-Pérès S., Paulli, J. (1997): "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search", Annals of Operations Research 70, 281-306.

[7] Dell'Amico M., Trubian M. (1993), "Applying tabu-search to the job shop scheduling problem", Annals of Operations Research 22, 15-24.

[8] Dongarra, j.j. (1998), Performance of various computers using standard linear equations software, Computer Science Department, University of Tennessee, Knoxville, Tennessee.

[9] H. Fisher and G.L. Thompson (1963), "Probabilistic learning combinations of local job shop scheduling rules", in: Industrial Scheduling, J.F. Muth and G.L. Thompson, eds., Prentice Hall, Englewood Cliffs, pp. 225-251.

[10] Hurink, E., Jurisch, B., Thole, M. (1994), "Tabu search for the job shop scheduling problem with multi-purpose machine", Operations Research Spektrum 15, 205-215.

[11] Van Laarhoven P.J.M., Aarts E.H.L., Lenstra J.K. (1992), "Job Shop Scheduling by simulated annealing", Operations Research 40, 113-125.

[12] Lawrence S. (1984), Supplement to "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques", GSIA, Carnegie Mellon University, Pittsburgh, PA.

[13] Nowicki, E., Smutnicki, C. (1996), "A fast taboo search algorithm for the job shop problem", Management Science 42, 797-813.

[14] Vaessens, Aarts, Lenstra (1996), "Job shop scheduling by local search", INFORMS Journal on computing, vol.8 n.3.

3 Tables

Algorithm	Total CI-CPU	B:E:W
MPM JSP	258750	102:27:0
MM JSP	531056	107:22:0
DP	428409.6	77:47:5
TS opt1	16568.36	

Table 2: Summary results for the Hurink et al. data.

Problem	(LB,UB)	C_{best}	Av(C)	DevStd(C)	Av(CPU)	Av(C_0)
Mk01	(36,42)	40	40.00	0.00	0.01	102.40
Mk02	(24,32)	26	26.00	0.00	0.73	93.20
Mk03	(204,211)	*204	204.00	0.00	0.01	511.60
Mk04	(48,81)	60	60.00	0.00	0.08	147.20
Mk05	(168,186)	173	173.00	0.00	0.96	310.80
Mk06	(33,86)	58	58.40	0.55	3.26	255.40
Mk07	(133,157)	144	147.00	3.00	8.91	361.80
Mk08	523	*523	523.00	0.00	0.02	867.40
Mk09	(299,369)	307	307.00	0.00	0.15	807.80
Mk10	(165,296)	198	199.20	0.84	7.69	730.60

Table 3: Results on Brandimarte's data.

Problem	(LB,UB)	C_{best}	Av(C)	DevStd(C)	Av(CPU)	Av(C_0)
01a	(2505, 2530)	2518	2528.00	6.89	28.17	4604.00
02a	(2228, 2244)	2231	2234.00	2.45	42.48	4762.40
03a	(2228, 2235)	2229	2229.60	0.55	64.44	4855.40
04a	(2503, 2565)	*2503	2516.20	10.38	27.66	4834.80
05a	(2189, 2229)	2216	2220.00	2.74	45.52	4503.40
06a	(2162, 2216)	2203	2206.40	2.41	54.08	4572.60
07a	(2187, 2408)	2283	2297.60	10.74	68.23	4935.80
08a	(2061, 2093)	2069	2071.40	1.52	53.32	4792.80
09a	(2061, 2074)	2066	2067.40	0.89	81.19	4961.80
10a	(2178, 2362)	2291	2305.60	12.14	56.11	5211.20
11a	(2017, 2078)	2063	2065.60	2.88	61.48	4960.60
12a	(1969, 2047)	2034	2038.00	3.94	81.93	4884.40
13a	(2161, 2302)	2260	2266.20	4.32	48.54	5564.60
14a	(2161, 2183)	2167	2168.00	1.00	110.68	5901.20
15a	(2161, 2171)	2167	2167.20	0.45	137.53	5766.00
16a	(2148, 2301)	2255	2258.80	5.26	74.66	5586.20
17a	(2088, 2169)	2141	2144.00	2.55	120.83	5458.40
18a	(2057, 2139)	2137	2140.20	2.77	149.49	5836.40

Table 4: Results on the data from Dauzere-Peres and Paulli.

Problem	(LB,UB)	C_{best}	Av(C)	DevStd(C)	Av(CPU)	Av(C_0)
mt10x	(655, 929)	918	918.00	0.00	4.31	1822.60
mt10xx	(655,929)	918	918.00	0.00	1.73	1642.60
mt10xxx	(655,936)	918	918.00	0.00	1.10	1723.60
mt10xy	(655,913)	906	906.00	0.00	4.02	1915.00
mt10xyz	(655,849)	847	850.80	2.49	5.50	1697.60
mt10c1	(655,927)	928	928.00	0.00	2.33	1795.60
mt10cc	(655,914)	910	910.00	0.00	10.04	1687.20
setb4x	(846,937)	925	925.00	0.00	7.45	2079.00
setb4xx	(846,930)	925	926.40	2.19	14.87	2031.80
setb4xxx	(846,925)	925	925.00	0.00	7.99	2186.00
setb4xy	(845,924)	916	916.00	0.00	3.15	1953.20
setb4xyz	(838,914)	905	908.20	1.79	7.35	2023.80
setb4c9	(857,924)	919	919.20	0.45	14.02	2111.80
setb4cc	(857,909)	909	911.60	5.81	12.95	2052.80
seti5x	(955,1218)	1201	1203.60	1.52	15.85	2710.80
seti5xx	(955,1204)	1199	1200.60	2.30	23.64	2728.40
seti5xxx	(955,1213)	1197	1198.40	0.89	23.51	2787.20
seti5xy	(955,1148)	1136	1136.40	0.55	11.91	2705.80
seti5xyz	(955,1127)	1125	1126.60	1.14	17.13	2391.80
seti5c12	(1027,1185)	1174	1174.20	0.45	19.49	2774.20
seti5cc	(955,1136)	1136	1136.40	0.55	11.91	2705.80

Table 5: Results on the data from Barnes and Chamber

Problem	(LB,UB)	C_{best}	Av(C)	DevStd(C)	Av(CPU)	Av(C_0)
m06	55	*55	55.00	0.00	0.00	101.40
m10	(871, 873)	*871	873.00	1.41	1.61	1812.00
m20	(1088, 1106)	*1088	1088.80	0.84	3.52	2034.20
la01	609	*609	609.00	0.00	0.01	1110.20
la02	655	*655	655.00	0.00	0.04	1181.40
la03	(550, 554)	*550	550.00	0.00	1.00	1086.80
la04	568	*568	568.00	0.00	0.36	1131.00
la05	503	*503	503.00	0.00	0.01	994.80
la06	833	*833	833.00	0.00	0.00	1540.40
la07	(762, 765)	*762	762.00	0.00	0.35	1438.40
la08	845	*845	845.00	0.00	0.02	1431.40
la09	878	*878	878.00	0.00	0.04	1493.20
la10	866	*866	866.00	0.00	0.01	1478.40
la11	(1087, 1103)	1103	1103.00	0.00	1.91	1935.00
la12	960	*960	960.00	0.00	0.02	1773.80
la13	1053	*1053	1053.00	0.00	0.02	1997.20
la14	1123	*1123	1123.00	0.00	0.03	1915.00
la15	1111	*1111	1111.00	0.00	0.30	2018.60
la16	(892, 915)	*892	892.00	0.00	0.25	1728.80
la17	707	*707	707.00	0.00	0.58	1570.00
la18	(842, 843)	*842	842.00	0.00	0.79	1624.60
la19	796	*796	796.00	0.00	1.53	1624.80
la20	(857, 864)	*857	857.00	0.00	0.88	1777.80
la21	(895, 1046)	1017	1023.20	5.26	2.83	2220.60
la22	(832, 890)	882	883.00	1.41	4.29	2123.40
la23	(950, 953)	*950	950.00	0.00	2.97	2156.60
la24	(881, 918)	909	911.60	2.61	3.88	2113.20
la25	(894, 955)	941	945.00	2.83	1.76	2217.60
la26	(1089, 1138)	1125	1127.00	3.39	5.48	2536.60
la27	(1181, 1215)	1186	1188.80	3.27	9.25	2760.80
la28	(1116, 1165)	1149	1149.00	0.00	3.44	2654.00
la29	(1058, 1157)	1118	1120.60	2.88	5.47	2564.20
la30	(1147, 1225)	1204	1213.20	6.46	9.22	2882.60
la31	(1523, 1556)	1539	1540.60	0.89	9.58	3390.60
la32	1698	*1698	1698.00	0.00	1.85	3497.20
la33	1547	*1547	1547.00	0.00	1.40	3640.20
la34	(1592, 1613)	1599	1599.20	0.45	9.35	3570.40
la35	1736	*1736	1736.00	0.00	0.41	3457.20
la36	(1006, 1171)	1162	1163.20	1.79	8.08	2891.40
la37	(1355, 1418)	1397	1397.00	0.00	3.48	3003.00
la38	(1019, 1172)	1144	1146.60	2.61	6.90	2719.20
la39	(1151, 1207)	1184	1185.60	1.34	8.68	2811.80
la40	(1034, 1150)	1150	1151.60	2.61	7.78	2597.60

Table 6 Results on the set edata from Hurink et al.

Problem	(LB,UB)	C_{best}	Av(C)	DevStd(C)	Av(CPU)	Av(C_0)
m06	47	*47	47.00	0.00	0.00	82.80
m10	(679, 686)	686	686.00	0.00	2.71	1572.40
m20	(1022, 1024)	*1022	1022.60	0.55	3.53	1907.40
la01	(570, 574)	571	571.80	0.84	1.97	1232.40
la02	(529, 532)	530	530.60	0.55	1.31	1133.80
la03	(477, 479)	478	478.20	0.45	1.36	1118.60
la04	(502, 504)	*502	503.00	0.71	0.62	1013.00
la05	(457, 458)	*457	457.60	0.55	1.78	889.00
la06	(799, 800)	*799	799.40	0.55	2.99	1588.60
la07	(749, 750)	750	750.00	0.00	1.13	1401.80
la08	(765, 767)	*765	765.80	0.45	0.35	1494.00
la09	(853, 854)	*853	853.40	0.55	2.29	1526.00
la10	(804, 805)	*804	804.60	0.55	1.32	1679.60
la11	(1071, 1072)	*1071	1071.00	0.00	2.56	1863.80
la12	936	*936	936.00	0.00	0.08	1638.40
la13	1038	*1038	1038.00	0.00	0.90	1998.60
la14	1070	*1070	1070.00	0.00	0.28	1949.60
la15	(1089, 1090)	1090	1090.00	0.00	1.76	1858.00
la16	717	*717	717.00	0.00	0.07	1780.80
la17	646	*646	646.00	0.00	0.03	1384.40
la18	(666, 669)	*666	666.00	0.00	1.79	1625.60
la19	(647, 703)	700	701.20	1.64	1.90	1732.40
la20	756	*756	756.00	0.00	0.03	1779.00
la21	(808, 846)	835	841.00	4.24	7.81	2436.60
la22	(737, 772)	760	763.60	2.07	5.14	2057.20
la23	(816, 853)	842	845.20	2.95	6.50	2230.20
la24	(775, 820)	808	813.80	3.35	4.06	2084.60
la25	(752, 802)	791	794.40	2.19	3.38	2074.80
la26	(1056, 1070)	1061	1063.80	1.64	7.69	2747.00
la27	(1085, 1100)	1091	1092.60	1.34	7.47	2619.20
la28	(1075, 1085)	1080	1081.60	1.14	7.54	2689.60
la29	(993, 1004)	998	998.60	1.34	4.03	2716.00
la30	(1068, 1089)	1078	1080.80	1.79	7.78	2679.80
la31	(1520, 1528)	1521	1522.00	1.00	8.61	3454.20
la32	(1657, 1660)	1659	1659.80	1.10	12.67	3773.80
la33	(1497, 1501)	1499	1500.00	1.00	11.48	3436.40
la34	(1535, 1539)	1536	1536.20	0.45	7.28	3356.00
la35	(1549, 1555)	1550	1550.60	0.55	15.28	3556.80
la36	(1016, 1030)	1030	1031.20	1.10	4.90	2818.00
la37	(989, 1082)	1077	1080.60	2.70	9.52	2860.60
la38	(943, 989)	962	968.00	3.67	9.32	2839.60
la39	(966, 1024)	1024	1033.20	13.29	2.78	2639.00
la40	(955, 980)	970	974.00	3.39	6.11	2836.00

Table 7: Results on the set rdata from Hurink et al.

Problem	(LB,UB)	C_{best}	Av(C)	DevStd(C)	Av(CPU)	Av(C_0)
mt06	47	*47	47.00	0.00	0.00	98.00
mt10	655	*655	655.00	0.00	0.02	1593.00
mt20	1022	*1022	1022.00	0.00	3.79	1766.60
la01	(570, 572)	*570	570.80	0.45	1.08	1072.00
la02	529	*529	529.40	0.55	2.29	1158.00
la03	(477, 479)	*477	477.60	0.55	0.58	1099.60
la04	(502, 503)	*502	502.00	0.00	1.15	1093.60
la05	(457, 460)	*457	458.00	0.71	1.40	977.20
la06	799	*799	799.00	0.00	0.97	1519.00
la07	(749, 750)	*749	749.80	0.84	3.55	1466.60
la08	(765, 766)	*765	765.20	0.45	1.81	1518.20
la09	853	*853	853.00	0.00	2.59	1545.40
la10	(804, 805)	*804	804.00	0.00	1.66	1493.20
la11	1071	*1071	1071.00	0.00	0.68	1790.40
la12	936	*936	936.00	0.00	0.59	1697.60
la13	1038	*1038	1038.00	0.00	0.65	1941.80
la14	1070	*1070	1070.00	0.00	0.53	1960.00
la15	1089	*1089	1089.40	0.55	1.55	2053.00
la16	717	*717	717.00	0.00	0.01	1730.20
la17	646	*646	646.00	0.00	0.01	1557.40
la18	663	*663	663.00	0.00	0.01	1590.40
la19	617	*617	617.00	0.00	0.12	1660.00
la20	756	*756	756.00	0.00	0.01	1747.20
la21	(800, 814)	806	807.60	1.52	4.66	2136.40
la22	(733, 744)	739	739.80	0.84	6.42	2103.80
la23	(809, 818)	815	816.00	1.22	5.66	2170.40
la24	(773, 784)	777	779.00	1.58	6.79	2103.20
la25	(751, 757)	756	756.40	0.55	6.27	2084.00
la26	(1052, 1056)	1054	1054.60	0.55	8.55	2746.60
la27	(1084, 1087)	1085	1085.80	0.45	6.44	2743.40
la28	(1069, 1072)	1070	1070.40	0.55	12.73	2794.80
la29	(993, 995)	994	994.60	0.55	13.06	2542.00
la30	(1068, 1070)	1069	1070.00	0.71	9.60	2668.00
la31	(1520, 1521)	*1520	1520.00	0.00	16.92	3643.20
la32	(1657, 1658)	1658	1658.00	0.00	13.00	3706.00
la33	(1497, 1498)	*1497	1497.80	0.45	17.14	3312.80
la34	(1535, 1536)	*1535	1535.20	0.45	13.50	3454.60
la35	(1549, 1550)	*1549	1549.00	0.00	20.69	3633.20
la36	948	*948	948.00	0.00	0.28	2733.00
la37	986	*986	986.00	0.00	0.44	2834.00
la38	943	*943	943.00	0.00	0.09	2671.20
la39	922	*922	922.00	0.00	0.22	2798.60
la40	955	*955	955.00	0.00	0.26	2810.60

Table 8: Results on the set vdata from Hurink et al.

Problem	(LB,UB)	C_{best}	Av(C)	DevStd(C)	Av(CPU)	Av(C_0)
mt06	55	*55	55.00	0.00	0.00	99.80
mt10	930	*930	932.50	2.76	27.69	1938.70
mt20	1165	*1165	1166.70	5.38	2.01	2035.30
la01	666	*666	667.20	3.79	0.01	1193.40
la02	655	*655	656.10	3.48	0.05	1189.30
la03	597	*597	597.60	1.90	0.39	1073.20
la04	590	*590	590.00	0.00	0.05	1127.30
la05	593	*593	593.00	0.00	0.00	981.20
la06	926	*926	926.00	0.00	0.00	1541.40
la07	890	*890	890.00	0.00	0.02	1475.40
la08	863	*863	863.00	0.00	0.00	1395.70
la09	951	*951	951.00	0.00	0.00	1620.30
la10	958	*958	958.00	0.00	0.00	1576.90
la11	1222	*1222	1222.00	0.00	0.00	1918.30
la12	1039	*1039	1039.00	0.00	0.00	1741.20
la13	1150	*1150	1150.00	0.00	0.00	1942.60
la14	1292	*1292	1292.00	0.00	0.00	1968.80
la15	1207	*1207	1207.00	0.00	0.04	1982.70
la16	945	*945	945.00	0.00	4.01	1745.40
la17	784	*784	784.00	0.00	0.19	1520.20
la18	848	*848	848.00	0.00	0.45	1671.40
la19	842	*842	842.00	0.00	1.29	1755.00
la20	902	*902	902.00	0.00	0.49	1761.60
la21	1046	*1046	1047.30	1.16	33.63	2292.90
la22	927	*927	929.60	2.76	27.93	2182.60
la23	1032	*1032	1032.00	0.00	0.09	2174.90
la24	935	*935	938.00	1.41	26.59	2080.30
la25	977	*977	977.60	0.97	14.10	2123.20
la26	1218	*1218	1218.00	0.00	0.91	2687.00
la27	1235	*1235	1249.30	9.65	8.18	2721.00
la28	1216	*1216	1216.70	2.21	3.48	2591.60
la29	(1120, 1157)	1160	1167.30	3.20	35.38	2454.30
la30	1355	*1355	1355.00	0.00	0.68	2769.60
la31	1784	*1784	1784.00	0.00	0.14	3456.40
la32	1850	*1850	1850.00	0.00	0.11	3675.20
la33	1719	*1719	1719.00	0.00	0.17	3353.40
la34	1721	*1721	1721.00	0.00	0.50	3533.40
la35	1888	*1888	1889.00	3.16	0.17	3642.40
la36	1268	1269	1272.70	3.09	58.13	2813.30
la37	1397	*1397	1399.90	3.60	60.70	2911.50
la38	(1184, 1196)	1196	1201.60	2.32	41.53	2561.60
la39	1233	*1233	1237.50	2.32	53.09	2749.90
la40	1222	1228	1229.70	1.70	74.61	2764.70

Table 9: Results on the data by Lawrence, Fisher and Thompson.

Problem	BV	TS opt1	NS	DT	BC
mt10	930	930	930	935	935
la02	655	655	655	655	655
la19	842	842	842	842	843
la21	1046	1046	1047	1048	1053
la24	935	935	939	941	946
la25	977	977	977	979	988
la27	1235	1235	1236	1242	1256
la29	1164	1160	1160	1182	1194
la36	1268	1269	1268	1278	1278
la37	1397	1397	1407	1409	1418
la38	1196	1196	1196	1203	1211
la39	1233	1233	1233	1242	1237
la40	1224	1228	1229	1233	1239

Table 10: Makespan for 13 hard problems.