On Minimizing L_{max} for Large-Scale, Job Shop Scheduling Problems

Scott R. Schultz
Department of Mechanical and Industrial Engineering
Mercer University
Macon, GA 31207

Thom J. Hodgson
Department of Industrial Engineering
North Carolina State University
Raleigh, NC 27695-7906

Russell E. King
Department of Industrial Engineering
North Carolina State University
Raleigh, NC 27695-7906
Phone: (919) 515-5186
FAX: (919) 515-1543
king@eos.ncsu.edu
(*corresponding author*)

Kristin A. Thoney
Department of Textile & Apparel Technology & Management
North Carolina State University
Raleigh, NC 27695-8301

February 2004

Keywords: scheduling: lateness: job shop: benchmark: simulated annealing: performance bounds

Abstract: In a recent paper by Hodgson *et al.* (2000) a procedure is presented for solving the job shop scheduling problem of minimizing L_{max}. Their iterative-adaptive simulation-based procedure is shown to perform well on large-scale problems. However, there is potential for improvement in closing the gap between best known solutions and the lower bound. In this paper, a simulated annealing post processing procedure is presented and evaluated on large-scale problems. A new neighborhood structure for local searches in the job shop scheduling problem is developed. The procedure is also evaluated using benchmark problems and new upper bounds are established.

1. Introduction

Consider the classic job shop scheduling problem. A set of jobs is to be processed on a set of machines without preemption. Each job has a given technological route, i.e., a specified sequence of operations and required processing times on the machines. All jobs are ready at time 0, and due dates are given for each job i (d_i). A job can only be processed on one machine at a time, and a machine can only process one job at a time. A job's lateness is its completion time minus its due date (C_i - d_i). The objective is to sequence the operations on the machines in order to minimize the maximum lateness over all jobs ($\min[\max_i\{C_i - d_i\}]$). Using the Graham, *et al.* (1979) scheduling notation, this problem is referred to as $J//L_{max}$.

The problem was first introduced by Muth and Thompson (1963). Since then, job shop scheduling has been a standard topic in scheduling textbooks (e.g. see Baker, 1974 and Morton and Pentico, 1993). Rinnooy Kan (1976) proved that the job shop problem with the makespan objective is NP-Hard. Furthermore, other authors (see e.g. Adams *et al.*, 1988 and Werner and Winkler, 1995) indicate that not only is the job shop problem NP-Hard, it is one of the worst NP-hard problems. One classical 10x10 job shop problem formulated by Muth and Thompson (1963) was not solved until Carlier and Pinson (1989).

Van Laarhoven, *et al.* (1992) and Matsuo, *et al.* (1998) used simulated annealing to solve job shop problems with the makespan objective. They compared their simulated annealing approach to three other job shop scheduling heuristics and found superior or equivalent results for most problems. However, the procedure requires significant computational effort.

Dell'Amico and Trubian (1993), Nowicki and Smutnicki (1996), and Pezzella and Merelli (2000) use various Tabu search heuristics finding optimal or near-optimal solutions on numerous benchmark problems.

Demirkol, *et al*. (1998) established a series of benchmark problems with the intent that researchers could use them to evaluate the effectiveness of their algorithms. For the $J//L_{max}$ problem, they posted 160 problems including the best known solutions (upper bounds) along with solution times. Balas, *et al*. (1998) utilized a shifting bottleneck strategy combined with a guided local search procedure to obtain significant improvement in the upper bounds originally posted. The guided local search procedure searches through a neighborhood tree structure to overcome local optima. To the best of our knowledge, the solutions found by Balas, *et al*. (1998) are the current best known solutions to the benchmark problems.

Hodgson, *et al*. (1998) presented a procedure (referred to as the Virtual Factory) for efficiently and effectively solving industrial-size job shop scheduling problem with the objective of minimizing maximum lateness, $J//L_{max}$. They presented results for an iterative adaptive simulation-based scheduling procedure based on an idea found in Morton and Pentico (1993). For each iteration of the procedure, jobs are ordered at each machine based on the notion of revised slack. Revised slack is a function of the job's due date, processing time on the current machine and all downstream operations, and estimated queuing times for all downstream operations. Downstream queuing times are estimated from the previous iteration of the procedure. Hodgson, *et al*. (2000) enhanced the procedure by accelerating hot jobs, i.e. those jobs whose lateness is equal or nearly equal to the L_{max} value. Acceleration is accomplished by inserting idle time so that when a hot job arrives at a machine, it begins processing immediately.

In this paper, a simulated annealing post processing procedure is added to the Virtual Factory. The goal is to develop a procedure that improves on the Virtual Factory solutions, and does so efficiently. The procedure is evaluated on large-scale problems and compared to a lower bound. In addition, the Dermikol, *et al*. (1998) benchmark problems are evaluated and the results are compared to the Balas, *et al*. (1998) solutions.

2. Algorithm Development

We experimented with a variety of local improvement procedures for the Virtual Factory solution. Based on these experiments, simulated annealing was selected for further evaluation since it yielded the most promising results. Simulated annealing requires a solution representation, a method to generate initial solutions, a method to generate neighboring solutions, and a method to evaluate neighboring solutions. Each of these is described below.

2.1 Solution Representation

A solution is represented by a sequence of jobs for each and every machine, referred to here as the machine-job sequence. For example, a possible solution to a 4-machine, 3-job problem is:

Machine	Job Sequence
1	{1,2,3}
2	{2,1,3}
3	{3,1,2}
4	{1,3,2}

2.2 Initial Solution

An initial solution is generated for the $J//L_{max}$ problem using the Virtual Factory (Hodgson, *et al.* 1998, 2000) described above. For the benchmark problems, the Virtual Factory produces solutions close to, and in some cases, better than the Demirkol, *et al.* (1998) upper bounds after only a few CPU seconds. However, they are generally not as good as the Balas, *et al*. (1998) upper bounds.

2.3 Neighboring Solutions

Given a solution, a neighbor is generated by switching or interchanging the order of a pair (or pairs) of adjacent jobs on a machine. The neighboring solution shown below results from the two pair-wise interchanges on the initial solution (shown above). Jobs 3 and 1 have been interchanged on machine 3, and jobs 3 and 2 have been interchanged on machine 4.

Machine	Job Sequence
1	{1,2,3}
2	{2,1,3}
3	{1,3,2}
4	{1,2,3}

For the makespan problem, Van Laarhoven, *et al.* (1992) chose a neighborhood space that only investigates pair-wise interchanges on the critical path. Here, for a given solution to the L_{max} problem, the critical path is obtained by tracing backwards from the last operation of the L_{max} job (i.e.,

the job whose lateness is L_{max}) and determining if that operation is constrained by the preceding operation on the same machine or the preceding operation of the same job. In some cases, the critical path will diverge when an operation is constrained by both operations. Other authors investigating search heuristics for the job shop problem use variants of the Van Laarhoven, *et al*. (1992) general neighborhood structure that an interchange must contain an operation on the critical path (Dell'Amico and Trubian, 1993, Nowicki and Smutnicki, 1996, and Pezzella and Merelli, 2000).

However, consider a schedule *S* for the L_{max} problem in figure 1. Assume all jobs have a common due date, (d_i = 10), and the layout in figure 1 represent each job's process times and does not violate any jobs' technological routes. For this schedule L_{max} = 1 and job 4 is the L_{max} job. The shaded jobs represent those on the critical path.

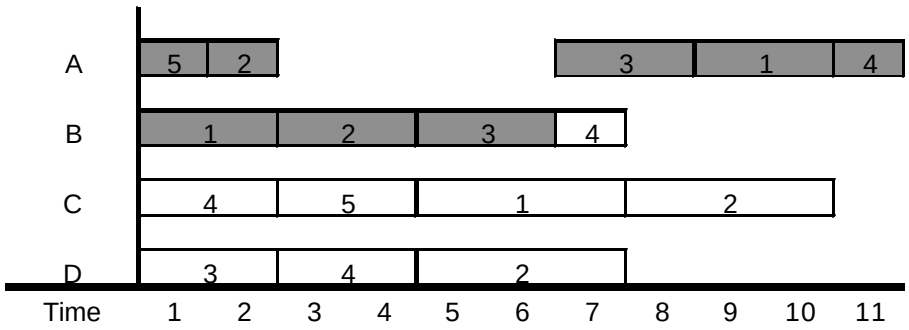

Figure 1 - Schedule *S*

Observe the following pair-wise interchanges on the critical path:

Machine	Interchange	Observation
A	1&4	Lmax = 1
A	3&1	Lmax = 2
A	2&3	Infeasible
A	2&5	Lmax = 1
B	2&3	Lmax = 2
B	1&2	Lmax = 3

From the observations of pair-wise interchanges on the critical path, no interchange leads to an immediate improvement. However, if jobs 3 and 1 are interchanged on machine A, with the simultaneous interchange of 5 and 1 on machine C, an improved L_{max} of 0 is obtained. Therefore, the

neighborhood investigated in this paper considers simultaneous pair-wise interchanges of jobs both on and off the critical path.

2.4 Neighborhood Generation

Experimentation with various interchange schemes resulted in the development of the following neighborhood generation scheme.

- Let *H* be the set of the machine/job combinations that lie on the critical path.
- Let *K* represent the maximum number of pair-wise interchanges to consider, *I* represent the mean number of interchanges to consider, and *C* (*C*≤*I*) represent the mean number of interchanges on the critical path to consider.
- Assuming that a neighboring solution is generated from at least one pair-wise interchange, the following probabilities are generated using the parameters defined above. Let α be the fraction of interchanges to accept, and ? be the fraction of interchanges on the critical path. Then,

  $\alpha = \frac{I-1}{K-1}$ and $? = \frac{C}{I}$.

- The following algorithm generates neighboring solutions using the parameters *K*, *I*, and *C*:
  1. Generate a U[0,1] random number, u_0.
     - if $u_0 \le ?$, then select a job/machine combination from *H* and perform a pair-wise interchange with the previous job.
     - else, select a job/machine combination at random that is not in *H* and perform an interchange.
  2. Perform the following *K*-1 times:
     - generate a U[0,1] random number, u_1. If $u_1 \le \alpha$ then:
       - generate a U[0,1] random number u_2.
         - if $u_2 \le ?$, then select a job/machine combination from *H* and perform a pair-wise interchange,
         - else, select a job/machine combination at random that is not in *H* and perform an interchange.

In summary, the neighborhood search is defined by parameters that specify the maximum number of pair-wise interchanges, the mean number of interchanges to consider, and the mean number of interchanges to be on the critical path. Through experimentation a robust parameter set can be determined. Figure 2 shows typical results of such an experiment for the Demirkol *et al.* (1998) problem 'r_20_15_1_1_2'. The figure indicates the improvement in L_{max} over time for various

combinations of parameters for the neighborhood generator. Note that the parameter sets that include both interchanges on and off the critical path (e.g. *K*, *I*, *C* = 4, 1.5, .8 or 2, 2, 1, respectively) have the best performance both in time and quality. The parameter set of *K*, *I*, *C* = 1,1,1, respectively, represents the Van Laarhoven, *et al*. (1992) neighborhood. These results are representative of all problem sets analyzed.

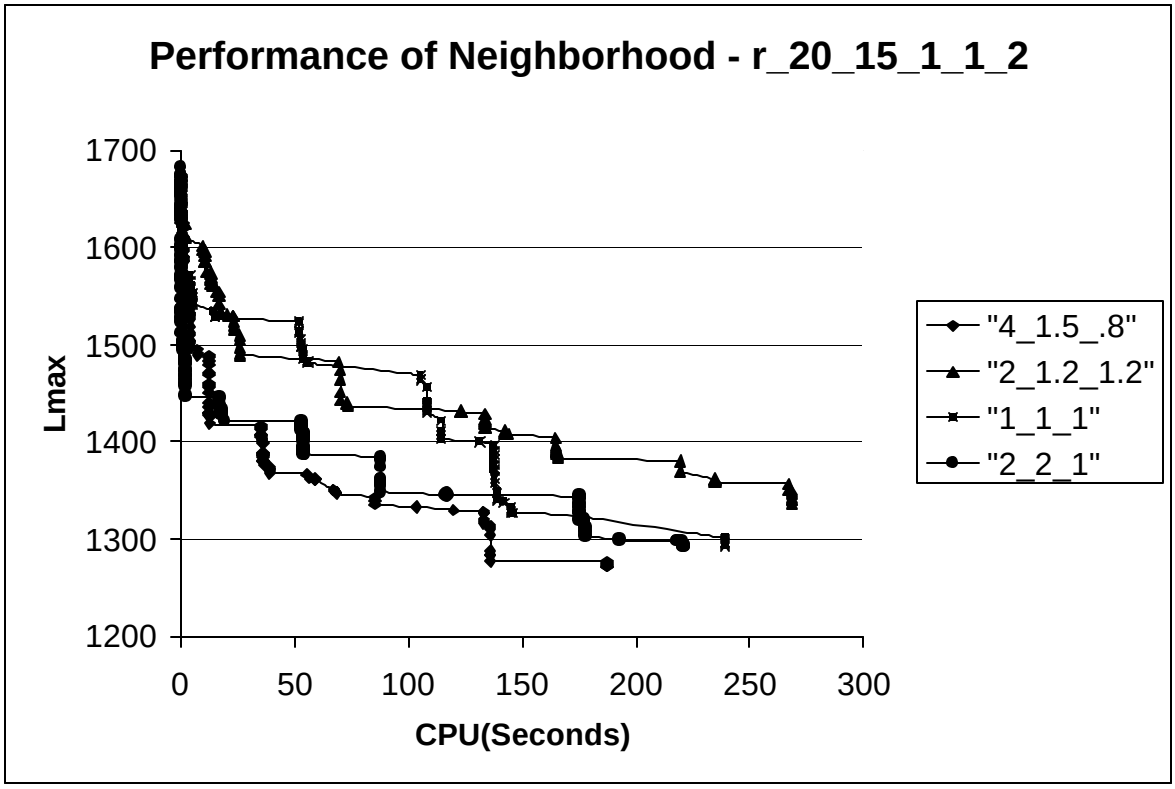

Figure 2 – Comparison of neighborhood generation parameter sets (*K_I_C*)

2.5 Solution Evaluation

Once a neighboring solution is generated, a computationally efficient evaluation method is necessary for the simulated annealing procedure to be of practical value. Given a fixed machine-job sequence, we were able to develop an efficient constructive algorithm for finding the minimum L_{max} schedule (see Appendix A). The algorithm is a key enabler for the approach taken here.

2.6 Simulated Annealing Procedure

The simulated annealing procedure described below is a modification of the classic Johnson *et al.* (1989) procedure.

1. Obtain an initial solution, *S*, using the Virtual Factory.
2. Start with an initial temperature, *T*. Set *S** (best solution found thus far) to *S*.
3. Perform the following loop until a maximum runtime *R* has been reached.
   - 3.1 Perform the following loop *G* times.
     - 3.1.1 Generate a random neighbor *S'* of *S*.
     - 3.1.2 Let δ = cost (*S'*) – cost (*S*).
     - 3.1.3 If $\delta < 0$ (downhill move), set *S* = *S'* and
       If(cost(*S'*) < cost(*S**), set *S** to *S'*.
     - 3.1.4 Else If $\delta \geq 0$ (uphill move), Set *S* = *S'* with probability $e^{-\delta/T}$
   - 3.2 If loop 3.1 has completed *B* times with no change in cost (*S*),
     go to 3.3 (frozen solution).
     Else, Set *T* = *rT* (reduce temperature), and return to 3.1.
   - 3.3 Set *T* = initial *T* (reset temperature).
     Set *S* to *S** (set current solution to best found thus far).
4. When maximum runtime is reached, return *S**.

Loop 3.1 is the classic Johnson *et al*. (1989) simulated annealing approach of generating neighbors at a given temperature, moving to any superior solutions, and allowing for the move to inferior solutions with some probability related to the temperature. The temperature is reduced after *G* iterations of Loop 3.1, thus reducing the chance of moving to inferior solutions as the process is cooled.

The classic simulated annealing approach is modified by enclosing it in Loop 3, which simply reheats the process to the initial temperature when the process has reached a temperature such that no new solutions are being accepted. The best solution found thus far is retrieved as the current solution. This, in effect, keeps the process from drifting away from this best solution. This approach of "reheating" the process is supported by Kolonko (1999) where he uses a reheating approach to obtain excellent results for the classic job shop problem of minimizing makespan. Furthermore, Kolonko shows that simulated annealing does not necessarily converge to the global optimum for the job shop problem because the standard neighborhoods used are not symmetric, thus a "reheating" approach is helpful to avoid becoming trapped in local optimum.

3. Experimentation

3.1 Industrial-sized problems

The objective in this paper is to determine if results obtained by Hodgson *et al*. (2000) could be improved in a timely manner. To this end, a series of experiments was performed using industrial-sized problems as defined in Hodgson *et al*. (2000). They examined problem sets consisting of 250 jobs, 50 machines, and 7 operations per job; and 1,000 jobs, 100 machines, and 7 operations. Processing times for the jobs were generated as Uniform [1-200]. Job routings are assumed to be Uniformly distributed across all machines, with the provision that a machine can appear only once on a route.

Demirkol, *et al*. (1998) noted that problem difficulty is a function of the due date range, i.e. the difference in the due dates between the jobs with the smallest and largest due dates.
Experiments were performed to determine the performance of the simulated annealing improvement procedure. For comparison, lower bounds are obtained using a procedure from Carlier and Pinson (1989). The difference between the lower bound and $L_{\max}$ found by the Virtual Factory was averaged across 30 problems for each range. $L_{\max}$ values were also found by applying the simulated annealing improvement procedure for 1, 5, and 10 minutes, respectively. Figures 3 and 4 depict the results. Each point on the charts represents an average of 30 problem instances. Note that these values ($L_{\max}$ – lower bound) are less than the processing time of a single operation. One can observe that the simulated annealing post processor improved the results, and that the benefits were obtained relatively early in the run. One can also observe that for problem sizes of 250 jobs, annealing for one minute provides the majority of improvement, with little additional value gained by longer annealing times. On average, the difference of the $L_{\max}$ values found from the lower bounds (shown in figure 3) were reduced by 67.6%, 80.0%, and 83.8% after one, five and ten minutes of annealing, respectively, for the 250 job problems. For the 1,000 job problems, annealing for five minutes provides the majority of improvement where 58.1%, 72.3%, and 77.1% reductions in the difference from the lower bound are found after one, five and ten minutes of CPU time, respectively, (figure 4).

3.2 Benchmark problems

As an additional validation, benchmark problems were obtained from Demirkol *et al*. (1998) (listed on the Internet site http://palette.ecn.purdue.edu/~uzsoy2/spssm.html). Various scheduling problems are posted on this site, along with the lower bounds (*LB*), best known solutions (*UB*) and the

CPU time required to obtain the best solution. Of interest here are the 160 problems under the section $J//L_{max}$.

For the 160 benchmark problems, the Virtual Factory/Simulated Annealing (VFSA) procedure was run using values of 0.5, 500,000, 2, and 0.8 for parameters *T, G, B,* and *r*, respectively. Values of 4, 1.5 and 0.8 were used for the neighborhood parameters *K*, *I* and *C*, respectively.

Table 1 lists each of the benchmark problems with its corresponding lower bound (LB), best known solution (UB) from Balas *et al.* (1998), and CPU time (SUNSparc-330 workstation) to achieve that solution. Results from the VFSA procedure are presented with the L_{max} values obtained at CPU times of 5, 30, 60, and 120 minutes. Runs were performed on a 1.4 GHz Intel processor. Also posted are the best solutions found during various trials of VFSA parameters. The first two parameters of the problem name specify the problem size by number of jobs and number of machines, respectively. Shaded cells represent the best known solutions.

Figure 5 is a graph of the time required by the VFSA to equal or surpass the best known solutions to the benchmark problems. For example, within 900 seconds, the VFSA found solutions equal to or better than Balas *et al.* (1998) for 90 of the benchmark problems. Of these 90 problems, the VFSA found superior solutions in 70 cases. The 20 problems with the same solution may be the result of both having obtained the optimal solution. In 120 minutes, the best known solution was obtained for 141 problems with an improved solution found for 116 of these problems.

Figures 6-8 are graphical depictions of the quality of the VFSA compared to the Balas upper bounds for the 5, 30, and 60 minutes of CPU time. Negative values indicate by how much the VFSA procedure improved the upper bound.

4. Observations and Conclusion

Simulated annealing coupled with the Virtual Factory is shown to be an effective approach to solving the $J//L_{max}$ job shop problem. For industrial-sized problems, the simulated annealing improvement procedure provided significant improvement to the schedule for additional run times of as little as 1 minute, with little additional improvement after 5 minutes. We found that the simulated annealing procedure benefits from a good starting point (i.e., the Virtual Factory) and found that the new parameter-driven neighborhood provided a more effective search mechanism over the traditional

approach of searching only along the critical path. Additionally, the reheat step in the annealing process helped re-center the search and helped to avoid becoming trapped in local optima.

Additionally, experimentation with benchmark problems validated the performance of the Virtual Factory with simulated annealing. Using a single parameter set, new or equivalent upper bounds were found for 36%, 69%, and 80% of the problems for run times of 5, 30, and 60 minutes, respectively. During our experimentation with a number of parameter settings, new or equivalent upper bounds were found for 159 of the 160 benchmark problems (see 'Best VFSA' column, table 1, and Schultz, 2001).

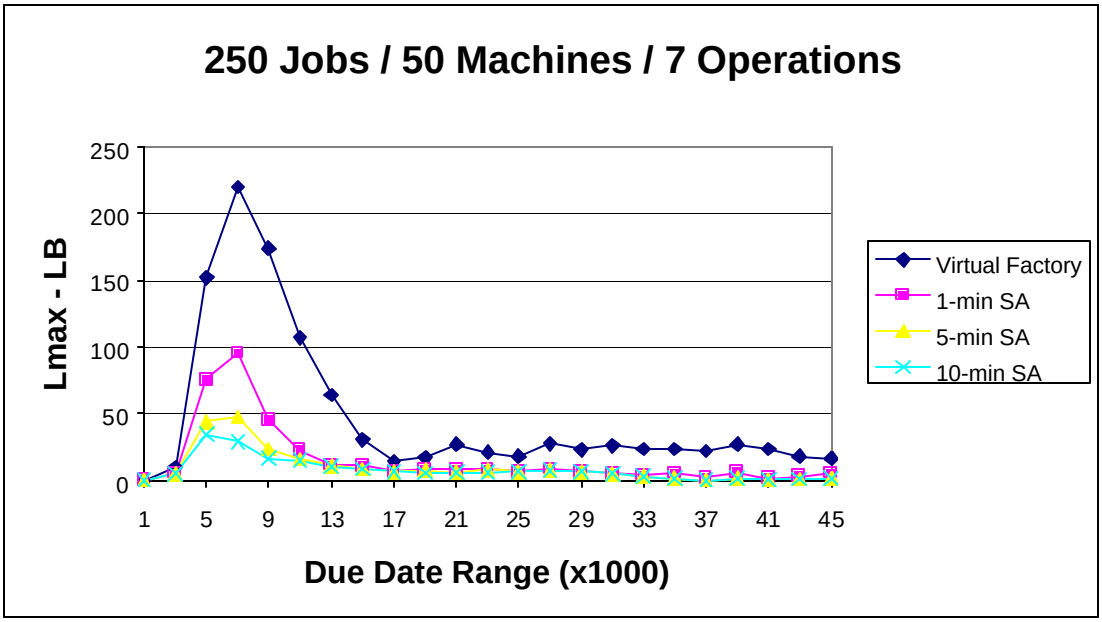

Figure 3: Performance to lower bound, 250 jobs/50 machines/7 operations

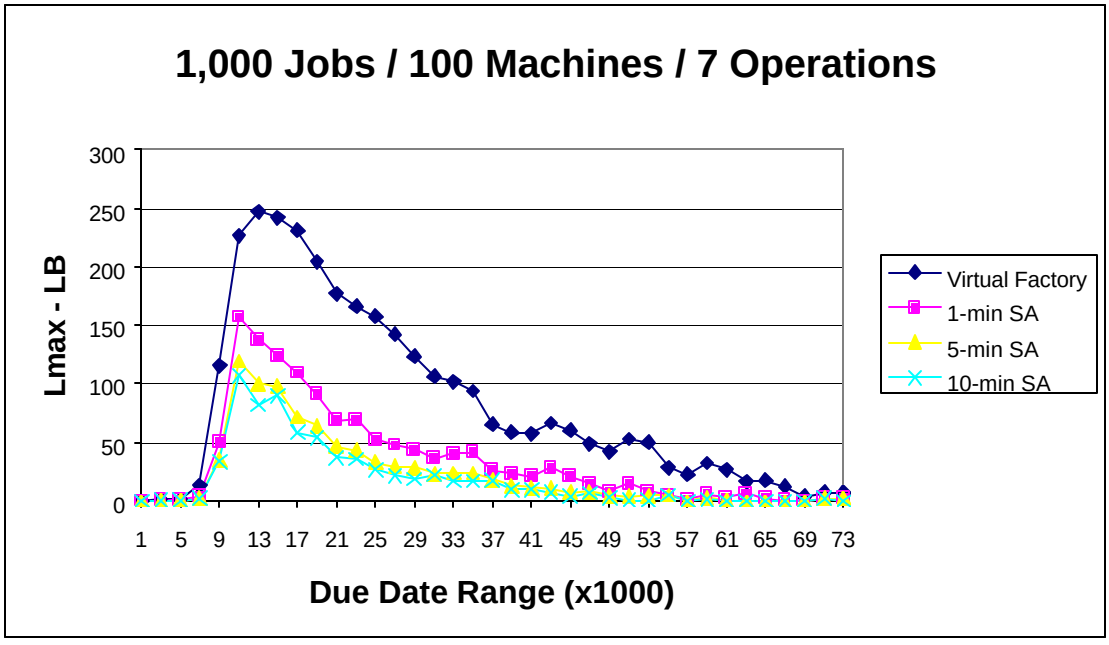

Figure 4: Performance to lower bound, 1,000 jobs/100 machines/7 operations

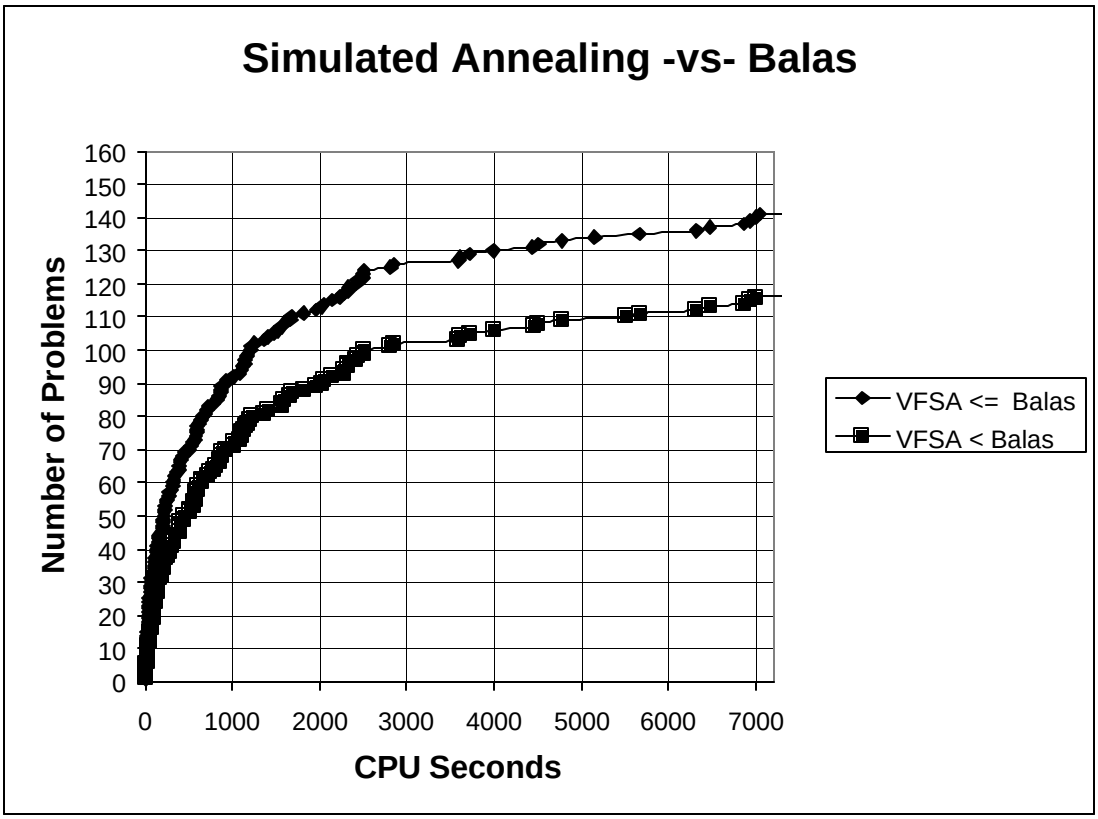

Figure 5: Performance of the VFSA versus computation time

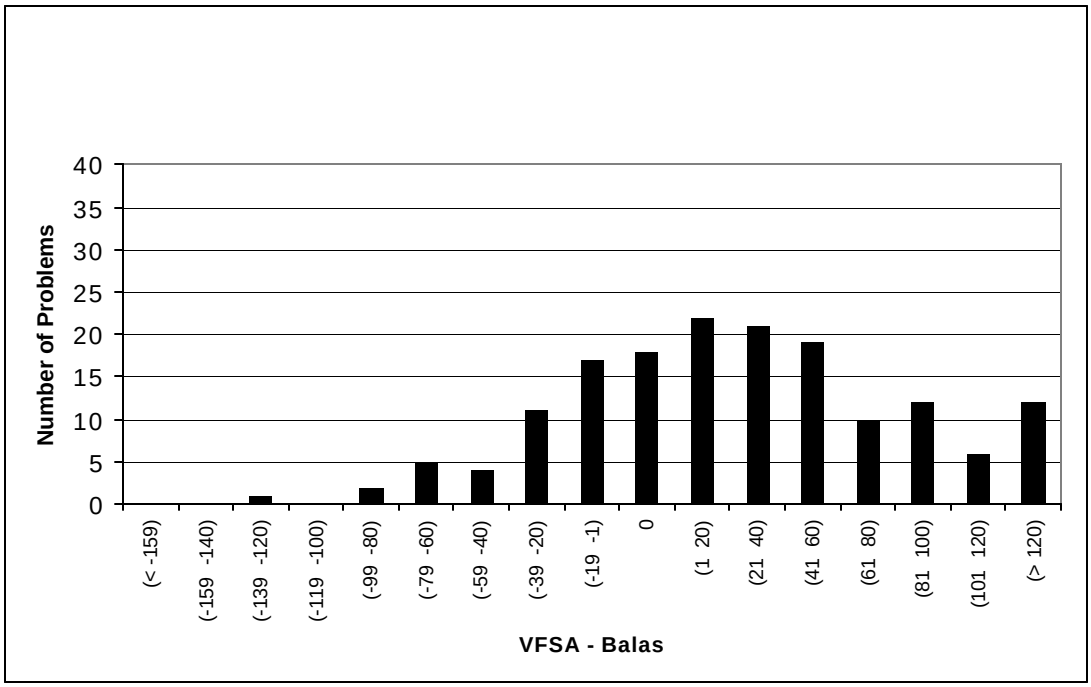

Figure 6: Comparison to Balas Solutions at 5 minutes

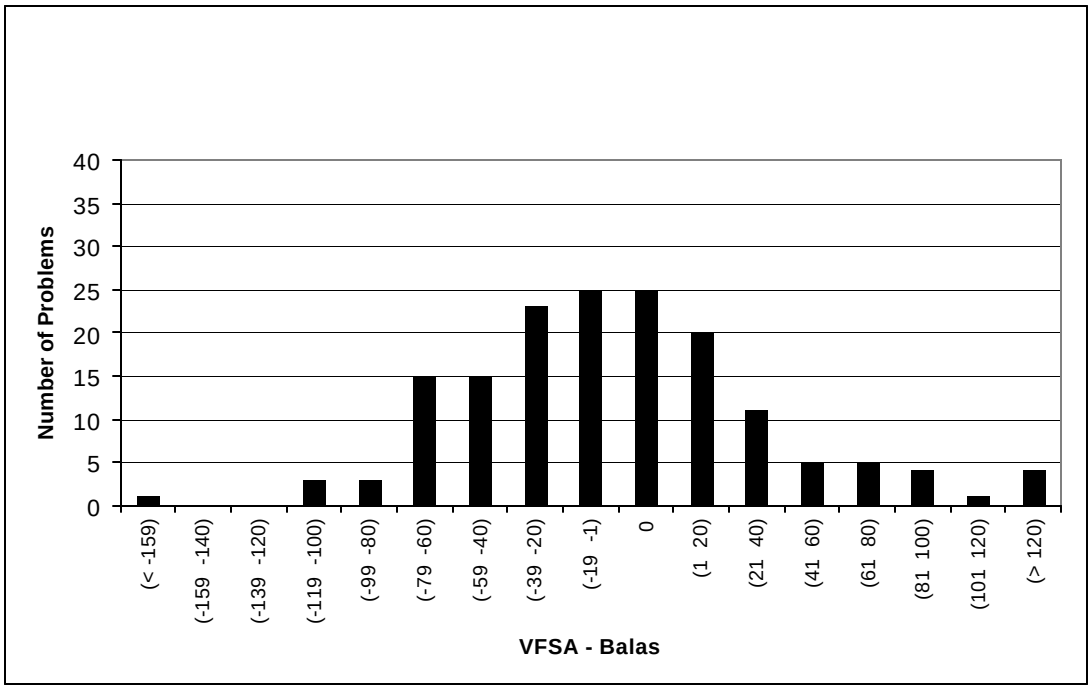

Figure 7: Comparison to Balas Solutions at 30 minutes

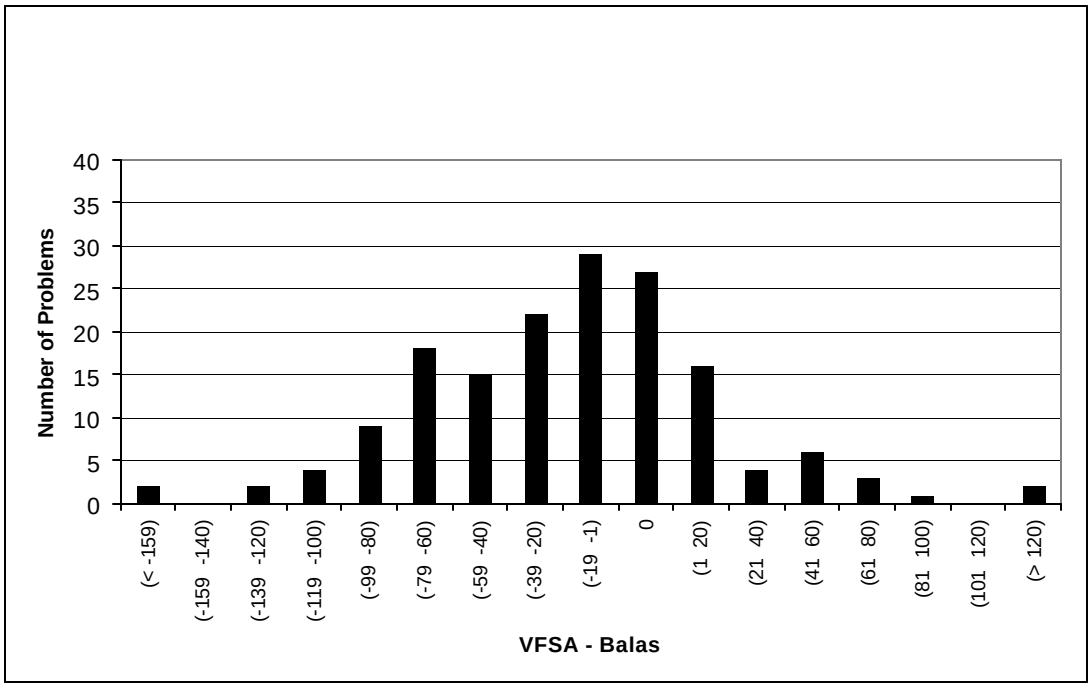

Figure 8: Comparison to Balas Solutions at 60 minutes

Table 1: Computational Results for Benchmark Problems

					VFSA: K=4,I=1.5,C=0.8				
Problem Name	LB	Previous UB (from Uzsoy)	UB (best known from Balas)	Balas CPU sec.	After 5 min	After 30 min	After 60 min	After 120 min	Best VFSA
r_20_15_1_1_2	1140	1464	1263	175	1299	1275	1275	1244	1244
r_20_15_1_1_3	1182	1501	1304	98	1271	1268	1268	1266	1258
r_20_15_1_1_4	1160	1492	1396	127	1367	1332	1332	1332	1326
r_20_15_1_1_6	1027	1448	1271	145	1229	1229	1229	1222	1208
r_20_15_1_1_8	1127	1552	1459	135	1449	1417	1388	1388	1388
r_20_15_1_2_1	1721	2090	1817	85	1818	1817	1817	1817	1817
r_20_15_1_2_10	1775	2092	1873	39	1873	1873	1873	1873	1873
r_20_15_1_2_5	1925	2181	1949	37	1930	1930	1930	1930	1930
r_20_15_1_2_8	1599	1785	1636	15	1636	1636	1636	1636	1636
r_20_15_1_2_9	1956	2246	2020	37	2020	2020	2020	2020	2020
r_20_15_2_1_1	1785	2165	2000	11	2014	1972	1970	1963	1960
r_20_15_2_1_3	1727	2100	1976	131	1971	1918	1901	1900	1898
r_20_15_2_1_5	1521	1839	1726	121	1697	1690	1684	1684	1683
r_20_15_2_1_7	1575	1957	1908	118	1846	1830	1824	1824	1824
r_20_15_2_1_9	1858	2143	1968	110	1955	1929	1914	1914	1913
r_20_15_2_2_10	1282	1682	1541	119	1525	1520	1520	1500	1475
r_20_15_2_2_2	1688	2174	1877	123	1918	1886	1869	1869	1865
r_20_15_2_2_3	1894	2381	2118	119	2173	2126	2126	2125	2117
r_20_15_2_2_4	1663	2018	1762	77	1730	1730	1730	1721	1721
r_20_15_2_2_7	1596	1943	1778	93	1776	1763	1763	1763	1759
r_20_20_1_1_10	1182	1708	1583	145	1642	1593	1585	1582	1574
r_20_20_1_1_3	1569	2248	2023	205	2056	1985	1985	1985	1974
r_20_20_1_1_4	1226	1753	1576	158	1571	1522	1507	1507	1506
r_20_20_1_1_6	1366	1962	1756	147	1771	1748	1748	1745	1719
r_20_20_1_1_7	1391	2013	1911	174	1830	1800	1799	1785	1785
r_20_20_1_2_2	2106	2465	2199	81	2192	2190	2186	2186	2186
r_20_20_1_2_4	2469	2835	2469	10	2469	2469	2469	2469	2469
r_20_20_1_2_6	2376	2842	2533	126	2542	2514	2504	2504	2504
r_20_20_1_2_8	2378	2712	2378	9	2378	2378	2378	2378	2378
r_20_20_1_2_9	2147	2631	2249	91	2249	2249	2249	2249	2245
r_20_20_2_1_2	1776	2638	2431	192	2336	2334	2334	2334	2310
r_20_20_2_1_4	1845	2535	2392	189	2395	2365	2354	2354	2336
r_20_20_2_1_6	1868	2647	2247	179	2275	2244	2244	2244	2230
r_20_20_2_1_7	1947	2640	2391	165	2476	2416	2402	2382	2382
r_20_20_2_1_8	1927	2627	2429	172	2428	2414	2414	2414	2408
r_20_20_2_2_1	1982	2617	2344	146	2324	2254	2254	2254	2219
r_20_20_2_2_3	2294	2851	2651	111	2623	2613	2599	2599	2599
r_20_20_2_2_5	2518	2966	2720	112	2701	2700	2685	2644	2644
r_20_20_2_2_6	2401	3029	2567	109	2560	2524	2512	2512	2512
r_20_20_2_2_7	2419	3118	2621	128	2677	2632	2632	2632	2621

Table 1 – cont.

					VFSA: K=4,I=1.5,C=0.8				
Problem Name	LB	Previous UB (from Uzsoy)	UB (best known from Balas)	Balas CPU sec.	After 5 min	After 30 min	After 60 min	After 120 min	Best VFSA
r_30_15_1_1_10	1185	1379	1295	133	1325	1275	1245	1231	1231
r_30_15_1_1_5	1205	1360	1209	134	1289	1289	1289	1289	1205
r_30_15_1_1_6	1255	1441	1270	108	1276	1255	1255	1255	1255
r_30_15_1_1_7	1263	1459	1308	167	1363	1293	1293	1293	1293
r_30_15_1_1_9	1320	1483	1386	142	1380	1380	1358	1358	1347
r_30_15_1_2_10	2240	2810	2455	108	2383	2383	2383	2383	2383
r_30_15_1_2_2	2436	3029	2526	85	2526	2526	2526	2526	2526
r_30_15_1_2_3	1935	2311	1982	59	1982	1982	1982	1982	1982
r_30_15_1_2_5	2475	2940	2601	115	2598	2598	2598	2598	2598
r_30_15_1_2_8	2169	2531	2320	77	2329	2302	2302	2302	2302
r_30_15_2_1_1	2042	2347	2157	192	2203	2158	2158	2158	2119
r_30_15_2_1_3	2189	2470	2339	124	2346	2269	2269	2269	2265
r_30_15_2_1_4	2401	2666	2549	184	2564	2485	2485	2485	2485
r_30_15_2_1_8	2252	2750	2493	191	2429	2429	2398	2381	2380
r_30_15_2_1_9	2224	2496	2366	194	2332	2289	2284	2284	2268
r_30_15_2_2_3	2068	2678	2377	133	2399	2357	2344	2344	2344
r_30_15_2_2_5	1960	2515	2212	141	2245	2217	2158	2141	2141
r_30_15_2_2_6	1734	2433	2149	183	2182	2142	2130	2126	2070
r_30_15_2_2_7	2075	2510	2196	149	2248	2215	2205	2205	2165
r_30_15_2_2_9	1922	2380	2195	133	2282	2179	2157	2157	2139
r_30_20_1_1_1	1611	2094	1855	317	1840	1753	1753	1742	1742
r_30_20_1_1_2	1575	2173	1846	303	1872	1806	1796	1773	1755
r_30_20_1_1_3	1268	1816	1587	335	1656	1566	1522	1522	1522
r_30_20_1_1_6	1412	1952	1656	269	1608	1607	1531	1531	1531
r_30_20_1_1_9	1710	2237	1880	202	1915	1883	1865	1859	1818
r_30_20_1_2_2	2386	3021	2641	246	2760	2710	2694	2688	2641
r_30_20_1_2_5	2292	2896	2315	42	2294	2292	2292	2292	2292
r_30_20_1_2_7	2817	3601	2897	142	2908	2884	2882	2882	2882
r_30_20_1_2_8	2713	3577	3023	226	3082	2989	2982	2965	2965
r_30_20_1_2_9	2314	3260	2527	223	2604	2541	2541	2476	2450
r_30_20_2_1_10	2496	3074	2873	272	2969	2872	2819	2819	2819
r_30_20_2_1_2	2178	2953	2669	318	2630	2593	2570	2553	2553
r_30_20_2_1_3	2298	3032	2780	361	2761	2722	2722	2722	2673
r_30_20_2_1_6	2461	3116	2766	303	2774	2739	2697	2683	2683
r_30_20_2_1_7	2562	3104	2833	285	2898	2878	2841	2816	2803
r_30_20_2_2_10	2570	3330	2713	224	2743	2743	2722	2720	2680
r_30_20_2_2_2	2254	3106	2560	211	2581	2581	2561	2553	2515
r_30_20_2_2_4	2061	2959	2349	340	2387	2386	2379	2349	2339
r_30_20_2_2_5	2485	3409	2781	228	2819	2810	2774	2743	2743
r_30_20_2_2_8	2412	3088	2514	198	2564	2448	2448	2448	2448

Table 1 – cont.

					VFSA: K=4,I=1.5,C=0.8				
Problem Name	LB	Previous UB (from Uzsoy)	UB (best known from Balas)	Balas CPU sec.	After 5 min	After 30 min	After 60 min	After 120 min	Best VFSA
r_40_15_1_1_1	1299	1468	1308	212	1358	1308	1308	1308	1308
r_40_15_1_1_10	1460	1527	1527	76	1460	1460	1460	1460	1460
r_40_15_1_1_3	1191	1431	1308	185	1368	1291	1291	1291	1291
r_40_15_1_1_4	1542	1648	1601	162	1696	1601	1601	1601	1601
r_40_15_1_1_8	1533	1787	1630	311	1619	1595	1563	1563	1563
r_40_15_1_2_10	1669	2459	1923	187	2063	2063	1976	1954	1917
r_40_15_1_2_3	1563	2397	1706	128	1777	1756	1746	1723	1712
r_40_15_1_2_5	1545	2053	1589	114	1686	1608	1589	1568	1563
r_40_15_1_2_6	1695	2191	1746	81	1754	1731	1731	1731	1731
r_40_15_1_2_7	1936	2420	1940	37	1940	1940	1940	1940	1940
r_40_15_2_1_10	3048	3120	3048	43	3048	3048	3048	3048	3048
r_40_15_2_1_2	2815	2894	2855	128	2856	2855	2855	2855	2855
r_40_15_2_1_6	2818	2875	2854	78	2856	2854	2854	2854	2854
r_40_15_2_1_8	2878	2924	2878	146	2929	2929	2924	2924	2878
r_40_15_2_1_9	2893	3093	2911	204	2911	2911	2911	2893	2893
r_40_15_2_2_1	1836	2617	2125	269	2169	2119	2086	2080	2080
r_40_15_2_2_10	1896	2539	2234	231	2326	2232	2224	2192	2186
r_40_15_2_2_2	2125	3042	2575	269	2575	2537	2536	2536	2517
r_40_15_2_2_3	2038	2641	2275	208	2320	2276	2258	2258	2258
r_40_15_2_2_6	2119	2767	2350	242	2358	2358	2339	2337	2337
r_40_20_1_1_1	1395	2058	1679	461	1690	1631	1606	1597	1590
r_40_20_1_1_10	1411	1834	1643	364	1606	1580	1580	1547	1547
r_40_20_1_1_2	1640	2143	1888	326	1942	1812	1802	1802	1802
r_40_20_1_1_4	1597	2337	1810	481	1846	1734	1712	1712	1706
r_40_20_1_1_6	1835	2212	1941	499	1895	1876	1876	1876	1835
r_40_20_1_2_2	2798	3565	2819	192	2870	2798	2798	2798	2798
r_40_20_1_2_3	2964	3862	3146	204	3146	3146	3146	3146	3146
r_40_20_1_2_4	2610	3444	2630	180	2630	2630	2630	2630	2630
r_40_20_1_2_8	2441	3064	2500	98	2505	2500	2500	2500	2500
r_40_20_1_2_9	3059	3895	3071	38	3059	3059	3059	3059	3059
r_40_20_2_1_1	2827	3430	3051	405	3152	3093	3050	3042	3001
r_40_20_2_1_3	3025	3572	3273	468	3194	3164	3159	3116	3116
r_40_20_2_1_4	2843	3366	3043	496	3156	3034	2993	2993	2993
r_40_20_2_1_5	3129	3535	3334	380	3208	3172	3162	3159	3111
r_40_20_2_1_8	3113	3691	3360	437	3398	3357	3341	3318	3317
r_40_20_2_2_1	2643	3560	3071	438	3208	3100	3063	3063	3059
r_40_20_2_2_5	2687	3985	3310	431	3338	3222	3222	3204	3163
r_40_20_2_2_7	2479	3469	2788	387	2885	2822	2764	2759	2752
r_40_20_2_2_8	2234	3154	2443	362	2629	2507	2487	2451	2439
r_40_20_2_2_9	2673	3540	3038	431	3041	2985	2985	2963	2942

Table 1 – cont.

					VFSA: K=4,I=1.5,C=0.8				
Problem Name	LB	Previous UB (from Uzsoy)	UB (best known from Balas)	Balas CPU sec.	After 5 min	After 30 min	After 60 min	After 120 min	Best VFSA
r_50_15_1_1_4	1707	1764	1707	267	1707	1707	1707	1707	1707
r_50_15_1_1_5	1418	1545	1418	330	1545	1545	1420	1420	1418
r_50_15_1_1_7	1050	1419	1072	381	1183	1080	1078	1050	1050
r_50_15_1_1_8	1957	2042	1983	201	1985	1985	1985	1985	1983
r_50_15_1_1_9	1757	1804	1757	150	1839	1839	1839	1807	1757
r_50_15_1_2_1	2284	3102	2346	133	2346	2346	2346	2346	2346
r_50_15_1_2_2	2137	2661	2195	107	2208	2195	2195	2195	2186
r_50_15_1_2_4	2085	2692	2154	207	2248	2095	2093	2087	2087
r_50_15_1_2_7	2192	2834	2301	122	2346	2278	2278	2269	2262
r_50_15_1_2_8	2217	3024	2364	251	2435	2370	2364	2364	2364
r_50_15_2_1_2	3181	3220	3181	389	3181	3181	3181	3181	3181
r_50_15_2_1_4	3277	3316	3281	237	3311	3277	3277	3277	3277
r_50_15_2_1_5	3216	3492	3216	273	3345	3345	3343	3343	3216
r_50_15_2_1_8	3391	3525	3397	228	3416	3397	3397	3397	3391
r_50_15_2_1_9	3396	3466	3396	332	3396	3396	3396	3396	3396
r_50_15_2_2_10	2345	3130	2619	342	2759	2697	2653	2615	2615
r_50_15_2_2_5	2381	3186	2615	272	2814	2622	2499	2499	2449
r_50_15_2_2_6	2486	3307	2775	434	2855	2820	2820	2759	2742
r_50_15_2_2_7	2464	3415	2812	329	2867	2825	2785	2753	2753
r_50_15_2_2_9	2323	3338	2724	375	2941	2941	2941	2740	2720
r_50_20_1_1_2	1794	2355	2007	551	2050	1966	1882	1882	1882
r_50_20_1_1_3	1746	2390	1948	743	1972	1914	1911	1811	1811
r_50_20_1_1_4	1786	2142	1881	608	1954	1905	1893	1893	1786
r_50_20_1_1_5	1591	2181	1757	551	1839	1781	1756	1710	1710
r_50_20_1_1_6	1845	2219	1912	464	1946	1861	1861	1861	1861
r_50_20_1_2_10	2824	3898	2914	305	3047	2981	2912	2887	2864
r_50_20_1_2_2	2440	3385	2445	176	2477	2444	2444	2444	2444
r_50_20_1_2_5	3205	4091	3229	287	3267	3262	3205	3205	3205
r_50_20_1_2_6	2363	3455	2595	361	2814	2677	2544	2530	2530
r_50_20_1_2_8	2918	3852	3044	327	3161	3154	3097	3052	3037
r_50_20_2_1_10	3407	3789	3522	435	3528	3476	3475	3476	3457
r_50_20_2_1_2	3189	3788	3277	467	3382	3288	3266	3230	3230
r_50_20_2_1_4	3527	3758	3562	547	3536	3527	3527	3527	3527
r_50_20_2_1_5	3419	3875	3504	552	3454	3432	3432	3432	3432
r_50_20_2_1_7	3642	3971	3708	246	3796	3746	3746	3746	3669
r_50_20_2_2_1	2500	3712	3020	473	3213	3106	3093	3020	2981
r_50_20_2_2_10	2628	4042	3142	602	3265	3241	3204	3117	3117
r_50_20_2_2_4	2551	3762	3072	614	3137	3045	3045	3028	3028
r_50_20_2_2_5	2774	4184	3347	550	3447	3365	3363	3324	3281
r_50_20_2_2_8	2472	3649	3099	636	3169	3058	3020	3017	3017

Appendix A – Constructive algorithm to find the minimum L_{max} schedule

If given a fully described machine-job sequence, the following is a constructive algorithm that efficiently finds the minimum L_{mx} schedule.

Definitions:

mach_ptr(*j*)	– pointer to a location in machine *j*'s job sequence list.
job_ptr(*i*)	– pointer to a location in job *i*'s operation list.
mach_time(*j*)	– current time on machine *j*'s schedule up to which jobs have been assigned.
job_time(*i*)	– current time for job *i*'s schedule up to which operations have been assigned.

Initialization:

Set *mach_ptr*(*j*) = 1.
Set *job_ptr*(*i*) = 1.
Set *mach-time*(*j*) = 0.
Set *job_time*(*i*) = 0.

Construction:

1. Set $j = 1$.
2. For machine *j*, identify job *i* in the machine's job sequence at the location identified by *mach_ptr*(*j*).
3. If *job_ptr*(*i*) contains machine *j*, then a match has occurred. Place job *i* on the schedule for machine *j* at the maximum of *mach_time*(*j*) and *job_time*(*i*) Set *mach_time*(*j*) and *job_time*(*i*) equal to this maximum value plus the process time for job *i*, operation *j*. Increase *mach_ptr*(*j*) and *job_ptr*(*i*) by 1.
4. Set $j = j+1$. If $j < M$ (the number of machines) go to 2. Otherwise return to step 1 as long as at least one match in step 3 has occurred during the complete pass through all machines, $j = 1$ to M.
5. Terminate construction if a match did not occur.

Return solution:

If the construction terminates with all job operations assigned, then the provided job sequence is feasible and the schedule is optimal for the given machine-job sequence. Otherwise, the solution is infeasible for the given job sequences on the machines.

References

ADAMS J., E. BALAS, and D. ZAWACK, 1988, The shifting bottleneck procedure for job shop scheduling. *Management Science*, **34**(3), 391-401.

BAKER, K.R., 1974, *Introduction to Sequencing and Scheduling*, John Wiley and Sons, New York.

BALAS, E., G. LANCIA, P. SERAFINI, and A. VAZACOPOULOS, 1998, Job shop scheduling with deadlines. *Journal of Combinatorial Optimization*, **1**, 329-353.

CARLIER, J. and E. PINSON, 1989, An algorithm for solving the job shop problem. *Management Science*, **35**(2), 164-176.

DELL'AMICO, M. and M. TRUBIAN, 1993, Applying tabu-search to the job shop scheduling problem. *Annals of Operations Research*, **4**, 231-252.

DEMIRKOL, E., S. MEHTA. and R. UZSOY, 1998, Benchmarks for shop scheduling problems. *European Journal of Operational Research*, **109**(1), 137-141.

GRAHAM R.L., E.L. LAWLER, J.K. LENSTRA, and A.H.G. RINNOOY KAN, 1979, Problem classification. *Annals of Discrete Mathematics*, **5**, 287-326.

HODGSON, T.J., D. CORMIER, A.J. WEINTRAUB, and A. ZOZOM JR., 1998, Satisfying due-dates in large job shops. *Management Science*, **44**(10), 1442-1446.

HODGSON, T.J., R.E. KING, K. THONEY, N. STANISLAW, A.J. WEINTRAUB and A. ZOZOM JR., 2000, On satisfying due-dates in large job shops: Idle time insertion. *IIE Transactions*, **32**(2), 177-180.

JOHNSON, D.S., C.R. ARAGON, L.A. MCGEOCH, and C. SCHEVON, 1989, Optimization by simulated annealing: An experimental evaluation; Part 1, Graph partitioning. *Operations Research*, **37**(6), 865-892.

Kolonko, M., 1999, Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research*, **113**, 123-136.

MATSUO, H., C.J. SUH, and R.S. SULLIVAN, 1998, A controlled search simulated annealing method for the general job shop scheduling problem, Working paper 03-04088, Department of Management, The University of Texas at Austin.

MORTON, T.E. and D.W. PENTICO, 1993, *Heuristic Scheduling Systems with Applications to Production Systems and Project Management*, John Wiley and Sons, New York.

MUTH, J.F. and G.L. THOMPSON, 1963, *Industrial Scheduling*, Prentice Hall, Englewood Cliffs, NJ.

NOWICKI, E. and C. SMUTNICKI, 1996, A fast taboo search algorithm for the job shop problem. *Management Science*, **42**, 797-813.

PEZZELLA, F. and E. MERELLI, 2000, A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, **120**, 297-310.

RINNOOY KAN, A.H.G., 1976, *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hauge.

SCHULTZ, S.R., 2001, Scheduling problems with due-date objectives and constraints, Unpublished Ph.D. dissertation, North Carolina State University.

VAN LAARHOVEN, P.J.M., E.H.L. AARTS, and J.K. LENSTRA, 1992, Job shop scheduling by simulated annealing. *Operations Research*, **40**(1), 113-125.

WERNER F. and A. WINKLER, 1995, Insertion techniques for the heuristic solution of the job shop problem. *Discrete Applied Mathematics*, **58**, 191-211.

Acknowledgements: This research was supported, in part, by the Furniture Manufacturing & Management Center, North Carolina State University, and by the Office of Naval Research, Contract #N00014-90-J-1009.