Allocating tools to index positions in tool magazines using tabu search

Diptesh Ghosh

W.P. No. 2016-02-06
February 2016

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD – 380015
INDIA

Allocating tools to index positions in tool magazines using tabu search

Diptesh Ghosh

Abstract

The arrangement of tools in tool slots of a tool magazine is an important problem in automated machining environments. This problem is called the indexing problem and has been widely studied in the literature. In this paper we propose a tabu search algorithm to solve the indexing problem. We perform computational experiments on instances which are of reasonable sizes.

Keywords: CNC tool magazine, indexing, tabu search

1 Introduction

In computer numerically controlled (CNC) machines, several operations using several tools can be performed on a job without altering the setup of the job. In order to achieve this, CNC machines arrange jobs in a tool magazine. When a particular tool is required for an operation, the tool is brought to a specific position called the index position, and is picked up by a tool change arm and sent to the work table. Once the operation is finished, the tool is returned to its position in the magazine by the arm, and the magazine rotates so as to bring the tool required for the next operation to the index position. Typical tool magazines have upward of 30 tools and 70 to 100 tool slots (see, e.g., Gray et al. 1993). The tool magazine can be visualized as a circular disk called a tool magazine, with tools in slots arranged at equal intervals on the disk (see Figure 1). The total processing time

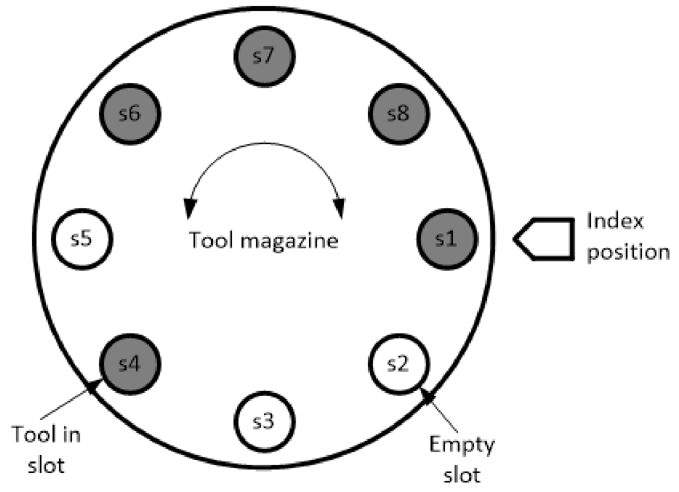

Figure 1: Schematic representation of a tool magazine

for a job on a machine is the sum of the actual processing time (which is constant) and the time required to bring tools to the index position. This process of bringing tools to the index position can be optimized (i.e., minimized) by a proper allocation of tools to slots in the tool magazine. The assignment of tools to slots in the tool magazine is called “indexing”, and the objective of this paper is to proposed an algorithm for efficient indexing.

The indexing operation has been studied in the literature in the past. Early studies, e.g., Gray et al. (1993) have drawn attention to tool management as an important part of automated manufacturing. They pointed out that industry data suggest that tooling accounts for 25% to 30% of both fixed and variable costs in automated machining environments. Wilson (1987) also presented a formulation of the indexing problem and used commercial solvers to solve indexing problems of small size. Dereli et al. (1998) and Dereli and Filiz (2000) were the first to address the indexing problem using a genetic algorithm. They did not allow duplication of tools in the magazine and solved an indexing problem instance with ten tools in a magazine with 16 slots. Recently Velmurugan and Victor Raj (2013) presented a particle swarm optimization algorithm for the problem and presented a solution to an instance with 14 tools and 28 slots. Baykasoğlu and Dereli (2004) and Baykasoğlu and Ozsoydan (2015) study a variant of the problem in which multiple copies of tools can be placed in the magazine. This variant is more complex since for each operation, one needs to decide which of the copies of the tool to bring to the index position. Some closely related problems have also been studied in the literature. For example, Aktürk and Ozkan (2001), Avcı and Aktürk (1996), Hertz et al. (1998), and Sinriech et al. (2001) have addressed the closely related tool loading problem. Saravanan and Ganesh Kumar (2013) provides a review of the large body of work in another closely related group of problems called loop layout problems. Kothari and Ghosh (2012) reviews the literature on the single row facility layout problem which has several elements in common with the indexing problem.

Summing up, we see that there are two broad variants of the indexing problem. In the first, there is only one copy of each tool required for a job, and that copy has to be assigned to a slot in the tool magazine. The number of slots in the tool magazine can either be equal to the number of tools or greater. In the second variant, there can be multiple copies of each tool. In this paper we present an algorithm for the first variant of the problem.

Let us now derive mathematical expressions for the cost of the indexing operation. Consider a tool magazine with n slots. The tool magazine has to rotate $2\pi/n$ degrees to move from one slot in the index position to the next slot in that position. We will call this an “operation” of the tool magazine. The effort required to move a tool from a position in the magazine to the index position is measured in terms of the number of operations required. For example, if the tools required for two consequent operations are in positions j and k in the magazine, the tool change operation would require $\min\{|k-j|, |j+n-k|\}$ operations. Note that the number of operations remain unchanged if the positions of the tools are reversed. Now assume that a particular job requires m tools ($m \leq n$). Let the vector $\Pi = (\pi(1), \pi(2), \ldots, \pi(n))$ denote an arrangement of tools in the magazine, such that the j-th entry of the vector, $\pi(j)$ has a value of $*$ if slot j is empty, and the number k ($1 \leq k \leq m$) if tool k occupies slot j. Assume that the frequency with which tool at positions j and k are used consequently f_{jk} times. Clearly, $f_{**} = 0$, $f_{*k} = f_{k*} = 0$ for all $k = 1, \ldots, m$. Let $d_{jk} = \min\{|k-j|, |j+n-k|\}$. The total number of operations required to process the job if Π denotes the tool assignment is

$$\sum_{j=1}^{n-1} \sum_{k=j+1}^{n} f_{\pi(j)\pi(k)} d_{jk}. \tag{1}$$

Our objective of our indexing problem is to find a permutation Π that minimizes the quantity (1).

The mathematical programming formulation of the problem is the following. We denote the set of slots by S and use indices i and j to traverse it. We denote the set of tools by T and use indices k and l to traverse it. We use binary variables y_{ik} which assume a value 1 if tool k is in position i and 0 otherwise. The problem data is input through the frequency matrix $F = [f_{kl}]$ where f_{kl} denotes the number of consecutive operations which requires tools k and l in any order. As defined earlier, $d_{ij} = \min\{|j-i|, |i+n-j|\}$. The model is

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{m} \sum_{\substack{l=1 \\ l \neq k}}^{m} d_{ij}\ f_{kl}\ y_{ik} y_{jl} \\
\text{Subject to} \quad & \sum_{i=1}^{n} y_{ik} = 1 \qquad k = 1, \dots, m \\
& \sum_{k=1}^{m} y_{ik} \leq 1 \qquad i = 1, \dots, m \\
& y_{ik} \in \{0, 1\} \qquad \text{for all } i \in S, k \in T.
\end{aligned}
$$

This is a quadratic assignment problem (QAP) formulation (see Loiola et al. (2007) for a survey of the QAP). However, since the QAP is NP-hard, we propose a tabu search algorithm to solve this problem. In the next section, we describe the tabu search algorithm that we use in this paper. Section 3 provides the results of our computational experiments with tabu search on the indexing problem. Finally Section 4 summarizes the contributions of this paper.

2 Tabu Search

Neighborhood search, also called local search forms the basis of several effective metaheuristics to solve computationally hard optimization problems. In neighborhood search, several feasible solutions which are in some way close to a solution forms its neighborhood. Typically, a neighborhood is defined as solutions that can be obtained by performing operations, called "moves" from a pre-specified set of moves on the solution. Starting with a solution as the initial "current" solution, neighborhood search finds increasingly better solutions by searching the neighborhood of the current solution. Once such a better solution is found, the search replaces the current solution with the better solution. It stops when no better solution is found in the neighborhood of the current solution and outputs the current solution as an approximation to an optimal solution for the problem.

Such a method often gets trapped in local optima which may be of poor quality. So extensions of the neighborhood search algorithms which can get out of local optima and continue the search have been proposed in the literature. Tabu search (see Glover (1989; 1990)) is arguably the most popular of such methods. It is an ierative algorithm that works by maintaining a list of moves that are generally forbidden at a particular iteration of the algorithm. This list is called a tabu list and includes the reverse of moves recently executed by the search. It is updated at every iteration of the algorithm. The only situation in which the tabu list is overridden is when a neighbor of a current solution obtained through a move in the tabu list meets an aspiration criterion. The usual aspiration criterion is that the neighbor thus obtained has a better objective than any solution encountered by the search thus far. Tabu search ends when a pre-specified criterion is met. This criterion is usually one among a pre-specified iteration limit, a pre-specified time limit, or a pre-specified number of iteration since the last improvement seen by the search. On termination the search outputs the best solution it has encountered as an approximation to an optimal solution for the problem.

For the indexing problem with n slots and m tools ($m \leq n$), a solution to the problem is a permutation of numbers from 1 through n. Of these, numbers 1 through m represent tools, and numbers from $m + 1$ through n represent empty slots. We consider an exchange neighborhood in which the only move allowed is an exchange move in which the numbers in two positions of the permutation are exchanged. If both the numbers being exchanged are between 1 and m, this represents an actual exchange of tools in the magazine. If one of the two is in the range from $m+1$ through n, then the exchange represents a re-positioning of a tool to an empty slot in the magazine. If both the numbers are in the range from $m + 1$ through n then the move represents a vacuous exchange.

The pseudocode of the tabu search algorithm we use in this paper is given below.

ALGORITHM TS

Input: An initial permutation Π, cost function $z(\cdot)$, tabu tenure `TENURE`, maximum number of iterations allowed `MAXITER`.

Output: A good solution to the indexing problem defined by $z(\cdot)$.

```
1.  begin
2.      BestSol ⟵ Π; TABU ⟵ ∅; iter ⟵ 1; CurrSol ⟵ Π;
3.      while (iter ⟵ MAXITER) do begin
4.          IterBestSol ⟵ CurrSol;
5.          for (every possible neighbor NbrSol of CurrSol) begin
6.              if ( (CurrSol ⟶ NbrSol) ∉ TABU and z(NbrSol) < z(IterBestSol) )
    then
7.                  IterBestSol ⟵ NbrSol;
8.                  add (NbrSol ⟶ CurrSol) to TABU for the next TENUREiterations;
9.                  if (z(IterBestSol) < z(BestSol)) then BestSol ⟵ IterBestSol;
10.             end
11.             else if ( (CurrSol ⟶ NbrSol) ∈ TABU and z(NbrSol) < z(BestSol) ) then
12.                 IterBestSol ⟵ NbrSol; BestSol ⟵ NbrSol;
13.                 TABU ⟵ ∅;
14.             end;
15.         end;
16.         CurrSol ⟵ IterBestSol;
17.         iter ⟵ iter + 1;
18.     end;
19.     output BestSol;
20. end.
```

After initializing the algorithm in Step 2, tabu search proceeds by performing `MAXITER` iterations (steps 3 through 18). At the beginning of each iteration the best solution obtained in the iteration is initialized to the current solution. Then all neighbors of the current solution are searched. If the neighbor improves the best solution obtained in the iteration and is not obtained through a move in the `TABU` list (steps 6 through 10), then the best solution is updated, and the reverse of the move is added to the `TABU` list for the next `TENURE` iterations. If however the neighbor is better than any solution reached so far even though the move to obtain the neighbor is tabu, then the neighbor is still considered (steps 11 through 14). In this case, we say that the aspiration criterion is satisfied and we reset the `TABU` list (step 13). After performing `MAXITER` tabu search iterations, the algorithm terminates and outputs the best solution it has encountered (step 19).

Computing the cost of a solution from scratch requires quadratic time, and since the number of exchange neighbors of a solution is $\mathcal{O}(n^2)$, the time complexity of a tabu seach iteration is $\mathcal{O}(n^4)$. This is prohibitively expensive. We show below that the cost difference between a solution and its exchange neighbor can be computed in linear time. We use this to reduce the complexity of a TS iteration to $\mathcal{O}(n^3)$.

Consider a solution Π and its neighbor Π' obtained by interchanging tools in positions r and s ($r < s$ w.l.o.g.).

$$\begin{aligned}\text{Then } z(\Pi') - z(\Pi) &= \text{cost of repositioning } r + \text{cost of repositioning } s\\ &= \Big(\sum_{\substack{i=1\\ i\neq r,s}}^{n} f_{\pi(i)\pi(s)}d_{ir} - \sum_{\substack{i=1\\ i\neq r,s}}^{n} f_{\pi(i)\pi(r)}d_{ir}\Big) + \Big(\sum_{\substack{i=1\\ i\neq r,s}}^{n} f_{\pi(i)\pi(r)}d_{is} - \sum_{\substack{i=1\\ i\neq r,s}}^{n} f_{\pi(i)\pi(s)}d_{is}\Big)\\ &= \sum_{\substack{i=1\\ i\neq r,s}}^{n} \Big(f_{\pi(i)\pi(r)} - f_{\pi(i)\pi(s)}\Big)\Big(d_{is} - d_{ir}\Big).\end{aligned}$$

This expression can clearly be computed in linear time.

We describe our computational experience with this algorithm in the next section.

3 Computational Experiments

We coded the tabu search algorithm described in Section 2 in C. We ran our experiments on a machine with Intel i-5-2500 64-bit processor at 3.30 GHz with 4GB RAM. Our algorithm was implemented as a multi-start algorithm and we report the best solution obtained from among all starts. From our initial experimentation with randomly generated problems we found out that it was sufficient to run the program with 20 starts. We ran each start with the value of `MAXITER` set to 10000, and found that the solution that was finally output was invariably obtained within the first 500 tabu search iterations. We experimented with different values of `TENURE`, and found out that the best results are obtained by letting `TENURE` be randomly generated from the interval [2,5]. Hence for our experiments we set the number of starts to 20, `TENURE`to be randomly generated between 2 and 5, and the value of `MAXITER` to 2000.

In the literature, computational experiments on the indexing problem has been carried out on small instances. One such instance is from Dereli and Filiz (2000). It has 10 tools, 16 operations, and 16 slots in the tool magazine. The sequence of operations is T1, T1, T5, T4, T2, T9, T7, T3, T3, T3, T9, T9, T9, T10, T8, T6. TS found an optimal solution that used 13 operations in 0.571 cpu seconds. The other instance is from ?. It has 14 tools, 27 operations and 28 slots in the tool magazine. The sequence of operations is 1, T1, T2, T3, T4, T2, T2, T2, T2, T5, T6, T5, T7, T5, T5, T8, T9, T5, T10, T5, T11, T5, T12, T5, T13, T5, T14. TS found an optimal solution that used 50 operations in 3.904 seconds. Since we could not find other instances for the problem, we decided to modify some other instances for related problems for our computations.

3.1 B-D instances

These are 30 instances from Baykasoğlu and Ozsoydan (2015) used for the indexing problem. They allowed duplication of tools in the tool magazine. The first ten instances (8-20-01 through 8-20-10) used 8 tools for 20 operations and the tool magazine had 12 slots. The second ten instances (8-30-01 through 8-30-10) used 8 tools for 30 operations and the tool magazine had 12 slots. The final ten instances (12-50-01 through 12-50-10) used 12 tools for 50 operations and the tool magazine had 16 slots. Table 1 presents our results for these instances. The first three colums of each row of the table provide the instance name, the number of tools and the number of slots in the tool magazine. The number of operations is not important for the solution algorithm and is not given in the table. The last two columns in each row present the number of operations required by the solution output by TS for the instance and the time in cpu seconds required by TS for the instance.

Table 1: Performance on B-K instances

Instance	Tools	Slots	Cost	Time
8-20-01	8	12	19	0.270
8-20-02	8	12	31	0.265
8-20-03	8	12	25	0.267
8-20-04	8	12	29	0.266
8-20-05	8	12	38	0.264
8-20-06	8	12	28	0.270
8-20-07	8	12	24	0.266
8-20-08	8	12	31	0.266
8-20-09	8	12	31	0.267
8-20-10	8	12	25	0.267
8-30-01	8	12	48	0.268
8-30-02	8	12	48	0.270
8-30-03	8	12	50	0.266
8-30-04	8	12	50	0.269
8-30-05	8	12	48	0.272
8-30-06	8	12	51	0.270
8-30-07	8	12	54	0.273
8-30-08	8	12	55	0.267
8-30-09	8	12	46	0.267
8-30-10	8	12	54	0.269
12-50-01	12	16	117	0.580
12-50-02	12	16	114	0.578
12-50-03	12	16	118	0.584
12-50-04	12	16	95	0.586
12-50-05	12	16	121	0.582
12-50-06	12	16	112	0.602
12-50-07	12	16	107	0.591
12-50-08	12	16	136	0.589
12-50-09	12	16	136	0.624
12-50-10	12	16	117	0.619

The solutions output by TS on these instances is given in Appendix A. In the solutions a '-' denotes an empty slot.

3.2 Anjos Instances

These instances are adapted from instances used in Anjos et al. (2005) for the single row facility layout problem (SRFLP). Each of the instances in Anjos et al. (2005) had a frequency matrix denoting the frequency of interaction between a pair of facilities in the SRFLP instance. We use the frequency data to denote the number of times a pair of tools are used in consequent operations. We also used a tool magazine with 100 slots for our experiments. Table 2 presents our results for these instances. The structure of the table is the same as that of Table 1. The solutions output by TS on these instances is given in Appendix B. In the solutions a '-' denotes an empty slot.

Table 2: Performance on Anjos instances

Instance	Tools	Slots	Cost	Time
an-60-01	60	100	54053	179.468
an-60-02	60	100	31278	180.109
an-60-03	60	100	23510	180.109
an-60-04	60	100	11592	178.235
an-60-05	60	100	15182	178.233
an-70-01	70	100	42297	178.452
an-70-02	70	100	51723	178.046
an-70-03	70	100	43795	178.500
an-70-04	70	100	27705	178.842
an-70-05	70	100	134269	181.483
an-75-01	75	100	66656	182.202
an-75-02	75	100	111806	181.454
an-75-03	75	100	38158	181.859
an-75-04	75	100	106341	179.531
an-75-05	75	100	47031	179.140
an-80-01	80	100	54463	178.858
an-80-02	80	100	52853	179.015
an-80-03	80	100	95091	182.609
an-80-04	80	100	100950	184.234
an-80-05	80	100	36227	185.530

3.3 sko Instances

These instances have been used in Anjos and Yen (2009) for computational experiments for the single row facility layout problem. Each problem in the set had five variants, but all the variants of a problem had the same frequency matrix. Therefore we had a total of seven instances for our experiments. We used a tool magazine with 60 slots when the number of tools were less than 60. Otherwise, we used a tool magazine with 100 slots. Table 3 presents our results for these instances. The structure of the table is the same as that of Table 1. The solutions output by TS on these instances is given in Appendix C. As usual, in the solutions a '-' denotes an empty slot.

Table 3: Performance on sko instances

Instance	Tools	Slots	Cost	Time
sko-42	42	60	24408	38.281
sko-49	49	60	36582	38.078
sko-56	56	60	52974	38.030
sko-64	64	100	95337	182.062
sko-72	72	100	133607	182.561
sko-81	81	100	185852	183.078
sko-100	100	100	290496	185.000

4 Summary

In this paper, we have studied the indexing problem. This is an important problem in automated machining, and optimizes costs that account for 25% to 30% of the total costs of the machining process. We presented a tabu search algorithm TS and our computational experience with this algorithm on indexing problems of realistic size. Apart from demonstrating the capability of TS to handle such problems, our results will serve as benchmarks for the performance of other solution methods for these problems.

References

M.S. Aktürk and S. Ozkan. Integrated Scheduling and Tool Management in Flexible Manufacturing Systems. International Journal of Production Research 39 (2001) pp. 2697–2722.

M.F. Anjos, A. Kennings, and A. Vannelli. A Semidefinite Optimization Approach for the Single-Row Layout Problem with Unequal Dimensions. Discrete Optimization 2 (2005) pp. 113–122.

M.F. Anjos and G. Yen. Provably Near-Optimal Solutions for Very Large Single-Row Facility Layout Problems. Optimization Methods and Software 24 (2009) pp. 805–817

S. Avcı and M.S. Aktürk. Tool Magazine Arrangement and Operations Sequencing on CNC Machines. Computers & Operations Research 23 (1996) pp. 1069–1081.

A. Baykasoğlu and T. Dereli. Heuristic Optimization System for the Determination of Index Positions on CNC Magazines with the Consideration of Cutting Tool Duplications. International Journal of Production Research 42 (2004) pp. 1281–1303.

A. Baykasoğlu and F.B. Ozsoydan. An improved approach for determination of index positions on CNC magazines with cutting tool duplications by integrating shortest path algorithm. International Journal of Production Research. DOI: 10.1080/00207543.2015.1055351

T. Dereli, A. Baykasoğlu, N.N.Z. Gindy, and İ.H. Filiz. Determination of Optimal Turret Index Positions by Genetic Algorithms. Proceedings of 2nd International Symposium on Intelligent Manufacturing Systems. (1998) pp. 743–750. Turkey.

T. Dereli and İ.H. Filiz. Allocating Optimal Index Positions on Tool Magazines Using Genetic Algorithms. Robotics and Autonomous Systems 33 (2000) pp. 155–167.

A.E. Gray, A. Seidmann, and K.E. Stecke. A Synthesis of Decision Models for Tool Management in Automated Manufacturing. Management Science 39 (1993) pp. 549–567.

A. Hertz, G. Laporte, M. Mittaz, and K.E. Stecke. Heuristics for Minimizing Tool Switches When Scheduling Part Types on Flexible Machine. IIE Transactions 30 (1998) pp. 689–694.

F. Glover. Tabu Search – Part 1. ORSA Journal of Computing 1 (1989) pp. 190–206.

F. Glover. Tabu Search – Part 2. ORSA Journal of Computing 1 (1989) pp. 4–32.

R. Kothari and D. Ghosh. The Single Row Facility Layout Problem: State of the Art. OPSEARCH 49 (2012) pp. 442–462.

E.M. Loiola, N.M.M. de Abreu, P.O. Buaventura-Netto, P. Hahn, and T. Querido. A Survey of the Quadratic Assignment Problem. European Journal of Operational Research 176 (2007) pp. 675–690.

D. Sinriech, J. Rubinovitz, D. Milo, and G. Nakbily. Sequencing, Scheduling and Tooling Single-stage Multifunctional Machines in a Small Batch Environment. IIE Transactions 33 (2001) pp. 897–911.

M. Velmurugan and M. Victor Raj. Optimal Allocation of Index Positions on Tool Magazines Using Particle Swarm Optimization Algorithm. International Journal of Artificial Intelligence and Mechatronics 1 (2013) pp. 5–8.

M. Saravanan and S. Ganesh Kumar. Different Approaches for the Loop Layout Problem: A Review. International Journal of Manufacturing Technology 69 (2013) pp. 2513–2529.

J.M. Wilson. Formulation and Solution of a Set of Sequencing Problems for Flexible Manufacturing Systems. Proceedings of the Institute of Mechanical Engineering 201 (1987) pp. 247–249.

Appendices

Appendix A: TS output for the B-D instances

8-20-01 6, 7, -, -, -, 3, 5, 4, 2, 8, 1, -

8-20-02 -, -, -, -, 7, 2, 6, 3, 1, 8, 4, 5

8-20-03 4, 6, 7, 3, 2, -, -, -, -, 8, 1, 5

8-20-04 -, -, -, -, 3, 6, 7, 5, 4, 8, 2, 1

8-20-05 6, 2, -, -, 3, -, -, 1, 4, 8, 7, 5

8-20-06 6, -, -, -, 3, 7, 4, 1, 2, 8, 5, -

8-20-07 -, -, -, -, 1, 2, 6, 7, 3, 5, 8, 4

8-20-08 -, -, -, 4, 3, 8, 7, 2, 6, 1, 5, -

8-20-09 -, -, -, 4, 3, 1, 7, 6, 2, 8, 5, -

8-20-10 -, -, -, -, 8, 3, 5, 2, 1, 6, 7, 4

8-30-01 -, -, -, -, 2, 5, 3, 1, 7, 8, 6, 4

8-30-02 6, 4, -, -, -, -, 2, 3, 7, 8, 5, 1

8-30-03 6, 2, 3, 7, -, -, -, -, 1, 8, 4, 5

8-30-04 3, 1, -, -, -, -, 8, 7, 2, 6, 5, 4

8-30-05 1, 5, -, -, -, -, 8, 6, 7, 2, 3, 4

8-30-06 6, 3, 4, 1, 2, -, -, -, -, 8, 5, 7

8-30-07 4, 2, -, -, -, -, 3, 1, 7, 8, 6, 5

8-30-08 4, 6, 2, 3, -, -, -, -, 1, 5, 8, 7

8-30-09 -, 4, -, -, 2, 3, 8, 7, 6, 1, 5, -

8-30-10 -, -, -, 5, 8, 3, 2, 6, 4, 1, 7, -

12-50-01 4, 11, 3, 5, 8, -, -, -, -, 7, 6, 9, 1, 10, 2, 12

12-50-02 6, 4, 10, 2, 3, 12, 5, 11, 1, 7, 9, 8, -, -, -, -

12-50-03 6, 12, 9, 5, 2, 11, -, -, -, -, 1, 3, 4, 7, 8, 10

12-50-04 1, 7, 12, 6, 10, 11, 8, 9, -, -, -, -, 5, 3, 2, 4

12-50-05 6, 12, 5, 8, 1, 9, 10, 7, -, -, -, -, 2, 3, 11, 4

12-50-06 7, 6, 11, 4, 1, -, -, -, -, 10, 8, 2, 3, 9, 12, 5

12-50-07 7, 1, 11, 2, 8, 10, 9, 12, -, -, -, -, 4, 5, 3, 6

12-50-08 6, 10, 2, 5, 7, 4, -, 3, -, -, -, 8, 11, 12, 9, 1

12-50-09 4, 6, 7, 3, 12, 11, -, -, -, -, 5, 1, 2, 8, 10, 9

12-50-10 -, 11, 7, 4, 6, 12, 3, 5, 9, 8, 1, 2, 10, -, -, -

Appendix B: TS output for the Anjos instances

an-60-01 -, 50, 1, 29, 18, 58, 39, 41, 20, 19, 60, 52, 23, 55, 45, 14, 10, 30, 42, 43, 22, 53, 59, 37, 31, 5, 38, 49, 2, 17, 26, 47, 57, 35, 44, 4, 46, 33, 51, 32, 8, 36, 15, 9, 7, 16, 3, 25, 54, 6, 24, 13, 12, 21, 56, 40, 11, 27, 48, 34, 28, -, -, -, -, -, -, -, -, -

an-60-02 40, 46, 41, 50, 34, 15, 30, 24, 14, 42, 9, 37, 21, 32, 33, 5, 36, 25, 18, 45, 56, 52, 17, 22, 31, 2, 4, 8, 47, 49, 48, 38, 58, 16, 23, 57, 39, 51, 59, 54, 43, 6, 27, 55, 7, 1, 12, 13, 3, 35, 26, 11, 20, 60, -, 53, 10, 28, 19, 44, 29

an-60-03 19, 3, 59, 42, 50, 24, 16, 57, 12, 37, 11, 48, 20, 44, 7, 17, 9, 41, 2, 21, 14, 35, 8, 31, 55, 18, 46, 33, 58, 28, 39, 10, 32, 56, 45, 25, 51, 30, 52, 60, 23, 54, 43, 22, 15, 49, 36, 53, 34, 47, 1, 27, 13, 26, 6, 40, -, 4, 5, 29, 38

an-60-04 1, 34, 49, 54, 38, 8, 25, 32, 41, 9, 46, 51, 43, 12, -, -, -, 13, -, 20, 19, 31, 22, 57, 36, 39, 10, 7, 55, 50, 24, 23, 44, 56, 33, 6, 4, 40, 5, 53, 45, 27, 59, 48, 3, 14, 17, 21, 15, 37, 58, 2, 35, 11, 42, 60, 29, 30, 26, 16, 52, 18, 47, 28

an-60-05 -, -, -, -, -, -, -, -, -, -, -, -, -, 33, 38, 49, 44, 11, 12, 57, 5, 51, 34, 43, 15, 23, 3, 10, 52, 45, 32, 1, 22, 29, 18, 14, 31, 58, 26, 42, 27, 21, 56, 24, 28, 50, 6, 2, 46, 48, 17, 8, 20, 54, 60, 59, 36, 16, 41, 25, 19, 35, 13, 47, 37, 7, 9, 4, 40, 30, 55, 53, 39, -

an-70-01 12, 64, 46, 20, 59, 25, 55, 26, 61, 38, 17, 11, 57, 30, 35, 29, 39, 8, 50, 22, 40, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, *, -, -, -, -, -, -, -, -, -, -, -, 53, 47, 31, 28, 62, 32, 63, 37, 69, 2, 65, 1, 15, 19, 41, 51, 5, 9, 68, 27, 49, 43, 4, 16, 48, 10, 13, 67, 56, 33, 45, 42, 34, 18, 36, 52, 60, 6, 54, 7, 23, 21, 58, 14, 3, 66, 44, 70, 24

an-70-02 43, 70, 46, 55, 2, 5, 14, 40, 20, 32, 42, 24, 51, 54, 17, 35, 34, 12, 22, 15, 68, 37, 66, 61, 27, 48, 31, 16, 65, 69, 59, 38, 62, 10, 53, 47, 52, 41, 57, -, 21, 39, 28, 56, 45, 64, 50, 36, 19, 63, 13, 58, 26, 49, 3, 67, 1, 4, 18, 25, 30, 9, 23, 7, 29, 8, 11, 33, 44, 6, 60

an-70-03 -, 22, 69, 44, 16, 34, 13, 62, 56, 57, 33, 31, 48, 12, 66, 43, 28, 15, 46, 38, 64, 53, 29, 3, 35, 45, 36, 60, 18, 59, 24, 25, 58, 39, 55, 40, 37, 41, 1, 5, 14, 23, 10, 4, 42, 6, 67, 68, 11, 9, 61, 70, 65, 27, 21, 7, 51, 49, 26, 20, 17, 2, 32, 19, 50, 47, 63, 54, 30, 52, 8, -

an-70-04 64, 29, 58, 18, 5, 54, 52, 34, 14, 9, 60, 43, 15, 12, 26, 49, 67, 7, 6, 30, 51, 46, 63, 25, 16, 66, -, 41, 35, 38, 8, 13, 37, 4, 19, 3, 65, 45, 31, 39, 59, 47, 20, 32, 10, 36, 55, 1, 23, 69, 44, 21, 50, 40, 17, 28, 27, 56, 70, 57, 48, 53, 62, 2, 68, 61, 11, 42, 24, 33, 22

an-70-05 10, 30, 45, 14, 34, 25, 50, 39, 16, 70, 12, 63, 35, 11, 5, 36, 3, 61, 66, 57, 4, 46, 48, 42, 52, 54, 49, 29, 60, 13, 7, 53, 15, 59, 65, 43, 69, 55, 56, 9, 68, 26, 21, 19, 38, 64, 17, 67, -, 28, 27, 20, 62, 44, 32, 22, 24, 58, 51, 23, 1, 47, 37, 2, 6, 41, 31, 18, 33, 8, 40

an-75-01 -, 9, 64, 67, 58, 38, 2, 60, 47, 48, 29, 33, 16, 59, 63, 5, 52, 54, 39, 73, 1, 51, 65, 30, 32, 10, 45, 26, 43, 4, 31, 66, 13, 68, 17, 7, 53, 3, 24, 19, 12, 55, 37, 71, 34, 62, 21, 8, 22, 75, 42, 18, 27, 74, 15, 35, 56, 41, 70, 25, 44, 11, 57, 40, 28, 50, 72, 49, 23, 14, 6, 61, 69, 46, 20, 36, -, -, -

an-75-02 25, 13, 44, 42, 55, 64, 43, 37, 54, 29, 21, 20, 16, 70, 73, 36, 63, 58, 50, 30, 41, 38, 69, 45, 40, 48, 31, 35, 68, 74, 59, 17, 9, 27, 15, 19, 11, 53, 8, 10, 4, 67, 33, 71, 52, 18, 24, 34, 46, 7, 23, 39, 66, 75, 5, 6, 14, 65, 1, 60, 62, 28, 47, 3, 12, 72, 22, 61, 56, 32, 57, 26, 49, -, 51, 2

an-75-03 -, 42, 15, 47, 69, 43, 63, 71, 50, 51, 14, 53, 1, 13, 29, 46, 32, 25, 70, 8, 6, 72, 66, 52, 49, 2, 18, 27, 73, 31, 16, 34, 39, 48, 58, 3, 12, 37, 59, 44, 67, 75, 9, 30, 24, 7, 11, 56, 28, 38, 62, 45, 55, 21, 57, 74, 65, 61, 35, 60, 5, 40, 41, 26, 64, 4, 54, 23, 19, 36, 33, 17, 20, 68, 22, 10, -, -, -, -

an-75-04 36, -, 8, 60, 56, 10, 14, 27, 15, 5, 7, 62, 37, 48, 50, 70, 52, 42, 57, 22, 41, 75, 47, 33, 16, 55, 3, 64, 65, 31, 30, 39, 29, 12, 40, 20, 46, 53, 35, 44, 28, 67, 34, 63, 19, 24, 13, 71, 58, 1, 4, 59, 54, 11, 18, 51, 25, 45, 32, 43, 74, 68, 49, 2, 72, 38, 73, 26, 17, 69, 61, 23, 6, 21, 66, 9

an-75-05 19, 75, 63, 9, 31, 52, 20, 37, 43, 17, 22, 24, -, 27, 13, 12, 26, 60, 54, 33, 57, 36, 67, 61, 62, 21, 45, 10, 70, 14, 69, 11, 15, 48, 18, 59, 53, 16, 1, 29, 46, 42, 68, 25, 74, 35, 7, 34, 71, 47, 28, 44, 73, 5, 39, 32, 49, 23, 55, 3, 30, 4, 41, 56, 51, 66, 65, 6, 40, 64, 58, 38, 72, 2, 8, 50

an-80-01 -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, 70, 78, 72, 25, 79, 18, 77, 28, 11, 4, 7, 57, 64, 30, 1, 71, 3, 31, 38, 50, 69, 26, 37, 74, 8, 59, 35, 22, 73, 36, 15, 42, 33, 5, 6, 53, 76, 23, 9, 58, 48, 66, 13, 24, 10, 32, 75, 52, 51, 47, 41, 45, 20, 43, 63, 39, 61, 54, 40, 56, 17, 27, 19, 49, 62, 44, 46, 68, 21, 2, 67, 12, 29, 14, 34, 65, 16, 80, 60, 55, -, -, -, -, -

an-80-02 20, 15, 32, 7, 54, 30, 80, 72, 60, 78, 73, 64, 2, 51, 75, 63, 13, 49, 69, 28, 74, 43, 45, 61, 68, 52, 42, 70, 66, 48, 11, 24, -, 57, 18, 17, 22, 77, 55, 46, 10, 29, 33, 5, 26, 23, 40, 12, 58, 44, 41, 53, 14, 47, 62, 50, 27, 9, 56, 79, 37, 38, 6, 1, 4, 39, 25, 67, 34, 3, 35, 71, 59, 8, 21, 65, 36, 16, 76, 19, 31

an-80-03 77, 25, 80, 23, 51, 78, 10, 14, 11, 7, 67, 55, 9, 19, 41, 15, 59, 1, 4, 12, 31, 73, 2, 79, 38, 28, 26, 60, 75, 62, 6, 5, 47, 45, 18, 42, 43, 37, 49, 16, 58, 20, 68, 33, 65, 46, 24, 66, 40, 63, 74, -, 57, 61, 54, 29, 34, 53, 21, 13, 36, 71, 70, 39, 69, 50, 35, 8, 52, 3, 27, 22, 32, 30, 48, 72, 56, 76, 64, 44, 17

an-80-04 9, 14, 62, 18, 16, 12, 15, 74, 2, 72, 27, 3, 77, 70, 8, 1, 25, 53, 33, 65, 23, 11, 73, 19, 45, 63, 68, 39, 13, 20, 31, 26, 52, 5, 10, 36, 55, 37, 51, 57, 22, 49, 58, 40, 48, 24, 50, 61, 44, 56, 41, -, 69, 47, 60, 59, 66, 29, 21, 64, 38, 71, 42, 7, 34, 80, 35, 28, 30, 75, 54, 43, 79, 67, 78, 32, 17, 46, 76, 6, 4

an-80-05 58, 60, 2, 52, 27, 68, 38, 74, 73, 6, 21, 10, 61, 47, 34, 79, 1, 54, 33, 70, 76, 30, 28, 9, 37, 22, 75, 49, 36, 31, 71, 55, 24, 67, 15, 44, 66, 19, 32, 80, 43, 65, 4, 62, 20, 8, 18, 14, 72, 50, 45, 77, 63, 16, 3, 35, 23, 51, 57, 5, 56, 29, 64, 42, 69, 40, 59, 13, 39, 78, 12, 41, 7, 46, 25, 48, 11, -, 26, 53, 17

Appendix C: TS output for sko instances

sko-42 -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, 41, 25, 29, 11, 7, 5, 34, 17, 42, 12, 27, 20, 10, 13, 31, 19, 32, 18, 26, 35, 9, 15, 14, 2, 4, 39, 30, 33, 21, 40, 28, 38, 37, 1, 24, 22, 3, 36, 23, 16, 8, 6, -, -, -

sko-49 19, 36, 47, 25, 28, 42, 13, 20, 18, 46, 14, 41, 35, 29, 7, 1, 9, 15, 10, 34, 12, 11, 45, 38, 43, 6, 37, 27, 2, 40, 48, 3, 23, 31, 49, 33, 22, 32, 5, 16, 30, 44, 8, 17, 39, 21, -, -, -, -, -, -, -, -, -, -, -, 24, 26, 4

sko-56 51, 3, 47, 29, 39, 18, 5, 35, 32, 20, 48, 34, 23, 17, -, -, 15, 4, 46, -, -, 19, 26, 55, 43, 33, 54, 38, 6, 24, 25, 45, 22, 31, 53, 21, 16, 50, 8, 2, 56, 9, 42, 12, 14, 27, 49, 11, 37, 36, 1, 28, 30, 7, 13, 44, 52, 40, 10, 41

sko-64 7, 35, 63, 18, 60, 31, 57, 34, -, 15, 41, 3, 61, 36, 27, 12, 46, 54, 32, 11, 28, 43, 30, 5, 6, 47, 51, 40, 39, 38, 48, 55, 62, 19, 8, 10, 53, 64, 16, 23, 20, 4, 42, 13, 49, 52, 29, 22, 24, 9, 58, 14, 21, 44, 59, 17, 26, 33, 25, 45, 2, 56, 1, 50, 37

sko-72 30, 70, 10, 2, 38, 35, 60, 27, 18, 8, 56, 31, 12, -, 53, 34, 29, 3, 25, 11, 45, 43, 24, 54, 5, 69, 62, 72, 4, 66, 50, 20, 58, 19, 6, 15, 63, 9, 40, 48, 71, 37, 13, 46, 33, 49, 59, 39, 36, 65, 41, 61, 7, 55, 44, 57, 16, 23, 67, 1, 68, 32, 22, 17, 64, 42, 51, 14, 47, 26, 52, 28, 21

sko-81 58, 77, 32, 5, 79, 42, 61, 22, 56, 35, 49, 38, 41, 4, 23, 15, 69, 21, 70, 71, 36, 6, 17, 48, 50, 67, 55, 33, 3, 34, 30, 29, 11, 65, 81, 80, 25, 26, 66, 74, 12, 2, 14, 9, 31, 19, 40, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, -, 28, 46, 64, 60, 20, 8, 72, 44, 78, 27, 68, 45, 10, 57, 59, 54, 53, 16, 47, 7, 39, 76, 43, 24, 62, 73, 1, 75, 52, 13, 18, 37, 51, 63

sko-100 72, 45, 38, 41, 76, 36, 82, 51, 73, 64, 90, 60, 81, 46, 20, 44, 67, 99, 49, 2, 85, 28, 43, 94, 14, 1, 18, 91, 97, 78, 63, 57, 77, 32, 34, 25, 95, 6, 19, 79, 37, 13, 11, 71, 31, 33, 83, 54, 89, 5, 75, 74, 15, 16, 52, 27, 22, 88, 80, 55, 9, 84, 65, 39, 62, 4, 96, 30, 86, 87, 68, 8, 98, 59, 42, 56, 93, 24, 69, 61, 12, 50, 21, 66, 29, 47, 100, 70, 58, 3, 35, 23, 26, 7, 40, 48, 92, 53, 17, 10