INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD • INDIA

Research and Publications

Exploring Lin Kernighan neighborhoods for the indexing problem

Diptesh Ghosh

W.P. No. 2016-02-13
February 2016

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD – 380015
INDIA

Exploring Lin Kernighan neighborhoods for the indexing problem

Diptesh Ghosh

Abstract

The indexing problem in automated machining environments aims to arrange tools in tool slots of a tool magazine for faster processing of jobs. This problem is called the indexing problem and has been widely studied in the literature. We present heuristics using Lin Kernighan neighborhood structures to solve the indexing problem. Two of the heuristics are local search heuristics and one is a tabu search heuristic. To the best of our knowledge this is one of the first implementations of tabu search on a Lin Kernighan neighborhood structure.
Keywords: CNC tool magazine, indexing, Lin Kernighan neighborhoods, local search, tabu search

1 Introduction

The indexing problem is one of deciding on the position of cutting tools in a tool magazine of a computer numerically controlled machine to reduce the processing time of jobs on the machine. The total processing time of a job on a CNC machine is the sum of the cutting times by various tools and the times required to change tools on the turret. The cutting times do not vary on the same job, but the tool changing times are dependent on the position of the tools in the tool magazine. Some studies, Gray et al. (1993) have pointed out that such tool changing operations account for up to a third of the total costs in automated machining environments. When tools need to be changed, the tool which has finished operation is brought to the indexing position in the tool changer and put back on the tool magazine, then the tool magazine rotates to bring the next tool to the indexing position of the tool changer for the next operation. We can consider the tool magazine as a circular disk with tools in slots arranged at equal intervals (see Figure 1).

The indexing problem has been studied in the literature in two variants. In the first variant, only one copy of any tool is allowed in the tool magazine. This has been studied in Wilson (1987), Dereli et al. (1998), Dereli and Filiz (2000), Velmurugan and Victor Raj (2013), and Ghosh (2016). For these problems, the indexing problem is essentially a quadratic assignment problem (see, e.g., Ghosh 2016). In the second variant, multiple copies of a tool is allowed in the magazine. This problem has been studied in Baykasoğlu and Dereli (2004) and Baykasoğlu and Ozsoydan (2015). In these problems, the computation of the indexing cost given a tool magazine configuration becomes non-trivial and involves computing a shortest path in a graph. Problems closely related to the indexing problem have been studied in Aktürk and Ozkan (2001), Avcı and Aktürk (1996), Hertz et al. (1998),

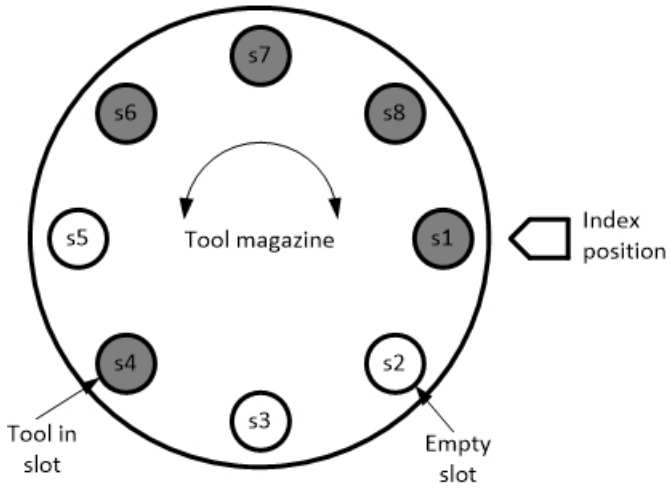

Figure 1: Schematic representation of a tool magazine

and Sinriech et al. (2001). 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.

Formally the indexing problem is defined as follows. We are given a tool magazine with n slots placed at equal angular intervals in the magazine. We are given the sequence of tools required for different operations on a job. The total number of tools required to complete the job is m, with $m \leq n$. Based on the sequence of jobs we can obtain, for each pair of tools i and j, the frequency f_{ij} with which they are used in consequent operations on the job. Since the slots are at equal intervals on the magazine, the rotation of the tool magazine to change tools located at positions k and k in the magazine can be quantified as an integer multiple of the rotation required to move one slot the magazine to the position occupied by an adjacent slot. Taking this amount of rotation as an unit, the rotation required to move between slots k and l is $d_{kl} = \min\{|l-k|, |l+n-k|\}$ units. We are required to find an (into) assignment of m tools $(1, \ldots, m)$ to slots $(\pi(1), \ldots \pi(n))$ such that the total amount of rotation $\sum_{i=1}^{m-1} \sum_{j=i+1}^{m} f_{ij}\, d_{\pi(i)\pi(j)}$ is minimized.

We are interested in metaheuristic techniques for the indexing problem. Most of the literature presenting metaheuristic techniques for this problem have used population based metaheuristics. For example, Dereli et al. (1998) and Dereli and Filiz (2000) have used genetic algorithms and Velmurugan and Victor Raj (2013) have used particle swarm optimization. Recently Ghosh (2016) presented the first neighborhood search algorithm for this problem. In this paper, a tabu search algorithm using a 2-exchange neighborhood function was proposed.

In the current paper, we plan to explore another more powerful neighborhood structure for the problem called the Lin Kernighan neighborhood structure. The Lin Kernighan neighborhood structure is a composite neighborhood structure comprising multiple moves from simple neighborhood structures. It was first proposed in Lin and Kernighan (1973) to solve the traveling salesman problem and was used to generate the best known solutions to large size traveling salesman problems for a long time. An efficient implementation of the neighborhood structure was topical even in 2000 (see, e.g., Helsgaun 2000). The neighborhood structure was also used to solve other problems effectively, e.g., for graph

partitioning and VLSI design (see, e.g., Kernighan and Lin 1970, Ravikumar 1995) and for the single row facility layout problem (see, e.g., Kothari and Ghosh 2013).

In the next section, we describe the Lin Kernighan neighborhood structure applied to the indexing problem and present two local search algorithms and a tabu search algorithm using the neighborhoods. Section 3 provides the results of our computational experiments with our algorithms on realistic size instances of the indexing problem. Finally we summarize the contribution of this paper in Section 4.

2 Lin Kernighan Neighborhood and our Algorithms

The 2-exchange neighborhood used in Ghosh (2016) is a natural neighborhood structure for the indexing problem. In this neighborhood, a solution is defined as an assignment of tools to slots in the tool magazine, and a neighbor of a solution is defined as a solution that is obtained by exchanging two tools in the tool magazine or by moving a tool to a vacant slot in the magazine. The Lin Kernighan (LK) neighborhood is one obtained by performing a number of such exchanges in one go to obtain a neighbor of the initial solution. Since a move in an LK neighborhood is achieved through performing multiple moves in a simple neighborhood (the 2 exchange neighborhood in this case) the LK neighborhood is said to be a composite neighborhood. The maximum number of constituent moves is generally specified when describing an LK neighborhood.

Let us describe the process of selecting a best neighbor of a solution in a LK neighborhood of a solution x_0 using a 2 exchange neighborhood as the simple neighborhood in which a maximum of $n_{\max}$ component moves are allowed. The problem we are addressing is a minimization problem. Let x_1 be a best 2 exchange neighbor of x_0. The cost of x_1 could be more, equal, or less than that of x_0. Similarly let x_2 be a best 2-exchange neighbor of x_1 and so on. Now let $z(x)$ denote the cost of solution x, and denote the difference $z(x_{i-1}) - z(x_i)$ by $\Delta(i)$ $i = 1, \ldots, n_{\max}$. Let $i^* = \arg \max_{1 \le i \le n_{\max}} \{\Delta(i)\}$. A best neighbor of x_0 then is x_{i^*}.

As an example, consider a solution x_0 for an indexing problem. Suppose we want to find a best LK neighbor of x_0 when the maximum number of simple 2 exchange moves (i.e., $n_{\max}$ is 6. The objective function values of solutions x_0 through x_6 as described in the paragraph above is depicted in Figure 2. Note that the best value of $\Delta(\cdot)$ is for $i^* = 5$. Hence we choose x_5 as the best neighbor of x_0 in the LK neighborhood.

In order to make the LK neighborhood search efficient, one needs to search the 2 exchange neighborhood of a solution efficiently. Note that there are $\mathcal{O}(n^2)$ 2 exchange neighbors of a solution to the indexing problem. Computing the cost of a solution is itself $\mathcal{O}(n^2)$, and hence a naïve approach to evaluating a best 2 exchange neighbor of a solution requires $\mathcal{O}(n^4)$ time. However, the following technique proposed in Ghosh (2016) reduces this time to $\mathcal{O}(n^3)$.

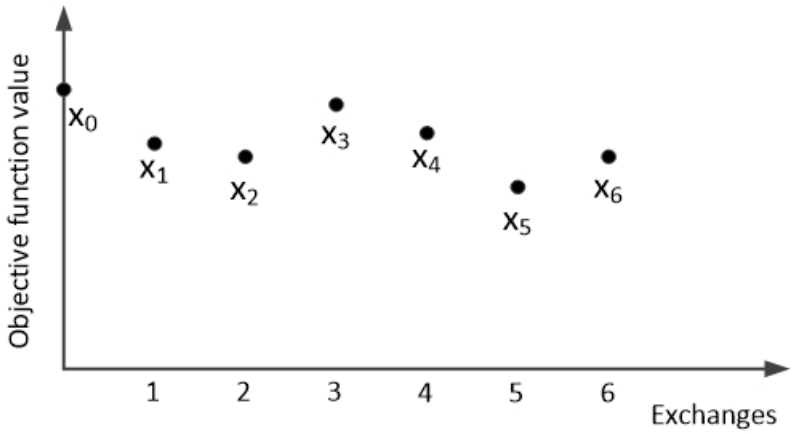

Figure 2: Illustrating a LK move for a minimization problem

Consider a solution x_0 and its neighbor x obtained by interchanging tools in positions r and s ($r < s$ w.l.o.g.). Then

$$\begin{aligned} z(x) - z(x_0) =& \text{cost of repositioning } r + \text{cost of repositioning } s \\ =& \Big(\sum_{\substack{i=1 \\ i \neq r,s}}^{n} f_{x_0(i)x_0(s)} d_{ir} - \sum_{\substack{i=1 \\ i \neq r,s}}^{n} f_{x_0(i)x_0(r)} d_{ir} \Big) + \Big(\sum_{\substack{i=1 \\ i \neq r,s}}^{n} f_{x_0(i)x_0(r)} d_{is} - \sum_{\substack{i=1 \\ i \neq r,s}}^{n} f_{x_0(i)x_0(s)} d_{is} \Big) \\ =& \sum_{\substack{i=1 \\ i \neq r,s}}^{n} \Big(f_{x_0(i)x_0(r)} - f_{x_0(i)x_0(s)} \Big) \Big(d_{is} - d_{ir} \Big). \end{aligned} \tag{1}$$

This expression can clearly be computed in linear time, which implies that computing a best (i.e., lowest cost) 2 exchange neighbor of a solution takes $\mathcal{O}(n^3)$ time. This in turn means that computing $\delta_i = z(x_{i-1}) - z(x_i)$ requires $\mathcal{O}(n^3)$ time. Since $\Delta_i = \sum_{j=1}^{i} \delta_j$, the solution x_{i^*} can be obtained in $\mathcal{O}(n^3)$ time.

The following is a pseudocode for a LK iteration using 2 exchange neighborhood as the simple neighborhood. It assumes a feasible solution x_0 and uses two vectors, `moves` that stores the sequence of simple moves made during the iteration, and `gains` that stores the cumulative gain made from the cost of x_0 during the iteration. LKDepth is the maximum number of simple 2 exchange moves allowed in a LK iteration. x_N is the best neighbor of x_0 in its LK neighborhood.

LK-Iteration

Input solution x_0, cost function $z(\cdot)$
Output best neighbor x_N of x_0 in its LK neighborhood

1. begin
2. set `moves` $\longleftarrow (\emptyset, \emptyset, \ldots, \emptyset)$, `gains` $\longleftarrow (-\infty, -\infty, \ldots, -\infty)$, $x \longleftarrow x_0$;
3. for `iter` from 1 to `LKDepth` do begin
4. set $\Delta^{\max} \longleftarrow -\infty$;

```
5.          for each 2 exchange neighbor x_ij of x obtained by interchanging contents
            of slots i and j in x do begin
6.              compute the cost difference δ_ij = z(x_ij) − z(x) using equation (1);
7.              if δ_ij > Δ^max then begin
8.                  set Δ^max ⟵ δ_ij; i* ⟵ i; j* ⟵ j;
9.              end
10.         end
11.         set moves[iter] ⟵ (i*, j*);
12.         if (iter = 1) then set gains[iter] ⟵ Δ^max;
13.         else set gains[iter] ⟵ gains[iter − 1] + Δ^max;
14.     end
15.     g^max ⟵ −∞, posn ⟵ −1;
16.     for iter from 1 to LKDepth do begin
17.         if gains[iter] > g^max then begin
18.             g^max ⟵ gains[iter]; posn ⟵ iter;
19.         end
20.     end
21.     x_N ⟵ x_0;
22.     for iter from 1 to posn
23.         interchange contents of the two components of moves[i] in x_N;
24.     return x_N;
25. end
```

When one implements the LK neighborhood search described above, one often sees the costs of consequent 2 exchange neighborhoods follow a pattern shown in Figure 3. In particular the search obtains a 2 exchange neighbor with a low value, and then the search oscillates between solutions with that value and with a value higher than that value. The

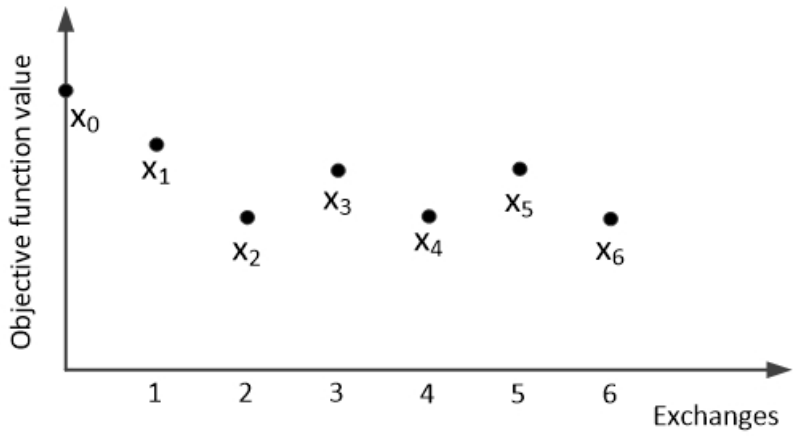

Figure 3: Pattern of best neighboring solutions

reason for this pattern is that the solution with the low value is a local minimum for the

2 exchange neighborhood. In the next iteration, the search finds the best solution in the 2 exchange neighborhood of the local minimum, which has an objective function value more than that of the local minimum. In the next iteration, then search finds the best neighbor of this solution and reaches the local optimum it reached two 2 exchange iterations back. This continues repeatedly giving rise to the pattern.

We modify the search in the LK neighborhood slightly to eliminate this problem. In our new search, once a pair of tools have been exchanged to generate a solution x_i which is a best neighbor of another solution x_{i-1}, then this pair of tools can not be interchanged again to obtain a best neighbor. It is clear that the oscillation that we mentioned in the previous paragraph cannot occur if we adopt this search process. In practice this is achieved by banning 2 exchange moves in `moves` from the loop from step 5 through 10 in the previous pseudocode. We call the modified neighborhood that arises as a result of this modification in the search process the LK1 neighborhood and shall assume a LK1-Iteration procedure that takes in a solution x_0 and returns a best neighbor x_N in the LK1 neighborhood of x_0.

We use both the neighborhoods in local search algorithms. We call the local search algorithms LS-LK and LS-LK1 based on the neighborhood that they use. Both the local search algorithms start with a randomly generated solution as the current solution, and iteratively improve it by searching its neighborhood for a better solution and replacing the current solution with the better solution. It stops when the neighborhood of a current solution does not contain a solution better than the current solution.

We also describe a tabu search algorithm based on the LK1 neighborhood structure. The tabu search algorithm is a generic algorithm (such as the one described in Ghosh (2016)). Since the LK1 neighborhood is a composite neighborhood comprising several simple 2 exchange moves, the maintenance of the tabu list is more complicated. A move in the LK1 neighborhood in our tabu search is said to be tabu if at least one of its component 2 exchange moves is in the tabu list. Once a move has been adopted in the tabu search, all its component 2 exchange moves are marked tabu for a number of iterations.

TS-LK1

Input: A solution x_0, 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` $\longleftarrow x_0$; `TABU` $\longleftarrow \emptyset$; `iter` $\longleftarrow 1$; `CurrSol` $\longleftarrow x_0$;
3. while (`iter` $\longleftarrow$ `MAXITER`) do begin
4. `IterBestSol` $\longleftarrow$ `CurrSol`;
5. for (every possible LK1 neighbor `NbrSol` of `CurrSol`) begin
6. if (none of the 2-exchange moves required to move from `CurrSol` to `NbrSol` are in `TABU` and z(`NbrSol`) $<$ z(`IterBestSol`)) then
7. `IterBestSol` $\longleftarrow$ `NbrSol`;
8. add the 2-exchange moves required to move from `CurrSol` to `NbrSol` to `TABU` for the next `TENURE` iterations;

```
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.
```

3 Computational Experiments

We coded the two local search algorithms (LS-LK and LS-LK1) and the tabu search algorithm (TS-LK) 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. The value of `LKDepth` in all three implementations was set to 6. For TS-LK1, the tabu tenures, i.e., the number of tabu search iterations for which a 2 exchange move was tabu was generated from a uniform distribution with support $[2, 5]$, and the maximum number of iterations `MAXITER` was set at 335. This was chosen for the following reason. The implementation of the TS algorithm in Ghosh (2016) which we compare our implementations with has `MAXITER` = 2000, and since we chose `LKDepth` as 6, we chose the maximum number of iterations for TS-LK1 as a value close to 2000/6. For each of the algorithms we choose the best among the outputs from 20 starts of the algorithms. (Each algorithm chooses the same starts though.)

We use two sets of benchmark instances to compare our implementations with the implementations in Ghosh (2016). These two sets are standard benchmark instances for the single row facility layout problem, but can be used without modification for the indexing problem. The advantage of these instances is that they are large enough to be realistic.

3.1 Anjos Instances

The first set of instances are the 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. The dataset has 20 instances, a set of five instances with 60 tools, a second set of five instances with 70 tools, a third set of five instances with 75 tools and a fourth set of five instances with 80 tools. In our computations we assume a tool magazine with 100 slots for our experiments. Table 1 presents our results for these instances. The first three columns in each row present the name of the instance, the number of tools in the instance and the number of slots in the

tool magazine. The next four columns report the costs of the best solutions found by four algorithms, the TS algorithm from Ghosh (2016), the LS-LK, the LS-LK1, and the TS-LK1 algorithms. The last four columns report the execution times for 20 starts of each of the four algorithms starting from randomly generated initial solutions on our machines.

From the results we see that TS-LK1 matches each of the solutions obtained by TS (from Ghosh 2016). The time taken by TS-LK1 is on average about two times the execution time of TS. The local search algorithms however are much faster. There is no significant difference in the times required by LS-LK and LS-LK1 and both of them require approximately 5% of the time required by TS. The quality of solutions output by LS-LK1 were marginally better than those by LS-LK for the Anjos instances; LS-LK generated solutions that were worse than those generated by TS in five Anjos instances, whereas the corresponding number was only two for LS-LK1.

3.2 sko Instances

The next set of instances are the sko instances. They have been used in Anjos and Yen (2009) for computational experiments for the single row facility layout problem. There are seven sko instances ranging from instances with 42 tools to instances with 100 tools. 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 2 presents our results for these instances. The structure of the table is the same as that of Table 1.

Our computational experience with the sko instances is very similar to that with the Anjos instances. LS-LK and LS-LK1 were equally fast for these instances, requiring approximately 5% of the time required by TS, while TS-LK1 took twice as much time as TS. LS-LK output the worst results, while LS-LK1 and TS-LK output the same solution as TS for all the sko instances.

Hence, our experience suggests that TS is still the best method for generating good quality solutions to indexing problems. LS-LK1 produces very good quality solutions in much less time, and so it may be an algorithm to use if there are time constraints.

4 Summary

In this paper, we study the performance of local search and tabu search algorithms for the indexing problem. We use the Lin Kernighan neighborhood structure for our algorithms. This neighborhood structure has been known to provide good solutions for the traveling salesman problem and the graph partitioning problem.

We introduced the indexing problem and provided a brief literature review in Section 1. In Section 2 we described the Lin Kernighan neighborhood, and local search algorithms (LS-LK and LS-LK1) and a tabu search algorithm (TS-LK1) using the Lin Kernighan neighborhood to solve the indexing problem. We described our implementations of the algorithms and our computational experience with these algorithms in Section 3. Based on our experiments, we feel that tabu search with 2-exchange neighborhoods is the best

Table 1: Performance of algorithms on Anjos instances

			Cost				time (cpu seconds)			
Instance	tools	slots	TS*	LS-LK	LS-LK1	TS-LK1	TS*	LS-LK	LS-LK1	TS-LK1
an-60-01	60	100	54053	54053	54053	54053	179.468	9.171	8.953	350.296
an-60-02	60	100	31278	31278	31278	31278	180.109	9.203	9.218	350.077
an-60-03	60	100	23510	23514	23510	23510	180.109	8.546	8.593	350.312
an-60-04	60	100	11592	11592	11592	11592	178.235	8.843	8.796	350.233
an-60-05	60	100	15182	15182	15182	15182	178.233	8.921	8.906	349.500
an-70-01	70	100	42297	42299	42297	42297	178.452	9.811	9.875	350.640
an-70-02	70	100	51723	51723	51723	51723	178.046	10.312	10.218	349.328
an-70-03	70	100	43795	43795	43795	43795	178.500	9.921	10.031	349.749
an-70-04	70	100	27705	27705	27705	27705	178.842	10.172	10.453	349.421
an-70-05	70	100	134269	134269	134269	134269	181.483	9.890	9.812	349.718
an-75-01	75	100	66656	66656	66656	66656	182.202	10.515	10.359	350.546
an-75-02	75	100	111806	111850	111850	111806	181.454	10.765	10.609	348.640
an-75-03	75	100	38158	38158	38158	38158	181.859	10.765	10.359	350.046
an-75-04	75	100	106341	106341	106341	106341	179.531	11.546	11.343	349.735
an-75-05	75	100	47031	47031	47031	47031	179.140	11.078	10.968	350.452
an-80-01	80	100	54463	54463	54463	54463	178.858	11.062	11.031	349.312
an-80-02	80	100	52853	52853	52853	52853	179.015	11.265	11.625	349.515
an-80-03	80	100	95091	95091	95091	95091	182.609	11.608	11.328	348.452
an-80-04	80	100	100950	100951	100950	100950	184.234	11.374	11.062	348.531
an-80-05	80	100	36227	36228	36228	36227	185.530	11.390	11.312	349.703

*: Tabu search algorithm from Ghosh (2016)

Table 2: Performance of algorithms on sko instances

			Cost				**time (cpu seconds)**			
Instance	tools	slots	TS*	LS-LK	LS-LK1	TS-LK1	TS*	LS-LK	LS-LK1	TS-LK1
sko-42	42	60	24408	24408	24408	24408	38.281	1.265	1.234	73.406
sko-49	49	60	36582	36582	36582	36582	38.078	1.421	1.484	73.109
sko-56	56	60	52974	52974	52974	52974	38.030	1.671	1.687	72.937
sko-64	64	100	95337	95337	95337	95337	182.062	10.390	10.281	349.734
sko-72	72	100	133607	133632	133607	133607	182.561	11.311	11.156	349.874
sko-81	81	100	185852	185852	185852	185852	183.078	12.531	12.109	349.655
sko-100	100	100	290496	290517	290496	290496	185.000	14.953	15.328	350.187

*: Tabu search algorithm from Ghosh (2016)

algorithm to use if we have enough time. For a fast and good solution we suggest local search using a modified Lin Kernighan neighborhood (LS-LK1).

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.

D. Ghosh. Allocating tools to index positions in tool magazines using tabu search. IIMA Working Paper Series, W.P. No. 2016-02-06 (2016) pp. 1–14.

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.

K. Helsgaun (2000). An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. European Journal of Operational Research 126 (2000) pp. 106–130

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.

B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 49 (1970) pp. 291–307.

R. Kothari and D. Ghosh. Insertion based LinKernighan heuristic for single row facility layout. Computers & Operations Research 40 (2013) pp. 129–136.

S. Lin and B.W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research 21 (1973) pp. 498–516.

C.P. Ravikumar. Parallel methods for VLSI layout design. Greenwood Publishing Group. p. 73.

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.