TABU SEARCH ALGORITHM FOR BATCH PLANT SCHEDULING - MATLAB IMPLEMENTATION

Martin Žďánský, Jaroslav Poživil

Department of Informatics and Control Engineering

Prague Institute of Chemical Technology

1 INTRODUCTION

Scheduling and sequencing is a part of the real-life requirements on batch plant management. Furthermore, a scheduling system should be an integral part of a control methodology, because it influences the production at many levels. At one end it provides the management with guidance for long- and medium term decisions, at the other end it should provide support for operational level decisions. Such a system requires a scheduling algorithm at its core, and the one we have tested is the tabu search. From the various categories of batch processes, in this paper we deal with a problem from the field of multiproduct plants. Such plant can be described as a periodical manufacturing of many batches of distinct products in a series of campaigns. Solving multiproduct batch plant production scheduling problem consists of two sub-problems. The first one, hierarchically a higher one, is that of finding an optimum sequence of batches. The second sub-problem is the problem of determining start-times and completion-times for all operations, i.e. creating a detailed schedule for a known sequence of batches.

Finding an optimum sequence of batches is a typical NP-complete problem. Computation time increases exponentially with increasing dimension of input. Tabu search algorithms, introduced by F. Glover in late 80's [Glover, 1989; Glover and Laguna, 1997], can be efficiently used to solve this problem. In contrast with simulated annealing and genetic algorithms, tabu search is often a deterministic method, although it should be noted that many methods containing randomizing elements have been proposed. For this reason, finding optimum solution is not, for many of the algorithm variants, guaranteed even if the number of iterations approaches infinity. Therefore, it is even more that with other methods necessary to find acceptable parameter values for different formulations of problems.

The goal of this work was to create in Matlab a tabu search algorithm for a specific class of combinatorial problems, to determine its effectiveness for solving flowshop sequencing problem, and to find good parameter values and algorithm configuration; the objective function we employ is the minimization of makespan.

2 MODEL OF A MULTIPRODUCT BATCH PLANT

Multiproduct batch plant can be modeled using different approaches; the model we use is a model of time requirements. Such models can vary in the level of details they incorporate; the model we use requires following input data:

- a set of n manufactured batches
- a set of m stages, forming an unchanging sequence of operations
- processing times matrix T_{ij}, whose elements correspond to processing times for operation j and batch i

- storage policy - results presented in this work were obtained under four different assumptions: unlimited intermediate storage, finite intermediate storage, no intermediate storage policies, zero wait
- vector *Storage*, containing numbers of temporary storage units between processing units, used only under FIS policy

Processing time consists of following sub-parts:

- batch *i* transfer time from preceding stage $a_{i,j-1}$
- operation t_{ij} itself
- transferring batch *i* into next stage a_{ij}; under FIS policy this is also used for transfer between stage and storage unit
- setting-up empty stage *j* for batch *i+1* (cleaning, adjusting etc.) $S_{i(i+1)j}$

It is important to bear in mind that this last sub-part is sequence-dependent, and therefore it increased the difficulty of the problem compared to problems with zero or fixed setup times.

The goal of optimization is to determine a batch sequence so as to minimize completion time. The sequence the batches are processed in on various units can vary, but then the resulting problem is with $(n!)^m$ possible solutions very difficult to solve. In this work we use only simplified, permutation schedules. The permutation schedule limits the search space, assuming that the sequence of batches is identical on all processing units, and therefore the optimization is less difficult, as the size of the search space drops from $(n!)^m$ for non-permutation schedules to $(n!)$ for permutation schedules.

3 TABU SEARCH HEURISTICS

To keep this article short, only a brief description of tabu search algorithm is included. It is possible to find some additional information on this subject in the works mentioned in references.

Tabu search is a methodology for searching a discrete solution space. The algorithm is descended from downhill search, and while deterministic (at least in its basic variant), it is often classified as stochastic because of its properties and behavior. Many of the variants of the algorithm incorporate some random elements and are truly stochastic. Unlike downhill search it descended from, the tabu search algorithm is able to leave a local optimum and to continue the search. Rather than being a single algorithm, the tabu search would be better described as a set of concepts that the algorithms falling under this heading share. For this reason, there is no single algorithm called tabu search, and the discussion here is limited to the algorithms we have used in our work. Tabu search heuristics starts from an initial solution, at each step such a move to a neighboring solution is chosen to hopefully improve objective criterion value. This is close to a local improvement technique except for the fact that a move to a solution worse than the current solution may be accepted. The algorithm tries to take steps to assure that the method does not re-enter a solution previously generated which serve as a way of avoiding becoming trapped in local optima. The variants of the algorithm we use accomplish this by recency-based or frequency-based data structures that contain the information on the moves that are discouraged at the current iteration. The algorithm is, in many variants, not guaranteed to find an optimum solution; however, even if not successful it usually does find good near-optimum solution. As finding an optimum solution is not guaranteed even if the number of iterations approaches infinity, the optimization must be stopped by means of some external computation termination criterion. From the point of actual use, the advantages of tabu search algorithm include easy implementation and an ability to find optimum or good near-optimum solution to NP-hard problem in polynomial time. The algorithm is fast enough to remain applicable even to fairly large problems.

Solutions to solved problem are encoded in the form of a string of unique identifiers, each identifier corresponding to one specific batch. The completion time, used as an objective criterion, can be for all listed interstage storage policies computed for all valid strings. As most constraints are already incorporated into functions computing these completion times, this means that only one condition must be met for a string to represent valid solution: every batch identifier must exist in the string in exactly one copy. Our tests show that even if tabu search algorithm starts from different initial solutions, it quickly moves towards optimum or good sub-optimum areas of search space. This allows us to use random solutions as initial solutions.

The neighborhood of a solution *y* is a group of solutions that can be arrived at by application of the defined transformation on the solution *y*. Different neighborhood definitions exist, depending on problem solved and author's choice, but all such definition should guarantee that the solutions obtained by the application of the transformation on a solution *y* yields valid solution. In our work, we have tested various neighborhood definitions, and based on these tests we use the general pairwise interchange as the transformation rule. Other tested rules were move of one identifier, adjacent pairwise interchange, swap of three identifiers, move of a segment and reversion of sequence in a segment. General pairwise interchange defines the neighborhood of solution *y* as all the solutions that can be generated from *y* by swapping two identifiers in the string encoding *y*. As the rule is applied to only one string, and all identifiers in the string are unique, it is guaranteed that resulting solutions are valid and different from solution *y*.

4 COMPUTATIONAL RESULTS

4.1 TEST PROBLEMS

A total of 110 test problems, consisting of 11 groups of 10 problems each, were created. These groups differed by problem dimension, i.e. number of batches and stages (*n*/*m*): 5/10, 5/15, 5/20, 8/4, 10/5, 10/15, 10/20, 15/5, 15/10, 20/5, and 20/10. In this paper we present only results of optimized algorithm, for larger problems; for problems of size 8/4 or less the algorithm was able to find optimum solution in all cases under all storage policies. Processing times were obtained using random number generation, as is common in published works. Processing times values were generated in range 0-24, set-up and transfer times in range 1-4, and the numbers of intermediate storage units allocated to stages in range 0-1. If a processing time is zero, the product in question is not processed in this stage. Every test problem was solved 10 times (i.e. 100 tests per group), using steepest descent technique, for tabu list length 7, and these tests were then repeated, with varying relevant parameter values, using fastest descent technique (the same tabu list length values), using long-term memory, and with reduced neighborhood.

4.2 COMPUTATION TERMINATION CRITERION

Two different criteria are often used for terminating calculation:

a) calculation stops when pre-set time limit is reached, best solution found during this time is considered to be an optimal one

b) calculation stops when the maximum number of consecutive steps not improving objective function value is reached

The second one of these criteria was selected, because it is not dependent on computer speed and better reflects the fact that the objective criterion of an optimal solution is not known. Tests showed us that for the problems we study the probability of objective function value improvement is almost zero after 500 unsuccessful iterations. Therefore, the computation was stopped after meeting the 500 unsuccessful steps criterion in all cases presented in this article. For better results, the simple rule we often used is to set this value to n*100.

4.3 RESULTS EVALUATION

Results obtained by solving test problems using different techniques and parameter values were classified using mean percentage deviation. Presented values correspond to mean percentage deviation in cases of problems with 5 and 8 batches where it's possible to find global optimum using complete enumeration technique. For more complex problems, best solution found during test runs of a problem is used in place of the global optimum. It should be noted that as we also used genetic algorithms and simulated annealing to solve these problems, some of the best solutions found were solutions obtained using these other approaches; see [4] and [5] for details on respective algorithms.

4.4 TABU LIST LENGTH AND LOCAL OPTIMIZATION TECHNIQUE

For practical purposes, correct setting of tabu list length L is of high importance. Setting this value too small can lead the algorithm to cycling through a fixed sequence of moves. Using too long a tabu list means that it often rejects promising moves, lengthening the computation as a result. Mean percentage deviations as functions of tabu list length are shown in Fig. 1. The figure suggests that, for the General Pairwise Interchange rule, the optimal tabu list length depends on the dimension of the problem. Taking into consideration the computation time increase related to lengthening the tabu list, and the need to compare the results for different problems, a length L=7 was chosen as a sufficient one and was used in all subsequent computations unless mentioned otherwise. This value gives good results for all problems from 8/4 to 20/10; increasing the L for larger problems leads to small improvement in algorithm performance, but such change makes comparing the results more difficult.

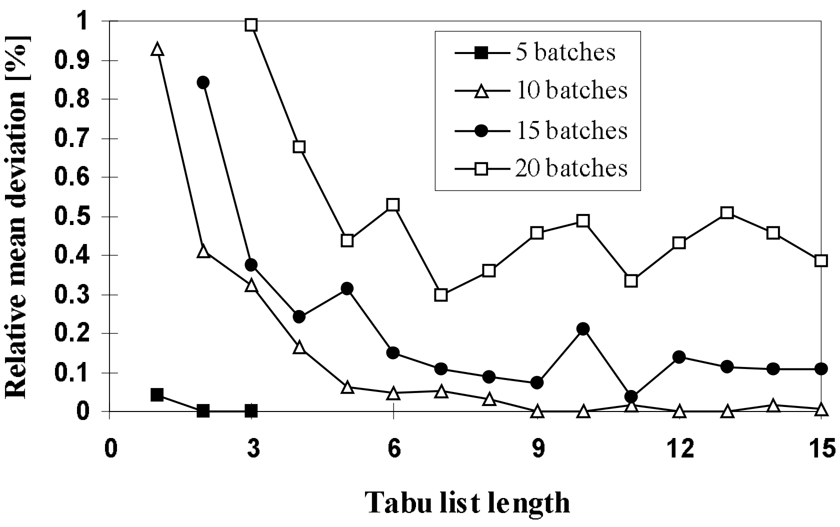

Fig. 1 Dependence of mean percentage deviation on the tabu list length

Selecting a next move from a neighborhood can be done using two different approaches. Steepest descent technique, common in local optimization algorithms, searches whole neighborhood and then selects the move with best objective function value. This means considering $n(n-1)/2$ possible moves for every algorithm step, which greatly increases computation time for higher values of n. An alternate method is the fastest descent technique, which means using the first move leading to objective function value improvement. Advantages of this technique are evident during the beginning steps of the algorithm, but computation time requirements later increase again as ways for quick improvements disappear. Tests show that steepest descent algorithm is marginally better for most tabu list lengths, and the technique was used in all subsequent computations, unless mentioned otherwise.

The question of number of stages influences the performance of the algorithm is can be divided into two issues: algorithm speed and solution quality. The influence on algorithm speed should be evident: while the algorithm itself uses only the sequence of batches, and therefore is not influenced, the objective criterion computation lengthens with increasing number of stages. The solution quality decreases with increasing number of stages: when the number of stages doubles from 5 to 10, the mean percentage deviation rises by approximately 50%.

4.5 LONG-TERM MEMORY

Inclusion of some form of long-term memory is an interesting modification of basic tabu search algorithm. Transformation usage frequency $\omega(t)$ is used as a replacement for short-term memory. This function is used as a penalty component of an objective function when heuristics search the neighborhood for next move. This modified objective function can be expressed as follows: $f(x)+\alpha\omega(t)$, where α is a weight to be set empirically. The weight α moves the algorithm from pure downhill search ($\alpha=0$) towards exploration of search space, its optimum value depending on problem in question. The main problem of long-term memory algorithm is the need to store usage frequencies of all transformations, number of which increases quadratically with the increase of n.

Choosing fitting value of the weight α is very important, because for a value of $\alpha=0$ heuristics degrades to a simple downhill local optimization, stopping in first local optimum found. Fig. 2 shows averaged results for all problem dimensions. It should be evident that selecting optimal value of α is very difficult. To get better resolution, results can be split into 4 groups according to problem dimension, as shown in Fig. 3. Nevertheless, no improvement in multiproduct batch plant scheduling results was observed, and based on this result, long-term memory concept was not tested in more detail.

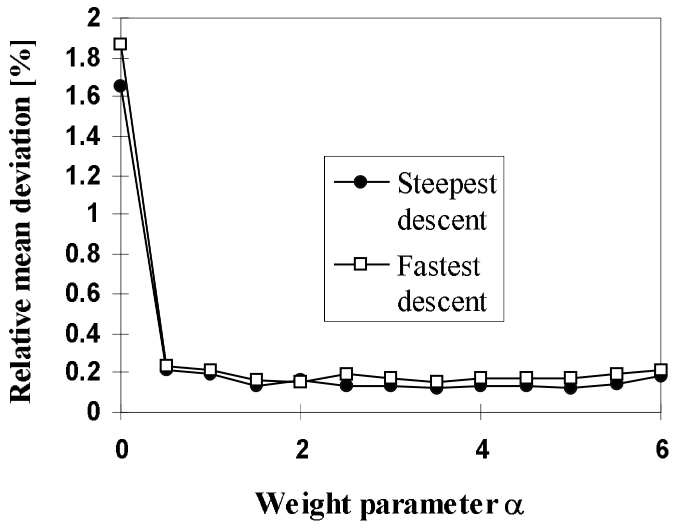

Fig. 2 Influence of the long-term coefficient α

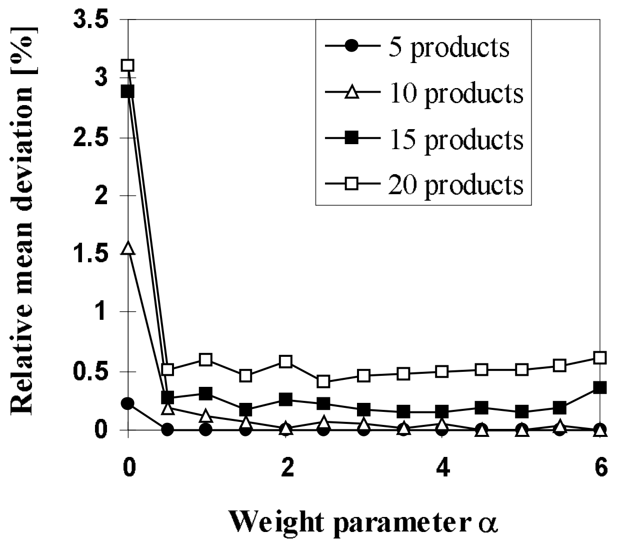

Fig. 3 Influence of long-term memory for various problem dimensions

4.6 SEARCHED NEIGHBORHOOD REDUCTION

An interesting idea for improving tabu algorithm efficiency was proposed by Reeves (1993). Using steepest descent technique for selecting next move requires searching whole neighborhood. This decreases algorithm efficiency in case of large-dimension problems, and since using fastest descent technique does not improve algorithm performance too, it seems to be worth it to somehow reduce searched portion of neighborhood, regardless of a certain increase in a number of iterations. Such reduction can be, when the principle is applied to flowshop sequencing problem, achieved by choosing a sub-set of p batch identifiers in the solution string $\{x_1, x_2, ..., x_p\}$, $p<n$. Steepest search technique is then performed on this sub-set only. In the next step, sub-set $\{x_{p+1}, x_{p+2}, ..., x_{2p}\}$ is used and this process is repeated until the last identifier x_n is reached. In addition to this Reeves suggests to randomly exchange variables in sub-sets after reaching set time. Simpler method was used in this work. The length p of the sub-set is set at the beginning of a computation, and in every iteration p positions in the solution string are randomly chosen (from the set of $\{x_1, x_2, ..., x_n\}$) to fill the searched sub-set. This approach leads to certain randomization, which seems to improve search efficiency in cases of complicated areas. It seems logical to expect that optimal value of p depends on the dimension of problem n, so a parameter $\varphi = p/n$, describing how large a portion of n variables the sub-set of length p contains, is often defined.

For a more precise description, we introduce another parameter defining the degree of reduction. This parameter, ψ, representing the number of evaluated sequences compared with whole neighborhood, is defined as follows:

$$\psi = \frac{p(p-1)/2}{n(n-1)/2} \qquad (1)$$

Mean percentage deviation as a function of the parameter ψ is shown in Fig. 4. Best results are achieved for values of ψ close to 1, which means almost no reduction in searched neighborhood. Results for problems with 15 and 20 batches together are in Fig. 5 compared to results for problems with 30 batches. The figure shows that reducing searched neighborhood to 1/2 is advisable for a problem with 30 batches, but this might be caused by influence of other algorithm settings.

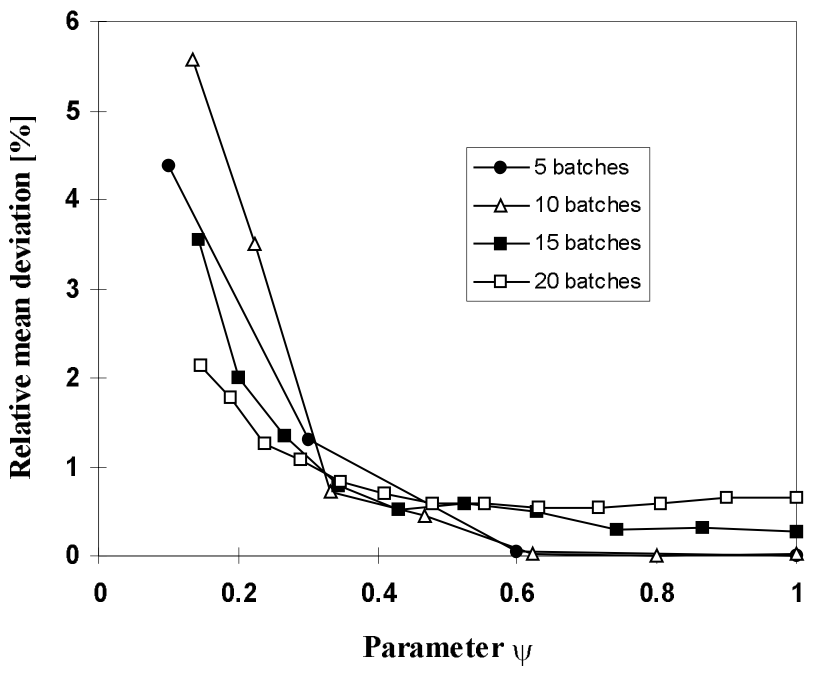

Fig. 4 Reduced neighborhood results for different values of coefficient ψ

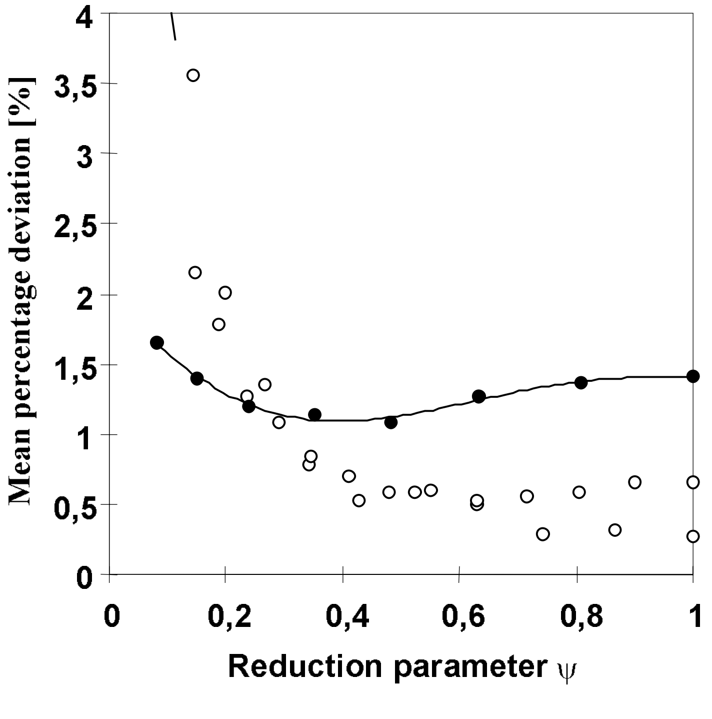

Fig. 5 Averaged reduced neighborhood results for different values of coefficient ψ

5 CONCLUSION

This work deals with multiproduct batch plants production sequencing. Tabu search method analyzed in this work was implemented in Matlab, its function was verified and the algorithm's configuration and parameters were set. In the chosen variant it proved to be successful, as it finds optimum schedules in cases of problems with small number of batches, and it finds optimum or good sub-optimum schedules in cases of large problems. Computation times are, when the algorithm is implemented in Matlab, significantly lower compared to standard MILP/MINLP techniques. The best results are obtained using steepest descent technique together with short-term memory in the form of tabu list of L=7, although lengthening tabu list is advisable for problems of larger size. Other modifications seemed not to offer better results and therefore no further research was directed that way.

Acknowledgements

This work has been supported by the Ministry of Education of the Czech Republic (program No. MSM 223400007).

References

1. Glover, F., Laguna, M. (1997) Tabu search, Kluwer Academic Publishers
2. Hubsher, R., Glover, F. (1994) Applying tabu search with influential diversification to multiprocessor scheduling, *Computers Ops. Res.* 21, 877-884
3. Kim, M., Jung, J. H., Lee, I. B. (1996) Optimal scheduling of multiproduct batch processes for various intermediate storage policies, *Ind. Chem. Engng*, 35, 4058-4066
4. Reeves, C. R. (1993) Improving the efficiency of tabu search for machine sequencing problems, *J. Opl. Res. Soc.* 44, 375-382
5. Stluka, P. (1998) Application of artificial intelligence methods in the scheduling of chemical batch plants, *Ph.D. Thesis*, Prague Institute of Chemical Technology
6. Widmer, M. (1991) Job shop scheduling with tooling constraints: a tabu search approach, *J. Opl. Res. Soc.* 42, 75-82

kontakty na autory:

Technická 1905, 166 28 Praha 6, Czech Republic

Email: martin.zdansky@seznam.cz, Jaroslav.Pozivil@vscht.cz

Tel.: +420 2 2435 4259, Fax: +420 2 2435 5053