Proceedings of ALGORITMY 2002
Conference on Scientific Computing, pp. 230–236

COMBINATION GENETIC/TABU SEARCH ALGORITHM FOR HYBRID FLOWSHOPS OPTIMIZATION *

M. ŽĎÁNSKÝ AND J. POŽIVIL†

Abstract. The paper describes an algorithm for solving the scheduling problem of a hybrid flowshop (flowshop with multiple processors, FSMP). The algorithm is a combination of genetic algorithm and tabu search, and the batch processes it is applied on are modeled by a model of time requirements. The paper describes the algorithm and compares its performance with other optimization techniques.

Key words. batch processes, scheduling, tabu search, genetic algorithms

AMS subject classifications. 68W99, 68M20, 90B36

1. Introduction. The highest portion of production of chemical commodities comes from continuous processes, but batch processes also have their place in the chemical industry. Batch processes are often found in low-volume manufacture of products such as pure chemicals, specialty chemicals, pigments, pharmaceuticals, etc. The advantages of batch processes include high flexibility that allows to quickly react to changes in demands, to change technology based on current situation, and to quickly introduce new or modified products. Batch processes also allow easier sharing of some of the resources (e.g. production units, storage capacities, manpower). Because a new product is likely to be manufactured on existing equipment, the emphasis is on process planning and control rather than on sizing the equipment. Current trends in abovementioned industries are towards products with shorter life cycles and higher functionality that are tailored to specific market niches, and consequently process development problems are encountered very frequently. The development of new or modified products "from scratch" would be too costly and time-consuming, and so would be a purchase of new equipment, and for these reasons new products utilize existing equipment. This means that the problem of production scheduling in these plants is one of high importance.

Two categories of chemical batch plants are widely recognized: multi-product plants and multi-purpose plants. In a basic serial multi-product plant, called flowshop, the production line consists of a single set of m processing stages, the plant has only one path for all products, and this path consists of a chain of stages where no branches or loops exist. Hybrid flowshops can be derived from the classical multistage flowshop, each stage being composed of one or more identical parallel machines (see Fig. 1.1). Each machine is able to process one job at a time, and at least one stage consists of more than one machine. This paper addresses the problem of finding optimum schedule for a set of n jobs on such a configuration, and applies combination genetic/tabu search algorithm to solve the problem. The objective function is the makespan of a schedule.

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

† Department of Computing and Control Engineering, Faculty of Chemical Engineering, Prague Institute of Chemical Technology, Technická 1905, CZ-166 28 Praha 6, Czech Republic, Tel.: +420 2 2435 4259, Fax: +420 2 2435 5053, (Jaroslav.Pozivil@vscht.cz Martin.Zdansky@vscht.cz)

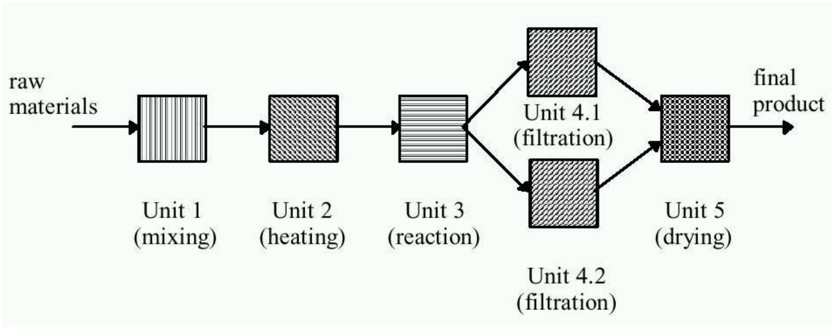

FIG. 1.1. *Example of a Hybrid Flowshop topology*

2. Problem Description. The flexibility of batch plants puts increased demands on production planning and control. A series of campaigns, during which only one or few of the whole range of products are manufactured, is usually a result of mid-term planning. The individual campaigns are the input for the discussed problem of production planning and scheduling (PPS). Efficient plant operation can be reached using different optimization criteria, and one of the most used ones is minimization of makespan; this requires that batches (more often called "jobs" outside chemical industry) enter the flowshop in such an order that operations thorough chain of units are as close to ,,lock step,, as possible. Finding optimum product sequence is an NP-hard problem even for simple flowshops - see Garey [4].

Most of the heuristic algorithms for the m-stage flowshop scheduling problem can be divided into four categories: applications of Johnson's two-machine algorithm, the use of a slope index for the batch processing times, the minimization of idle time on machines, and stochastic techniques such as tabu search, genetic algorithms and simulated annealing algorithms.

Palmer [9] first proposed a heuristic for minimizing makespan in a flow shop scheduling problem. The heuristic generates a slope index for jobs and sequences them in descending order of the index. Campbell et al. [3] developed a heuristic that is a generalization of Johnson's algorithm. Gupta [6] presented the minimum idle time (MINIT) algorithm based on the minimization of idle time at the last machine. Nawaz et al. [7] proposed that a job with larger total processing time should have higher priority in the sequence. More recently, Ogbu and Smith [8] used simulated annealing and Taillard [16] applied tabu search algorithm for makespan criteria.

The hybrid flowshop scheduling has attracted considerable attention in recent years. Both optimizing and heuristic techniques have been used as a solution methodology. The optimizing techniques used so far are mostly branch and bound and mixed integer programming. However, because of the NP-completeness of the problem, heuristics have been, in our view, more popular.

Salvador [11] proposed the branch and bound approach to solve a special case of permutation hybrid flowshop. Brah [2] formulated a mixed integer programming hybrid flowshop model. Brah [1] also discussed the complexity of the problem and establishes that the hybrid flowshop scheduling is indeed an NP-complete problem. Sridhar and Rajendran [14] used simulated annealing approach to minimize the total flow time. Santos et al. [12] developed a global lower bound for the FSMP makespan problem, and the same group of authors published an overview of various heuristics [13].

Most recent developments usually use variants of branch and bound approach, and try to combine them with other methods (e.g. Portmann et al. [10]), but other approaches (e.g. neural networks) are also used.

The difficulty of the problem leads to the fact that most works that try to solve such problems operate under some simplifying assumptions. The assumption we make is limiting the search space to permutation schedules, an approach also used by most other works on this problem.

3. Process Model. Batch processes can be modeled at many different levels of abstraction, and each of these levels is best suited for different purpose. The simplified models we use are based on time requirements of different operations occurring during manufacture. Based on the degree of simplification, different types of models of time requirements are recognized, but most of them are thanks to their nature flexible, applicable to description of most batch processes, because the amount of process-specific elements is zero or minimal. Hybrid flowshop, as defined here, is described by following input data and assumptions:

- a set of n batches manufactured
- a set of m processing stages, arranged into a fixed sequence
- vector U, describing the topology of the plant; in hybrid flowshop at least one processing stage contains more than one processing unit
- processing times matrix T, its elements t_{ij} corresponding to processing time for operation j and product i
- a unit may only process one batch at a time
- once started, a unit must complete the processing of a batch
- all processing units in a single stage are identical in performance
- storage policy - algorithm was tested under one of the commonly used inter-stage storage policies - unlimited intermediate storage (UIS)

A schedule for this problem is an assignment of units to batches that meets all the constraints described above. The makespan of a schedule is the time of completion of the last operation performed on a given set of batches minus the time the first operation began at.

4. Combined Tabu Search/Genetic Algorithm. Our previous works on scheduling problems showed us that both the tabu search and genetic algorithm are suitable tools for solving such problems. Each algorithm, however, has some disadvantages. The algorithm we have developed is a combination of both approaches, and tries to combine the positives of the two methods.

4.1. Tabu Search Principles. Rather than being a single algorithm, the tabu search (TS) would be better described as a set of concepts that the algorithms falling under this heading share [5]. For this reason, there is no single algorithm called tabu search, and the discussion here is limited to algorithms we have used in our work. The algorithm we use is descended from the local improvement techniques, 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 local optimum and continue the search. Tabu search heuristics starts from an initial solution, and 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. Algorithm tries to take steps to assure that the method does not re-enter a solution previously generated which serve as a way to avoid becoming trapped in local extreme. The variant of the algorithm we use accomplishes this by recency-based data structure called tabu list that contains the moves that are discouraged at the current iteration. A move remains a restricted one only for a limited number of iterations. Algorithm is not guaranteed to find optimum solution; however, experimental results show that even if not successful it does find good near-optimum solution.

4.2. Genetic Algorithms. Genetic algorithms (GAs) are a general methodology for searching a discrete solution space in a way that is similar to process of natural selection procedure in biological systems. The algorithm is a remarkably general one, and it can be applied to different problems if following conditions are met:

a) solutions to the problem can be expressed in form of a string of characters
b) a ,,fitness" criterion, which in some way quantifies the quality of a solutions, can be computed for any valid string
c) strings in which ,,part" of a good solution is present are rewarded by allocation of a higher fitness than ,,average" strings

Genetic algorithms, as the name implies, are a type of algorithms, not a single one. This means that many variants of the basic idea exist, and that individual applications may be highly different. However, every variant should include following operations: (1) a method for encoding solutions to the problem into a string of characters; (2) an evaluation function which takes a string as an input and returns a fitness value which measures the quality of the solution the strings describes; (3) an adaptive plan, whose purpose is to produce new, improved generation of solutions from the current one.

The strings encoding the solutions are often binary coded. This encoding, however, is not well suited for our purpose. Instead, the string is composed of a sequence of unique identifiers. Each identifier is represented by an integer number, and identifies a corresponding batch. The use of such an alphabet does not violate the principles of GAs. Encoding the solution in this way is enabled by the fact that the optimization is performed under the assumption of permutation schedules. The strings containing substrings with small makespan generally have smaller total makespan compared to average strings, as required in c), allowing the use of GAs.

4.3. Combination Tabu search/Genetic algorithm. The algorithm combines the principles of the two approaches. At initiation it creates a set of random valid solutions, and for several iterations it optimizes them using tabu search-based method. Then the algorithm applies genetic principles to the set of solutions, and this creates a new generation of solutions. The solutions retained from the previous generation keep the associated tabu lists; new solutions begin with clear tabu lists. The process of several tabu-principles iterations followed by a genetic-principles iteration continues until computation termination criteria are met.

This approach combines the advantages of the two algorithms and mitigates the disadvantages. Pure tabu search that uses only one solution can easily miss some promising areas of the search space, and a larger set of parallel solutions does not exchange information. Genetic algorithms, thanks to the nature of the problem solved, show lower solution quality with increasing problem size; the most prominent cause is the damage to solutions that occurs during solution crossover. The combined algorithm we propose combines the parallelism and information-exchange of genetic algorithms with a strong local optimization of the recency-based tabu search.

Tabu search components of the algorithm are similar to the pure tabu search algorithm we have used in our previous works. We use fixed-length recency based tabu list. The neighborhood generation method we use is pairwise exchange: it exchanges positions of two batches in the schedule, and the move is selected using fastest descent technique.

Genetic components are similar to GA described in [15]. Makespan of solutions in a generation is transformed into fitness values in range 5-100. Parents of a new generation are selected using deterministic reproduction with single-string elitism and stochastic remainder sampling; the process uses the best makespan found since the solution in question was evaluated in genetic iteration, not the current makespan of the current solution. We use the crossover operator proposed in [15], with the addition of a safeguard that allows crossover should all the chosen parents be copies of one solution. In this case the algorithm selects a random other solution in the current generation as the second parrent. Mutation operator used is random pairwise exchange.

In this work the algorithm stops after a pre-set number of consecutive unsuccessful genetic iterations. This means that computation stops when maximum number of consecutive genetic-based steps that do not improve objective criterion value is reached; the best solution that was found during this time is used as the result.

The evaluation function is based on the objective function, i.e. the computation of makespan. Considering that the ranges of processing times, as well as other values, can change for different applications, it is hardly possible to use the raw makespan value, and it must be somehow transformed to allow better algorithm function. Details on this transformation are included later in this paper.

$$Fitness\_new = Makespan_{max} * 1.2 - Makespan \tag{4.1}$$

This new fitness is then transformed to a value in range $< 5, 100 >$ using linear interpolation, with the minimum fitness in a generation equal to 5 and the maximum one to 100.

One of the advantages of heuristic algorithms such as the one we propose is the way the constraints of the problem are treated. As long as neighborhood generation, crossover, and mutation operators guarantee that only valid solutions will be generated, the algorithm itself does not have to take any constraints into account, because all are incorporated into the objective function. This means that for similar problems with different constraints the algorithm, aside from minor changes of parameters, requires only rewriting of objective function calculation and associated code to become applicable.

The random elements in the genetic parts of the algorithm guarantee asymptotic convergence towards the global optimum, because the combination of stochastic remainder sampling and mutations ensures that in infinite number of iterations the algorithm visits all valid solutions (of which there is a finite number), including the optimal ones.

5. Algorithm Evaluation. This paper present the results of preliminary tests of the algorithm; the algorithm presented is still being developed, and we expect to test it on a greater variety of problems (e.g. different intermediate storage policies, larger problems)

	Search success rate [%]			Average computation time [%]		
Problem dimension	TS/GA	TS	GA	TS/GA	TS	GA
10/5	100	94	77	100	100	94
12/8	87	91	57	100	120	77
15/10	77	64	17	100	115	51

Table 5.1
Statistics of algorithm performance for various problem dimensions.

5.1. Test problems. As it was impossible to predict the exact nature of specific real problems should the algorithm be applied to solution of such, we used a method common in similar studies. The problems the program was tested on were sets of randomly (within defined parameters) generated input data matrices. The dimensions of solved problems varied, and the results presented in this paper are the ones for the problems with $n/m = \{10/5, 12/8, 15/10\}$. The range of processing times was 1-50; each stage contained 1 or 2 processing units.

5.2. Algorithm performance. The algorithm was tested against pure TS and pure GA that use the same principles as the algorithm we propose. The results indicate that performance of the combined algorithm is better than that of either pure TS or pure GA.

Low performance of the GA is, we believe, caused by fitness scaling function we use. We tried linear fitness scaling, and while the search success rate for 10/5 problems went up to 97%, the program occasionally failed because of problems caused by linear fitness scaling and associated code (one of the reasons for such errors were problems with floating point operations precision).

6. Conclusions. The algorithm we have proposed is able to optimize schedules for a hybrid flowshop, minimizing the makespan, and the performance of this algorithm is better than of both pure tabu search and pure genetic algorithm. The algorithm is, thanks to its composite nature, more adaptable to changes, and retains the advantages commonly associated with heuristics. Solution quality is higher than for tabu search, and the random elements in the genetic parts of the algorithm guarantee asymptotic convergence towards the global optimum.

REFERENCES

[1] S. A. Brah: *Complexity of the flow shop with multiple processors scheduling problem, and some dominance conditions*, in: Phua, K.H. et al. (Ed.), Optimization: Techniques and Applications, 1 (1992), World Scientific, Singapore, pp. 538-545

[2] ———: *Scheduling in a Flow Shop with Multiple Processors*, Dissertation Abstracts International 50 (1988), 1587B (University Microfilms No. 89-122667)

[3] H. G. Campbell, R. A. Dudek, and M. L. Smith: *An heuristic algorithm for the* n *job* m *machine sequencing problem*, Management Science, **16/B** (1970), 630–637.

[4] M. R. Garey and D. S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.

[5] F. Glover and M. Laguna: Tabu search, Kluwer Academic Publishers, 1997.

[6] J. N. D. Gupta: *Heuristic algorithms for multistage flowshop scheduling problem*, AIIE Transactions, **4** (1) (1972), 11–18.

[7] M. Nawaz, E. Enscore, and I. Ham: *A heuristic algorithm for the* m *machine,* n *job flow shop sequence problem*, OMEGA, **11** (1) (1983), 91–95.

[8] F. A. Ogbu and D. K. Smith: *The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem*, Computers & Operations Research, **17** (3) (1990), 243–253.

[9] D. S. Palmer: *Sequencing jobs through a multi-stage process in the minimum total time - A quick method of obtaining a near optimum*, Operations Research Quarterly, **16** (1) (1965), 101–107.

[10] M.-C. Portmann, A. Vignier, D. Dardilhac, and D. Dezalay: *Branch and bound crossed with GA to solve hybrid flowshops*, European Journal of Operational Research, **107** (2) (1998), 389–400.

[11] M. S. Salvador: *A solution to a special case of flow shop scheduling problems*, in: Elmaghraby, S.E. (Ed.), Symposium of the Theory of Scheduling and Applications, 1973, Springer, New York.

[12] D. L. Santos, J. L. Hunsucker, and D. E. Deal: *Global lower bounds for flow shops with multiple processors*, European Journal of Operational Research, **80** (1995), 112–120.

[13] ———: *Evaluation of sequencing heuristics in flow shops with multiple processors*, Computers & Industrial Engineering, **30** (4) (1996), 681–692.

[14] J. Sridhar and C. Rajendran: *Scheduling in a cellular manufacturing system: A simulated annealing approach*, International Journal of Production Research, **31** (12) (1993), 2927–2945.

[15] P. Stluka: *Využití prvků umělé inteligence při rozvrhování vsádkových výrob (Use of artificial intelligence for scheduling of batch operations)*, Ph.D. thesis at VŠCHT Praha (1998).

[16] E. Taillard: *Some efficient heuristic methods for the flow shop sequencing problem*, European Journal of Operational Research, **47** (1) (1990), 65–74.