SCHEDULING OF MULTIPLE PRODUCTS ON PARALLEL UNITS WITH TARDINESS PENALTIES USING SIMULATED ANNEALING

M. TANDON, P. T. CUMMINGS and M. D. LEVAN
Department of Chemical Engineering, University of Virginia, Charlottesville, VA 22903-2442, U.S.A.

(Received 16 March 1992; final revision received 3 November 1993; received for publication 31 August 1994)

Abstract—We present a solution methodology to obtain near-optimal solutions for the problem of scheduling multiple products on a network of single-stage, unrelated parallel units using simulated annealing (SA) with a performance criterion based on tardiness minimization. The approach incorporates sequence dependent clean-up times, varying processing rates for different products on different units, and constraints on feasible product-to-unit assignments. Near-optimal solutions are obtained for problems with 20–100 products and 3–11 units. A comparison is made between solutions obtained by SA, solutions obtained by the heuristic improvement method of Musier and Evans for similar computational effort, and solutions obtained by a list scheduling algorithm.

1. INTRODUCTION

Batch scheduling determines the order in which products are processed by the various units of a plant. Some chemical plants consist of numerous serial stages with parallel units at each stage, and it is often appropriate to optimize each stage individually.

Mathematically, the batch scheduling problem addressed in the paper is characterized by the following elements (Reklaitis, 1982): (i) a set of N products to be processed; (ii) a set of M single-stage processing units; (iii) the amount of each product to be manufactured; (iv) the processing rates of the various products on the available units; (v) a set of feasible product-to-unit assignment constraints; (vi) a matrix of sequence-dependent clean-up times $c_{ij}$; (vii) a performance criterion with respect to which the schedule is optimized; and (viii) a set of rules governing the production process. The problem is deterministic (all parameters are known in advance) and static (the production requirement remains unchanged for the duration of the sequencing), and has nonpreemptive operations (once a unit begins processing a particular product, it continues to do so until the processing requirement is completed). The restriction to nonpreemptive operation is appropriate for the chemical process industry, where parallel units will often be continuous production lines.

We classify networks into four categories based on the dependence of processing time as follows: (i) product independent unit independent (PIUI); (ii) product independent unit dependent (PIUD); (iii) product dependent unit independent (PDUI); and (iv) product dependent unit dependent (PDUD). For example PIUI means that the processing time is independent of both the product and the unit. This nomenclature differs from that usually used in the literature. What we call PIUI is usually referred to as identical units, PIUD as uniform units, and both PDUI and PDUD as unrelated units. It should be noted that problems that have unit independent processing networks can be defined by specifying only the processing time of each product. On the other hand, problems that have unit dependent processing networks are defined by specifying the processing time for each product in each unit. Unit independent networks usually consist of processing units of similar sizes while unit dependent networks consist of units of different sizes. Product-to-unit assignment constraints are incorporated due to the presence of processing units of varying specification and equipment sizes. Therefore, many products can only be produced on a subset of the parallel units, defined as feasible units.

Concerning the performance criterion, Musier and Evans (1989) concluded from their survey that the major focus of a process industry is to meet the due date requirements of the customer. Thus, the problem of scheduling multiple products with the objective of minimizing tardiness, defined as the difference between the completion times of late products and their prior due dates, is especially relevant.
In this paper, we solve the problem of scheduling multiple products on single-stage PDUI and PDUD parallel units networks using simulated annealing (SA) with a dimensionless objective function obtained by dividing the tardiness of each product by its processing time. We compare the solutions obtained by SA with solutions obtained by the heuristic improvement method (HIM) of Musier and Evans (1989) and a list scheduling algorithm.

2. SIMULATED ANNEALING

SA is an algorithm for single objective multi-variable optimization based on the Monte-Carlo method used in statistical mechanics (Kirkpatrick, 1983). As an approximate algorithm, SA provides near-optimal solutions in polynomial time (Aarts and van Laarhoven, 1985). The algorithm for SA consists of repeatedly making a move to change the system configuration and accepting the move based on a mathematically derived acceptance probability, while periodically reducing the annealing temperature. In chemical engineering, SA has been applied to the synthesis of heat exchanger networks, pressure relief header networks (Dolan et al., 1987, 1989) and batch scheduling (Das et al., 1990; Ku and Karimi, 1991a, b; Tandon et al., 1991).

In the present study, we have used the Metropolis move acceptance criterion (Metropolis et al., 1953) with the Aarts and van Laarhoven annealing schedule (Aarts and van Laarhoven, 1985). The initial and final temperatures are computed by the method of Aarts and van Laarhoven (1985). The reader is referred to Das et al. (1990) for details on computation of the various parameters required.

3. HEURISTIC IMPROVEMENT METHOD

HIM (Musier and Evans, 1989) is the best published heuristic for the problems considered in this paper. It is an evolutionary strategy consisting of two parts: (i) generation of an initial sequence using a heuristic; (ii) subsequent improvement of the initial schedule by generating new schedules until a termination criterion is satisfied. Musier and Evans have outlined the method for generating initial configurations; it relies on a heuristic which creates numerous partially random feasible sequences. New schedules are obtained using two procedures, IMPROVE-1 and IMPROVE-2. IMPROVE-1 moves single products by relocating them at any of the $O(N+M)$ distinct positions in the schedule. IMPROVE-2 swaps pairs of products. Each time a procedure is used it performs a complete pass through the schedule, considering all relocations of single products or all exchanges of pairs of products. The procedures are used alternatively until no further improvement is obtained. In implementation, each new configuration is evaluated, and, if it results in an equal or lower objective function value, then the most recent new configuration is retained for later exploration, after all perturbations involving the $N$ products have been investigated once, and the algorithm continues.

Musier and Evans assumed that the processing rate and, consequently, the processing time is the same for each type of product on all of the feasible units. Thus, they actually consider the PDUI problem rather than the more general PDUD case. However, HIM can easily be extended to solve the PDUD problem.

4. RESULTS

The solution to the problem of minimizing a dimensionless tardiness function for PDUI and PDUD parallel units network is completely specified by the sequence of products for each of the units. In our implementation of SA, two different moves, similar in spirit to the moves of IMPROVE-1 and IMPROVE-2 of HIM, are used to generate new system configurations. The first move picks a product at random and moves it to a random position in the same unit or in a different unit which can process the product. The second move picks two random products from the same unit or different units and exchanges them; if the randomly chosen products are from different units, a new configuration is generated only if both of the units are feasible units after the exchange of the two products. The total number of moves attempted at each temperature is $10N$.

The list scheduling algorithm considered in addition to SA and HIM is straightforward. It involves simply the construction of a priority list in which the products with the largest processing times are processed first, thereby attempting to minimize tardiness in the overall schedule.

We first consider problems with PDUI networks, similar to those considered by Musier and Evans (1989). The solution obtained by SA, HIM and list scheduling are compared by solving 20 randomly generated problems of varying sizes. The parameters which specify a single problem are (i) $N$ products having processing times randomly assigned from a uniform integer distribution between 1 and 100 time units with the assumption that the processing time for each product is the same on all of the units on which it can be processed; (ii) $M$ processing units with the feasible units for each type of product
randomly assigned with a probability of 0.5; (iii) sequence dependent clean-up times randomly assigned with a probability of 0.5 that a particular sequence of two products requires interim clean-up and the time randomly assigned in the range of 1–20 time units; and (iv) due dates assigned as the product completion times of a saturated schedule created by assigning each product in turn to the unit on which it could be completed the earliest. For each test problem, the due dates have been constructed so that there exists at least one optimal schedule with zero tardiness. Results are obtained by solving problems of varying sizes ranging from 20 to 100 products and 3 to 11 units. Industrial problems are often within these broad ranges, as are the problems considered by Musier and Evans.

We begin by comparing the quality of the solutions obtained by SA and HIM when similar computational effort is expended by the two algorithms. Musier and Evans (1989) have compared the best solution obtained by HIM for 50 trials of a problem with the list scheduling algorithm. In this paper, we compare solutions by computing both the overall average deviation from the optimum for a fixed number of trials and the average deviation from the optimum for the best solution for a fixed number of trials. We define the overall average deviation from the optimum by:

$$\mu = \frac{1}{PT} \sum_i^{P} \sum_t^{T} \frac{\alpha_{PT}}{\phi_t}.$$  \hspace{1cm} (1)

where $\alpha_{PT}$ is the tardiness for a particular trial of a particular problem, $\phi_t$ is the processing time required for the particular problem, $T$ is the number of trials and $P$ is the number of problems. We define the average deviation from the optimum for the best result for each problem by:

$$\mu_B = \frac{1}{P} \sum_{i=1}^{P} \frac{\gamma_p}{\phi_p},$$  \hspace{1cm} (2)

where $\gamma_p$ is the best result for $T$ values of $\alpha_{PT}$. HIM, being deterministic in nature, obtains a number of different final solutions by generating different initial configurations, thus linearly increasing the computational effort with an increasing number of attempts to solve the problem. The computational effort required by SA to solve a problem depends on the parameter $\delta$ and the number of times each problem is solved using different initial configurations. We report computational effort measured in CPU seconds on an IBM RS/6000 Model 320 engineering workstation. The results are obtained for 20 problems with 50 products and 7 units.

Figures 1 and 2 are plots of $\mu$ and $\mu_B$ vs the computational effort expended to solve the problem by HIM and SA at fixed values of $\delta$ and trials, respectively. For the case of HIM, each problem is solved 500 times and the values of $\mu$ and $\mu_B$ are plotted at intervals of 10 trials. For the case of SA with fixed values of $\delta$, shown in Fig. 1, each problem is solved 50 times with $\delta = 0.3$ and 0.1 and 16 times
with $\delta = 0.03$, the values of $\mu$ and $\mu_B$ being plotted after each trial. For the case of SA with a fixed number of trials, shown in Fig. 2, the value of $\delta$ is varied between 0.06 and 1.5. It is observed that lower values of $\mu$ and $\mu_B$ are obtained for SA compared to HIM. For SA, it can be concluded that for a fixed number of trials, the additional computational effort required by decreasing the value of $\delta$ improves both $\mu$ and $\mu_B$ significantly. As expected, for a fixed value of $\delta$, the additional computational effort resulting from increasing the number of trials does not give significantly better solution for $\mu$. However, the value of $\mu_B$ consistently improves with an increase in the number of trials at a fixed $\delta$ until an asymptotic limit is reached. It can also be concluded that for a fixed computational effort, fewer SA trials at a low value of $\delta$ give a lower value of $\mu$ and $\mu_B$ than a greater number of SA trials at a higher $\delta$. This is consistent with the idea that slow annealing (small $\delta$) should result in solutions closer to the global optimum, so that fewer runs at smaller $\delta$ should be more effective than more runs at higher $\delta$. For the case of HIM, the value of $\mu_B$ improves as the number of trials increases because the chances of a trial obtaining a good solution increases with an increasing number of solution attempts.

Figures 3 and 4 show various values of $\mu$ and $\mu_B$ for problems with a varying number of products and a varying number of units, respectively. The values of $\mu$ and $\mu_B$ for SA are averaged for 10 trials with a $\delta$ of 0.1. The solution for HIM is averaged for 100 and 200 trials of each test problem. The figure presents results using roughly comparable computation times. It is observed that in general, SA gives better solutions than HIM and that both $\mu$ and $\mu_B$ increase as the problem size increases for SA and HIM. However, the increase in the values of $\mu$ is small. As expected, the value of $\mu$ does not improve when the computational effort is doubled for HIM from 100 to 200 trials; the same is not true for $\mu_B$ as the chance of obtaining a good solution increases with the number of solution attempts. However, 10 trials of SA finds a better solution than 200 trials of HIM. The results obtained by the list scheduling technique are very poor compared to the results obtained by both SA and HIM, because list scheduling is a single pass heuristic relying on the optimum placement of a single product and overlooks the scheduling problem as a whole.

Next, we consider problems with a PDUD network. The solution obtained by SA and HIM are compared for problems having units with processing rates randomly chosen in the range of 96–104 kg (time unit)$^{-1}$ manufacturing products randomly chosen from the range of 1000–4000 kg in multiples of 200 kg. The clean-up requirement was chosen randomly to be in the range of 1–10 time units. Results are obtained by solving problems with the number
of products ranging from 20 to 100 and the number of units ranging from 3 to 9. It may be noted that we chose a wide range of amounts of products to be manufactured and a narrow range of processing rates. Of course, it is really these two factors in combination that set the required processing times and, hence, largely define the problems. Similar results could be expected for problems with a narrow range of amounts and a wide range of rates.

Table 1 shows the results for the solutions obtained by SA and HIM. It is observed that for problems of all sizes, the value of $\mu$ is lower for SA compared to HIM. The value of $\mu_B$ is the same with no deviation from the optimum for both SA and HIM for the case of the smallest problems with 20 products being scheduled on 3 or 5 units. This
indicates that both algorithms find at least one global optimum out of 10 and 200 trials of SA and HIM, respectively. For problems with 20 products and 7 or 9 units and with 40 or more products and 3 units, HIM performs better than SA. At this stage, the size of the problem is within reasonable limits for HIM to find near-optimal solutions, while for similar computational effort, the value of $\delta$ is not low enough for SA to converge on solutions of comparable quality. However, for all problems with 40 or more products and 5 or more units, the values of $\mu_B$ obtained by 10 trials of SA are better than those for 200 trials of HIM. In general, the values of $\sigma$ are lower for SA compared to HIM while the

![Graph](image)

**Fig. 4.** Comparison of $\mu$ and $\mu_B$ for SA, HIM and list scheduling for 60 products and various numbers of units. Results are evaluated for 10 trials of SA with $\delta = 0.1$ and 100 and 200 trials of HIM on 20 problems.
values of $\sigma_B$ are comparable for both of the solution techniques. A lower value of $\mu$ or $\sigma_B$ indicates that the final solutions lie in a small range and if the corresponding value of $\mu$ and $\mu_B$ is close to optimal, we can conclude that the solution repeatedly reaches a near-optimal value. It should also be noted that for problems with 40 products SA takes comparable time to HIM and for problems with 60 or more products SA takes less time than HIM.

5. CONCLUSIONS

The problem of scheduling multiple products on a network of single-stage parallel units incorporating product to unit assignment constraints and clean-up times is solved using SA as the optimization algorithm. The method is applicable to both PDUU and PDUD networks. For the case of PDUU networks, the algorithm developed was found to be superior to HIM and yielded better solutions for similar computation time expended by both algorithms. For the case of PDUD networks, HIM performs better for small problems while SA yields better results for large problems. It was also found that the quality of the solutions deteriorates with increasing complexity of the problems.

NOMENCLATURE

**Arabic symbols**

$M$ = Number of units
$N$ = Number of products
$P$ = Number of problems
$T$ = Number of trials

**Greek symbols**

$\alpha$ = Tardiness for a trial of a problem
$\gamma$ = Best tardiness of a problem
$\phi$ = Processing time for a problem
$\sigma$ = Standard deviation of $\sigma_{pT}$
$\sigma_B$ = Standard deviation of $\gamma_p$
$\mu$ = Average deviation from optimum
$\mu_B$ = Average best deviation from optimum

**Abbreviations**

SA = Simulated annealing
HIM = Heuristic improvement method
PDUD = Process dependent units dependent
PDUU = Process dependent units independent
PIUD = Process independent units dependent
PIUI = Process independent units independent

REFERENCES


