PRODUCTION PLANNING & CONTROL, 2002, VOL. 13, NO. 3, 307–316

Taylor & Francis
Taylor & Francis Group

The permutation flow shop problem with just in time production considerations

PATRICK R. MCMULLEN

Keywords Scheduling, heuristics, optimization, search

Abstract. This research presents a variation to the permutation flow shop problem where Just In Time (JIT) production requirements are taken into account. The model developed in this research employs dual objectives. In addition to the traditional objective of minimizing the production makespan, minimization of Miltenburg's material usage rate is also incorporated. In this model, multiple units of any product are permitted in the production sequence. However, the minimization of material usage rates attempts to prevent batch scheduling of products and allows unit flow of products as required in demand flow manufacturing. A solution method is proposed for determining an optimal production sequence via an efficient frontier approach and Simulated Annealing (SA). Test problems and specific performance criteria are used to assess the solutions generated by the proposed method. Experimental results presented in this paper show that the use of the efficient frontier and SA provide solutions that approach the optimal solution for the performance measures used in this research.

1. Introduction

Much effort has gone into finding production sequences minimizing makespan – the amount of time required to complete a production run. While such efforts are important, they have essentially ignored an additional schedule attribute associated with stability of materials usage rate. In the context here, stability of materials usage rate is the amount of product intermixing that occurs when there are multiple units of the same item needing to be scheduled. This material usage rate was first discussed by Monden (1998) and later formalized by Miltenburg (1989). Optimization of this material usage rate is quite important for Just in Time (JIT) production systems because this material usage rate is a measure of how well the unique items are intermixed in the production sequence. This intermixing of products is frequently referred to as mixed-model sequencing and is described by Costanza (1996) as 'a means of smoothing the flow and the work content between products.' Costanza also suggests that this mixed-model sequencing will reduce quality problems because fewer numbers of the same item will be produced at one time. In a JIT environment, responsible production sequencing is necessary, much in the same way that small batches and quick changeovers are important for successful implementation.

For this research, then, there are two objectives of interest: minimization of production makespan and optimization of product intermixing, which is achieved by minimization of Miltenburg's usage rate. To the author's best knowledge, there has been no published research that addresses simultaneous optimization of both makespan and stability of material usage rate. The material usage rate, as defined in this research, has a higher value when the products are scheduled in batches and decreases when unit product batch size is allowed.

Authors: P. R. McMullen, Auburn University, College of Business, Department of Management, Auburn University, Alabama 36849, USA, e-mail pmcmullen@business.aubum.edu

PATRICK MCMULLEN is an Associate Professor of Management at the Auburn University College of Business, specializing in Operations Management and Quantitative Analysis. Prior to an academic career, McMullen was an Industrial Engineer in the food and automotive industries. McMullen received his PhD in Operations Management from the University of Oregon in 1995. He has held teaching positions at the University of Oregon, the University of Maine and Harvard University. He has published articles in *International Journal of Production Research*, *Communications of the ACM*, *European Journal of Operational Research*, *IIE Transactions on Scheduling and Logistics*. His research interests currently lie in the areas of optimization and search heuristics.

Production Planning & Control ISSN 0953–7287 print/ISSN 1366–5871 online © 2002 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/09537280110086758

In dealing with such a problem, there are two complicating forces requiring attention: dual objective functions and combinatorial complexity. To address the dual objective functions, an efficient frontier is exploited to find production sequences having minimal usage rates for each possible makespan value. To address the combinatorial complexity of such problems, the popular search heuristic of Simulated Annealing (SA) is employed in an attempt to find production sequences having near optimal solutions without performing complete enumeration. In short, the SA approach is used to find production sequences providing values of makespan and usage rates that are near optimal with a reasonable amount of computational effort.

The following sections detail the two objectives of minimization of makespan and usage rate, the combinatorial complexity of such problems, and the SA heuristic. This collection of makespan and usage rate values obtained via SA (in the form of an efficient frontier) is compared against the efficient frontier obtained via complete enumeration of all sequences for several test problems. The performance of the SA approach is then assessed. General observations and conclusions are then offered.

2. Objective functions and combinatorial complexity

The makespan is the amount of time required to complete a production sequence. If there are a total of n jobs requiring processing through m machines, the following additional variables are defined so that minimization of makespan can be modelled:

C_{ij} = the latest completion time of job j on machine i.

t_{ij} = the processing time for job j on machine i.

s_{ijk} = the sequence dependent setup time for changing from jobs j to k on machine i.

D_{jk} = a binary variable representing unity if job j is scheduled before job k and zero otherwise.

M is a large positive value.

The objective function to minimize makespan is as follows:

$$\text{Minimize:} \quad C_{\max} \tag{1}$$

Subject to the following constraints:

$$C_{\max} \geq C_{mj} (j = 1, \ldots, n) \tag{2}$$

$$C_{1k} \geq t_{1k}, (k = 1, \ldots, n) \tag{3}$$

$$C_{ik} \geq C_{i-1,k} + t_{ik}, (i = 2, \ldots m; k = 1, \ldots, n) \tag{4}$$

$$C_{ij} - C_{jk} + MD_{jk} \geq s_{ijk} + t_{ij} \tag{5}$$

$$C_{jk} - C_{ij} + M(1 - D_{jk}) \geq s_{ijk} + t_{ik} \tag{6}$$

For equations (5) and (6), $i = 1, \ldots, m$; $j = 1, \ldots, n-1; k = 2, \ldots, n; k > j$.

Constraint (2) requires the objective function value to be no less than the completion time of any jobs on the last machine. Constraint (3) forces all jobs to be processed on machine 1, while constraint (4) forces all jobs to be produced on machines 2 through m, once processing on the prior machine has been completed. Constraints (5) and (6) are mutually exclusive constraints which govern precedence restrictions in the sequence. Stafford and Tseng (1990) offer this formulation as a correction of an error in the original Srikar and Ghosh formulation (1986). It is assumed here that the production run will commence after any required initial setups (to process the first job) are completed. It is appropriate to note that both the Safford and Tseng formulation and the Srikar and Ghosh formulation are setup-oriented extensions of the work by Manne (1960).

If there are a unique products and the demand for product h is represented by d_h, then the total demand for all products is as follows:

$$n = \sum_{h=1}^{a} d_h \tag{7}$$

It is appropriate to note here that the total number of products (n) is the same as the total number of jobs from the makespan minimization model above. Given the fact that there are problems here requiring sequencing with multiple units of each unique item, the variable x_{hg} is used to represent the number of units of item h placed in the sequence through the first g positions in the sequence. With this known, the usage rate objective is as follows:

$$\text{Minimize:} \sum_{g=1}^{n} \sum_{h=1}^{a} \left(x_{hg} - g \cdot \frac{d_h}{n} \right)^2 \tag{8}$$

It is desired that a production sequence provide minimal levels of both makespan and usage rate. The problem with this desire, however, is twofold. First, there is no known direct way of simultaneously minimizing both. Second, it can be demonstrated that these two objectives are essentially in opposition to one another – in other words, they are inversely correlated. To illustrate this second thought, consider the following example: suppose there are five units of item A demanded ($d_1 = 5$), three units each of items B and C ($d_2 = d_3 = 3$). One possible sequence would be AAAAABBBCCC. This sequence would result in a relatively low value of makespan, which is desirable, and high usage rate, which is undesir-

able. The makespan value would be low because there would only be three required changeovers for each machine. The usage rate would be high because there is no product intermixing. Another possible sequence would be ABACABACABC. Here the makespan would be high (undesirable) due to the 11 required changeovers for each of the m machines. The usage rate, however would be low (desirable) due to the large degree of product intermixing. This example should shed some light on the inverse relationship between makespan and usage rate.

One way to address the dual objectives as well as their inverse relationship is through the use of an efficient frontier. In the context of this research, the efficient frontier is the set of points defined by the minimum usage rate for each unique makespan value. To determine the optimal efficient frontier for a problem, all possible sequences are completely enumerated. For each sequence, the makespan and usage rate values are determined. If the usage rate for the current sequence is less than the minimum usage rate associated with that makespan value, then the usage rate becomes the minimum usage rate associated that makespan value. In other words, that particular sequence, and its associated makespan and usage rate values are on the efficient frontier. Otherwise, the sequence does not exist on the efficient frontier, and is considered inefficient. Use of an efficient frontier permits the decision-maker to find a sequence yielding a makespan value considered tolerable. The frontier then subsequently informs the decision-maker of the minimum usage sequence for the makespan value of interest. It should be noted that one could use some sort of composite objective function to address such problems, but that forces the decision-maker to determine weighting schemes for the components of the objective function, which complicates matters.

Figure 1 shows the efficient frontier for the example problem described above: $a = 3,\ d_1 = 5,\ d_2 = d_3 = 3$. For this example, there are two machines $(m = 2)$. Process times (t_{ij}) and setup times (s_{ijk}) are random numbers from the discrete-uniform distribution in the interval $[1, 99]$. These random numbers are also rounded to the nearest integer.

There are three interesting things to note for this example problem (and all similar problems in general). First, the efficient frontier is not strictly convex like efficient frontiers from other applications. Second, there is no guarantee that there exists a sequence at every value of makespan between the maximum and minimum values of makespan – the frontier is discontinuous. Third, the inverse relationship between makespan and usage is noticeable. The objective, again, is to find sequences having the minimum usage rate for each existing value of makespan.

The example problem above is a small one. Larger, more realistic problems require more enumeration effort. The number of unique sequences for problems of this type is as follows:

$$\text{Unique Sequences} = \left(\sum_{h=1}^{a} d_h\right)! \Big/ \left(\prod_{h=1}^{a} d_h!\right) \qquad (9)$$

For the example problem above, there would be 9240 unique sequences:

$$\frac{(5+3+3)!}{(5!)(3!)(3!)} = 9.240$$

If one additional unit were added to each item, there would be 210 210 unique sequences:

$$\frac{(6+4+4)!}{(6!)(4!)(4!)} = 210\,210$$

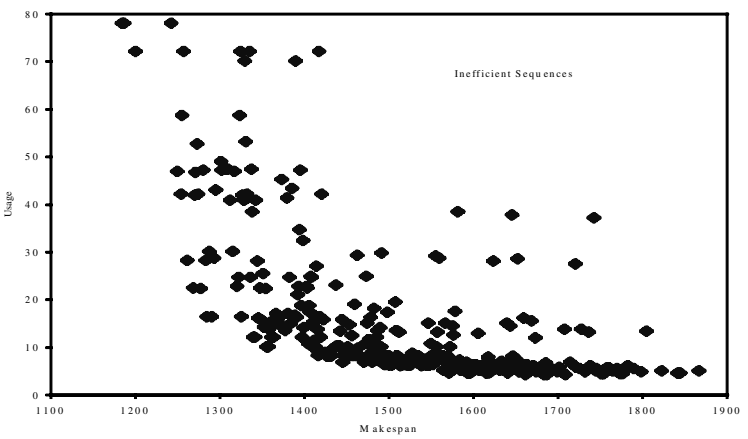

Figure 1. Optimal efficient frontier for example problem.

This illustrates that small increases in problem size result in large increases in computational effort required to find optimal solutions (in this case efficient frontiers). As a result, complete enumeration is not possible, and search heuristics must be employed, so that desirable, although not necessarily optimal, solutions can be found with a reasonable amount of computational effort.

3. Simulated annealing

Simulated Annealing (SA) is a search heuristic that can be used to address problems having the combinatorial complexity as exhibited by the problem of interest here. SA randomly selects feasible solutions, makes random and minor modifications to the existing solution. The objective function value of the modified solution is compared to the original solution. If the modified solution has a more desirable objective function value than the original solution, the modified solution replaces the current solution. Otherwise, the modified solution may still occasionally replace the current solution if a certain criterion is met. This occasional replacement of a current solution with a relatively inferior solution is done in an attempt to avoid being 'trapped' at local optima – a common characteristic of some modern search heuristics like Tabu Search (Glover 1990 and 1993) and Genetic Algorithms (Goldberg 1989). Although general details of the SA heuristic are not presented here, the interested reader is referred to some informative and readable articles on the subject (Kirkpatrick *et al*. 1983; Eglese, 1990).

2.1. *SA heuristic steps*

The following variables are defined as they relate to the SA heuristic:

T_I = initial temperature.

T_F = final temperature.

T = current temperature.

Iter = iterations for each value of T.

Counter = iteration counter.

CR = cooling rate.

ΔE = energy change

$P(A)$ = probability of acceptance.

K_b = Boltzman Constant

Rand = Uniformly distributed random number on the interval [0, 1].

Usage[*Makespan*] = Best usage rate found associated with the value of makespan.

TUsage[*Makespan*] = Usage rate of the newly found test solution associated with the value of makespan.

The first step of the heuristic is initialization, where values of T_1, T_F and CR are chosen. The current temperature, T, is set equal to T_1. Large usage rate values are assigned to the efficient frontier for each associated makespan value. An initial sequence of a product schedule is also chosen and it is defined as the current sequence. The makespan and usage rate associated with this sequence are assigned to the efficient frontier.

The current sequence undergoes a modification. The modification used here is a simple pairwise swap of unique sequence members. Here is an example:

AA<u>A</u>AABBB<u>C</u>CC (BeforeSwap)

The unique members of the sequence randomly chosen for swapping are underlined. After the swap, the resultant sequence is as follows:

AA<u>C</u>AABBB<u>A</u>CC (After Swap)

It should be noted that there are many different types of possible modifications – for example, three-way swaps. Prior research, however, has demonstrated that the type of swapping used has no effect on performance (McMullen and Frazier 2000). The makespan and usage rates are determined for the modified sequence (also referred to as the test solution). If *TUsage*[*Makespan*] < *Usage*[*Makespan*], then the value of *TUsage*[*Makespan*] becomes the value of *Usage*[*Makespan*], and the test solution essentially becomes the current solution. The usage rate of the newly accepted current solution is then compared with the usage rate of the 'best' solution for the associated makespan. If the usage rate of the 'new' current solution offers an improvement over the usage rate of the best solution, then the best solution is replaced with the current solution here as well.

If *TUsage*[*Makespan*] ≥ *Usage*[*Makespan*] the Metropolis criterion is examined for acceptance of an inferior solution. The change in 'energy' is first determined:

$$\Delta E = 100^{*}\left(\frac{TUsage[Makespan] - Usage[Makespan]}{Usage[Makespan]}\right) \tag{10}$$

Using the energy change, ΔE, the probability of accepting an inferior solution is then determined:

$$P(A) = \exp(-\Delta E/K_b T) \tag{11}$$

The value K_b is the Boltzman constant, which provides the decision-maker some control over the probability of acceptance. A random number, *Rand*, uniformly distributed on the interval [0, 1], is then generated. The inferior solution replaces the current solution if the following condition is true:

$$Rand < P(A) \tag{12}$$

Otherwise no replacement is made.

Regardless of whether or not replacement occurs, the value of the iteration counter, *Counter*, is incremented by one:

$$Counter = Counter + 1 \tag{13}$$

The above steps (with the exception of initialization) continue until the value of *Counter* has been reached. At this point, the temperature value is adjusted according to the relation:

$$T = T^{*}CR \tag{14}$$

When the value of T_F has been reached, the search process ends, and the 'best' frontier is reported.

Pseudocode is presented to simplify description of the above steps:

```
Initialize (current and best frontier, CR, T1, TF, T, Kb)
While T > TF
  For Counter = 1 to Iter
    Modify current solution
      If TUsage[Makespan] < Usage[Makespan]
        Replacement
      Else
        If Rand < P(A) Replacement
      Endif
  Next Counter
  T = T*CR
End While
Report best frontier
End
```

2.2. *Example problem*

The example problem from above is used again to illustrate the SA approach. Parameter values used are as follows: $T_1 = 25$, $T_F = 1$, $CR = 98\%$, $Iter = 10$. The Boltzman constant, K_b, was established such that an inferior solution with a resultant energy level of 10% would have a 25% probability of acceptance at the value of T_1. Figure 2 shows the efficient frontier obtained via SA compared with the optimal efficient obtained via complete enumeration.

It may be noticed from figure 2 that the SA efficient frontier is generally 'northeast' of the optimal frontier, which suggests the relative inferiority of SA as compared to the optimal solution. The SA frontier can only match the optimal frontier, but never surpass it.

4. Experimentation

4.1. *Performance measures*

For this research, there are three performance measures of interest with respect the SA search heuristic: the number of 'voids' in the efficient frontier, the average inferiority of the efficient frontier and the percentile of the efficient frontier.

The above problem has makespan range of 366. The makespan range here is the difference between the minimum and maximum makespan values of the optimal sol-

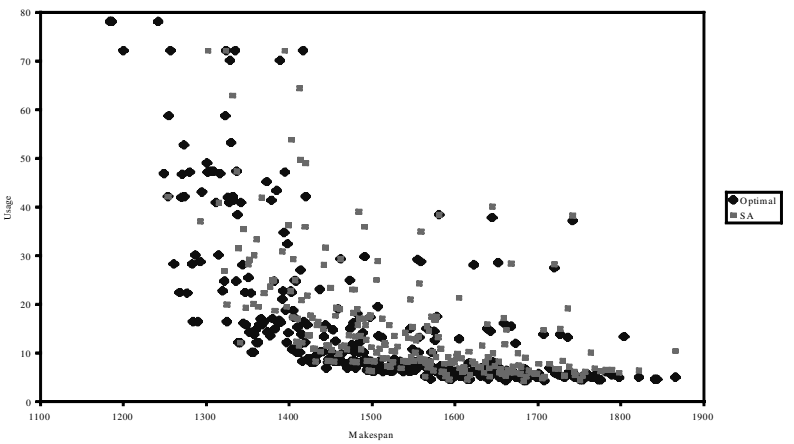

Figure 2. Efficient frontier comparison of SA (●) and optimal (◆).

ution. Also, there were 12 makespan values in this range of 366 where no solution existed. The value of 366 is therefore adjusted to 354 (366–12) to account for the absence of makespan values within this range. There were (40) occasions where the SA heuristic was unable to find sequences having makespan values that were found by the optimal solutions. These are referred to as 'voids,' and is essentially a measure of the SA heuristic's ability to find complete (but not necessarily optimal) frontiers – minimal voids are desired. Figure 3 represents a hypothetical SA-obtained solution with (2) voids.

From figure 3, the two voids are associated with makespans of equations (4) and (7) – here, the SA approach does not provide sequences when the optimal solution does. The way voids are presented here is by showing the ratio of SA voids to adjusted optimal makespan range. This way, the number of voids is placed in perspective relative to problem size. The void ratio for the above example is then $40/354 = 0.1130$.

The second performance measure presented is the average level of the SA solution's relative inferiority to the optimal solution in terms of usage for each possible makespan value. That is, the extent (percentage) to which the SA frontier is 'above' the optimal frontier. SA voids are not a part of this calculation – their inclusion would confound interpretation. For the example frontiers shown in figure 3, the average inferiority is as follows:

$$\left(\frac{10-8}{8}+\frac{7.5-6}{6}+\frac{5.5-4.5}{4.5}+\frac{3.9-2.9}{2.9}\right.$$
$$\left.+\frac{3.5-2.5}{2.5}+\frac{3-2}{2}+\frac{2.9-1.9}{1.9}+\frac{2.3-1.8}{1.78}\right)\Big/8 = 33.98\%$$

It is, of course, desired to minimize this relative inferiority measure.

The third performance measure is percentile performance. This is the percentage of all solutions at the specific makespan level to which the SA solution is superior (in terms of usage rate). When the SA solution frontier is compared to the optimal solution frontier, a count is kept of the number of times the optimal solution usage rate is superior to the usage rate of the frontier obtained via SA. Table 1 shows an example of this.

Table 1 shows that there were a total of 49 solutions found via enumeration which were superior to the SA frontier in terms of usage rate for each makespan value. Assuming that this example problem has a total of 4500 feasible solutions, the percentile performance of the SA approach would be as follows:

$$\text{Percentile Performance} = 100 - 100*(49/4500)$$
$$= 98.9111$$

The above calculation suggests that the SA approach is superior to 98.9111% of all feasible solutions. It should be noted that SA voids do not interfere with this calculation. For makespan values of four and seven, SA did not find solutions, but enumeration did find two solutions with makespans of four, and three solutions with makespans of seven. It is desired to maximize this percentile performance measure.

Table 1. Example of percentile performance measure.

Makespan	Optimal	SA	Solutions found superior to SA frontier value
1	8	10	5
2	6	7.5	6
3	4.5	5.5	3
4	3.5	Void	2
5	2.9	3.9	3
6	2.5	3.5	5
7	2.2	Void	3
8	2	3	9
9	1.9	2.8	6
10	1.8	2.3	7

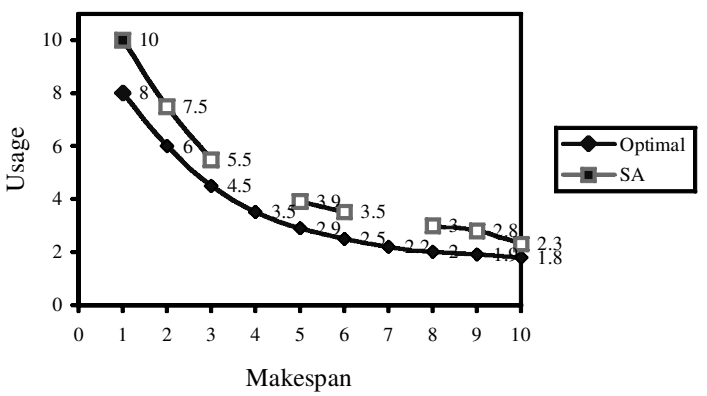

Figure 3. Efficient frontiers with SA yielding (2) voids. (◆) Optimal; (■) SA.

4.2. *Experimental problems*

Table 2 shows the problems used to evaluate the SA approach for comparison to optimal. Problem Set 1 has a total demand of ten units, Problem Set 2 has a total demand of 20 units, Problem Set 3 has a total demand of 12 units, and Problem Set 4 has a total demand of 15 units. For each successive problem within each set, the product mix becomes more diverse, which subsequently results in more permutations (total solutions), thereby complicating attainment of optimal solutions.

For Problems Sets 1 and 2, two machines are used ($m = 2$), and for Problem Sets 3 and 4, three machines are used ($m = 3$). For all problem sets, all setup times (s_{ijk}) and process times (t_{ij}) are random numbers from the discrete-uniform distribution in the interval [1, 99]. For each problem solved, several cooling rates are examined so that sensitivity to search space increases can be studied (the smallest problems were solved four times, while the larger problem solved six times). The first cooling rate used for each problem (CR_1) is proportional to the number of feasible solutions. Therefore, problems with more total solutions have higher cooling rates. For all problems, CR_1 was set to the following value:

$$CR_1 = \exp(\ln(1/T_1)/200*\ln(\text{No. of feasible solutions})). \tag{15}$$

Subsequent cooling rates are interpolated midpoints between the prior cooling rate and unity. Mathematically, this is as follows:

$$CR_i = CR_{i-1} + (1 - CR_{i-1})/2 \qquad i = 2, \ldots, 6 \text{ (or } i = 1, \ldots, 4) \tag{16}$$

As the cooling rate increases, more solutions are processed. The smaller problems use only four cooling rates due to the limited search space relative to the larger problems. Values of CR_1 are shown in table 2.

The research questions of interest are as follows:

- Is the void ratio offered by the SA approach statistically negligible?

Table 2. Problems used for experimentation.

Prob. set	Prob.	Item A	Item B	Item C	Item D	Item E	Total solutions	CR_1
1	1	6	1	1	1	1	5040	95.7462%
1	2	5	2	1	1	1	15 120	96.2225%
1	3	4	2	2	1	1	37 800	96.5451%
1	4	4	3	1	1	1	25 200	96.4094%
1	5	3	3	2	1	1	50 400	96.6353%
1	6	3	2	2	2	1	75 600	96.7548%
1	7	2	2	2	2	2	113 400	96.8661%
2	1	16	1	1	1	1	116 280	96.8727%
2	2	15	2	1	1	1	930 240	97.3395%
2	3	14	2	2	1	1	6 976 800	97.6757%
2	4	13	3	2	1	1	32 558 400	97.8805%
2	5	12	3	3	1	1	141 086 400	98.0445%
2	6	10	5	2	2	1	1 396 755 360	98.2555%
3	1	8	1	1	1	1	11 880	96.1273%
3	2	7	2	1	1	1	47 520	96.6173%
3	3	6	3	1	1	1	110 880	96.8601%
3	4	6	2	2	1	1	166 320	96.9644%
3	5	5	3	2	1	1	332 640	97.1275%
3	6	5	2	2	2	1	498 960	97.2150%
3	7	4	3	2	2	1	831 600	97.3179%
3	8	4	4	2	1	1	415 800	97.1763%
3	9	3	3	2	2	2	1 663 200	97.4461%
4	1	11	1	1	1	1	32 760	96.4984%
4	2	10	2	1	1	1	180 180	96.9841%
4	3	9	3	1	1	1	600 600	97.2533%
4	4	7	5	1	1	1	2 162 160	97.4914%
4	5	7	3	2	2	1	10 810 800	97.7378%
4	6	6	3	3	2	1	25 225 200	97.8491%
4	7	5	3	3	3	1	50 450 400	97.9323%
4	8	4	3	3	3	2	126 126 000	98.0328%
4	9	3	3	3	3	3	168 168 000	98.0624%

- Is the efficient frontier provided by the SA approach statistically inferior to the efficient frontier as obtained via the optimal solution?
- Is the percentile performance of the efficient frontier provided by the SA approach statistically inferior to 100%?
- Do cooling rates closer to unity provide SA solutions that approach an optimal condition in terms of the void ratio, average inferiority and percentile performance?
- Do the number of feasible solutions for each have an effect on the three performance measures?

4.3. *Computational experience*

Both the SA heuristic and complete enumeration approaches were coded using C/C++, and run on a Windows platform with Dual Pentium III processors at 500 MHz. For all cooling rates, the maximum CPU time for the SA heuristic was 4 seconds. The maximum CPU time for complete enumeration was 16.8 hours. This essentially means that the ratio of SA CPU time to complete enumeration CPU time is negligible.

It is also worth mentioning that the integer programming formulation presented earlier to find sequences with minimal makespan *only*, was used successfully for a two-machine, four job problem. Larger problems (in terms of either n and/or m) quickly become unmanageable in terms of both constraint formulation and solution effort – even larger to a small degree (LINDO was used here on the same computing platform as described above).

5. Experimental results

Table 3 presents the means of the performance measures organized by problem. The first research question is addressed via the following hypotheses:

$\mathbf{H}_0$: Mean Void Ratio $= 0$

$\mathbf{H}_A$: Mean Void Ratio > 0

Table 3. Means (and standard deviations) of performance measures by problem.

Problem set	Problem	Void to makespan range ratio	Average inferiority (%)	Percentile performance (%)
1	1	0.0933 (0.0925)	4.39 (5.49)	99.127 (0.884)
1	2	0.1200 (0.1109)	11.94 (13.09)	99.008 (0.655)
1	3	0.1441 (0.1333)	18.61 (19.11)	99.499 (0.280)
1	4	0.1496 (0.1092)	18.09 (14.38)	99.237 (0.402)
1	5	0.1631 (0.1387)	20.41 (20.00)	99.577 (0.223)
1	6	0.1570 (0.1169)	20.53 (20.99)	99.706 (0.1381)
1	7	0.1943 (0.1114)	15.68 (13.30)	99.895 (0.052)
2	1	0.1656 (0.1293)	6.47 (7.40)	99.914 (0.034)
2	2	0.1835 (0.1395)	19.2 (14.7)	99.971 (0.003)
2	3	0.1661 (0.1459)	23.0 (26.3)	99.996 (0.001)
2	4	0.2374 (0.1557)	35.4 (24.6)	>99.999 (<0.001)
2	5	0.2477 (0.1258)	45.4 (28.9)	>99.999 (<0.001)
2	6	0.2912 (0.1431)	76.10 (48.30)	>99.999 (<0.001)
3	1	0.1600 (0.0729)	0.00 (0.00)	99.137 (0.060)
3	2	0.2016 (0.0941)	13.12 (11.97)	99.547 (0.043)
3	3	0.2076 (0.1234)	22.03 (14.21)	99.701 (0.027)
3	4	0.2892 (0.1084)	26.74 (15.37)	99.809 (0.019)
3	5	0.2971 (0.1110)	42.25 (21.85)	99.880 (0.016)
3	6	0.3478 (0.1093)	40.22 (18.03)	99.923 (0.010)
3	7	0.3307 (0.1084)	55.98 (23.87)	99.949 (0.008)
3	8	0.2972 (0.0989)	46.43 (22.79)	99.901 (0.011)
3	9	0.3457 (0.0988)	59.89 (20.94)	99.973 (0.004)
4	1	0.2816 (0.0780)	4.05 (3.41)	99.703 (0.182)
4	2	0.3708 (0.0628)	14.76 (7.14)	99.896 (0.007)
4	3	0.3822 (0.055)	12.30 (13.94)	99.966 (0.007)
4	4	0.4197 (0.0675)	9.01 (11.42)	99.990 (0.003)
4	5	0.4438 (0.0801)	47.85 (21.60)	99.996 (<0.001)
4	6	0.4370 (0.0839)	69.05 (29.46)	99.998 (<0.001)
4	7	0.4391 (0.0949)	60.88 (23.25)	99.999 (<0.001)
4	8	0.3511 (0.1117)	62.34 (40.57)	>99.999 (<0.001)
4	9	0.3253 (0.1146)	50.09 (19.88)	>99.999 (<0.001)

Testing of these hypotheses is done via a one-tailed t-test. The mean void ratio of the SA approach sequences is 0.2494, with a standard deviation of 0.1434, which is statistically greater than zero ($t = 21.58, p < 0.001$).

The second research question is addressed via the following hypotheses:

$\mathbf{H}_0$: Mean Inferiority = 0

$\mathbf{H}_A$: Mean Inferiority > 0

A one-tailed *t*-test is also used for these hypotheses. The mean inferiority is 30.29%, with a standard deviation of 28.79%. This mean inferiority is also statistically greater than zero ($t = 13.06, p < 0.001$).

The third research question is addressed via the following hypotheses:

$\mathbf{H}_0$: Mean Percentile Performance = 100%

$\mathbf{H}_A$: Mean Percentile Performance < 100%

A one-tailed *t*-test is also used for these hypotheses. The mean percentile performance is 99.773%, with a standard deviation of 0.378%. This percentile performance is statistically less than 100% ($t = -7.46, p < 0.001$).

The fourth research question is first addressed via a multivariate analysis of variance – the overall multivariate effect of the cooling rate on the three performance measures is explored. MANOVA shows that these performance measures are sensitive to the cooling rate – Wilks' Lambda = 0.2283, which is statistically significant ($F = 19.031, p < 0.001$). Table 4 shows results of the three performance measures organized by the six cooling rates.

Table 4 makes it clear that cooling rates approaching unity result in solutions with lower void ratios, less average inferiority and higher percentile performance. Univariate ANOVA generally gives further support to the above claim with relevant statistics showing that the void ratio is sensitive to cooling rate ($F = 42.23, p < 0.001$). ANOVA also shows that the cooling rate also has a significant effect on the relative inferiority of the SA approach ($F = 15.20, p < 0.001$). ANOVA, however, shows that the cooling rate does not have a significant effect on percentile performance at the $\alpha = 0.05$ level of significance ($F = 1.51, p = 0.19$), although table 4 above does strongly suggest that cooling rates closer to unity do in fact provide percentile performance closer to 100%.

To address the fifth and final research question, it is important to consider a few things. While larger search spaces do provide solutions closer to optimal, increases in the size of the problem do make it more difficult to obtain near-optimal solutions. Table 3 shows performance measures organized by problem.

The means of the three performance measures are generally sensitive to the size of the problem. Simple linear regression shows that average inferiority increases (worsens) as the number of feasible solutions increase ($F = 25.51, p < 0.001$), but conversely, the percentile performance actually improves as the number of feasible solutions increase for each problem – this was also determined via simple regression ($F = 4.42, p = 0.037$). This is an interesting finding – as the problem gets larger in terms of feasible solutions, the efficient frontier obtained via SA 'moves further away' from the optimal frontier, but the percentile performance simultaneously improves. This suggests that the very few solutions not found via the SA search are important ones in terms of minimizing the relative inferiority of the SA efficient frontier. It is also important to note that linear regression shows that the void ratio is not sensitive to the number of the problem's feasible solutions ($F = 1.31, p = 0.255$).

6. Discussion and concluding remarks

A technique has been presented to obtain production sequences which provide tolerable results in terms of makespan and stability of material usage. The value of this technique is perhaps most beneficial when JIT considerations must be made – optimize product intermixing while simultaneously minimizing the length of the production run. Obtaining production sequences which simultaneously consider both minimal makespans and desirable levels of flexibility would definitely be important to many practitioners. It should be emphasized here that an important managerial implication is that to use this frontier approach properly, decision-makers must inspect the frontier for a sequence with a tolerable

Table 4. Means (and standard deviations) of performance measures by cooling rate.

Cooling rate	Void ratio	Average inferiority (%)	Percentile performance (%)
1	0.4130 (0.08)	57.44 (34.45)	99.70 (0.53)
2	0.3150 (0.10)	39.50 (27.31)	99.71 (0.45)
3	0.2361 (0.10)	26.20 (21.86)	99.74 (0.36)
4	0.1905 (0.11)	16.06 (14.29)	99.80 (0.26)
5	0.0999 (0.07)	14.85 (15.18)	99.89 (0.15)
6	0.0749 (0.07)	8.43 (12.24)	99.96 (0.04)

makespan, which subsequently results in the sequence's minimal makespan (or at least the one found via the SA heuristic).

Experimental results generally show desirable solutions to the problem sets used here. Whichever way the data is examined, percentile performance is always in excess of 99% – a strong argument for near-optimality. The void ratio and average inferiority only become desirable when the cooling rate is closest to unity (CR_4 or CR_6). On the other hand, experimentation also shows that as the problem size becomes larger, the level of average inferiority starts to deteriorate. This is reason for concern given that the experimental problems used here are somewhat small and that more 'real-world' problems might be larger (it is desired to address larger, more realistic problems here, but determination of optimal efficient frontiers becomes CPU prohibitive). Improvement of this situation is an opportunity for future research. It should be noted that some people could consider the average inferiority and void ratio to be more important performance measures than the percentile performance. The reason for this is that average inferiority and void ratio are more direct measures than percentile performance – percentile performance could be considered 'at the mercy' of the problem size. Nevertheless, all of these three performance measures are taken seriously here.

Simulated Annealing is one of many possible search heuristics. Others which have been demonstrated to provide similar promise are: Tabu Search, Genetic Algorithms and Beam Search. Exploring the use of goal programming could also be considered to find sequences resulting is desirable level of these two objectives. For this particular research effort, the author is not necessarily interested in finding the best heuristic for the specific application, but only in showing how near-optimal solutions can be obtained using the efficient frontier and SA heuristic approach. Nonetheless, use of these other types of heuristics does provide additional research opportunities.

References

Costanza, J. R., 1996, *The Quantum Leap…In Speed to Market* (Englewood, Colorado, USA: John Costanza Institute of technology).

Eglese, R. W., 1990, Simulated annealing: a tool for operational research. *European Journal of Operational Research*, **46**, 271–281.

Glover, F., 1990, Tabu search: a tutorial. *Interfaces*, **20**, 74–94.

Glover, F., 1993, A user's guide to tabu search. *Annals of Operations Research*, **41**, 3–28.

Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimizaiton & Machine Learning* (Reading, MA, USA: Addison-Wesley).

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983, Optimization by simulated annealing. *Science*, **220**, 671–679.

Manne, A. S., 1960, On the job-shop scheduling problem. *Operations Research*, **8**, 219–223.

McMullen, P. R., and Frazier G. V., 2000, A simulated annealing approach to mixed-model sequencing with multiple objectives on a JIT line. *IIE Transactions*, **32**, 679–686.

Monden, Y., 1998, *The Toyota Production System: An Integrated Approach to Just-In-Time* (Norcross, GA: Engineering and Engineering Management Press).

Miltenburg, J., 1989, Level schedules for mixed-model assembly lines in just-in-time production systems. *Management Science*, **35**, 192–207.

Srikar, B. N., and Ghosh, S., 1986, A MILP model for the n-job, m-stage flowshop with sequence dependent setup times. *International Journal of Production Research*, **24**, 1459–1474.

Stafford, E. F.,, and Tseng, F. T., 1990, On the Srikar-Ghosh MILP model for the $n \times m$ flowshop problem. *International Journal of Production Research*, **28**, 1817–1830.