

Workforce-Constrained Preventive Maintenance Scheduling using Evolution Strategies

ABSTRACT

Heavy equipment overhaul facilities such as aircraft service centers and rail-yards face the challenge of minimizing the makespan for a set of preventive maintenance (PM) tasks, requiring single- or multiple-skills, within workforce availability constraints. In this paper, we examine the utility of evolution strategies (ES) to this problem. Comparison of the computational efforts of ES with exhaustive enumeration (EE) to reach optimal solutions for sixty small problems illustrated the ability of ES to yield optimal solutions increasingly efficiently with increasing problem size. A set of 852 large-scale problems was solved using ES to examine the effects of task-related problem characteristics (number of tasks N , skills required per task SPT , and workers required per task WPT), workforce-related variables (number of workers W , skills per worker SPW), and ES population size (μ) on CPU time. Due to unavailability of optimal solution for comparison, each experiment was stopped at a quasi-optimal solution using a convergence criterion. In the estimated multiplicative power expression, the CPU time requirements using ES were proportional to $N^{1.34}$ (versus e^N for the EE approach). The maximum CPU time was 55 minutes for a 1000-task problem. The results empirically supported practical utility of ES to solve large-scale, complex PM problems involving single and multiple-skilled workforce. Finally, comparison of convergence speeds of ES and simulated annealing (SA) for the 852 experiments indicated that, on the average, SA needed to search about 12 times the number of solutions searched by ES to converge to the quasi-optimal solution, demonstrating the superiority of the ES approach to the SA approach for this class of PM problems.

Subject Area: Preventive Maintenance, Evolution Strategies, Workforce Scheduling

1. INTRODUCTION

In today's competitive environment, the need for effective maintenance of operational equipment can hardly be ignored. In manufacturing sector, downtime of machines has been identified as one of the major causes of reduced productivity (Spencer and Guide, 1995; Vuppalapati, Ahire, and Gupta, 1995). Improved maintenance of manufacturing equipment has been recognized as a critical element of overall quality improvement efforts (Flynn, Sakakibara, and Schroeder, 1994; Saraph, Benson, and Schroeder, 1989). In the service sector also, the primary equipment used to provide service must be kept in good condition. This can not only avoid major disasters, but also defer expensive capital investments. An excellent case in point is the maintenance of aircraft in the airline

industry. It is estimated that the maintenance function of various airlines accounts for about 10 percent of their operating costs, about the same as the fuel costs (Lam, 1995). In June 2000, the U.S. Federal Aviation Administration threatened to shut down major aircraft overhaul maintenance at Alaska Airlines in the wake of several jet crashes linked to inept maintenance execution and reporting practices (Wald, 2000). In summary, competent and effective maintenance has a direct impact on the profitability and reputation of an airline, and generally, of any organization. Research into better methods of scheduling preventive maintenance (PM) certainly represents a worthwhile contribution to enhancing operations management effectiveness.

In this paper, we study the PM scheduling problem faced by all major overhaul maintenance facilities where aircraft, ships, locomotives, or other heavy equipment needs to be turned around as early as possible. However, available time window is not the explicit restricting factor in scheduling, and the servicing of the unit must be completed in its entirety before the unit is returned to field usage. Thus, we consider the problem of assigning single- or multiple-skilled workers to a set of PM tasks requiring single- or multiple-skills. The objective is to compute the assignment, which will lead to a minimum makespan (time to complete all tasks). We examine the applicability of an evolution strategy (ES) algorithm to this NP-hard problem. The efficiency of the algorithm vis-à-vis exhaustive enumeration is illustrated for small problems. The computational efficiency of ES vis-à-vis simulated annealing (SA) is further confirmed through a large-scale experimental design covering 852 experiments.

The paper is organized as follows. Section 2 surveys the past PM research. Section 3 formally describes our PM problem. Evolution strategies (ES) are introduced and conceptually compared with genetic algorithms and simulated annealing (SA) in Section 4. Section 5 presents the mechanics of ES application to our specific PM problem. Computational efficacy of the ES approach is compared to exhaustive enumeration (EE) in Section 6. Computational performance of the ES approach for a systematic design of 852 large-scale PM problems is analyzed in Section 7. The results from ES and SA approach are compared for the 852 problems in Section 8. Finally, conclusions and recommendations for future research are offered in Section 9.

2. PM RESEARCH REVIEW

From a managerial perspective, different scheduling methodologies suit different contexts including system characteristics (continuous versus intermittent operations), scope of the problem (only maintenance schedules versus production and maintenance schedules), and goals (minimizing total expected maintenance costs versus maximizing total effectiveness in terms of breakdown avoidance). Over the last two decades, researchers have focused on various aspects of the PM function in different contexts (Table 1). PM research has been well documented in various seminal surveys (McCall, 1965; Pierskalla and Voelker, 1975; Sherif and Smith, 1981; Valdez-Flores and Feldman, 1989). Some classics have also pioneered mathematical treatment of the subject (Barlow and Hunter, 1960). Researchers have adopted various approaches to model PM scheduling problems. For example, Cornell, Lee, and Tagaras (1987) use the markov chain approach to modeling deterioration of equipment and analyze the long-term performance of a combination of scheduled and on-demand maintenance policies. Ram and Olumolade (1987) fit a Weibull distribution to time between failures on past equipment data, and use it to optimize the total expected maintenance (scheduled and breakdown) cost in presence of a production plan.

Researchers have studied a wide range of PM scheduling contexts. Golabi, Kulkarni, and Way (1982) discuss scheduling of pavement maintenance using Markov approach. Recently, Golabi and Shepard (1997) present a combination of Markovian state prediction and dynamic cost minimization approach for scheduling maintenance/overhaul of road bridge networks. Various highway associations in the U.S. have adopted both of these approaches.

A few recent research papers have developed models to maximize some measure of PM productivity or effectiveness in a limited maintenance resource context. For example, Dijkstra et al. (1994) describe a decision support system (DSS) for scheduling aircraft maintenance for KLM Royal Dutch Airlines within a finite time-window. The DSS is based on two integer programming models. The first model uses workload estimates, shift times, the number of teams per shift, and desired service level as inputs to

calculate the size and composition of the teams so that the total number of engineers is minimized, the service level constraint is satisfied, and jobs are matched with engineer skills. The second model assigns a given workforce (number of engineers per team and their skill combinations, shift times, and workload estimates) to workload in an optimal manner so as to maximize the service level. They limit the scope of the problem to single-skilled engineers within the specific organizational context.

Ulusoy, Or, and Soydan (1992) present an actual system of maintenance planning and control in a foundry setting. The heart of the system is a maintenance task priority function based on three factors: recommended task frequency (F), task criticality index (C) based on a composite of various machine-specific and production-relevance criteria, and actual maintenance delay beyond the recommended frequency (D). The authors use the conventional utility function approach to develop the function for various groups of PM tasks of different frequencies incorporating these three factors. Finally, they present a ratio-analysis based set of performance measures to evaluate the PM effectiveness in terms of the ratios between breakdown maintenance and PM, among other indices.

Dekker and Smeitink (1994) analyze the problem of scheduling preventive replacement tasks at randomly occurring opportunities of restricted duration. They present an approach to determining the priorities of various replacement packages (or sets of replacement tasks) based upon the cost of delaying the package beyond their optimal control limit for time since last execution. The scheduling criterion is based upon an opportunistic block replacement model as an underlying long-term cost optimization model.

Finally, Gopalakrishnan, Ahire, and Miller (1997) model PM task priorities using logistic regression using past machine usage and breakdown. They feed these priorities to a task scheduling (binary integer programming) model which incorporates workforce man-hours and skills availability constraints. Deterministic heuristics are developed to provide good solutions to the task-scheduling problem. Note that, with the exception of Gopalakrishnan et al. (1997), most other models do not explicitly address scheduling of PM activities using multiple-skilled workforce. Dijkstra et al. (1994) further specify that work-teams of workers with skills required for particular PM tasks constitute a unit of

workforce. All of these recent studies assume restricted time availability for conducting PM.

Insert Table 1 About Here

3. THE WORKFORCE-CONSTRAINED PM PROBLEM

We consider the problem of scheduling a set of PM tasks with a given available workforce (Table 1). For each task, the skills required and number of persons with these skills needed to work on the task is known. The tasks are to be performed by the available workforce, defined by number of persons with each skill or multiple skills. All of the skills required for the set of tasks to be performed must be available in the given workforce (i.e., there must be at least one person with each of the required skills). The objective is to determine the schedule (sequence in which the PM tasks should be executed) with the minimum makespan (the time of completion of the final task). This will be referred to as the *workforce-constrained preventive maintenance problem*. Note that the optimal solution to this problem will yield a PM schedule in which: (a) all PM tasks in the set will be performed, (b) the available workforce characteristics (number of persons, and single- versus multiple-skill availability) will be utilized in an efficient manner, and (c) minimum possible maintenance window (time to perform PM between production or equipment usage) will be determined. Hence, solution to this problem is critical to reducing maintenance downtimes in manufacturing/service production systems.

This problem differs from that addressed by Ulusoy et al. (1992), Dekker and Smeitink (1994), Dijkstra et al. (1994), and Gopalakrishnan et al. (1997) in terms of its focus and objective. Instead of executing those tasks that will maximize the total net savings in a specific time-bucket subject to workforce availability constraints (skills, persons, and man-hours), our goal is to minimize the makespan of a set of PM tasks within the workforce availability constraints (skills and persons). Thus, we schedule all tasks. Since all tasks will be scheduled, the logic of assigning workers to tasks is not driven by previously determined priorities. Instead, our approach consists of matching

the workers of required skills to tasks in an efficient manner, so as to complete the tasks in the shortest possible makespan.

In practice, this problem occurs routinely in large-scale, complex maintenance environments, such as aircraft heavy maintenance facilities, ship maintenance yards, and railroad yards. For example, while Dijkstra et al. (1994) model the time-constrained scheduling of inspection and PM activities for aircraft on ground between consecutive flights, aircraft heavy maintenance facilities receive aircraft for major overhauls at various phases in their life. Aircraft type and design dictates the intervals (in terms of mileage and/or age) for these overhauls. Although there is a realistic expectation about the maximum makespan within which the aircraft must be turned around to resume its flights, maintenance facilities always seek to minimize the turnaround time within their available workforce. All of the overhaul tasks must be completed before the aircraft can be returned to the field. Thus, the problem that we consider is not that of prioritizing tasks or maximizing service level (the number of tasks completed within the finite frame as a percentage of the total number of candidate tasks) within a finite time frame. Rather, the objective is to minimize the turnaround time or makespan.

Structurally, our PM problem differs from the earlier works as follows. The goal of the earlier problems (especially, Gopalakrishnan et al., 1997) was to maximize some major of PM effectiveness (number of tasks, composite priority of tasks completed, etc.) within the constraints of the available time and workforce. While our problem does incorporate explicit workforce constraints, it does not have explicit time-constraint. Thus, the possible savings in our problem come from “faster completion of all candidate required tasks” instead of “completion of a subset of the candidate tasks within the available limited time-window”. While the common essence of the goals is to perform PM more efficiently, the context is different. The applicability of our research is to contexts (such as overhaul maintenance of heavy equipment) that are fundamentally different from those time-constrained contexts (such as machinery maintenance between production batches or aircraft routine maintenance between successive flights) where the earlier models could be useful.

Our problem could also be extended to the multiple aircraft overhaul case by aggregating the tasks to which PM workforce can be assigned so that the overall makespan for all of the aircraft is minimized. On the other hand, if aircraft are serviced on a first-come, first-serve basis, the multiple aircraft situation is simply an extension where the scheduling problem is solved for each aircraft sequentially, and where each aircraft's PM schedule is affected by the preceding aircrafts' schedules in terms of workforce availability while those aircraft are being serviced (Lam, 1995). The same scheduling problems are inherent in ship and railroad yards.

Mathematically, this problem can be characterized as a variation of n -jobs, m -machines job shop problem where, (1) each job (PM task) has only one processing step, and (2) the job ties up one or more machines of different types (PM workers) for its processing duration. This problem is more complex than the n -jobs, m -machines job shop due to simultaneous utilization of multiple machines by a job. Since the standard job-shop scheduling problem has been known to be NP-hard (Adams, Balas, and Zawack, 1988), our problem is clearly NP-hard.

Traditionally, optimization algorithms have been tried on the standard job shop scheduling problem (Lageweg, Lenstra, and Rinnooy Kan, 1977; Carlier and Pinson, 1989), with some success, to smaller size problems. However, the NP-hard nature of the problem precludes the possibility of exact algorithms for large-size, real-world problems. For larger problems, deterministic heuristics have been traditionally applied using some form of priority rules (e.g., Adams et al., 1988). Since the late 1980s, new approaches such as simulated annealing (SA) and evolution techniques such as genetic algorithms have been applied to the problem (Biegel and Davern, 1990; Looi, 1992; Van Laarhoven, Aarts, and Lenstra, 1992). Glover and Greenberg (1989) provide a good review of simulated annealing, genetic algorithms, neural networks and other new approaches for heuristic search in combinatorial optimization problems.

Recently, a new paradigm called evolution strategies (ES) has attracted researchers' attention for application to complex computational problems (Back and Schwefel, 1993; DeJong, 1993; Michalewicz, 1994). They have been applied very recently to a diverse set of domains including task scheduling in distributed systems

(Greenwood, Gupta, and McSweeny, 1994) and maintenance scheduling (Greenwood et al., 1995). However, none of these papers including Greenwood et al. (1995) has applied ES algorithms to our complex resource-constrained PM problem. ES algorithms, like genetic algorithms (GA), belong to the general class of algorithms called Evolution Algorithms (EA).

4. EVOLUTION STRATEGIES

The 1990s have seen a growing interest in the use of evolution algorithms (EA) to computationally complex problems. Two major paradigms of EA, namely, genetic algorithms (GA) and evolution strategies (ES), have been applied in a wide range of domains. Both, GA and ES, emulate biological principles of adaptive selection found in nature. Both are capable of obtaining good solutions to computationally hard problems (Davis, 1991; Back, 1996). Each generation (iteration of an ES) takes a population of individuals (potential solutions) and modifies the genetic material (problem parameters) to produce a new offspring. While both the parents and the offspring are evaluated, only the fittest individuals (better solutions) survive over multiple generations. This means an ES simultaneously investigates several regions of the search space, thus finding comparable solutions to those found by other heuristic techniques (such as simulated annealing) but in far less computation time. Additionally, an ES can also directly incorporate deterministic heuristics leading to even better solutions (Michalewicz, 1994; Nissen, 1993). The particular genetic encoding for an individual is referred to as the *genotype*. New genotypes are created by special mutation operators which modify the genetic material. Decoding this genetic material gives observed characteristics of the individual, referred to as the *phenotype*. The fitness of the genotype quantifies how closely the observed characteristics meet a desired objective. Highly fit individuals display highly desirable characteristics. The ES terminates after a fixed number of generations (Γ) have been produced and evaluated, or earlier if an acceptable solution is found.

GA and ES were developed independently. While each maintains a population of trial solutions, imposes random changes to these solutions, and incorporates selection to

determine which solutions survive, there are a number of differences. The most critical difference is that GA are concerned with the genotype level, and attempt to model genetic operators that exist in nature (such as crossover). ES are more concerned with phenotypic effects and emphasize mutational transformations (Fogel, 1995-a). Recent research has shown that macro-mutations such as crossovers are not always necessary to produce satisfactory performance of an evolution algorithm. Indeed, it has been suggested that overemphasizing low-level operators and components (which is done in GAs) causes the loss of the selection of good behaviors in evolutionary processes (Fogel, 1995-b). Finally, GA has been found to be less efficient as compared to ES in solving problems with many constraints (Sumichrast, Oxenrider, and Clayton, 2000).

Simulated Annealing (SA) has also been tried on several combinatorial optimization problems (Eglese, 1990; Johnson et al., 1991; Van Laarhoven et al., 1992). Simulated annealing attempts to find out the global minimum of a cost function f over a finite solution space S as follows. The method starts from a random initial solution, and a neighboring solution is examined. Better solutions are always selected while worse solutions may be accepted with an exponentially decreasing probability. The probability of accepting a move which causes an increase δ in f is called the acceptance function and is normally set to $\exp(-\delta/T)$ where T is a control parameter corresponding to temperature in an annealing process. Thus, small increases in f are more likely to be accepted than large increases, and when T is high most moves will be accepted. This helps to escape local optima (Eglese, 1990).

SA can find good results in certain types of optimization problems though its excessive computation time has been cited as a major disadvantage (Bollinger and Midkiff, 1994). This has led some researchers to suggest parallel SA algorithms. Unfortunately, even these can produce unsatisfactory results. Researchers have demonstrated that parallel random search can outperform parallel SA (Braschi, Ferreira, and Zerovnik, 1989).

In summary, the relative strengths of the ES approach over GA and SA have been demonstrated theoretically and empirically. Further discussion on these comparisons can be found in Back (1996). Based on these comparisons, we use ES algorithms to solve the

resource constrained PM scheduling problem. We later provide a direct empirical comparison between SA and ES.

Generic Steps in the ES Algorithm

The algorithm starts by randomly generating an initial population of μ individuals, representing feasible PM schedules (step 1). From this initial population, each individual is mutated to generate one offspring. All of the offspring are added to the population (step 2). All of the 2μ individuals are evaluated to determine their fitness (step 3). Of these individuals, μ fittest individuals are selected for survival while others are discarded (step 4). Steps 2 through 4 are continued until an acceptable solution (per predetermined criteria) is found or Γ generations have been evaluated.

In step 2, each individual is mutated to produce an offspring. The mutation operator produces this offspring by making a small, random perturbation to the problem parameters encoded by the genotype. Exactly how a mutation operator creates an offspring depends on the genotype data structure, thus it is application dependent. It is also possible to have multiple mutation operators. In such cases, the k th mutation operator is applied to an individual with probability p_k . Since $\sum p_k = 1.0$, all individuals will be subjected to a mutation operator. In the next section, we discuss the mutation operators used for solving the workforce-constrained PM scheduling problem. While mutation operators are stochastic in nature, ES is not merely a random search. It concentrates the search in those regions of the search space where good solutions have previously been found. Regions with little promise are pruned out, because individuals with high fitness survive over the weaker ones. Also, since only the fittest μ individuals (parents and offspring) survive after each generation, ES is able to converge monotonically over generations to an acceptable solution. As problem size increases, higher number of generations is required as more mutations are required for ES convergence. Generally, the quality of solutions is proportional to μ and Γ .

5. APPLYING ES TO THE WORKFORCE-CONSTRAINED PM PROBLEM

The application of ES to the workforce-constrained PM problem is described in the following sections: identification of genotype (candidate sequence of tasks), computation of schedule and makespan (phenotype), and description of mutation operators.

PM Problem Genotype

To apply ES to the workforce-constrained PM problem, we first specify the genotype, that is, the data structure that encodes the problem parameters. Recall that for the PM problem, the N PM tasks (labeled 1, ..., N) must be scheduled for execution. Each task has a known completion time and a known set of skills that technicians must possess to perform the task. For example, a PM task may require the assignment of an electrician for three hours and a mechanical technician for one hour. Given a set of technicians (with specific skills), the goal is to assign them to the N PM tasks such that all tasks can be completed in minimal time. Thus, the genotype for ES implementation is an N element ordered list of integers corresponding to the N tasks. The left-to-right order of the list specifies the sequence in which the tasks are to be assigned available maintenance personnel. The ordered list represents a candidate sequence.

Computation of Schedule and Makespan (Phenotype)

L is the set of technicians available. Individual elements of this set represent individual technicians, who may have one or more of the skills required for executing the N tasks. Note that the cardinality of this set varies as PM tasks are executed because all personnel assigned to a task may not be needed for the entire task execution time. For example, a mechanic could be required to work 5 time units while an electrician might be required to work for only 2 time units. Once a technician has completed a PM task, (s)he is “returned” to L to await assignment to another task.

The *schedule* for a specific genotype is computed as follows. The first task in the genotype task list is selected and all needed personnel for this task are taken from L . If L is not empty, all remaining tasks are scanned (from left to right) to check if any other task could be started. Tasks cannot hoard resources. This means all needed personnel must be available in L at the instant the task is scheduled to begin. Technicians are assigned to other PM tasks whenever they have completed their current task. Thus, for the above example, we could re-assign the electrician to another tasks after 2 time units while the

mechanic could be assigned to another task after 5 time units. The process of assignment of PM personnel to the tasks in the list continues until all PM tasks are completed. The completion time for the last task determines the schedule length or makespan. The lower the makespan, the higher is the fitness of the phenotype.

Mutation Operators

The ES uses two genetic operators to produce offspring from parents. The *insertion operator* selects a task from the task list and inserts it at some other randomly chosen position in the task list. This operator has the effect of either advancing or delaying (in time) the chosen task. The execution order of all other tasks remains unchanged. The *recombination operator* randomly selects two points in a task list and perturbs the execution order of tasks between these points. Again, the execution order of all other tasks remains unchanged. A simple example of these operators is shown in Figure 1. Part (a) demonstrates the insertion operator and Part (b) shows the recombination operator. Note that in a generation, the insertion operator was applied to a parent for producing offspring with a probability of 0.3, and the recombination operator was applied with a probability of 0.7.

Insert Figure 1 About Here

6. COMPARISON OF THE ES APPROACH WITH THE EE APPROACH

ES represents an economical method of selectively examining solutions in the solution space of the PM problem. On the contrary, exhaustive enumeration (EE) examines each feasible solution in the solution space. Thus, for an N task problem, the EE approach needs to examine $N!$ solutions. Thus, the ES approach should yield the optimal solution more efficiently than the EE approach in a majority of cases. The magnitude of the economy should increase significantly with an increase in problem size.

Experimental Design

To validate this ES computational economy vis-à-vis the EE approach, a set of sixty problems was executed using both EE and ES. Of these, problems 1 through 20 entailed

five tasks, problems 31 through 40 had ten tasks, and problems 41 through 60 were eleven-task problems. The analysis was limited to smaller problems due to computational efforts required for EE. The notation and problem specification are explained below.

N = number of tasks

TS = total number of skills available in the workforce = 2 (say, a = mechanic and b = electrician)

W_a = number of workers with skill a

W_b = number of workers with skill b

$W_{a\&b}$ = number of workers with both skills a and b

W = total number of workers = $W_a + W_b + W_{a\&b}$

SPT = maximum number of skills required per task

SPW = skills per worker (1=single-skilled workers, 2 = double-skilled workers)

Specifications for problems 1-10 and 11-20 were identical except that 1-10 used single-skilled workers ($W_a=3$, $W_b=5$, $W_{a\&b}=0$) while 11-20 used double-skilled workers ($W_a=0$, $W_b=0$, $W_{a\&b}=8$). Similar distinction was designed between problems 21-30 and 31-40, and between 41-50 and 51-60. For each of problems 1 through 10, 21 through 30, and 41 through 50, the specific task information was individually specified. For example, for problem 6, the following data was specified:

(A) General Problem Parameters

$N = 5$

$TS = 2$ (i.e., skills a and b); where a = mechanic, b = electrician

$SPT = \text{Maximum Number of Skills Required Per Task} = 2$

$SPW = \text{Skills per Worker} = 1$ (i.e., single-skilled workforce)

$W = \text{Number of Workers} = 8 = W_a + W_b$

$W_a = \text{Number of Workers with Skill } a = 3$ (i.e., persons a_1, a_2, a_3)

$W_b = \text{Number of Workers with Skill } b = 5$ (i.e., persons b_1, b_2, b_3, b_4, b_5)

(B) Specific Tasks Skill Requirement Information

<i>Task</i>	<i>Task Time</i>	<i>Skills (# of Workers) and Duration Required</i>
1	6 hrs	b(2) - 6 hrs
2	15 hrs	a(2)- 5 hrs, b(3) - 15 hrs
3	10 hrs	a(2) - 8 hrs, b(1) - 10 hrs
4	20 hrs	a(1) - 20 hrs, b(4) - 10 hrs
5	5 hrs	a(2) - 5 hrs, b(3) - 4 hrs

The worker assignment was determined using two rules. First, a task can't be started unless all of the workers with requisite skills are available. Second, if a person is used for part of a task, he works from the start of the task for that amount of time, and is released for assignment to another job. For example, suppose a task starts at time 26 and requires 8 hrs to do. If it requires a(2) - 5 hrs and b(3) - 8 hrs, then the 2 workers with skill 'a' will work on it from 26 to 31. They will be available for another task at the end of 31. The three workers with skill 'b' will work for all 8 hours from 26 through 34.

Execution

All of the sixty problems were run using EE first. The optimum solution makespan and CPU time (CPUEE) were noted. The number of solutions examined by EE (SOLNEE) corresponded to $N!$. Next, each problem was run using ES to converge to optimal makespan, and the CPU time (CPUES) and number of solutions required to attain the optimal makespan (SOLNES) were noted. The following two indices of ES efficiency (vis-à-vis EE) were computed for each problem:

$$\begin{aligned} \text{CPUPARS} &= \text{CPU parsimony index} \\ &= \text{CPUES} * 100 / \text{CPUEE} \\ \text{COMPARS} &= \text{Computational parsimony index} \\ &= \text{SOLNES} * 100 / N! \end{aligned}$$

Results

Table 2, Table 3, and Table 4 summarize the results of ES comparison with EE. For five-task problems, the ES approach had to screen 18.12 percent of the feasible solutions on average to find the optimal solution. A maximum search effort corresponded to screening of 75.00 percent of the feasible space of 120 solutions. The CPU time required was

negligible for both ES and EE. For ten-task problems, the computational efficiency from ES was much more significant. The ES approach had to screen only 0.45 percent of the feasible solutions on average to find the optimal solution. The CPU time required by ES to find the optimal solution was only 1.87 percent of that required by the EE approach. While the maximum search effort required by the ES corresponded to screening of 2.61 percent of the feasible space of 3,628,800 solutions, the CPU time required was 11.77 percent of that required by the EE approach. For eleven-task problems, the computational efficiency from ES was even more significant. The ES approach had to screen only 0.091 percent of the feasible solutions on average to find the optimal solution. The CPU time required by ES to find the optimal solution was only 0.12 percent of that required by the EE approach. While the maximum search effort required by the ES corresponded to screening of 0.35 percent of the feasible space of 39,916,800 solutions, the CPU time required was only 0.48 percent of that required by the EE approach. These results clearly demonstrate the practical utility of the ES approach even for relatively small practical problems, and its increasing relative efficiency vis-à-vis the EE approach with increasing problem size.

Insert Table 2, Table 3, and Table 4 About Here

In addition, we solved a sample of PM problems of eight different sizes ($N = 5, 6, 7, 8, 9, 10, 11, 12,$ and 13) using the EE approach. The average CPU times (seconds) for these eight problem sizes with EE and ES execution were as follows: CPUEE=0, CPUES=0 (for $N=5$); CPUEE=0, CPUES=0 (for $N=6$); CPUEE=1.0, CPUES=0 (for $N=7$); CPUEE=9.0, CPUES=0 (for $N=8$); CPUEE=81.0, CPUES=0 (for $N=9$), CPUEE=810, CPUES=0 (for $N=10$); CPUEE=10,000, CPUES=1 (for $N=11$); CPUEE=130,000, CPUES=3 (for $N=12$); and CPUEE=1,690,000, CPUES=5 (for $N=13$). The computational requirements indicate an exponential relationship between the number of tasks and CPU time for the EE approach. The corresponding CPU times for ES execution of the same problems demonstrates that the efficiency of the ES approach

should increase tremendously as we approach real-world PM problems of the size and complexity experienced in heavy maintenance facilities for aircraft or railroad cars.

7. ES PERFORMANCE FOR LARGE-SCALE PROBLEMS

To further demonstrate the utility of ES algorithms to the workforce constrained PM problem, we analyzed the quality of solutions offered by ES under a wide range of problem settings and at different levels of ES parameters. A total of 852 experiments were conducted for various ranges of problem characteristics and ES execution parameters. For each experiment, two critical solution quality measures, namely, *CPU time* (CPU) and *makespan* (MSPAN) were recorded. The results were analyzed to examine the effects of the task-related and workforce-related problem characteristics and the ES parameter μ on the two solution quality measures. While problems described in literature can serve as a basis for experimental designs for well-documented problems such as assembly line balancing (Leu, Matheson, and Rees, 1994) or traditional job-shop scheduling (Rachamadugu, Nandkeolyar, and Schriber, 1993), the problem addressed by us in this paper has not been studied in the past. Hence, the experimental design had to be developed from scratch. The PM problem parameters were chosen based on our prior knowledge of similar maintenance environments and discussions with operations research analysts at a commercial airline company. The ES parameters were based on considerations of reasonable computational efforts. We ran sample experiments with different parameter settings and arrived at the chosen parameters upon the preliminary findings. This procedure is generally followed for new categories of algorithms that have not been applied in a particular context (e.g., Sumichrast et al. 2000).

Experimental Design

Following the scheme used by Leu et al. (1994), we divided the factors for experimental design into two types: problem characteristics and ES parameters. Problem characteristics were grouped into two types: task-related and workforce-related. These reflected problem size as well as complexity of scheduling. Five task-specific factors were identified: number of tasks to be scheduled (N), distribution of task times (Y), total skills required for all tasks (TS), skills required per task (SPT), and number of persons

with a required skill needed to perform the task (WPT). Three levels of N were used (N = 100, 500, and 1000). For each of the N tasks, task times had to be generated. These were generated from binomial distributions following Lue et al. (1994) using two different levels of variance among task times (Binomial probability Y = 0.2 and 0.5). A total of 852 experiments were identified using the factors and levels presented in Tables 5-A and 5-B. For the sake of brevity, the derivation of these 852 experiments is detailed in the Appendix.

Table 5-B shows three measures of solution quality. Each of the 852 experiments was executed using the stopping rule of “no improvement over the solution at a generation for ten subsequent generations”. The solution at which computations were stopped was denoted as the quasi-optimal solution from ES. The number of generations required to attain the quasi-optimal solution was noted as the required number of generations (GEN). CPU execution time for each run (CPU) represented another direct measure of computational requirement. The makespan (MSPAN) corresponding to the final solution was also noted. For each experiment, the actual number of solutions examined by ES was also recorded.

Insert Table 5-A and Table 5-B About Here

Statistical Results

Each of the 852 runs was executed on an Ultra Sparc I machine (167 Mhz, 128 MB RAM). Results obtained by applying the ES algorithm to the 852 runs were analyzed statistically using SPSS statistical package (Norusis, 1994). The CPU time required varied between 0.02 and 3,248.00 seconds (mean = 86.02). Number of generations required for solution convergence (GEN) ranged from 10 to 172 (mean = 19.3). Schedule makespan (MSPAN) varied between 10.00 and 11,297 minutes (mean = 678). From these summary statistics of computational efforts, it is evident that ES approach can be easily implemented for real-life PM problems involving hundreds of tasks.

Regression analysis was used to estimate power expressions for the two outcome measures, CPU time and Makespan, as functions of the various task-related and workforce-related factors. These are summarized in Table 6. The estimations provide several interesting insights. First, Model 1 indicates that CPU time requirements using ES were proportional to $N^{1.34}$ in contrast to the fact that the CPU time from EE varied exponentially with the number of tasks (e^N). This comparison provides an empirical evidence of the computational efficiency and practical utility of ES for real-world PM problems. The function also showed that CPU time was proportional to $SPT^{0.96}$, $WPT^{0.35}$, $W^{1.5}$, and $SPW^{-0.4}$. Thus, as the task complexity increases, the computational requirement increases. One explanation could be that with growing complexity of the problem, an existence of multiple optimal solutions may be less probable. Hence the probability of rapidly reaching the optimal solution is reduced. CPU time proportionality to $W^{1.5}$ is attributable to an increase in the number of feasible solutions with increasing W . Finally, CPU variation with $SPW^{-0.4}$ indicates that as multi-functionality of a worker increases, the flexibility of assignment of workers to span a set of tasks increases. Thus, there is a higher probability of quickly finding an optimal assignment.

In Model 2, makespan was found to be proportional to $N^{0.97}$, $SPT^{1.18}$, and $WPT^{1.07}$, while it varied with $W^{-1.05}$ and $SPW^{-0.17}$. Thus, larger problem size (N) and higher complexity of tasks (SPT and WPT) should result in higher makespan, whereas the use of larger workforce (W) and multi-functionality (SPW) should reduce the makespan. Both these findings have practical implications for management of such a PM program.

Finally, the number of solutions examined to arrive at the final solution (SOLNES) was regressed on the task-related and workforce-related factors (Model 3). The number of solutions examined was found to vary with $N^{0.11}$, $SPT^{0.51}$, and $WPT^{0.26}$. Thus, the task complexity rather than the number of tasks seemed to affect the number of solutions examined. This computational effort measure was also proportional to $W^{-0.24}$. It was not influenced by SPW in a statistically significant manner. However, these results should be interpreted with caution because the factors and the multiplicative form of the regression expression did not estimate SOLNES satisfactorily ($R^2 = 0.11$).

Insert Table 6 About Here

8. ES AND SA PERFORMANCE COMPARISON

The complexity in solving the PM problem necessitates use of heuristic search algorithms. It is therefore natural to ask if any one particular algorithm outperforms others for the PM problem. The No Free Lunch (NFL) theorem (Wolpert and Macready, 1997) suggests that good performance of a particular search algorithm for a certain class of problems is no guarantee that it will perform equally well for any other class of problems. This means one cannot prefer, for example, simulated annealing (SA) over other algorithms for the PM problem just because it works well for, say, the traveling salesman problem.

Direct comparisons between algorithms are insightful. However, without a conscientious attempt to make the comparisons fair, the results may be inconclusive, or in the worst case, completely wrong (Greenwood, 1997). While the focus of our paper is on testing the performance of ES for a wide range of specifications for the particular PM problem under consideration, we also compared the relative performance of ES against SA for this problem.

To compare the speed of convergence of the two algorithms under a wide spectrum of problem specifications, all of the 852 problem instances (described in Section 7) were again solved using SA. Specifically, the target makespan (quasi-optimal solution) attained from the prior ES run was noted for each problem. As mentioned before, this was the makespan obtained using the stopping rule of no improvement over ten successive generations. Next, SA iterations were executed for each problem until either the exact target makespan or an even better value is obtained. For each SA experiment, the following scheme was used:

```
temp = 1.0; // lower makespan represents higher solution fitness
```

```
while exit criteria not met do
```

```
{
```

```
    zz = prob(); // get random number between 0 and 1
```

```

pr = exp(-(chromosome[1].fitness – chromosome[2].fitness)/temp);
if (pr > zz)
    chromosome[1] = chromosome[2]; // take the weaker chromosome
temp = temp * 0.98;
}

```

end do while

The variable “temp” is the cooling temperature that controls the annealing process in the SA algorithm. This temperature is reduced each iteration by multiplication with a factor of 0.98, which was selected based on multiple SA runs conducted for sample problems. The number of SA iterations represent the number of solutions examined by SA to attain the target quasi-optimal solution and can be directly compared with the corresponding value of SOLNES (number of solutions examined by ES) to evaluate the relative convergence speeds of the two algorithms. The ratio of the number solutions examined by SA and ES (defined as ρ) was computed for each experiment. A comparative analysis across the 852 runs yielded an average ρ value of 11.89. Thus, SA had to examine almost “12 times” as many solutions as those examined by ES. These results clearly demonstrate the superiority of the approach over the SA approach for the specific class of PM problems.

9. CONCLUSIONS

In this paper, we considered the PM scheduling problem faced by all major overhaul maintenance facilities where aircraft, ships, locomotives, or other heavy equipment needs to be turned around as early as possible. However, available time window is not the explicit restricting factor in scheduling, and the servicing of the unit must be completed in its entirety before the unit is returned to field usage. Thus, we have considered the problem of assigning single- or multiple-skilled workers to a set of PM tasks requiring single- or multiple-skills. The objective is to compute the assignment, which will lead to a minimum makespan (time to complete all tasks). We examined the applicability of an evolution strategy (ES) algorithm to this NP-hard problem. The efficiency of the algorithm vis-à-vis exhaustive enumeration was illustrated for sixty small problems. The computational efficiency of ES vis-à-vis exhaustive enumeration was further confirmed through a large-scale experimental design covering 852 experiments. ES comparison to

SA across the 852 experiments indicated that SA had to examine almost 12 times as many solutions as those examined by ES to reach the quasi-optimal solutions.

These results clearly demonstrate the utility of the ES approach to solve large-scale complex PM scheduling problem faced by heavy maintenance facilities within realistic computing efforts, and the superiority of this approach over the SA approach for the specific class of PM problems. Various regression expressions also provided insights on the impact of task-related and workforce-related factors on computational efforts and makespan. The makespan expression indicated that makespan increases with increased problem size and task complexity while it decreases with a larger workforce and multiple-skilled workers. Also, CPU time increased with increased problem size and complexity while it decreased with multiple-skilled workforce.

The results reported in this paper of direct relevance to the heavy maintenance overhaul functions in aircraft service centers, shipyards, and rail-yards. The research could be extended in several directions. On one hand, the ES algorithms could be used for many operations management problems which have been known to be NP-hard. Many scheduling problems apart from the one considered here fall into this class of problems (e.g., job-shop scheduling). The basic ES paradigm is general enough to be applicable to these other problems. Of course, as illustrated in the previous section, the genotype must change to capture the new problem's parameters and the solution fitness measure must also be suitably altered. At the present time, no other researchers have addressed the type and size of problem discussed in this paper. Hence we have no other results to compare against. The comprehensive experimentation implemented in this work should make it a good benchmark for comparing the performance of other algorithms.

REFERENCES

- Adams, J., Balas, E., and Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3), 391-401.
- Barlow, R. and Hunter, L. (1960). Optimum preventive maintenance policies. *Operations Research*, 8, 90-100.

- Back, T. (1996). *Evolutionary algorithms in theory and practice*. Oxford, England: Oxford University Press.
- Back, T. and Schwefel, H.P. (1993). An overview of evolution algorithms for parameter optimization. *Evolutionary computation*, 1(1), 1-23.
- Biegel, J.E. and Davern, J.J. (1990). Genetic algorithms and job shop scheduling. *Computers and Industrial Engineering*, 10(1-4), 81-91.
- Bollinger, S. and Midkiff, S. (1994). Heuristic techniques for processor and link assignments in multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 5(2), 113-120.
- Braschi, A., Ferreira, A., and Zerovnik, J. (1989). On behavior of parallel simulated annealing. *Parallel Computation*, 263-268.
- Carlier, J., and Pinson, E. (1989). An algorithm for solving the job-shop problem. *Management Science*, 35(2), 164-176.
- Cornell, P., Lee, H., and G. Tagaras (1987). Warnings of malfunctions: The decision to inspect and maintain processes on schedule or on demand. *Management Science*, 33(10), 1277-1290.
- Davis, L. (Ed.) (1991). *Handbook of genetic algorithms*. New York, NY: Van Nostrand Reinhold.
- DeJong, K.A. (1993). *Evolutionary computation*. Boston, MA: MIT Press.
- Dekker, R. and Smeitink, E. (1994). Preventive maintenance at opportunities of restricted duration. *Naval Research Logistics*, 41, 335-353.
- Dijkstra, M.C., Kroon, L.G., Salomon, M., Van Nunen, J., and Van Wassenhove, L.N. (1994). Planning the size and organization of KLM's aircraft maintenance personnel. *Interfaces*, 24(6), 47-58.
- Eglese, R.W. (1990). Simulated annealing: A tool for operational research. *European Journal of Operational Research*, 46, 271-281.
- Flynn, B.B., Schroeder, R.G., and Sakakibara, S. (1994). A framework for quality management research and an associated instrument. *Journal of Operations Management*, 11, 339-366.
- Fogel, D. (1995-a). *Evolutionary computation*, (2nd Ed.) IEEE Press, Piscataway, NJ

- Fogel, D. (1995-b). Phenotypes, genotypes, and operators in evolutionary computation. *Proceedings of the IEEE Conference on Evolutionary Computation*, 193-198.
- Glover, F. and Greenberg, H. (1989). New approaches for heuristic search: A bilateral linkage with artificial intelligence. *European Journal of Operational Research*, 39, 119-130.
- Golabi, K., Kulkarni, K.R., and Way, G. (1982). A statewide pavement management program. *Interfaces*, 12(6), 5-21.
- Golabi, K. and Shepard, R. (1997). Pontis: a system for maintenance optimization and improvement of US bridge networks. *Interfaces*, 27(1), 71-88.
- Gopalakrishnan, M., Ahire, S., and Miller, D. (1997). Maximizing the effectiveness of a preventive maintenance system: an adaptive modeling approach. *Management Science*, 43(6), 827-840.
- Greenwood, G. (1997). So many algorithms: so little time. *ACM software Engineering Notes*, 22, 92-93.
- Greenwood, G., Gupta, A., and McSweeney, K. (1994). Scheduling tasks in multiprocessor systems using evolutionary strategies. *Proceedings of the IEEE international conference on evolutionary computing*, 345-349.
- Greenwood, G., Ahire, S., Gupta, A., and Munnangi, R. (1995). Parallel implementation of evolutionary strategies. *Proceedings of the international conference on high performance computing*, 469-474.
- Johnson, D.S., Aragon, C.R., McGeoch, L.A. (1991). Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning. *Operations Research*, 39, 378-406.
- Lageweg, B.J., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1977). Job shop scheduling by implicit enumeration. *Management Science*, 24(10), 441-450.
- Lam, M. (1995). An introduction to airline maintenance. In *The Handbook of Airline Economics*, 1st edition, New York, NY: McGraw-Hill, 397-406.
- Leu, Y., Matheson, L., and Rees, L. (1994). Assembly line balancing using genetic algorithms with heuristic-generated initial populations and multiple evaluation criteria. *Decision Sciences*, 25(4), 581-606.
- Looi, C. (1992). Neural network methods in combinatorial optimization. *Computers and Operations Research*, 19(3/4), 191-208.

- McCall, J. (1965). Maintenance policies for stochastically failing equipment: A survey. *Management Science*, 11(5), 493-524.
- Michalewicz, Z. (1994). *Genetic algorithms + Data structures = Evolutionary algorithms*. (2nd Ed.), Berlin, Germany: Springer-Verlag.
- Nissen, V. (1993). Evolutionary algorithms in management science: An overview and list of references. *European study group for evolutionary economics*.
- Norusis, M.J. (1994). *SPSS Advanced Statistics (Version 6.1)*. Chicago, IL: SPSS, Inc.
- Pierskalla, W.P. and Voelker, J.A. (1976). A survey of maintenance models: The control and surveillance of deteriorating systems. *Naval Research Logistics Quarterly*, 23(3), 353-388.
- Rachamadugu, R., Nandkeolyar, U., and Schriber, T. (1993). Scheduling with sequencing flexibility. *Decision Sciences*, 24(2), 315-341.
- Ram, B. and Olumolade, M. (1987). Preventive maintenance scheduling in the presence of a production plan. *Production and Inventory Management, 1st Quarter*, 81-87.
- Saraph, J.V., Benson, P.G., and Schroeder, R.G. (1989). An instrument for measuring the critical factors of quality management. *Decision Sciences*, 20(4), 810-829.
- Sherif, Y. and Smith, M. (1981). Optimal maintenance models for systems subject to failure- A review. *Naval Research Logistics Quarterly*, 28, 47-74.
- Spencer, M.S. and Guide, D.V. (1995). An exploration of the components of JIT: Case study and survey results. *International Journal of Operations and Production Management*, 15(5), 72-83.
- Sumichrast, R.T., Oxenrider, K.A., and Clayton, E.R. (2000). An evolutionary algorithm for sequencing production on a paced assembly line. *Decision Sciences*, 31(1), 149-172.
- Ulusoy, G., Or, I., and Soydan, N. (1992). Design and implementation of a maintenance planning and control system. *International Journal of Production Economics*, 24, 263-272.
- Valdez, F. and Feldman, R. (1989). A survey of preventive maintenance models for stochastically deteriorating single-unit systems. *Naval Research Logistics Quarterly*, 36, 419-446.
- Van Laarhoven, P.J.M., Aarts, E.H.L., and Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, 40(1), 113-125.

- Vuppalapati, K., Ahire, S.L., and Gupta, T. (1995). JIT and TQM: A case for joint implementation. *International Journal of Operations and Production Management*, 15(5), 84-94.
- Wald, M.L. (2000). F.A.A. threatens action that could shut airline. *New York Times*, June 3, A-20.
- Wolpert, D.H. and Macready, W.G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1, 67-82.

TABLE 2: Comparison of Evolutionary Strategy with Exhaustive Enumeration (5 Task Problems, N = 5)

Expt.	Optimum Makespan ¹	ESOPTGEN ² ($\mu = 5$)	COMPARS ³ (%)
1	37	2	8.333
2	35	2	8.333
3	33	2	8.333
4	36	2	8.333
5	40	1	4.166
6	29	4	16.666
7	40	2	8.333
8	40	1	4.166
9	48	2	8.333
10	50	1	4.166
11	27	17	70.833
12	25	16	66.666
13	30	2	8.333
14	30	1	4.166
15	50	1	4.166
16	25	1	4.166
17	35	10	41.666
18	35	1	4.166
19	40	18	75.000
20	40	1	4.666
			COMPARS Summary
		Average	18.125
		Minimum	4.166
		Maximum	75.000
		Median	8.333

Notes

1. The CPU time to attain optimum makespan for all experiments using both ES and EE was negligible.
2. ESGENOPT = Number of generations required by ES (with $\mu = 5$) to reach the optimal makespan.
3. COMPARS = Computational Parsimony Index (%)
 = Total number of solutions examined by ES to reach the optimum makespan * 100 / N!
 = ESGENOPT * μ * 100 / N!

TABLE 3: Comparison of Evolutionary Strategy with Exhaustive Enumeration (10 Task Problems, N = 10)

Expt.	Optimum Makespan	CPUEE (sec)	CPUES ¹ (sec)	CPUPARS ² (%)	COMPARS ³ (%)
21	80	803	0	0	0.008
22	80	813	0	0	0.006
23	110	805	1	0.124	0.007
24	75	806	1	0.124	0.009
25	90	847	0	0	0.006
26	105	888	2	0.225	0.015
27	93	858	62	7.226	1.830
28	87	818	1	0.122	0.013
29	76	839	79	9.416	2.150
30	90	811	0	0	0.006
31	70	782	12	1.535	0.292
32	69	783	20	2.554	0.790
33	90	772	0	0	0.007
34	70	784	1	0.127	0.006
35	78	812	3	0.370	0.086
36	90	845	21	2.485	0.781
37	85	808	9	1.114	0.271
38	80	790	93	11.772	2.610
39	73	826	1	0.121	0.010
40	85	779	1	0.128	0.021
	Makespan Summary	EE CPU Summary	ES CPU Summary	CPUPARS Summary	COMPARS Summary
Average	83.8	813.5	15.35	1.8726	0.446
Minimum	69	772	0	0	0.006
Maximum	110	888	93	11.772	2.610
Median	80	811	1	0.128	0.015

Notes

1. $CPUPARS = \text{CPU Parsimony Index (\%)} = CPUES * 100 / CPUEE$
2. $COMPARS = \text{Computational Parsimony Index (\%)} = \text{Total number of solutions examined by ES to reach the optimum makespan} * 100 / N!$

TABLE 4: Comparison of Evolutionary Strategy with Exhaustive Enumeration (11 Task Problems, N = 11)

Expt.	Optimum Makespan	CPUEE (sec)	CPUES ¹ (sec)	CPUPARS ² (%)	COMPARS ³ (%)
41	86	9471	0	0	0
42	80	11190	0	0	0
43	118	10850	0	0	0
44	75	11041	2	0.018	0.007
45	95	11580	0	0	0
46	111	10719	0	0	0
47	96	10383	48	0.463	0.350
48	90	9817	5	0.051	0.040
49	76	10163	2	0.020	0.008
50	96	10292	0	0	0
51	76	10301	1	0.010	0.008
52	73	10701	0	0	0.002
53	93	10408	0	0	0
54	72	10612	45	0.424	0.350
55	80	11085	47	0.424	0.350
56	93	10206	0	0	0.002
57	88	9805	1	0.010	0.003
58	83	9478	45	0.475	0.350
59	73	9845	47	0.477	0.350
60	91	9972	0	0	0
	Makespan Summary	EE CPU Summary	ES CPU Summary	CPUPARS Summary	COMPARS Summary
Average	87.25	10395	12.15	0.119	0.091
Minimum	72	9471	0	0	0
Maximum	118	11580	48	0.477	0.350
Median	87	10342	0.500	0.005	0.002

Notes

1. $CPUPARS = CPU \text{ Parsimony Index } (\%) = CPUES * 100 / CPUEE$
2. $COMPARS = Computational \text{ Parsimony Index } (\%) = \text{Total number of solutions examined by ES to reach the optimum makespan} * 100 / N!$

TABLE 5-A: Experiment Design: The Basic Thirteen Experiments

Run Number	Number of Skills per Person (SPW)	Total Skills Required (TS)	Skills Required per Task (SPT)
1	1	2	1
2	1	2	2
3	1	5	1
4	1	5	2
5	1	5	3
6	2	2	1
7	2	2	2
8	2	5	1
9	2	5	2
10	2	5	3
11	3	5	1
12	3	5	2
13	3	5	3

TABLE 5-B: Experimental Design: Other Problem Characteristics, ES Parameters, and Performance Measures

Factor	Description	Levels		
<i>Problem-Characteristic Factors</i>				
N	Number of tasks	100	500	1,000
W	Number of workers	10	30	50
WPT	Number of workers needed of a specific skill per task	1	3	
Y	Binomial probability for task time (hour)	0.2	0.5	
<i>Evolution Strategies Parameter¹</i>				
μ	ES Population	5	20	
<i>Performance Measures</i>				
CPU	CPU Time	not applicable	not applicable	not applicable
GEN	Number of Generations	not applicable	not applicable	not applicable
MSPAN	Makespan	not applicable	not applicable	not applicable
SOLNES	Number of Solutions Examined by ES	not applicable	not applicable	not applicable

- Each experiment was executed using the stopping rule of “no improvement over the solution at a generation for ten subsequent generations”. Thus, the number of generations Γ at which this occurs was noted as the required number of generations (GEN).

TABLE 6: Results of Multiplicative Regressions for CPU Time, Makespan, and Number of Solutions Examined by ES Approach¹

	Model 1	Model 2	Model 3
Dependent Variable (y)	CPU	MSPAN	SOLNES
Regression Coefficients () for Tasks-Related Variables			
N	1.34*	0.97*	0.11*
SPT	0.96*	1.18*	0.51*
WPT	0.35*	1.07*	0.26*
Regression Coefficients () for Workforce-Related Variables			
W	1.50*	-1.05*	-0.25*
SPW	-0.40*	-0.17*	-0.15*
Model Fit			
Adjusted R ²	0.68	0.88	0.11
F-Statistic	302 (p=0.00)	1025 (p=0.00)	21 (p=0.00)

¹ All regressions are of the form: $y = c \times N^1 \times WPT^2 \times SPT^3 \times W^4 \times SPW^5$.
Estimated using logarithms of terms of both sides.

* Significant at $p < 0.05$.

APPENDIX

Design of Experiments for Large Scale PM Problems

We divided the factors for experimental design into two types: problem characteristics and ES parameters. Problem characteristics were grouped into two types: task-related and workforce-related. These reflected problem size as well as complexity of scheduling.

Task-Related Factors

Five task-specific factors were identified: number of tasks to be scheduled (N), distribution of task times (Y), total skills required for all tasks (TS), skills required per task (SPT), and number of persons with a required skill needed to perform the task (WPT). Three levels of N were used (N = 100, 500, and 1000). For each of the N tasks, task times had to be generated. These were generated from binomial distributions following Lue et al. (1994) using two different levels of variance among task times (Binomial probability $Y = 0.2$ and 0.5).

Two levels of total skills required for performing all N tasks (TS) were considered, namely, 2 and 5. This means, when $TS = 2$, each of the N tasks may need either one or both of the skills from TS. Thus, the number of skills required per task (SPT) is another factor. It was set at three levels (SPT = 1, 2, and 3). Note that when $TS = 2$, a value of $SPT = 3$ is not feasible. However, when $TS = 5$, all three levels of SPT are possible. In fact, for an experiment with $TS = 5$ and $SPT = 2$, the specific two skills required for a task could be any combination of two tasks from the 5 tasks listed in TS. Similarly, when $TS = 5$ and $SPT = 3$, the three specific skills required for a task could be any combination of three tasks from the 5 tasks listed in TS. Finally, once the specific skills needed for a task are known, the number of persons needed of each skill required for the task (WPT) must also be specified. We assigned two practical levels of WPT, namely, 1 and 3 workers.

For example, let us say we have a PM system which needs either one or both of only two skills (mechanics and electricians) for performing any of a total of 100 tasks.

Here, $N = 100$, and $TS = 2$ (skill 1 = mechanic, skill 2 = electrician). Thus, if a task requires 2 mechanics and 2 electricians, it will have $SPT = 2$, and $WPT = 2$. For reasons of practicality of experimentation efforts, we did not consider the possibility of different numbers of persons required for each skill used in the task. Thus, for the current example, the possibility of a task requiring 2 mechanics and 3 electricians was not considered. Also, it was assumed that the persons working on a task would be tied up for the entire duration of the task. Partial engagement of the workers skilled in a particular skill on a task was ruled out in the experimental design. However, this could be easily incorporated in the ES algorithm.

Workforce-related Factors

Two workforce-related factors were identified: number of workers available (W) and number of skills available per person (SPW). Three practical levels of W were chosen, namely, 10, 30, and 50 workers. Literature on quality management and maintenance scheduling indicates that cross-training PM technicians in multiple skills can be used as a strategy to improve maintenance productivity (Vuppalapati et al., 1995; Gopalakrishnan et al., 1997). Hence, we evaluated ES performance under three different skill-levels. Thus, the factor SPW had three different levels, namely, single-skilled workers, double-skilled workers, and triple skilled workers. It was assumed that in an experiment, only one of these three types of workers would be employed (i.e., all workers W will be either single-skilled, or double-skilled, or triple-skilled), though the algorithm accounted for this possibility in task scheduling.

ES Parameters

Two ES parameters can affect the quality of the final solution, namely, population size (μ) and number of generations (GEN). This is because, the larger the population, the higher the number of solutions examined, and the higher the probability of finding a better solution. Similarly, the higher the number of generations, the more extensive the search, and thus, the higher the probability of finding a superior solution. However, if we specify a stopping rule, then for a given μ , we can find out the value of GEN after which

the final solution will be recorded. Hence, we used two levels of the population size ($\mu = 5, 20$), leaving Γ to be determined as a performance measure.

Experiments

A basic set of 13 runs was designed using the three problem characteristic factors and their levels: SPT, TS, and SPW (Table 5-A). For single-skilled workers ($SPW = 1$), when the total number of skills required for performing a set of tasks is 2 ($TS = 2$), the skills required per task must be either one or two ($SPT = 1$ or 2). Thus, we identified two runs for $SPW = 1$ and $TS = 2$. Next, for single-skilled workers, when $TS = 5$, we could identify three runs for the three different levels stated above for SPT ($SPT = 1, 2$ or 3). Thus, there were five basic runs for $SPW = 1$ (Table 5-A: runs 1 through 5). Again, for double-skilled workers ($SPW = 2$), when $TS = 2$, SPT could have only two levels ($SPT = 1$ or 2). For $TS = 5$, SPT could have three levels ($SPT = 1, 2$ or 3). Thus, there were five basic runs for $SPW = 2$ (Table 5-A: runs 6 through 10). Finally, for triple-skilled workers ($SPW = 3$), $TS = 2$ is not practical. That is, if only two skills at most are required for performing a set of tasks, it doesn't make sense to have a triple-skilled workforce. Hence, for triple skilled workers, only three basic runs were identified, corresponding to $TS = 5$, and $SPT = 1, 2$ and 3 (Table 5-A: runs 11 through 13).

Table 5-B shows the levels of the other problem-specific characteristics and ES parameters. Note that, for the problem-specific factors, there are three levels of the problem size (N), three levels of the number of workers (W), two levels of binomial distribution probabilities for generating task times (Y), two levels of number of persons with each skill required for performing a specific task (WPT). This yields a total of 468 ($13 \times 3 \times 3 \times 2 \times 2$) experiments.

However, of these, 42 experiments are not feasible. They correspond to the specifications where more persons are required of a skill for a task than are available for a specific setting. For example, for experiments with $W = 10$, $SPW = 1$, and $TS = 5$, we assume that the five skills will be equally distributed among the 10 single-skilled workers. Thus, there will be two single-skilled workers possessing each one of the five skills. Hence, for this setting, experiments with $WPT = 3$ are not feasible for any level of

SPT (1, 2, or 3), any number of tasks ($N = 100, 500, 1000$), or any level of Y (0.2, 0.5). Thus, there are 18 ($3 \times 3 \times 2$) such infeasible experiments corresponding to single-skilled workers. Now let us consider those double-skilled workers ($SPW = 2$) experiments where $W = 10$ and $TS = 5$. Within these, runs corresponding to the two levels of SPT (2 and 3) are not feasible for any level of N (100, 500, 1000) or Y (0.2, 0.5). Thus, 12 ($2 \times 3 \times 2$) of the double-skilled workers experiments are not feasible. Finally, consider those triple-skilled workers experiments where $W = 10$ and $TS = 5$. Again, within these, runs corresponding to the two levels of SPT (2 and 3) are not feasible for any level of N (100, 500, 1000) or Y (0.2, 0.5). Thus, 12 ($2 \times 3 \times 2$) of the triple-skilled workers experiments are also not feasible. Hence, of the total 468 experiments, 42 (18 for single-skilled workers, 12 for double-skilled workers, and 12 for triple-skilled workers) are not feasible and were not executed. This resulted in 426 feasible experiments for any specification of ES parameters. Since, we considered two levels of the parameter μ , the experimental design yielded a total of 852 experiments.