ISSN 0511-5728
The West Indian Journal of Engineering
Vol.41, No.2, January 2019, pp.4-12

A Comparison of Memetic Algorithms in a Generator Maintenance Scheduling Problem for Trinidad and Tobago

Neil Ramsamooj [a,Ψ], Laura Ramdath[b], Sanjay Bahadoorsingh[c] and Chandrabhan Sharma[d]

[a] Office of the Dean, Faculty of Engineering, The University of the West Indies, St Augustine Campus, Trinidad and Tobago, West Indies; E-mail: Neil.Ramsamooj@sta.uwi.edu

[b] Ministry of Works and Transport, Main Administrative Building, Corner Richmond and London Streets, Port-of-Spain, Trinidad and Tobago, West Indies; E-mail: lauraramdath@hotmail.com

[c,d] Department of Electrical and Computer Engineering, Faculty of Engineering, The University of the West Indies, St. Augustine, Trinidad and Tobago, West Indies; [c]E-mail: Sanjay.Bahadoorsingh@sta.uwi.edu; [d]E-mail: Chandrabhan.Sharma@sta.uwi.edu

[Ψ] *Corresponding Author*

(Received 29 May 2017; Revised 08 May 2018; Accepted 29 June 2018)

Abstract: *Power generation companies have to meet consumer demand and reliability criteria. The maintenance of generators needs to be performed regularly to ensure a reliable supply of electricity. An increase in repair frequency will result in greater maintenance cost together with an expected decrease in capital loss due to generator failure. An optimal maintenance schedule is therefore needed to find the best suited trade-off and lower the overall operation cost. Maintenance scheduling is a combinatorial optimisation problem for which an exhaustive search is typically infeasible due to the large size of the solution space. Metaheuristic optimisation techniques are therefore used to approximate global optimum schedules in finite time. This paper compares the uses of the local search methods (hill climbing, tabu search, simulated annealing) against genetic and memetic algorithms in the solution of the generator maintenance scheduling problem. The solution methods improve the previously implemented solution of generator scheduling for the Power Generation Company of Trinidad and Tobago.*

Keywords: *Generator, maintenance scheduling, tabu search, simulated annealing, genetic algorithm, memetic algorithm*

1. Introduction

Regular maintenance of the generators is required for the stable operation of a power system. The scheduling of generator maintenance is central to power system planning as this schedule affects unit commitment, production cost, fuel scheduling, reliability calculations and other aspects of system planning. A non-optimal schedule will negatively affect these functions. The objective of a maintenance schedule is to minimise or maximise the reserve. The determination of an optimal generator maintenance schedule is a complex combinatorial optimisation problem which has been researched over several decades (Patton and Ali, 1972; Egan et al., 1976; Mukerji et al., 1991; Yellen et al., 1992; Burke and Smith, 2000; Dahal and Chakpitak, 2007).

We briefly outline the following sections of our paper. Section 2 outlines previous solution methods to the generator maintenance scheduling problem. Section 3 of this paper shows that the GMS problem may be modelled as an optimisation problem by specifying the constraints, objective function and solution space. In Section 4, the local search, genetic and memetic algorithms that are used to solve our GMS formulation are described. In Section 5, the results obtained from the different solution methods are compared. In Section 6, a case study is illustrated and a method is demonstrated to improve on a previously implemented solution of generator scheduling problem in industry in Trinidad and Tobago.

2. Solution Methods to the Generator Maintenance Scheduling Problem

The following summarises different solution methods to the generator maintenance scheduling (GMS) problem.

1) *Heuristic techniques* were among the first methods used to determine a generator maintenance schedule. Generators are first ranked by their power outputs and maintenance requirements. A typical heuristic method is a trial and error process that determines a maintenance schedule by applying a sequence of rules to this ranking of generators. Heuristic methods are best suited for small generator systems and cannot ensure a true optimal solution (Schlunz, 2011).

2) Dynamic programming is a general method of reducing a complex problem into sub-problems and using the storage of the sub-problem solutions to obtain a solution of the initial problem. The direct use of dynamic programming methods in the GMS problem is considered infeasible as the complexity of the resulting

models increases greatly with the number of generators. The *modification dynamic programming with successive approximations* (DPSA) has however been successfully applied to the generator maintenance scheduling problem (Ahmad and Kothari, 1998). The DPSA method provides close approximations to a global optimum solution.

3) An integer programming problem is an optimisation problem in which some of the solution variables are constrained to take only integer values (Rao, 2009). The maintenance durations of a GMS problem are specified in integers (weeks), and therefore generator maintenance scheduling problems are necessarily integer programming problems. The main techniques used to solve integer programming technique are the branch-and-bound, cutting plane and Balas methods (Rao, 2009). Of these, the branch-and-bound (Egan et al., 1976) and Balas methods (Mukerji et al., 1991) have been successfully applied to the GMS problem. These methods are computationally expensive (Schlunz, 2011) but provide true optimum solutions. Benders decomposition is an iterative approach to the above standard integer programming methods which has also been applied to GMS (Yellen et al., 1992).

4) A metaheuristic is a high level approach that may be applied to the solution of optimisation problems. These approaches have been widely applied to the GMS problem (Ahmad and Kothari, 1998) as they usually provide close approximations to a global optimum solution within acceptable time. Metaheuristics may be broadly classified as either local searches or as population based. These categories and their application to the GMS problem are discussed in Section 4.

3. Problem Formulation

The generator maintenance scheduling (GMS) problem is an optimisation problem, and therefore consists of the selection of an optimal solution chosen from a solution space. The solution space for the GMS formulation consists of *start time vectors*, subject to various constraints as discussed below. As with the optimisation problems, the choice of an optimal or best solution is achieved by the use of an objective function. We first define the variables in order to precisely describe the model constraints, solution space and objective function. These variables are:

I = number of generators
i = generator index
J = number of periods in planning horizon
j = period index
M_i = maintenance duration of generator i
X_i = start time of maintenance of generator i
ST_i = start time of maintenance window for generator i
ET_i = end time of maintenance window for generator i
C_i = operating capacity of generator i
D_j = demand at period j
R_j = reserve capacity at period j
P_{ij} = output of generator i at period i

$$y_{ij} = \begin{cases} 1 & \text{if generator } i \text{ is in maintenance at period } j \\ 0 & \text{otherwise} \end{cases}$$

ST_i = start time of maintenance window for generator i

Table 1 gives a simple example of input data used in the implementation of the solution of the GMS problem. We use this data example to illustrate the use of the variables given above.

Table 1. Sample Input Data

i	C_i	DS_i	ST_i	ET_i	M_i
1	40	1	2	4	1
2	50	1	1	6	5
3	45	1	3	5	2
4	55	1	2	5	2

In this case, the number of generators is $I = 4$. From further user input, we assumed that the start and end times of the planning horizon are 1 and 6, respectively, so that the number of periods (weeks) in the planning horizon is $J = 6$. Maintenance of generator i is assumed to start at the beginning of a period. Hence, we have:

$$X_i \in \{1,2, \ldots, J\}. \quad (1)$$

However, maintenance must also be done within a time window that is specified by periods ST_i and ET_i within the planning horizon. We therefore have the stronger condition that,

$$ST_i \leq x_i \leq ET_i \quad (2)$$

3.1 Model Constraints

The constraint (Equation 2) may be further strengthened as maintenance of generator i must continue for M_i weeks without interruption. This implies that,

$$ST_i \leq x_i \leq ET_i - M_i + 1 \quad (3)$$

which may be referred to as a *continuous maintenance constraint* for generator i. The input data (for example, see Table 1) in our implementation is validated, so that the constraints specified by Equation (3) are not vacuous, that is,

$$1 \leq ST_i \leq ET_i - M_i + 1 \leq J \quad (4)$$

for each i.

An illustrated example is given here. It is easily verified that $(x_1, x_2, x_3, x_4) = (2,1,3,4)$ is a potential schedule solution as it satisfies the constraint (3) for each generator $i = 1,\ldots,4$ where ST_i, ET_i and M_i are as specified in Table 1. This solution may equivalently be written as an $I \times J$ binary matrix,

$$(y_{ij}) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

where the variable y_{ij} is defined above.

A generator i that is undergoing maintenance may be capable of an output that is a fraction of its full operating capacity C_i. This partial output can be quantified by the use of the fraction,

$$0 < DS_i \leq 1 \quad (5)$$

where $DS_i = 1$ means that generator i is totally unavailable. The output P_{ij} of generator i during period j is, therefore,

$$P_{ij} = C_i(1 - y_{ij}DS_i) = \begin{cases} C_i(1 - DS_i) & \text{if in maintenance} \\ C_i & \text{if fully available} \end{cases} \quad (6)$$

The collective power output of all generators must satisfy the load demand D_j at each period j, and therefore

$$\sum_{i=1}^{I} P_{ij} \geq D_j \quad (7)$$

which from Equation (6) implies that

$$\sum_{i=1}^{I} C_i(1 - y_{ij}DS_i) \geq D_j \quad (8)$$

The Equation (8) describes a *load constraint* for period j.

3.2 Solution Space

As is illustrated in the example, any tuple

$$x = (x_1, x_2, \ldots, x_I) \quad (9)$$

where each x_i satisfies the constraints (Equation 3) may be equivalently written as an $I \times J$ binary matrix (y_{ij}). In this paper, **x** is called a *start time vector* if each x_i satisfies (Equation 3) and the equivalent matrix form (y_{ij}) of the tuple **x** satisfies the constraints (Equation 8). The set of all such start time vectors is the *solution space*, $\mathcal{S}$ of the formulation of the GMS problem. As each x_i is an integer that satisfies (Equation 3), it follows that the size of our solution space is finite and bounded above by the product

$$\prod_{i=1}^{I} (ET_i - M_i - ST_i + 2) \quad (10)$$

and hence the formulation of GMS is necessarily a combinatorial optimisation problem.

3.3. Objective function

Objective functions in most formulations of the GMS problem may be classified using one of three (3) criteria (Schlunz, 2011): *convenience*, *economic* or *reliability*. In this paper, we use the following reliability objective function.

$$\text{Minimise } \sum_j^J \{(\sum_i^I P_{ij}) - D_j\}^2 \quad (11)$$

Equation (11) is a standard objective function used in GMS formulations (Froger et al., 2015), and may be interpreted as the leveling of reserves on the horizon. We briefly explain this interpretation. The reserve capacity R_j is defined as total power output minus load demand for period j, that is,

$$R_j = (\sum_i^I P_{ij}) - D_j \quad (12)$$

and therefore Equation (11) may be rewritten as:

$$\text{Minimise } \sum_j^J R_j^2 \quad (13)$$

Recall from Function (7) that our solution space of start time vectors is constrained so that $R_j \geq 0$ for all periods j. We assume that load demand D_j that forms part of the input data incorporates a sufficiently large reserve margin in addition to actual forecasted load. The objective function (Equation 11) may therefore be viewed as reducing the variability of the reserve capacity R_j with change in period j by minimising the sum of squares as stated in Equation (13). A large value of the sum of squares (Equation 13) (given that each R_j is nonnegative) will usually be the result of a few large outliers together with several R_j values which are approximately zero – such a situation is clearly not desirable from a reliability viewpoint. Conversely, a minimal value for Equation (13) corresponds to the more reliable situation in which the reserves R_j are approximately equal and sufficiently nontrivial for each period j across the horizon.

4. Optimisation Methods

The discussion in Section 3 shows that the GMS formulation is a combinatorial optimisation problem or, more precisely, a nonlinear integer programming problem (Rao, 2009). There are formulations of GMS which have been solved by integer programming methods (Mukerji et al., 1991). For this study, we do not use these methods. Instead, the heuristic techniques are used to obtain optimal solutions of the formulation.

4.1 Local Searches

In any iterated local search method, an initial point x_0 in the solution space is chosen and is then assigned as the current solution point **x** of the first iteration. An improved subsequent point is then chosen from the neighborhood $N(\mathbf{x})$ of this current point **x** where improvement is measured by an objective function. This process is then iterated and this results in a walk through the solution space. A well-defined neighborhood $N(\mathbf{x})$ is therefore required by any local search method for each point **x** in the solution space $\mathcal{S}$.

In this paper, the neighborhood $N(\mathbf{x})$ of a point **x** is defined as those points in the solution space $\mathcal{S}$ that are at a Hamming distance of at most one from the point x, that is, $N(\mathbf{x})$ consists of the points of $\mathcal{S}$ that differ from $\mathbf{x} = (x_1, x_2, \ldots, x_I)$ in at most one position. For example, the point $\mathbf{x} = (x_1, x_2, x_3, x_4) = (2,1,3,4)$ can be shown to be an element of the solution space $\mathcal{S}$ for the input data specified by Table 1 and where demand $D_j = 30$ MW for each period j = 1,…,6. The neighborhood N(x) of this x consists of the start time vectors

(2,1,3,4), (3,1,3,4), (2,2,3,4)
(2,1,4,4), (2,1,3,2), (2,1,3,3)

Local search methods are used as solution techniques for the GMS formulation.

4.1.1 Hill Climbing

This iterated procedure is perhaps the simplest of the local search methods. Each iteration consists of selecting a most optimal point from the neighborhood of the

current point; this selected point becomes the current point in the following iteration. Hill climbing methods are sometimes not suitable for determining global optima as they may potentially become trapped at local optima (Michalewicz and Fogel, 2013).

Algorithm 1: Basic hill climbing pseudocode

```
1  select initial solution x_0
2  x := x_0; x* := x_0
3  while not (Termination criterion) do
4    identify neighborhood N(x) of current solution x
5    select local best solution x' of N(x)
6    replace current solution x with x'
7    update global best candidate solution x*
8  end
9  return global best candidate x*
```

4.1.2 Tabu Search

Consider the case in a hill climbing procedure in which the current point **x** in the iteration is locally optimal, that is, the most optimal solution in the neighborhood $N(\mathbf{x})$ is **x**. The subsequent point in the hill climbing iteration will remain as **x** and hence the procedure becomes trapped at this point. The tabu search method may be regarded as a modification of hill climbing that avoids such pitfalls by using a *tabu list* $\mathcal{T}$ as a memory device in which previously visited solution points are stored. A subsequent point in a tabu search iteration is chosen from $N(\mathbf{x})\setminus\mathcal{T}$, that is, it is chosen from neighbors of the current point **x** that have not been previously visited. In this way, iterations are not trapped at **x** and exploration of the solution space is encouraged.

Algorithm 2: Tabu search pseudocode

```
1  select initial solution x_0
2  initialise tabu list T: = []
3  x :=x_0; x* := x_0
4  while not (Termination criterion) do
5      identify neighborhood N(x) of current solution x
6      select local best solution x' of N(x)
7      replace current solution x with x'
8      insert x' into tabu list T
9      remove oldest entry from T if necessary
10     update global best candidate solution x*
11 end
12 return global best candidate x*
```

4.1.3 Simulated Annealing

In this local search method, exploration of the solution space is forced by the use of a parameter T called *temperature* which is reduced at a predetermined rate. Recall that in a hill climbing iteration, a subsequent point **x**' is necessarily an improvement on the current solution point **x**. However, in simulated annealing it is possible that a subsequent point **x**' may be worse than the current solution point **x**. This is done to encourage exploration of the solution space. The probability of deteriorating moves is higher for larger initial temperature values with a decrease in temperature resulting in this probability being lowered. At small temperatures, this probability is virtually zero with only improving moves being allowed and hence at low temperatures the simulated annealing procedure behaves as a hill climbing iteration.

Algorithm 3: Simulated annealing pseudocode

```
1  select initial solution x_0
2  initialise temperature T
3  set number of iterations l
4  x:= x_0; x* := x_0
5  while not (Termination criterion) do
6    for i = 1 to l do
7      randomly select x' from neighborhood N(x)
8      determine ΔC = cost(x') – cost(x)
9      if ΔC ≤ 0 then
10       replace current solution x with x'
11     else if rand [0,1] ≤ exp(-ΔC/T) then
12       replace current solution x with x'
13     end
14   end
15 update global best candidate solution x*
16 reduce temperature, T
17 end
18 return global best candidate x*
```

4.2 The Genetic Algorithm

In each of the local search methods considered above, a single initial point is chosen from the solution space. This single point is then used to generate a sequence of points with a global optimum candidate being selected from this sequence. A genetic algorithm (GA) differs from a local search in that multiple solution points (collectively referred to as a *population*) are initially selected. This initial population is then used to create a subsequent population via an improvement process that is analogous to the mechanism of natural selection from biology.

This process is then repeated for a predetermined number of iterations (called *generations* in the notation of genetic algorithms), hence creating a sequence of populations. A globally optimal candidate is then selected from these populations. In a genetic algorithm, the improvement process that is used to iteratively create subsequent populations may be subdivided into three (3) stages (Simon, 2013) – *selection*, *crossover* and *mutation*. The implementation of these stages in the solution of the GMS formulation via a genetic algorithm is discussed below.

4.2.1 Selection

In each generation, several solution points (called *parents*) are chosen from the current population; information from these selected points is used in the creation of the elements of the subsequent population (referred to as *children*). Two parents are used to produce two children via the crossover process. To achieve improvement in successive generations, more optimal solution points are chosen as parents with higher probability than those with lower optimal values – this is similar to the evolutionary process in biology in which less fit individuals are unlikely to reproduce.

4.2.2 Crossover

In this paper, we utilise the *single-point crossover* method in the formation of children from parents; other crossover methods are described in Simon (2013). The single point crossover method can be illustrated by the use of the following example. Assume that the input data is as given in Table 1 and consider the following two parents,

$$(2,1,4,3)$$
$$(4,2,3,2)$$

belonging to a population which is a subset of the solution space $\mathcal{S}$. A crossover point is then randomly chosen from $\{0,1,\ldots,I\}$; assuming that the chosen value (indicated by Δ below) is two.

A single point crossover is simply the formation of two children,

$$(2,1_{\Delta}3,2)$$
$$(4,2_{\Delta}4,3)$$

by switching all information of the above parents beyond the crossover point.

4.2.3. Mutation

Small random changes called mutations are made to children after the crossover process. This is done to introduce some measure of diversity between successive populations and to prevent premature convergence of solutions.

Consider the possible child

$$(x_1, x_2, x_3, x_4) = (2,1,3,4)$$

that is the result of crossover. Mutation of this solution point is achieved by changing each x_i with some small probability p (usually $p \approx 0.02$). If this probability occurs then x_i will be changed to a value such that the modified solution satisfies the constraints (Equations 3 and 8).

Algorithm 4: Genetic algorithm outline

1 select initial population P_0
2 choose best individual **x*** from P_0
3 set number of generations N
4 **for** $i = 0$ **to** $N - 1$ **do**
5 select parents from current population P_i
6 use crossover to determine children
7 apply mutation to children
8 form new population P_{i+1} from children
9 choose best individual from P_{i+1} and update global best **x*** if necessary
10 **end**
11 **return** *global best candidate* **x***

4.3. Memetic Algorithms

According to Dawkins (2006), a *meme* is a cultural replicator such as an idea, belief or song. Memes, in a manner similar to the genetic material of individuals, undergo replication, propagation and mutation. The notion of fitness or optimality may also be applied in that fitter memes are more likely to survive and propagate within a society. It follows that memes may be considered as being subject to a cultural evolutionary process (Heylighen and Chielens, 2009) with the goal of *memetic algorithms* being to model such processes.

A memetic algorithm usually carries the same replication aspects as a genetic algorithm but also incorporates problem specific or local data. The inclusion of local data may be done by hybridisation (Moscato and Cotta, 2003), and such hybrids form a subclass of memetic algorithms. Our method of hybridisation follows the method used by Burke and Smith (2000) in which a local search is applied to each member of the population generated after each of the i^{th} iterations of a genetic algorithm.

Algorithm 5: Memetic algorithm outline

1 select initial population P_0
2 choose best individual **x*** from P_0
3 set number of generations N
4 **for** $i = 0$ **to** $N - 1$ **do**
5 select parents from current population P_i
6 use crossover to determine children
7 apply mutation to children
8 form intermediate population $\acute{P}_i$ from children
9 use each optimal solution vector of the local search applied to each member of $\acute{P}_i$ to form new population P_{i+1}
10 choose best individual from P_{i+1} and update global best **x***
11 **end**
12 **return** *global best candidate* **x***

In this paper, we consider three (3) hybrids

Hybrid HC, Hybrid TS, Hybrid SA

that are formed by respectively applying the distinct local search methods of hill climbing, tabu search and simulated annealing to the populations generated by a genetic algorithm.

5. Results

Numerical results are presented below for the seven (7) solutions methods considered (see Section 4). These

solutions may be classified as local searches (hill climbing, tabu search, and simulated annealing) and evolutionary methods (genetic and memetic (Hybrid HC, Hybrid TS, and Hybrid SA)). We also compared the methods to Automated Optimised Generator Outage Scheduler (AOGOS) which is a previously implemented solution of the GMS problem by Sharma and Bahadoorsingh (2004). These solutions were implemented in MATLAB and were executed on a Precision 7710 laptop. Certain code segments of the solutions were derived from Simon (2015).

5.1 Validation

The size of the solution space $\mathcal{S}$ of the GMS formulation increases exponentially with increase in the number of generators I. Therefore, an enumeration of $\mathcal{S}$ is typically not feasible for realistic input data in which the number of generators I usually exceeds twenty (20). However, for sample input data with small I, it is possible to enumerate the solution space $\mathcal{S}$. An enumeration (with corresponding fitness values) was implemented for the sample input data given in Table 1.

Optimal solution vectors were easily obtained from this enumeration and these results were used to validate various implemented solutions of the GMS problem. Further validation may be noted in the case of large I as the optimal values obtained from each of the solution methods (for fixed input data) were approximately equal to each other within reasonable tolerance.

5.2. Comparison of Solution Methods

We use input data from the IEEE Reliability Test System (Power Market Subcommittee, 1979) to graphically compare the convergence of our solution methods. This data set (also used in Sharma and Bahadoorsingh (2004)) describes a $I = 32$ generator system with a $J = 52$ week horizon and has the same tabular form as the sample input data from Table 1. A constant load demand $D_j = 2700$ MW is assumed for each week; note however that our solution implementation may be easily modified to accommodate variable load demands. The minimum fitness value achieved from among all of our solution methods for this input data was 1.068547×10^7. This value was obtained by Hybrid SA after an execution time of 1479 seconds for a population size of 30 and a generation number of 25. This value is used as a baseline for the convergence of the solution methods, and heuristically considers fitness values less than 1.07×10^7 as acceptable for the above input data.

Figure 1 illustrates typical convergence of fitness values of the three (3) local search methods and AOGOS. These methods usually converge within approximately 25 seconds under the hardware conditions. Parameter tuning of three (3) local search methods by increasing the search depth in the cases of hill climbing and tabu search or increasing the number of iterations in the case of simulated annealing did not result in significant improvement in the convergence values. It is to note that the convergence values of

$$1.08 \times 10^7,\ 1.075 \times 10^7 \text{ and } 1.077 \times 10^7$$

for the examples of hill climbing, tabu search and simulated annealing (illustrated in Figure 1) improve on the convergence value of 1.125×10^7 for AOGOS. The bar chart shown in Figure 2 compares the convergence of Hybrid HC with the GA for varying numbers of generations and a fixed population size of 30 (where the input data described above is used).

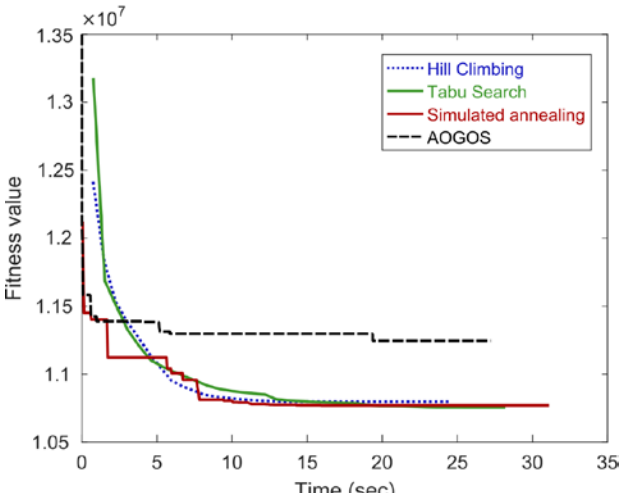

Figure 1. Comparison of local search methods with AOGOS

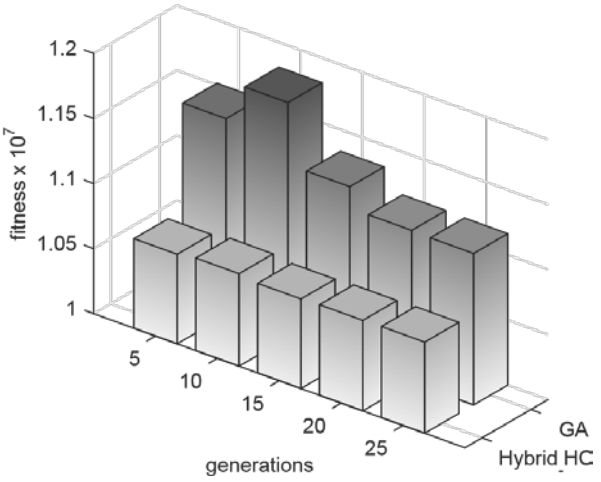

Figure 2. Comparison of fitness convergence values from Hybrid HC against the genetic algorithm (GA)

All convergence values of the GA for the given input data exceed 1.1×10^7 which is not acceptable when compared to the heuristic baseline of 1.07×10^7. It is to note that there is some marginal improvement in the GA convergence values with an increase in the number of generations. However, further increase in the number of

generations does not result in significant increase in the GA convergence values.

On the other hand, the performance of Hybrid HC is acceptable with the convergence values of all generation numbers being approximately equal to the baseline of 1.07×10^7. The cost of the improved convergence of Hybrid HC over the GA is execution time. The GA executes in 1.3 seconds for a generation number of five, while Hybrid HC requires 1285.4 seconds to complete for the same number of generations.

5.3. Adjustable Parameters and Execution Times

The execution time for each of the seven (7) solution methods will depend on the value of their associated variable parameters. These parameters include:

1) **search depth** in the case of the local search methods of hill climbing, tabu search and simulated annealing,
2) **initial temperature** and **cooling rate** in the case of simulated annealing ,
3) **population size** and **number of generations** in the case of the evolutionary methods of the genetic algorithm and the three (3) hybrids (discussed in Section 4.3).

The associated parameters of the local search component of a hybrid will also influence its execution time and performance. It is to note that, despite the previously discussed parameter influence, there is sufficient variance in the execution times of our seven (7) solution methods to allow a meaningful comparison to be made. The execution times stated in Table 2 correspond to the input data described in Section 5.2 and parameter values for which further modification does not result in significant improvement of convergence.

Table 2. Representative execution times in seconds for the GMS solution methods

Method	Execution Time
genetic algorithm	12.3
hill climbing	24.6
AOGOS	27.2
tabu search	28.2
simulated annealing	31.1
Hybrid SA	893
Hybrid TS	3625.4
Hybrid HC	3836.2

The representative execution times shown in Table 2 indicate that each of our local search methods generally executes in approximately 30 seconds. Recall that a hybrid method requires that a local search is performed for each element of each generation. These large number of subsidiary local searches for a hybrid method result in an execution time which is significantly higher than the local search times.

5.4. Parameter Independence of the Memetic Algorithm

As discussed in the Section 5.3 above, there are several parameters in the implementation of three (3) hybrid algorithms which may be adjusted as a possible means of improving the convergence of fitness values. It is to note that the performance of the hybrid solutions is good for relatively low population size and number of generations.

Further increase in these parameters does not significantly increase performance of the hybrid methods. An example of independence of convergence performance with change in parameter value may be seen in Figure 2. The optimal fitness for Hybrid HC does not improve with increase in the number of generations.

The optimal fitness values for Hybrid SA provide a second example of this relative parameter independence (see Table 3). These values illustrate that there is little change in convergence with increase in population and generation number after the thresholds of a population size of 30 and a generation number of 15 have been reached.

Table 3. Optimal fitness values $\times 10^7$ for Hybrid SA with respect to population size and number of generations

		Number of generations				
		5	**10**	**15**	**20**	**25**
Population	**10**	1.0703	1.0701	1.0723	1.0689	1.0691
	20	1.0717	1.0717	1.069	1.069	1.0689
	30	1.0731	1.0687	1.0686	1.0694	1.0685
	40	1.0686	1.0704	1.0697	1.0689	1.0696
	50	1.0731	1.0695	1.0689	1.0689	1.0689

6. Case Study

In this section, the GMS implementation is applied to obtain a maintenance schedule $\mathbf{x}^*$ that improves upon an actual schedule $\mathbf{x}$ utilised in 2013 (Sharma and Bahadoorsingh, 2004). Recall from Sections 3.2 and 3.3 that we may assign a fitness value to each schedule belonging to the solution space by the use of our objective function (Equation 11). Let F and F^* denote the respective fitness values of the schedules $\mathbf{x}$ and $\mathbf{x}^*$. If $F^* < F$ then (as Equation (11) is a minimisation problem), we regard the schedule $\mathbf{x}^*$ as an *improvement* of $\mathbf{x}$. This improvement may be interpreted as a reduction in the variability of the reserves, R_j (see Section 3.3).

In 2013, the Power Generation Company of Trinidad and Tobago (PowerGen) fulfilled the conditions of a Power Purchase Agreement to meet a constant forecast load demand D_j = 819 MW. The maintenance schedule $\mathbf{x}$ of the eighteen (18) PowerGen generators used in 2013 was recorded in (Sharma and Bahadoorsingh, 2004). As discussed above, we use the objective function given in Equation (11) to assign the fitness value of F = 282148 to this schedule $\mathbf{x}$.

In order to improve on this schedule, the requisite generator information as stated in Sharma and Bahadoorsingh (2004) forms the necessary input data for the GMS implementation (see Table 1 for an example of input data for a simple four generator case). Applying the Hybrid SA solution method to this input data gives an improved maintenance schedule $\mathbf{x}^*$ that is described in Table 4. The fitness value for this schedule is $F^* = 265496$ which is a 5.9% improvement on the fitness value $F = 282148$ of the actual PowerGen maintenance schedule **x** utilised in 2013.

Table 4. An improved generator maintenance schedule for PowerGen in the year 2013

Generator #	**Duration** (Week *i* to Week *j*)
1	30 to 33
2	21 to 28
3	44 to 52
4	1 to 5
5	6
6	37 to 43
7	8 to 20
8	44 to 46
9	28 to 33
10	48 to 52
11	47
12	34 to 36
13	21 to 23
14	2 to 4
15	none
16	24 to 27
17	6 to 7
18	52

7. Conclusion

The numerical results given in Section 5 indicate that the local search methods rapidly converge to acceptable solutions of the GMS problem. The hybrid methods converge to marginally better solutions than those obtained by the local searches at the cost of significantly longer execution times. This suggests the following methodology in the application of our solution methods – local searches should be initially applied to obtain estimates of the fitness values of optimal solutions. The hybrid methods would then be used to obtain improved solutions.

In addition, the performance of the genetic algorithm is generally unacceptable when compared to our local search and hybrid methods. A similar comparison is made in Dahal and Chakpitak (2007). It is possible that further tuning of crossover and mutation parameters may improve the performance of a GA solution of the GMS problem. These modifications would be considered in future work.

References:

Ahmad, A., and Kothari, D.P. (1998), "A review of recent advances in generator maintenance scheduling", *Electric Machines and Power Systems*, Vol.26, No.4, pp.373-387.

Dahal, K.P. and Chakpitak, N. (2007), "Generator maintenance scheduling in power systems using metaheuristic-based hybrid approaches", *Electric Power Systems Research*, Vol.77, No.7, pp.771-779.

Dawkins, R. (2006), *The Selfish Gene. 199*, Oxford University Press.

Egan, G.T, Dillon, T.S. and Morsztyn, K. (1976), "An experimental method of determination of optimal maintenance schedules in power systems using the branch-and-bound technique", *IEEE Transactions on Systems, Man, and Cybernetics*, No. 8. IEEE, pp.538-547.

Froger, A., Gendreau, M., Mendoza, J.E., Pinson, E. and Rousseau, L.M. (2015), "Maintenance scheduling in the electricity industry: A literature review", *European Journal of Operational Research*, Vol. 251, Issue 3, pp. 695-707.

Heylighen, F. and Chielens, K. (2009), "Evolution of culture, memetics", In: *Encyclopedia of Complexity and Systems Science*, Springer, p.3205-3220

Michalewicz, Z., and Fogel, D.B. (2013), *How to Solve It: Modern Heuristics*, Springer Science & Business Media.

Moscato, P., and Cotta, C. (2003), "A gentle introduction to memetic algorithms", In: *Handbook of Metaheuristics*, Springer, p.105-144.

Mukerji, R., Merrill, H.M., Erickson, B.W., Parker, J.H. and Friedman, R.E. (1991), "Power plant maintenance scheduling: Optimising economics and reliability", *IEEE Transactions on Power Systems*, Vol.6, No.2, pp.476-483.

Patton, A.D., and Ali, J. (1972), "Comparison of methods for generator maintenance scheduling", *IEEE PES Summer Meeting, Paper*, C 72 452-1, San Francisco, CA, July

Power Market Subcommittee (1979), "IEEE reliability test system", *IEEE Transactions on Power Apparatus and Systems*, No. 6, pp.2047-2054.

Rao, S.S. (2009), *Engineering Optimisation: Theory and Practice*. John Wiley & Sons.

Schlünz, E.B. (2011), *Decision Support for Generator Maintenance Scheduling in the Energy Sector*, PhD Thesis, Stellenbosch University.

Sharma, C. and Bahadoorsingh, S. (2004), "A MATLAB-based power generator maintenance scheduler", *Proceedings of the 2004 Power Systems Conference and Exposition*, IEEE PES, New York City, October, pp.1344-1348

Simon, D. (2013), *Evolutionary Optimisation Algorithms*, John Wiley & Sons.

Simon, D. (2015), *Evolutionary Optimisation Algorithms*, available at http://academic.csuohio.edu/simond/EvolutionaryOptimisation/.

Yellen, J., Al-Khamis, T.M., Vemuri, S., and Lemonidis, L. (1992), "A decomposition approach to unit maintenance scheduling", *IEEE Transactions on Power Systems*, Vol.7, No.2, pp.726-733.

Authors' Biographical Notes:

Neil Ramsamooj *is a Lecturer at the Office of the Dean at the Faculty of Engineering at The University of the West Indies. In 2005, he completed the Ph.D. degree in Mathematics at the University of Wisconsin-Madison. His current research interests include optimisation techniques and applications of econometric methods.*

Laura Ramdath *is an Electrical Engineer at the Ministry of Works and Transport. She holds a BSc. in Electrical Engineering (Hons.) from The University of the West Indies, St. Augustine. Her*

employment experience includes an engineering position of the Ministry of Public Administration and an internship at the Trinidad and Tobago Electricity Commissions.

Sanjay Bahadoorsingh *is a Senior Lecturer at the Department of Electrical and Computer Engineering at The University of the West Indies. In 2009, he completed the Ph.D. degree at The University of Manchester. His areas of research are power systems, asset management and dielectric aging.*

Chandrabhan Sharma *is the Professor of Energy Systems with the Faculty of Engineering, The University of West Indies. He is the Head of the Centre for Energy Studies and the Leader of the Energy Systems Group. He has served as a member of the Board of Directors of the local Electric Utility for over 10 years and as a member of the Board of Directors of the largest bank in the country. Prior to joining the Academic staff at the university, he was attached to the petrochemical industry in Trinidad. His interests are in the area of power system operations and control.*

■