

Unit Commitment for BC-Hydro's MICA Dam Generating Plant using a Genetic Algorithm Approach

Shabnam Khatami*, Jeff Breadner** and John A. Meech**

University of British Columbia,

* Department of Mechanical Engineering and

** Department of Mining and Mineral Process Engineering,

Vancouver, B.C., V6T 1Z4

Email: shabnam@mech.ubc.ca , breadner@telus.net , jam@mining.ubc.ca

Abstract

In this paper we describe a Genetic Algorithm to derive the best unit commitment for the MICA Dam Generating Plant in the BC Hydro Electric Power System for a 24-hour schedule. This plant consists of 4 turbines (units) and the optimal "combo" for each hour must meet the forecast load for that particular hour while using the least amount of water subject to certain constraints. The combination of all 24-hour combos must also attempt to minimize the number of on/off switches over the full 24 hours.

Introduction

One of the operating decisions required at a hydroelectric power plant having more than one turbine (unit) is which of the units should be operating. The question of which unit to operate is termed the "unit commitment" (UC) problem. For a plant with multiple units, it is generally possible to generate a specific plant load with the reservoir at a particular forebay elevation using different unit commitments and individual unit loadings. So the issue arises as to which of these solutions is best.

BC-Hydro use a program called SPUC to calculate the optimal loading for each unit for a given availability of units. The name SPUC is an acronym for "Static Plant Unit Commitment"; the word "static" referring to the fact that the results produced by SPUC are valid for a particular instant. So if a "snapshot" can be taken of the plant at a particular point in time, SPUC calculates the most efficient unit commitment and load (UC&L). SPUC has been run for the MICA plant in the BC Hydro system (see Fig. 1), and the results are stored in the SPUC database. In SPUC, a particular unit commitment is known as a "combo". The combo is the decimal equivalent of a binary number representing the unit commitment in which a value of one indicates a unit is committed and a zero represents a unit that is off. In calculating the optimal sequence of unit commitments for the MICA plant, we made use of the stored SPUC output to find the amount of water used for a particular combo given the unit availability, plant load and forebay level by using a Genetic Algorithm approach. Table 1 presents a selected number of combos from the SPUC database from a total of more than 25,000 possibilities.

Genetic Algorithms (GAs) are adaptive evolutionary methods that can be used to solve search and optimization problems such as this one. They are based on the genetic processes of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and "survival of the fittest". By mimicking this process, A GA is able to evolve solutions to real-world problems. They work with a population of individuals, each representing a possible solution. Individual is assigned a "fitness score" according to how good the solution solves the problem. Individual solutions with high fitness values are given an opportunity to reproduce, by breeding with other "fit" solutions. This produces new individuals as offspring, which share some features taken from each parent. The least fit members of the population are less likely to be selected for reproduction, and so, they die off. Selecting the best individuals from the current generation and mating them to produce a set of offspring produces a new population of possible solutions. If the GA is well-designed, the population will converge quickly to the optimal solution to the problem.

Crossover proceeds in two steps after reproduction. First, members of the chosen strings in the mating pool are mated at random. Second, each pair of strings undergoes crossover as follows: an integer position k along the string is selected at random between 1 and the string length less one $[1, l-1]$. Swapping all characters between position $k+1$ and l inclusively creates two new strings. For example, if there are two strings: 1011011 and 110100 chosen for reproduction and the crossover position is 3, then the offspring created will be 101100 and 110011.

Mutation generally plays a secondary role in the operation of genetic algorithms. Mutation is needed because, unlike the reproduction and crossover operations that search and recombine existent notions, they occasionally may lose some potentially useful genetic material. The mutation operation involves periodically selecting one individual at random, selecting one position on the chromosome string and transposing it from 0 to 1 or vice-versa. Mutation restores diversity but does not provide a logical approach to optimization. Its use is particularly important with situations in which a local minima (or maxima) has trapped the algorithm and a new population member is required to trigger the crossover operator on to a better result.

Description of the Application of GA for the MICA Plant

GA's have received considerable attention regarding their potential to deal with constrained problems. In calculating the optimal sequence of unit commitment for the MICA plant, we are trying to apply GA to find the best solution which uses the minimum amount of water and has the minimum number of unit switches over a 24-hour period given unit availability, plant load and forebay level as inputs for each hour subject to the following constraints:

1. A minimum of 2 units must always be on line.
2. Units G1 and G2 are identical.
3. G4 is the last unit on or off to minimize operation of its unit breaker.

In the above constraints, G1, G2, G3 and G4 indicate unit 1 to 4 for MICA plant. G1 and G2 are similar units with a maximum capacity of 435 MW. The maximum capacity for unit 3 is 465MW while G4 is limited to 400 MW due to vibration problems.

Using GA, we need to define each possible solution for 24 hours as a string or "chromosome". As we have a combo of 4 binary numbers indicating the status of each unit for each hour of the day, we have to consider a rather large chromosome size of 96 genes. This makes convergence very slow unless appropriate population size, crossover choice, and mutation rate are chosen. To achieve a faster and easier approach, we reduced the chromosome size to 16 to allow the algorithm to model a 4-hour period. For each successive run, we can use the state of the last hour of the 4-hour segment in our previous run to solve for the next 4-hour period. In this way, the problem is being solved in the manner of dynamic programming. The minimization of switches calculation requires an initial state in order to determine the switch value for hour 1. For the first run, we ask the user to enter the state of the units at the end of the previous 24-hour period as an input variable.

To allow the GA to satisfy all the constraints, we have placed them into data tables to allow the calculation of a fitness score for each possible solution. These look-up tables serve to establish the functional relationship to be used under the current circumstances in each part of the overall fitness function.

To find the minimum use of water, which is our first goal, GA uses the SPUC table as a database in EXCEL to find the water used in each hour. Considering this first constraint: we make all combos that have less than two units on-line to be infeasible. We can achieve this state by simply changing the value of the water use of combos 1, 2, 4, and 8 for all forebay and unit load inputs to a very high value. This high number has been set to four times the maximum water use in the table to generate a zero (0)

fitness score for all those strings which have at least one of those infeasible combos. We can also satisfy this constraint by modifying the table containing the number of switches as shown in Table 2. The assignment of 16 in this table for certain sequences of combos, is simply a way to endure that that particular combination will never be successfully retained in the system and allowed to reproduce.

Table 2. Number of switches and identification of feasible combos.

		combo k														
		1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111
Combo		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
10	2	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
11	3	16	16	0	16	2	2	1	16	16	16	16	16	16	16	16
100	4	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
101	5	16	16	2	16	0	2	1	16	16	16	16	16	16	16	16
110	6	16	16	2	16	2	0	1	16	16	16	16	16	16	16	16
111	7	16	16	1	16	1	1	0	16	16	16	16	16	16	16	16
1000	8	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
combo k+1	1001	9	16	16	16	16	16	16	16	0	2	1	1	1	3	2
	1010	10	16	16	16	16	16	16	16	2	0	1	2	3	1	2
	1011	11	16	16	16	16	16	16	16	1	1	0	3	2	2	1
	1100	12	16	16	16	16	16	16	16	1	2	3	0	1	1	2
	1101	13	16	16	16	16	16	16	16	1	3	2	1	0	2	1
	1110	14	16	16	16	16	16	16	16	3	1	2	1	2	0	1
	1111	15	16	16	16	16	16	16	16	2	2	1	2	1	1	0

Feasible sequences of combos for each two successive hours are shown in white

Applying the given constraints makes the following changes:

- 1- Units G1 & G2 are identical. This reduces the number of switches between combos as shown
- 2- There must always be a minimum of 2 units on line which eliminates combos 1, 2, 4 & 8.
- 3- G4 is the last unit on/off to minimize operation of unit breaker

Mapping the Objective Functions To A Fitness Function

A fitness function must be a non-negative figure of merit. As a result, it is often necessary to map the underlying natural objective function to a fitness function through one or more mappings. In normal operation research work, to transform a minimization problem to a maximization problem, we simply multiply the function by minus one. In GA, this operation alone is insufficient because the measure thus obtained is not guaranteed to be non-negative in all instances. In our case, the two objectives must be considered together and mapped into a common fitness function. In this project, the objective with the highest priority is to minimize the amount of water used. If we consider Q_i as the water use for each hour and find Q_{max} from the SPUC table (which is the maximum amount of water used), we can define the first element of the fitness function from the use of water for each chromosome in the population as follows:

$$Q_t = 4 Q_{\max} - \sum_{i=1}^4 Q_i \quad 1.$$

where $4 Q_{\max}$ indicates the worst string with the lowest fitness score.

Since we are considering two different fitness function elements, so they must be normalized. For the above equation, this is accomplished by dividing by $4Q_{\max}$ as follows:

$$Q_t / 4 Q_{\max} = 1 - \sum_{i=1}^4 (Q_i / 4Q_{\max}) \quad 2.$$

which creates a number between 0 and 1. Higher values for $Q_t / 4 Q_{\max}$ indicates lower amounts of water used for a particular chromosome during a 4-hour period.

Note that we have assigned $4 Q_{\max}$ to all infeasible combos in the SPUC table to ensure that the presence of even one infeasible combo during a 4-hour period will give a fitness value higher than that of the worst feasible string which has the maximum use of water. So if we have an infeasible combo, this will give a negative number for $Q_t / 4 Q_{\max}$. To prevent this, we define the following function as the first fitness score element based on water use for the whole string of combos for 4 hours:

$$Q = \text{Max} (0, Q_t / 4Q_{\max}) \quad 3.$$

which gives a fitness score of 0 to all infeasible combos. According to the SPUC table, $Q_{\max} = 1902$, hence,

$$Q = \text{Max} (0, Q_t / 7608) = \text{Max}[0, (1 - \sum_{i=1}^4 (Q_i / 7608))] \quad 4.$$

Our second goal is to minimize the number of switches as we move from one hour to the next. Finding the second fitness score uses Table 2 for the number of switches for each pair of successive combos:

$$N_t = 4N_{\max} - \sum_{i=1}^4 N_i \quad 5.$$

in which N_t is the total number of switches over 4 hours and N_{\max} is the maximum number of switches shown in the switch table.

After normalizing we have:

$$N_t / 4N_{\max} = 1 - \sum_{i=1}^4 (N_i / 4N_{\max}) \quad 6.$$

As indicated for the first fitness function element, a set with an infeasible combo gives a negative number. So we need to modify this score as well as:

$$N = \text{Max}(0, N_t / 4N_{\max}) \quad 7.$$

Obviously, the maximum number of switches between 4 units is 4 per hour, so we define this element of the fitness score as:

$$N = \text{Max}(0, N_t / 16) = \text{Max}[0, 1 - \sum_{i=1}^4 (N_i / 16)] \quad 8.$$

The fitness function F will be the sum of these two elements multiplied by a weight based on the priority of each objective:

$$F = W_1Q + W_2N \quad 9.$$

Assuming $W_1=0.6$ for the goal of minimizing water use and $W_2=0.4$ for minimizing the number of switches, we obtain the following formula for the fitness function.

$$F = 0.6\text{Max} [0, 1 - \sum_{i=1}^4 (Q_i/7608)] + 0.4\text{Max}[0, 1 - \sum_{i=1}^4 (N_i/16)] \quad 10.$$

After calculating a fitness value for each member of the population, we can commence reproduction. A common selection approach assigns a probability of selection, P_i , to each individual based upon its fitness such that the better individuals have an increased chance of selection.

$$P_i = F_i / \sum_{i=1}^N F_i \quad (\text{where } N = \text{population size}) \quad 11.$$

For crossover, we allow the point of crossover to vary randomly between 4, 8 or 12 for a 16-gene chromosome. On each iteration, we generate a random number between 0 and 1 that will switch between these three crossover points. If the number is below 0.33, then 4 is chosen, if the number is between 0.34 and 0.66 then 8 is chosen, if the number is above 0.66 then 12 is chosen.

An EXCEL worksheet has been set up to implement these procedures automatically. The spreadsheet can begin to generate new populations of solutions and find the best solution for each generation. The GA simulation terminates when the fitness of the best or average population remains the same for more than 2 successive generations.

The input/output worksheet is shown in Fig. 2. Note that the system also includes information about the previous 2-hour state of the plant in order to properly assess the number of switches.

Fig. 3 presents the form used to provide interim results. The user has the ability to lock certain combos for another optimization run. In addition, the form contains a drop down menu of alternative feasible solutions for each hour that can be viewed and assessed by the user.

Fig. 4 presents the increase in average fitness value as a function of generation number. Note that the starting fitness is a random result of the initially selected 100 solutions. The range of values in the initial population of solutions ranged between 22 and 80 percent. Note the substantial improvement in the next generation after the mutation operator was applied to generation 5 providing a jump to a new optimum. Conceivably better solutions are possible with increased numbers of generations or with a changing probability of mutation as the number of generations increases. Diversity is a useful property to restore to an evolving system in order to allow the system to find a new minima (or maxima).

Conclusion

Using the GA approach, we can perform a stochastic global search to find the best combination of unit status for a 24-hour schedule of the MICA plant having as input data – the unit availability, the forebay level and the plant loading for each hour that also satisfies the given constraints. By simplifying the program to six dynamic 4-hour segments, we were able to obtain a faster convergence to the problem. The robustness and power of GA to find the solution compared to other traditional optimization procedures has been apparent.

For this application, the GA fitness function needs further refinement. Consideration of unique conditions for each unit needs to be built into the fitness function via a third constraint component. As well, the GA Engine can be completely rewritten so that it does not actually manipulate gene strings as in a classical GA. In Excel, it would be more efficient to manipulate arrays of combo numbers internally. It is surmised that with this approach in which the engine handles arrays of numbers instead of gene strings, longer chromosomes are feasible (perhaps, as large as the entire 24 hours period).

				Past Unit Commitment:						
Run Optimizer				Unit Loading						
				Hour	Combo	G1	G2	G3	G4	
				-2	11	Off	Off	Off	296	
				-1	11	Off	Off	Off	306	
				0	11	Off	Off	Off	316	(Current Status)
Clear Outputs				Output						
Input				Unit Loading						
			Available	Hour	Combo	G1	G2	G3	G4	
Hour	ForeBay	GPInt	Combos	1	8	Off	Off	Off	280	
1	711	280	15	2	4	Off	Off	290	Off	
2	713	290	15	3	4	Off	Off	300	Off	
3	715	300	15	4	4	Off	Off	310	Off	
4	717	310	15	5	4	Off	Off	320	Off	
5	719	320	15	6	4	Off	Off	330	Off	
6	721	330	15	7	4	Off	Off	340	Off	
7	723	340	15	8	4	Off	Off	350	Off	
8	725	350	15	9	4	Off	Off	360	Off	
9	727	360	15	10	4	Off	Off	370	Off	
10	729	370	15	11	4	Off	Off	380	Off	
11	731	380	15	12	4	Off	Off	390	Off	
12	733	390	15	13	4	Off	Off	400	Off	
13	735	400	15	14	4	Off	Off	410	Off	
14	737	410	15	15	4	Off	Off	420	Off	
15	739	420	15	16	4	Off	Off	430	Off	
16	741	430	15	17	4	Off	Off	440	Off	
17	743	440	15	18	4	Off	Off	450	Off	
18	745	450	15	19	4	Off	Off	460	Off	
19	747	460	15	20	12	Off	Off	340	130	
20	749	470	15	21	12	Off	Off	350	130	
21	751	480	15	22	12	Off	Off	360	130	
22	753	490	15	23	12	Off	Off	370	130	
23	755	500	15	24	12	Off	Off	380	130	
24	757	510	15							

Fig. 2. EXCEL Spreadsheet used to input SPUC data and obtain output schedule.

UBC AI EULR Replacement [X]

Hour	Combo, Q_Turb, Efficiency, Power Generated	G1	G2	G3	G4	Lock
1	3, 225.92, 88.2, 280	Off	Off	Off	280	<input type="checkbox"/>
2	4, 230.4, 88.3, 290	Off	Off	290	Off	<input type="checkbox"/>
3	4, 234.9, 88.4, 300	Off	Off	300	Off	<input type="checkbox"/>
4	4, 239.33, 88.4, 310	Off	Off	310	Off	<input type="checkbox"/>
5	4, 243.61, 88.5, 320	Off	Off	320	Off	<input type="checkbox"/>
6	4, 247.81, 88.5, 330	Off	Off	330	Off	<input type="checkbox"/>
7	4, 252.02, 88.6, 340	Off	Off	340	Off	<input type="checkbox"/>
8	4, 256.14, 88.5, 350	Off	Off	350	Off	<input type="checkbox"/>
9	4, 260.3, 88.5, 360	Off	Off	360	Off	<input type="checkbox"/>
10	4, 264.37, 88.4, 370	Off	Off	370	Off	<input type="checkbox"/>
11	4, 268.3, 88.4, 380	Off	Off	380	Off	<input type="checkbox"/>
12	4, 272.24, 88.3, 390	Off	Off	390	Off	<input type="checkbox"/>
13	4, 276.37, 88.2, 400	Off	Off	400	Off	<input type="checkbox"/>
14	4, 280.39, 88, 410	Off	Off	410	Off	<input type="checkbox"/>
15	4, 284.69, 87.8, 420	Off	Off	420	Off	<input type="checkbox"/>
16	4, 288.83, 87.6, 430	Off	Off	430	Off	<input type="checkbox"/>
17	4, 292.76, 87.4, 440	Off	Off	440	Off	<input type="checkbox"/>
18	4, 296.83, 87.2, 450	Off	Off	450	Off	<input type="checkbox"/>
19	4, 300.58, 87, 460	Off	Off	460	Off	<input type="checkbox"/>
20	12, 315.77, 83.7, 470	Off	Off	340	130	<input type="checkbox"/>
21	12, 319.15, 83.7, 480	Off	Off	350	130	<input type="checkbox"/>
22	12, 322.61, 83.6, 490	Off	Off	360	130	<input type="checkbox"/>
23	12, 326.16, 83.4, 500	Off	Off	370	130	<input type="checkbox"/>
24	12, 330.13, 83.2, 510	Off	Off	380	130	<input type="checkbox"/>

Number of Switches: 0 0 1 2

Run AI Optimization OK Cancel

Fig. 3. EXCEL Form used to produce the optimized daily schedule. The Lock box allows the user to select a particular combo or set of combos and rerun the Optimizer.

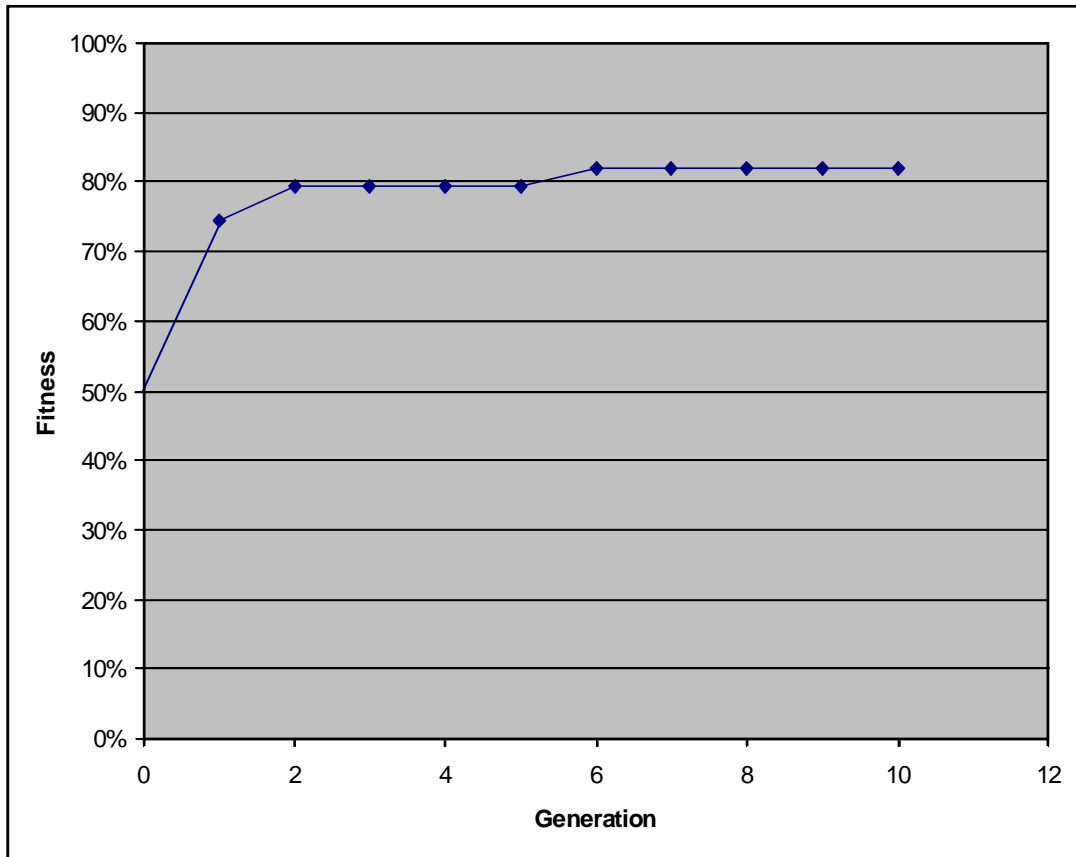


Fig. 4. Average fitness value as a function of the generation number.

Acknowledgement

The authors would like to express their appreciation to Dr. Thomas K. Siu, Head of Resource Management for BC Hydro who set up this problem as a Class Project for a course in Industrial Expert Systems at the University of British Columbia and who contributed his expertise to the development of the system. Dr. Ziad Shawwash from Civil Engineering at UBC also contributed considerably to the research performed.

References

1. D.L. Goldberg, 1989. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley.
2. T. Siu, G. Nash, Z. Shawwash, 2000. Expert Systems for Dynamic Unit Commitment and Loading (DUCL) Program, B.C. Hydro.
3. M. Gen, R. Cheng, 1997. Genetic Algorithms and Engineering Design, Wiley-Interscience.

