IASET: Journal of Mechanical Engineering (IASET: JME)
SSN(P): Applied; ISSN(E): Applied
Vol. 1, Issue 1, Jan – Jun 2016, 43-52

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

OPTIMIZATION AND APPLICATION OF SOLAR ABSORPTION CHILLER BY USING GA

REZA BAKHTIARI[1] & ARMEN ADAMIAN[2]

[1]Master of Science, Department of Mechanical Engineering, College of Engineering, Azad University of Tehran Center, Branch Azad University, Tehran Iran

[2]Assistant Professor, Department of Mechanical Engineering, College of Engineering, Azad University of Tehran Center, Branch Azad University, Tehran Iran

ABSTRACT

This study employs genetic algorithm (GA) to solve optimal chiller loading (OCL) problem. GA overcomes the flaw that with the Lagrangian method the system may not converge at low demand. This study uses the part load ratios (PLR) of chiller units to binary code chromosomes, and executes reproduction, crossover and mutation operation. After analysis and comparison of the two cases studies, we are confident to say that this method not only solves the problem of convergence, but also produces results with high accuracy within a rapid timeframe. It can be perfectly applied to the operation of air-conditioning systems.

KEYWORDS: Optimization and Application of Solar Absorption

INTRODUCTION

The designers of air-conditioning systems often develop multiple-chiller systems because they provide operational flexibility, standby capacity and less disruption maintenance. Such a system has a reduced starting in-rush current, reduced power cost under partial-load conditions and a set of chillers that can be operated at the best efficiency [3]. Figure 1 depicts the structure of a decoupled chilled water system (called a decoupled system), which includes multiple chillers [3]. A decoupled system offers constant flow on the primary side (the chiller side), which prevents excessively low temperatures from impairing the evaporator and prevents the chillers from being frequently shut down. On the secondary side (the load side), the two-way valves can regulate the chilled water that flows into the cooling coils in accordance with the load variation. Therefore, the decoupled system yields stable control and is applied in air-conditioning systems.

The cooling load of an air-conditioned room can be removed to the chiller unit using the chilled water system; it is then emitted to the atmosphere using the cooling water system. Therefore, the cooling load (in refrigeration tons) of a chiller can be calculated from [4], where f is the flow rate of chilled water (kg s−1), m the specific heat of chilled water (J kg−1 K−1), TCHr = the return temperature of chilled water (K), and TCHs = the supply temperature of chilled water (K).

The water flow in a chiller is constant, so its cooling load can be determined by measuring the supplied temperature and the return temperature of the chilled water. The input power (kW) can be measured using a power meter and then the kW–PLR curve can be obtained by regression. The variation in the demand can also be determined by measuring the water flow and the temperature in the main pipe on the load side.

OCL BY LAGRANGIAN METHOD

$$Q = \frac{fm(T_{\mathrm{CHr}} - T_{\mathrm{CHs}})}{3517}$$

In a system with all-electric cooling, the best performance occurs when the kW of a chiller is minimized while the load demand is satisfied. In general, the cooling load is expressed as a PLR which is the chiller cooling load divided by its design capacity. The partial-load energy consumption of centrifugal chillers are higher at low loads due to motor losses, but the increased input powers at high load are due to thermal heat exchange inefficiencies [5]. That is, the kW of a centrifugal chiller is a convex function of its PLR for a given wet-bulb temperature:

$$\mathrm{kW}_i = a_i + b_i\mathrm{PLR}_i + c_i\mathrm{PLR}_i^2 + d_i\mathrm{PLR}_i^3$$

Where ai, bi, ci, di are coefficients of kW–PLR curve of it chiller. The OCL problem is to find a set of chiller output which does not violate the operating limits while minimizing the objective function:

$$J = \sum_{i=1}^{1} \mathrm{kW}_i$$

Simultaneously, the balance equation must be satisfied:

$$\sum_{i=1}^{1} \mathrm{PLR}_i \times \overline{RT}_i = \mathrm{CL}$$

Where RTi = capacity of ith chiller, CL = system cooling load. The Lagrangian method [6] is adopted to find the optimal solution of the convex function. For a system's cooling load CL, the Lagrangian multiplier λ can be evaluated by:

$$\lambda = \frac{b_i + 2c_i\mathrm{PLR}_i + 3d_i\mathrm{PLR}_i^2}{\overline{RT}_i}$$

For a system cooling load CL, the Lagrangian multiplier λ and PLRi can be evaluated from Eqs. [7]. The kW of each chiller is finally calculated by above Eq But, the lambda-iteration method will not converge at low demand. This short coming can be overcome by GA method

GENETIC ALGORITHMS

GA was proposed by John Holland in 1975 [8]. It is a "search algorithm" based on Darwin's theories of natural selection and natural genetics. GA digitalizes an organism's genetic information or chromosome structure by coding the information as combinations of binary digits. The result is strings of two numbers: 0s and 1s. Through the process of evolution, GA promises to produce better offspring by randomly choosing the most favorable digits and fragments from the parents. The process continues until the most fitted chromosomes are obtained.

Preparation

Before GA's operators have been used to optimize a problem, we must consider several things: the coding and decoding of the strings, the size of the strings, and size of the population, and decision in choosing a fitness function. These preparations are explained as follows.

The process of coding and decoding when using GA to optimize the genetic information, the method of coding has to be decided first. One widely regarded method is through binary coding. During the process of binary coding, we need to first understand the range of variables (x min ≤ x ≤ x max, where x min and x max represent the minimum and maximum values of the variable "x", respectively) and to what extend do we need the accuracy to be (i.e. how many digits of decimal places, d, do we need).Thus, the variance Xr of variable 'x' is:

$$x_r = (x_{max} - x_{min}) \times 10^d$$

And Xr is an integer between 2n and 2n−1

$$2^{n-1} \leq x_r \leq 2^n$$

From above Eq we can decide that the binary coded string for variable 'x' will have 'n' number of digits. The higher the number of "n", the higher the accuracy of variable 'x' will be. After the strings have evolved under GA, we must determine how well the strings adapt to the system. To do this we need to decode the strings into actual variable value. The process of decoding can be understood as the reverse calculation of coding. For example, to convert binary digit strings into decimal (10 digit) integers,

$$(\langle b_{n-1} \cdots b_0 \rangle)_2 = \left(\sum_{i=0}^{n-1} b_i \times 2^i \right)_{10} = x^S$$

Where "n" is the number of binary digit. Then convert xS to the actual variable "x"

$$x = x_{min} + x^S \frac{x_{max} - x_{min}}{2^n - 1}$$

The longer the binary code, the higher the accuracy will be. Long binary code increases the time to code and decode, and waste computer memory resource; short binary code, however, may produce less desirable result.

Placing Fitness Function

In GA evolution, the most important valuation indicator is the fitness function. Fitness function is designed to indicate how every string adapts each other. The most difficult factor in designing the fitness function is how we can put penalty function into objective function to develop the whole fitness function,

Fitness function = objective function + penalty function

To evolve the optimized solution set, the design of fitness function, many constraints have to be considered. Also, fitness function should eliminate unfavorable chromosomes rapidly and produce new ones in order to speed up convergence.

OPERATORS

The operators in GA are the reproduction, crossover, and mutation operators. Based on the calculations of these operators, we can change or exchange chromosomal information of the old generation in order to produce more favorable off springs. The three operators are explained in details in the following.

Reproduction Operator

The number of chromosomes that will be reproduced depends on the value of the fitness function. More adapted chromosomes will produce a higher proportion of off springs, while less adapted chromosomes are eliminated. The most widely used reproduction method is the roulette wheel method. The size of each slot in the roulette wheel corresponds to the chromosomes' adaptive value, in other words, the larger the fitness function value, the larger the area size on the roulette wheel, and the higher the chance of the chromosome being reproduced. The probability of the *i*th chromosome being reproduced is where the numerator is the value of the ith chromosome's fitness function. And the denominator is the sum of the value of the fitness function of all the chromosomes in the population.

Crossover Operator

Strings in the crossover pool are selected according to the pre-determined crossover rate, and then two different genes are randomly selected from the strings. By exchanging gene digits, the original chromosomes produce two new chromosomes. The new chromosomes will be replicated in large quantity back into the crossover pool in the next replication process. When done repetitively, this method will generate more favorable digit information. There are three crossover methods: one-point, two-point and uniform crossover. Because the method used to reproduce will affect the result of GA reproduction, smaller population should use uniform crossover to obtain a better result; and a larger population should utilize two-point crossover. This study deals with two-point crossover method to promote execution results. In a two-point crossover, we randomly select two crossover points from two parent chromosomes, and then exchange all the digits between the two corresponding points, as illustrated in Figure 2.

Mutation Operator

Mutation operator will generate new chromosomes. Although reproduction and crossover can effectively search from the population, they cannot produce enough new chromosomes, and would result in redundancy between the populations. Mutation can eliminate this problem and prevent evolution process from premature convergence that results in local optimum. In other words, mutation generates new search direction in the whole population instead of in just part of the population.

APPLYING GENETIC ALGORITHMS TO OCL PROBLEM

The first step in solving OCL is the process of encoding. Encoding is the process to code variables by binary system, and then link the binary codes into strings (i.e. chromosomes).The variables that need to be processed are the PLR of chiller unit and the number of units running in parallel. After the variables are encoded into chromosomes, the information built into the chromosomes is the total PLRs of the units running in parallel. For example, as in Figure 4, three chiller units have been connected into a system. Each unit uses a 10 digit binary coding to represent its PLR. Three coding from the three units will form a string (i.e. chromosome).Then through initialization, chromosomes become a population. The number of chromosomes in a population is referred to as the population size.GA takes these binary chromosomes through reproduction, crossover and mutation operators to exchange or relay digital information on the chromosomes. To determine if every chromosome evolves towards convergence, after exchanging or relaying information every chromosome is decoded to obtain the actual PLR of the unit. And then the PLR and constraints are substituted into the objective function to calculate the corresponding result.

The remaining chromosomes will go through the same decoding process. The PLRs for units 2 and 3 are 0.7591 and0.6264, respectively. After every chromosome has been decoded to chiller unit PLR, use the objective function to calculate the corresponding result of the chromosomes. To solve OCL in a HVAC system, minimizing power consumption is always desirable. Eq. (3) can be modified into the following equation,

$$\mathrm{PLR}_1 = 0.3 + \left(730 \times \frac{1-0.3}{1023}\right) = 0.7995$$

Because the total chiller output has to fulfill the conditions of total system capacity in order to minimize power consumption, Eq. (4) can be modified into the following equation,

$$\mathrm{OBJ} = \mathrm{Min}\sum_{i=1}^{I} \mathrm{kW}_i(\mathrm{PLR}_i)$$

Where, ER represents discrepancy. When the result for Eq. (16) is zero, then capacity requirement has been fulfilled. The absolute value of Eq. (16) has a close relationship with the evolution process. There cannot be any negative value in the absolute item, in other words, the total chiller unit capacity output has to be greater or equal to the system capacity, and otherwise evolution does not fulfill the capacity requirement. After defining objective function and penalty function, fitness function can be designed. Suppose a chromosome has a minimum objective function for its total power consumption, and that penalty function is equal to zero, then the optimum fitness function for that chromosome can be achieved. The degree of how well a chromosome adapts is known by comparing other chromosomes in the same population. Therefore, by normalizing the discrepancy between objective function and penalty function, we can find how well each chromosome adapts. The process of normalization can be done as follows,

$$\%\mathrm{OB} = \frac{\mathrm{OB}_S - \mathrm{OB}_{min}}{\mathrm{OB}_{max} - \mathrm{OB}_{min}}$$

And

$$\%\mathrm{ER} = \frac{\mathrm{ER}_S - \mathrm{ER}_{min}}{\mathrm{ER}_{max} - \mathrm{ER}_{min}}$$

where %OB is the percentage value of objective function after the string has been normalized, OBS the value of the objective function for that string, OB min the smallest value of objective function in that population, OB max the largest value of objective function in that population, %ER the discrepancy percentage value of the of normalized string, ERS the discrepancy calculated of the constraints for the string, ER min the smallest discrepancy of the constraints in the population, ER max the largest discrepancy of the constraints in the population. After normalization, because minimized values of %OB and %ER are desirable, the value of the fitness function is also best minimized,

$$\text{Min Fit} = \text{Min}[\%\text{OB} + \%\text{ER}]$$

The quality of chromosomes is judged by values of their fitness functions. In other words, if the smaller the Min Fit from Eq. (19) gets, the lower the total system power consumption is. Eq. (19), however, aims for minimization, which stands against the roulette wheels selection process. In the roulette wheel method, the larger the value of fitness functions, the larger the slot area the chromosome gets, and therefore the more likely the chromosomes are to be replicated to the offspring. Eq. (19), then, should be rearranged in order to obtain the largest fitness function, as follows [9],

$$\text{Max Fit} = \text{Max}\{sf_1[(1 - \%\text{OB})^{\text{SP}_1}] + sf_2[(1 - \%\text{ER})^{\text{SP}_2}]\}$$

Where sfi is the scaling factor for emphasizing objective function or penalty function, and spi the scaling power factor for emphasizing the quality of a particular string compared with other strings during reproduction, this study uses roulette wheel method, where probability of a chromosome being reproduced is directly proportional to the value of its fitness function. The larger the value of its fitness function, the higher the probability of being selected. The roulette wheel method that we use can be explained as follows [10]:

Step 1 Calculate the value of fitness function Fit (ci) for each chromosome ci (i = 1. . . S) In the population.

Step 2 Calculate the sum of the value of fitness function for all the strings in the population.

$$F = \sum_{i=1}^{S} \text{Fit}(c_i)$$

Step 3 Use Eq. (13) to calculate the probability Pi of being selected for each string. Ci (i = 1. . . S).

Step 4 Calculate the accumulated probability qi = I j=1Pjfor each string ci(i = 1, . . . , S).

Step 5 randomly produce the number r of [0 . . . 1].

Table 1: Chillers Data

System	Chiller	a_i	b_i	c_i	d_i	Capacity(RT)
Case 1	CH-1	100.95	818.61	−973.43	788.55	800
	CH-2	66.598	606.34	−380.58	275.95	800
	CH-3	130.09	304.50	14.377	99.80	800

View in the table above and the nominal capacity of the chiller kW-PLR curve is given. In this simulation, the parameters of the PLR chillers should be chosen so that in addition to supply the required cooling load, low power consumption is required. To simulate, with ten bits of each parameter will display. Therefore, each chromosome contains genes will be thirty. To decode the binary PLR following formula is used.

$$PLR = 2^{-(n+1)} + \sum_{i=1}^{n} B_i \times \left(2^{-i}\right)$$

RESUTS

N = number of bits selected in the above equation for each parameter. The proposed method is based on the genetic algorithm was applied before the optimal solution according to the required capacity of the system were obtained. The table below shows the percentage of time required with respect to the overall capacity chillers listed, and then PLR has been optimized for each chiller. Below is listed the required cooling load.

And finally cooling load supplied through a chiller system is given. As seen once funded by chillers, cooling time is almost required to endure.

Table 2: Measured Loads

Load	PLR_1	PLR_2	PLR_3	U_{needed}	$U_{chillers}$
20%	0.0181	0.145	0.437	480	**480.0781**
30%	0.0562	0.2056	0.6361	720	**719.1406**
40%	0.1587	0.3188	0.7222	960	**959.7656**
50%	0.2661	0.3735	0.8599	1200	**1199.6**
60%	0.3813	0.4829	0.9351	1440	**1439.5**
70%	0.4790	0.6216	0.9995	1680	**1680.1**
80%	0.4976	0.9009	0.9976	1920	**1916.8**

In this simulation 100 times the number of chromosomes in each generation, and mutation rate is 5% and the number of generations have been 100.The following charts will be required to set the required cooling chillers in terms of number of generations in genetic algorithms is given. As seen by the desire to provide a minimum.

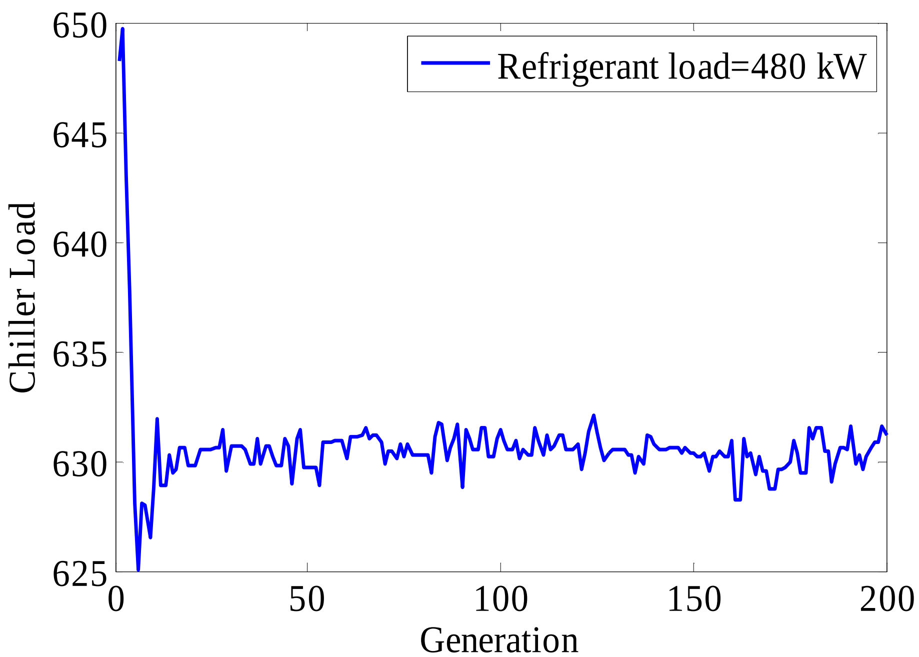

Figure 1: Chart of Production and Consumption 480kw Chiller Load

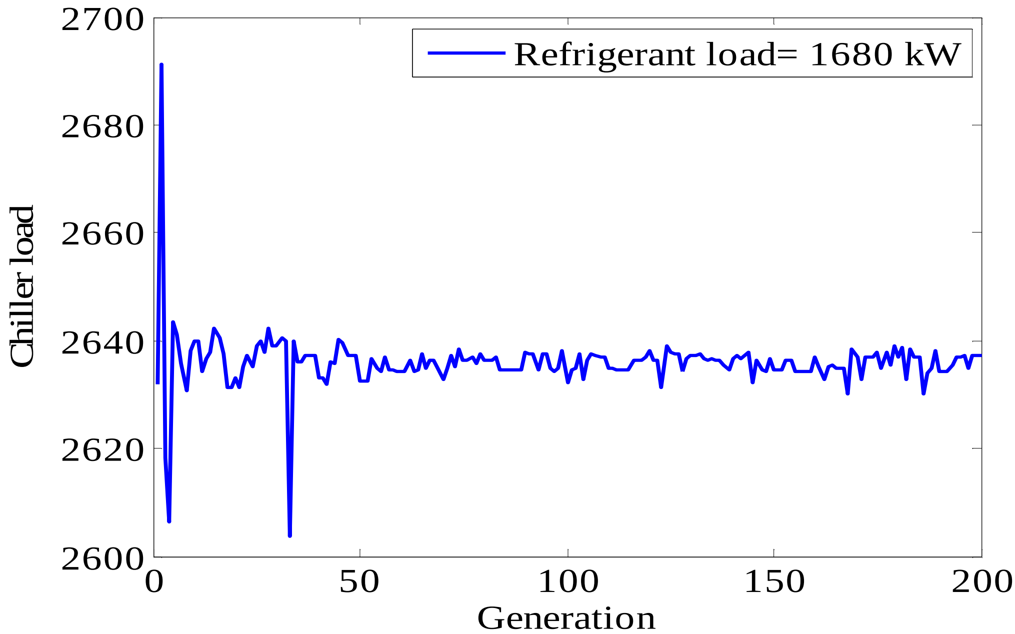

Figure 2: Chart of Production and Consumption 1680kw Chiller Load

CONCLUSIONS

In this paper, we first calculated the using heating and cooling load calculation and optimization are going to go after it. Results of load the building is as follows:

Table 3: Results of Load the Building

load	Sensible Cooling	Latent Cooling	Total Heat
Value	244437	54331	168220

Genetic algorithm is used to simulating With regard to the three units installed in the building diagram of kW-PLR and they are all characterized by nominal capacity, PLR optimal values for each of the heating and cooling loads are calculated to obtain.

Table 4: All Characterized by Nominal Capacity

Chiller	a_i	b_i	c_i	d_i	Capacity(RT)
CH-1	22209	180094.2	-214154.6	173481	150000
CH-2	14651.56	133394.8	-83727.6	60709	150000
CH-3	28619.8	66990	3162.94	21956	150000

After applying the genetic algorithm, the values of PLR in the heating and cooling loads are obtained as follows.

Table 5: The Values of PLR in the Heating and Cooling Loads

	Needed Load	PLR_1	PLR_2	PLR_3	$U_{chiller}$
Heating	298768	0.4067	0.5884	0.9966	298750
Cooling	168220	0.144	0.2954	0.6812	168090

REFERENCES

1. S.C. Hu, Y.K. Chuah, A study of energy and electricity consumption for high-tech industry focused on the semiconductor industry in Taiwan area, Report of NSC. 89-TPC-7-02-002, 2000.
2. Y.C. Chang, H.C. Tu, An effective method for reducing power consumption—optimal chiller load distribution, power system technology, in: Proceedings of the International Conference on Power, vol. 2, 2002, pp. 1169–1172.
3. Liquid Chilling System, ASHRAE Handbook, 2000 (Chapter 38).
4. Roy J. Dossat, Principles of Refrigeration, Wiley, 1991.
5. J.I. Levenhagen, D.H. Spethmann, HVAC Controls and Systems, McGraw-Hill, New York, 1993.
6. W.L. Winston, Operations Research: Applications and Algorithms, Duxbury Press, Boston, 1987.
7. A.J. Wood, B.F. Wollenberg, Power Generation Operation and Control, Wiley, New York, 1984.
8. J.H. Holland, Genetic algorithms, Sci. Am. July (1922) 66–72.
9. G.B. Sheble, K. Brittig, Refined genetic algorithm economic dispatch example, IEEE Trans. Power Syst. 10 (1) (1995) 117–123.
10. Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin, 1995.