Heuristics for the economic dispatch problem

*B. Carpio Flores, A. L. Laureano Cruces, R. López Bracho, J. Ramírez Rodríguez**

***Abstract*--This paper presents GRASP (Greedy Randomized Adaptive Search Procedure), Simulated Annealing (SAA), Genetic (GA), and Hybrid Genetic (HGA) Algorithms for the economic dispatch problem (EDP), considering non-convex cost functions and dead zones the only restrictions, showing the results obtained. We also present parameter settings that are specifically applicable to the EDP, and a comparative table of results for each heuristic. It is shown that these methods outperform the classical methods without the need to assume convexity of the target function.**

.

Key words: **Genetic algorithms, search methods, simulated annealing.**

I. INTRODUCTION

The EDP seeks to determine the power that each generating unit in an electrical power system must supply to meet expected demand at the lowest possible cost, without considering power losses from the transmission network [1], [10], [12]. The exact solution of the EDP can be obtained by listing possible generating combinations for each unit, which could be a very large number, making it impossible to complete the list in a reasonable computing time.

Heuristic methods have been used to solve many problems of optimization, and have proven highly efficient [3]-[5],[7],[11] and continuing to design new algorithms and combinations.

Genetic algorithms (GA) have become increasingly popular for solving variants of the EDP [4], [5], [7], [11] among other search methods [12]. However, heuristics such as tabu search [2], [8] and simulated annealing [2], among others, that have proven to be very efficient in finding good solutions have not been used in the EDP.

In order to show that there are other good options in heuristics, in addition to a Genetic Algorithm [3], this paper proposes the following: a) a hybrid genetic algorithm (HGA) that combines a Genetic Algorithm with a constraint satisfaction algorithm (CSA) [6], a GRASP algorithm [9] and a local search (LS) algorithm b) a simulated annealing algorithm (SAA), and c) a GRASP algorithm.

Benjamin Carpio Flores is with, CENACE, Departamento de Planificación Económica de Largo Plazo, México D. F. (e-mail: benjamin.carpo@cfe.gob.mx).

A. L. Laureano Cruces is with Departamento de Sistemas, UAM, México D. F. (e-mail: clc@correo.azc.uam.mx.

R. López Bracho is with Departamento de Sistemas, UAM, México D. F. (e-mail: rlb@correo.azc.uam.mx.

J. Ramírez Rodríguez is with Departamento de Sistemas, UAM, México D. F. (e-mail: jararo@correo.azc.uam.mx.

Heuristic methods were implemented based on general algorithms with the parameters recommended in general terms in the literature. A summary of the parameters used in each heuristic is included, with a brief description of the parameters inherent to its application to the EDP.

The paper is organized as follows: based on the definition of the problem in section II and the description of the study cases in section III, we describe the algorithms used, followed by the computational results obtained, and examine comparisons with one another and with other methods in section IV. Section V outlines the conclusions.

II. DEFINITION OF THE PROBLEM

The formulation [1], [10] is as follows:

$$\text{Minimize}: \sum_{i=1}^{n} F_i\ (P_i)$$

subject to

$$\sum_{i=1}^{n} P_i = demand$$

$$\text{Pmin}_i \le \text{P}_i \le \text{Pmax}_i$$

Where:

n: number of generating units

F_i : Cost function for generator *i*

P_i : Power of generator *i* $Pmin_i$: *Minimum power generator i can supply*

$Pmax_i$: Maximum power generator *i* can supply

The cost function used to optimize the problem is the input-output curve for each thermal unit. This curve represents the quantity or cost of fuel the unit consumes per hour for the power at which the unit is generating.

The cost functions of thermal units are discontinuous, because they present jumps at certain points due to the closing and opening of valves (valve point) to control generator power output [10].

In addition, combined cycle stations present discontinuities when changing the configuration in the number of turbo gas units. Combined cycle units are formed by several turbo gas units where steam at the turbine exhaust is reused by means of a heat exchanger that feeds the boiler of a steam unit. Input-output curves are obtained for each combination of gas turbines with a steam unit.

For some generating units, the first two points give rise to a non-convex input-output curve in its operating range of 122 to 245 MW. Also, this unit has a dead zone of 260 to 320 MW, which is an operating range in which it is undesirable to

operate due to the technical difficulties and high costs that working in that operating range entails, given that the turbogas unit must be started or shut down.

III. HEURISTICS

Three study cases were used to evaluate the performance of the heuristic methods applied to the EDP:

1. The first two cases are formed by 7 and 36 units. Second degree convex cost functions are considered to set the parameters of each algorithm because the solution under this condition can be obtained easily with the classic methodology. The optimum solution was found using the solver function in Microsoft Excel 2003.
2. The third case is made up by 3 units where the valve points are modeled. This case was taken from Walters [11] and consists of 3 units with cost functions that model the valve points.

The best solution found by Walters is presented below, and, as reported, coincides with the optimum solution obtained by the dynamic programming method.

P1 [MW]	P2 [MW]	P3 [MW]	Demand [MW]	Cost [$]
300.0	400.0	150.0	850.0	8,237.6

The objective function used is: $F = aP^2 + bP + c + | e \text{ sen}(f * (P_{min}-P)) |$; this is the general function that models the valve points [10]. If the coefficients *e* and *f* are zero, then the cost function is that commonly used to simplify the problem.

A. Genetic Algorithm

The pseudo code for the GA, followed by the general parameters used [2], [3], [5] and a description of how the dead zone was handled.

TABLE I
SIMPLE GENTIC ALGORITHM

start
Accurately represent the solutions to the Problem
Generate initial population with individuals or solutions to the Problem
Define an fitness function for individuals in the population
do while (the shutdown criterion has NOT been satisfied)
Choose pairs of individuals as parents
Crossover the parents chosen with probability p*c* to obtain two children
Replace the parents chosen with their children
Mutate some characteristics of each individual with probability p_m
The fitness function of individuals in the population
Choose individuals that survive to the next generation
enddo
end

Parameters: p_c= 0.8; p_m = 0.01; breeding points: 2 (random); population size: 40; method of selection: competition; elitism = yes; aptitude: objective function.

Penalty factor: The penalty factor PF_IBP penalizes solutions that fail to comply with the balance of power BP. This factor should be set properly for each instance of the problem. Small values cause the solution to converge on a lower-than-optimum cost, at the expense of a negative BP (the sum of the powers assigned is less than demand). On the contrary, large PF_IBP values lead to solutions that comply with the BP but result in above-optimum costs.

A PF_IBP value of 1000 produces good results, both in cost and in BP. The best values were obtained under the following reasoning: an error of 1 MW in the BP should be comparable with the highest cost a unit has on increasing or reducing the same unit power, so that the error is suitably penalized.

The cost of a 1 MW increase or reduction in each unit from an operating point is called incremental cost IC and is obtained by deriving the cost function *F*. For example, assuming that there are 50 inexpensive units operating with IC= 1 $/MW, and one costly unit operating with IC = 50 $/MW, the result is the same cost for the 50 units increasing 1 MW, with a total increase of 50 MW, as if the costly unit operates at 1 MW. The penalty for a 1 MW difference in the BP should be similar to the cost of increasing 1 MW in the unit with the highest IC. $PF\_IBP = \max\{IC_i\}$, i = 1,…, U's.

Dead zone. The dead zone is an operating range where a unit cannot be operated for technical or economic reasons. These zones can be restricted from encoding or by penalizing solutions that include operation in a dead zone. When encoding the dead zone an additional bit *bit_zm* is added to the subchain to indicate operation above (1) or below (0) the dead zone. When the unit operates above the dead zone, the whole value encoded increases by 2^L units. L is the number of bits in the subchain used to represent the power of each unit. When the integer value is decoded, the unit should be checked for dead zone operation.

The expression to decode the integer value is

$$Pi = P\min i + \frac{Pcod(P\max i - P\min i)}{2^{Li}-1}$$

Another option to treat the dead zone is by permitting power values in the dead zone but penalizing the target function. A value of 1% of the total cost of generation gave good results. Mutation and crossover operators may modify bit_zm from one generation to another.

Neighborhood generation algorithm

The neighborhood generation algorithm below, was used for the LS, the HGA, SAA and GRASP algorithms. This algorithm generates k_0 solutions from among which the lowest cost is kept.

TABLE 2
NEIGHBORHOOD GENERATION ALGORITHM

Start

```
Sol_ini = {PU1, PU2, …,PUn}
randomly select u, u∈ U={1,2,.., n}; n = number of units
movement = up (+) , if r>=0.5; down (-), if r <0.5,  r=
random (0,1)
number of MW= random(P,Pmax), if movement=(+);
random (Pmin,P), if movement = (-)
Pu = Pu +  movement * number of MW
U = U - {u}
difUs = [U] -1; [U] = cardinality of 1
pot_resto=number of MW
for i= 1 to difUs
     pot = random(0,pot_dif)
     pot_dif = pot_dif - pot
     PUi = PUi - movement * pot
endfor
PUn = pot_dif
end
```

The neighborhood to a given solution are defined by increasing or reducing the power of a randomly selected generating unit; the movement upward or downward is also random, a random number x is generated with a uniform distribution, so that the power: rises if $x \geq 0.5$ and falls if $x < 0.5$. The number of MW that changes the power is random and is limited by the maximum and minimum powers of the units. The change in power of the unit selected should be covered by the other units so that the value of demand supplied is not changed. The last unit should absorb the difference in demand that needs to be covered.

Dead zone. When there is a dead zone in a unit, the neighboring solution respects the unit's operating zone in the current solution, above or below the dead zone.

B. Hybrid Genetic Algorithm

The HGA is made up by the GA and some heuristics that are added to the input and output of the GA with the intention of improving the rate of convergence and the quality of the final solution [4], [7]. These heuristics are:

1. Constraint Satisfaction Algorithm (CSA) to create the initial population
2. GRASP construction phase algorithm to create the initial population
3. BL algorithm to improve the quality of the solution and the rate of convergence.

Constraint Satisfaction Algorithm (CSA) to create initial population

As possible power values for a unit, at the outset its power constraints (initial domain) are considered, but applying the CSA this initial domain is modified on confirming that the randomly assigned power value allows subsequent values to be chosen for the other units. In the EDP, participating units have been previously selected and should be dispatched with a power value, at least at their minimum power limit. To better understand why power limits should be restricted in the process of random assignment of power, the following example is provided:

TABLE 3

CSA TO CREATE A FEASIBLE SOLUTION

```
Start
 dr ← dem
 i=1
 Do
   NDi = new_domain (i, dr)
   Pi ← rand(NDi)
   dr ← dr - Pi
    i  ← i + 1
    to i=n
    Pn ← dr
  end
   New_domain(i, dr)
   start
   NPmaxi ← dr - Sum (Pmink),  k>i
   NPmini ← dr - Sum(Pmaxk),  k>i
 end
```

$$NP\max_i = \begin{cases} NP\max_i & if \quad NP\max_i < P\max_i \\ P\max_i & otherwise \end{cases}$$

$$NP\min_i = \begin{cases} NP\min_i & if \quad NP\min_i > P\min_i \\ P\min_i & otherwise \end{cases}$$

dem - demand
n - number of units
P_i - power assigned to unit i, $1 \leq i < n$
D_i - domain of unit *i*, defined by the power limits of each unit, $D_i = [Pmin_i, Pmax_i]$, $1 \leq i < n$
dr – remaining demand
ND_i – New domain for unit i by CS, NDi = [$NPmin_i$, $NPmax_i$], $1 \leq i < n$

Example. Find an initial solution from:

	Pmin	Pmax	demand
U1	35	210	400
U2	130	325	
U3	125	315	

Based on the data, it is not possible to choose a power value of 150 for the first unit, as the minimum power that the other 2 units could supply would be 255, giving a total of 405 and the demand would not be fulfilled.

Thus, the maximum value *NPmax* that can be assigned to unit U1 is defined by:

$NPmax_1 = demand – Pmin_2 - Pmin_3 = 400 – 130 – 125 = 145$

and the minimum value *$NPmin_1$* is defined as follows:

$NPmin_1 = demand – Pmax_2 - Pmax_3 = 400 – 325 – 315 = - 240$

Because the value is less than its power limit $Pmin_1$, then that limit is not changed, and the new limits for U1 are:

$Pmin_1 \leq P_1 \leq NPmax_1$ in other words, $35 \leq P_1 \leq 145$

Similarly, the limits of the other units are modified, except for the last unit, which simply absorbs the remaining demand to be fulfilled.

LS algorithm to improve the quality of the solution and the rate of convergence.

The GA directs the search to promising zones where the BL algorithm expedites convergence toward a better quality solution. The LS algorithm is based on the generation of vicinities described above. After a number of initial generations (NGA_{ini}), the GA has substantially reduced the aptitude of the individuals; then a solution is sought with a small error in BP, depending on the precision required. If no individual is found in the population that meets this criterion, the GA is activated to produce another generation, until the BP criterion is met. The error in BP is assigned to one of the units U_{sel} with sufficient capacity.

TABLE 4
HYBRID GENETIC ALGORITHM

```
BEGIN HGA
  Generate initial population
  n = NGA_ini
  DO
      FOR i = 1 TO n DO
          Evaluate population
          Select parents for new population
          Crossover parents to generate children
          Mutate children (there is a new population)
      END FOR
      Find individual with lowest DP
      n = 1
   WHILE (DP > 0.1)
   P_Usel ← P_Usel + DP
   BL algorithm
END HGA
```

It has been observed that the LS algorithm makes the HGA solutions independent of the quality of the individuals in the initial population and that the performance of the GA is inferior when using good quality solutions at the start, because the percentage of improvement is much less than when unfeasible solutions are used.

The solutions provided by cp_GRASP are of such good quality that the GA is unlikely to improve them, after 150 generations the improvement of the best individual was only 0.1%. However, the work of the GA can be observed if the average aptitudes of each generation are reviewed, in this case there is a 5.5% improvement. We can say that the sequence cp_GRASP + GA + BL is practically the same as cp_GRASP + BL = GRASP.

The GA performs best when it works in its simplest form, without adding operations that help to purge the random solutions ensuring feasibility. The percentages of improvement are 8.5% for the best individual and 38.6% for the entire generation.

The intensity of the local search is controlled with parameter k_0. Tests were performed with different values for this parameter to obtain a minimum acceptable value for cases 1, 2, and 3. For the three cases the best results were obtained with $k_0 = 500$.

C. Simulated Annealing Algorithm

The pseudo code for the SAA is:

TABLE 5
SIMULATED ANNEALING ALGORITHM

```
start
    initialize (sini, T, K0)
    k ←0
    s ← sini
    do while (T> minimum temperature (shutdown criterion)
       for k=1 to K0 do
             generate s' ∈ N(s))
             if f(s') < f(s) then s ←s'
              else
                 generate_random n in [0,1]
                 if (exp((f(s) – f(s'))/ c T) > n) then s ←s'
              end if
        end for
        T ← αT , α ∈ (0,1)
    end do
end
```

Temperature Parameter t.
The Metropolis criterion of the SAA uses Boltzman's distribution function to accept a new solution.

$$P_t(accept \quad j) = \begin{Bmatrix} 1 & if \quad C(j) < C(i) \\ \exp\left(\frac{C(i) - C(j)}{t}\right) & if \quad C(j) > C(i) \end{Bmatrix}$$

The value of t should be such that the exponential term can be comparable with *r*, whose range is [0, 1]. The value that approaches 1 is obtained when the cost functions of the solutions compared are nearly equal. The value 0 is obtained when $t \rightarrow \infty$. A good option is to set the value of t based on the maximum difference in cost (*mdc for short)* in the space for solutions. This value is obtained from the difference in cost of the units operating at maximum and minimum power.

Boltzman's function was evaluated for values of t referred to *mdc*. t = mdc/4 covers most of the range [0,1] and the possibility of accepting solutions with large differences in cost is less than the other assignments of t. A solution with a difference of 300 thousand pesos compared with the cost of the current solution would be accepted with a probability of 0.1 with t= mdc/4, and with a probability of 0.74 with T=mdc*2. The total number of neighborhood generated depends only on the initial temperature T, if the temperature drop factor α, the number k_0 of acceptances, and the shutdown criterion are fixed. On reducing T, a smaller number of solutions are reviewed, in addition to having less variety among them. Therefore, a suitable balance with the value of T should be found that allows variety in the solutions and generates a sufficient number of solutions.

Ten tests were performed using different values of t, referred to mdc. The minimum cost, the best average, and the smallest standard deviation do not coincide for the same value of T. However, it is desirable to choose the value of T that has the best combination of average cost and variability. Thus, t= mdc/4 will be used for the case of 36 units, and t =mdc/4 for the case of 3 U´s

Ten tests were also performed for each study case and to determine the best value of the parameter k_0 of BL. The values chosen were; for case 1: T= 500 000 and k_0 = 50, for case 2: T= 500 000 and k_0 = 800, and for case 3: T= 2 500, k_0 = 200.

D. GRASP

The GRASP method consists of a constructive phase and a local search phase [9]. The constructive phase creates a feasible solution, which is improved by the local search algorithm and the best solution is saved in the iterative process.

TABLE 6
GENERAL GRASP

```
f* ← ∞
  for i ≤ imax do
     constructive phase(g(.), α, x)
     x ← local search(x)
     if f(x) < f* then
       f* ← f(x)
       x* ← x
     end if
  end for
return x*
```

where
i_{max} is the maximum number of iterations.
g(.) is the myopic function that evaluates the incorporation of each element
x is the solution set, which contains the powers assigned to the N units
f(x) is the cost of solution x
f* is the cost of the best solution found
x* is the best solution found
$\alpha \in [0,1]$ establishes the degree of randomness or myopia of the algorithm

The LS algorithm is the same used for the SAA, and the constructive phase algorithm for EDP is shown below:

TABLE 7
CONSTRUCTIVE PHASE GRASP

```
start
  x = {};
  dem_dif ← Demand
  Initialize list of candidates C
     while |C| ≠ 1 do
        calculate New limits NPmini, NPmaxi | i∈ C
        pi ← random(NPmini, NPmaxi) | i ∈ C
        calculate pmyopic(pi) | i ∈ C
        f ← min{ pmyopic(pi) | i ∈ C }
        f+ ← max{ pmyopic (pi) | i ∈ C }
        RCL ← { p ∈ C | pmyopic (pi) ≤ f + α( f+ – f) }
        random selection of p for RCL
        x ← x ∪ {p}
        C ← C – {p}
        dem_dif ← dem_dif – p
     end while
     p {C} ← dem_dif
end
```

Where
Demand: what all units should generate
C: list of candidates
RCL: restrictive list of candidates
x: solution set
pi: element i in the feasible solution
p {c} : last element in the list of candidates

The algorithm creates a feasible solution $x = \{p_1, p_2, \ldots p_i, \ldots, p_n\}$. At first, the solution set for x is empty (1), the variable *rem_dem* starts with a value for the demand the N units must meet and lowers its value as powers are assigned to each element in the solution. The initial list of candidates C is made up by all the units $C = \{ U_1, U_2, \ldots, U_i, \ldots, U_n\}$. then, each unit is assigned a random power (6) in the restricted domains, calculated (5) by means of a constraint satisfaction algorithm. The application of this algorithm allows values to be assigned to a unit so that in subsequent stages it is possible to assign powers to the remaining units. The myopic function (7) is the generating cost function that evaluates each generator based on its randomly assigned power. The restrictive list RCL will contain the lowest cost units that are in the range defined in en 10. The element that is added to the solution is chosen randomly from the RCL (11). Then the solution set (12), the list of candidates (13), and the demand to meet are updated (14). In each cycle an element is added to the solution until the list of candidates contains only one element. The remaining demand *rem_dem* is assigned to this last unit.

IV. COMPARATIVE RESULTS OF HEURISTICS

SUMMARY, CASE 1
In the results of the GA, the solutions have high variability (3,915) and the average error compared with the cost of the optimum solution (487,018) is 1.95%.

TABLE 8 shows that the quality of the solutions is good for the GA and excellent for the algorithms HGA, SAA, and GRASP. The most stable algorithm is GRASP with a standard deviation of $1, although it uses more than 10 times the time reported for the HGA and the SAA. As we have mentioned, the GA rapidly improves the initial solutions, but the quality of the final solution is poor and highly variable. The statistics for the GA shown are very similar from generation number 100, but the result of 300 generations is shown to show that the quality of the solution no longer improves significantly. The algorithm that performed best was the SAA, due to the quality of its solutions in a shorter execution time.

TABLE 8
SUMMARY OF THE APPLICATION OF HEURISTIC METHODS TO CASE 1 WITH CONVEX COST FUNCTIONS

	GA	HGA	SAA	GRASP
	Generations=300	NAG_{ini} = 75	T = 500 000	Nitr=300
		k_0=500	k_0=50	K_0=400
Minimum cost	490,719	487,027	487,025	487,019
Maximum cost	503, 999	487,073	487,082	487,022
Average cost	496,491	487,061	487,043	487,021
Standard deviation	3,915	15	19	1
% error[1]	0.76	0.002	0.001	0.000
%average error	1.95	0.01	0.01	0.00
Time	0.45	0.09	0.03	0.89

[1]minimum cost compared with cost of the optimum solution
[2] average cost compared with cost of the optimum solution

The results for case 1 when considering the dead zone and the non-convex function in U7 are shown in Table 9.

TABLE 9
SUMMARY OF THE APPLICATION OF HEURISTIC METHODS TO CASE 1, WITH DEAD ZONE AND NON-CONVEX COST FUNCTION FOR UNIT 7

	GA	HGA	SAA	GRASP
	Generations=300	NAG_{ini} = 75	T = 500 000	Nitr=300
		k_0=500	k_0=50	K_0=400
Minimum cost	487,843	486,003	486,004	486,003
Maximum cost	501,288	486,018	489,018	486,003
Average cost	494,925	486,007	487,514	486,003
Standard deviation	5 271.2	5.2	1 573	0.03
% error[1]	0.379	0.000	0.000	
%average error	1.84	0.00	0.31	
Time	0.44	0.09	0.05	1.52

[1]minimum cost compared with cost of the optimum solution
[2] average cost compared with cost of the optimum solution

In this case, the optimum solution is not known, due to the restrictions imposed on unit 7. Again, GRASP provided the best solutions and the solutions in all tests were practically the same. The HGA performed better here than the SAA.

Case summary Unit 36
Table 10 shows the best performance of the HGA when it has the lowest cost and the best average cost with an acceptable variability compared with the other heuristics. The GRASP presented similar results to the HGA but with a time four times greater.

TABLE 10
SUMMARY OF THE APPLICATION OF HEURISTIC METHODS TO CASE 2 WITH CONVEX COST FUNCTIONS

	GA	HGA	SAA	GRASP
	Generations=300	NAG_{ini} = 75	T = 500 000	Nitr=300
		k_0=750	k_0=800	K_0=1000
Minimum cost	4,449,024	**4,443,442**	4,456,305	4,444,245
Maximum cost	4,455,664	4,447,401	**4,458,577**	4,448,312
Average cost	4,452,219	**4,445,964**	4,457,366	4,446,539
Standard deviation	2,454	1,284.2	**719**	1,115
% error[1]	0.16	**0.03**	0.32	0.05
%average error	0.23	**0.09**	0.35	**0.10**
Time	3.1	3.2	4.0	12.4

[1] compared with cost of the optimum solution

In Table 11 the variability of the solutions is increased for the heuristics GA, HGA, and SAA because when considering the dead zone for unit 36, the solutions present jumps in the power of unit 36, around the limits of the dead zone.

TABLE 11
SUMMARY OF THE APPLICATION OF HEURISTIC METHODS TO CASE 2, WITH DEAD ZONE AND NON-CONVEX FUNCTION FOR UNIT 36

	GA	HGA	SAA	GRASP
	Generations=500	NAG_{ini} = 500	T = 500 000	Nitr=500
		k_0=750	k_0=800	K_0=1000
Minimum cost	3,850,923	**3,843,912**	3,850,720	3,845,360
Maximum cost	3,863,661	3,853,344	**3,870,634**	3,847,561
Average cost	3,856,027	3,848,348	3,857,610	**3,846,342**
Standard deviation	4,725	3,860	7,593	**831**
% error[1]	0.14	-0.04	0.14	0.000
%average error	0.28	0.08	0.32	**0.03**
Time	3.1	3.3	4.1	16.4

[1] compared with best cost obtained (HGA)

Results for Case 3
The parameter settings for each heuristic for case 3 are summarized in Table 12.

TABLE 12
PARAMETER SETTING OF HEURISTICS, CASE 3

	Generations	Nitr	FP_IBP	k_0
AG	400	-	25	-
AGH	150	-	25	500
ARS	-	-		200
GRASP	-	100		100

Table 13 shows the performance of the heuristics for case 3. In this case all the heuristics provided fairly acceptable solutions, in particular the SAA due to the quality of the solutions as regards variability and execution time.

TABLE 13
SUMMARY OF THE APPLICATION OF HEURISTIC METHODS TO CASE 3

	GA	HGA	SAA	GRASP
	Generations=400	AG = 150	T =2 500	Nitr=100
		k_0=500	k_0=200	k_0=100
Minimum cost	8,241.1	8,234.2	8,234.1	8,234.08
Maximum cost	8,500.2	8,254.1	8,252.0	8,234.2
Average cost	8,327.4	8,241.7	8,241.81	8,234.11
Standard deviation	89.4	7.4	5.1	0.02
% error[1]	0.085	0.001	0.000	0.000
%average error	1.133	0.093	0.094	0.000
Time	0.26	0.10	0.03	0.04

Fig. 1, shows the cost functions analyzed in this case. As a reminder, the functions in thicker colored lines represent a behavior closer to the cost of fuel due to the opening and closing of and valves in the unit's power control. Heuristic methods can evaluate this kind of functions in the EDP and the comparative results with solutions offered by the classical method which uses the second-degree functions

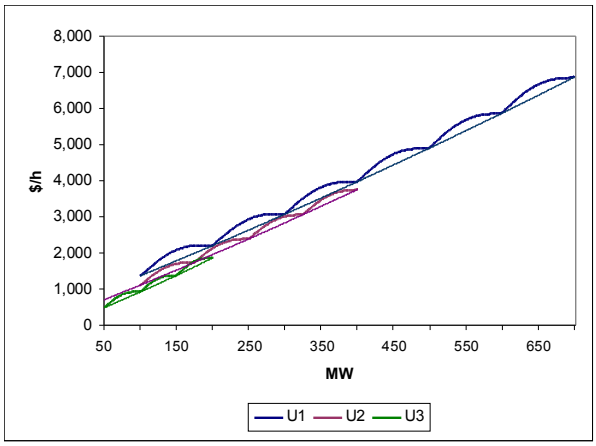


Fig. 1. Cost functions for case 3

V. CONCLUSIONS

We have presented the application of four heuristic methods to the EDP without losses. The study cases include an evaluation of non-convex cost curves and the consideration of dead zones in some thermal units. These are conditions that cannot be evaluated with classical optimization methods.

The results obtained for the EDP solution were fairly satisfactory and the four heuristics presented have shown consistency in obtaining good solutions, allowing for non-convex cost functions and the discontinuities that dead zones represent.

It has been made clear that the being able to evaluate the real cost curves of combined cycle units in PIEs and the closest cost functions for thermal units that account for valve points, with heuristic methods, has allowed us to obtain a lower cost of generation for generation dispatch in the study cases presented. For the first case, the differences in cost are minimal and we conclude that the second degree cost function used at present is acceptable, given that the improvement in cost was only 0.03 % under extreme conditions, when the real and adjusted curves are farthest apart. When considering valve points in cost functions, in the case of 3 units, the GRASP considerably improved generation dispatch, by 3 %.

There is no comprehensive approach to solving the economic dispatch problem capable of accounting for all the restrictions that may appear; however, it is necessary to combine and iterate with different problem solving techniques that treat some restrictions in isolation to achieve better solutions. First, we need to show that these techniques work and can solve such isolated problems. In this case of application of heuristics, they can be used to evaluate non-convex cost functions and discontinuities that arise, to combine them with other problem solving techniques.

VI. REFERENCES

[1] F. Aboytes, J. J Guerrero, "Despacho económico, flujos óptimos y control de generación". Curso de Capacitación en Sistemas de Potencia, CENACE. México, D. F., 2001.

[2] F. Glover and M. Laguna, *Tabu Search*, Ed. Kluwer, London, 1997.

[3] D. E Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, The University of Alabama, Adison-Wesley publishing Company, 1989.

[4] F. Li, A comparison of genetic algorithms with conventional techniques on a spectrum of power economic dispatch problems. *Expert Systems with Applications* 15, 133-142, 1998.

[5]Li F, R. Morgan, and D. Williams, "Hybrid genetic to ramping rate constrained dynamic economic dispatch". *Electric Power Systems Research* 43, 97-103, 1997.

[6] F. Manyà, and C. Gomes, "Técnicas de resolución de problemas de satisfacción de restricciones", *Revista Iberoamericana de Inteligencia Artificial*: 19: 168-180.

[7] A. Olachea, "Aplicación de la técnica de algoritmos genéticos al problema de despacho económico". M Sc dissertation, Dept of Electric Eng. Universidad Autónoma de Nuevo León, 2003.

[8] C. R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. Mc Graw Hill, 1995.

[9] M. Resende, "Greedy Randomized adaptive search procedure (GRASP)", AT&T Labs Research Tech. Rep. 98.41.1. December 22, 1998.

[10] A. J. Wood, and B. F. Wollenberg, *Power generation operation and control*. Ed. John Wiley and Sons Inc, 1984.

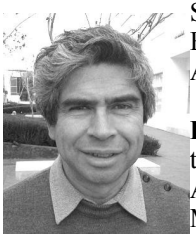
[11] D. Walters, and G. Sheble, Genetic algorithm solution of economic dispatch with valve point loading. *IEEE Transactions on Power Systems*, Vol. 8, No. 3, 1993.

[12] M. Zarei, A. Roozegar, R. Kazemzadeh, and Kauffmann J.M, "Two Area Power Systems Economic Dispatch Problem Solving Considering Capacity Constraints Transmission". *International Journal of Electrical, Computer, and Systems Engineering* 1;3, pp 155-160, 2007.

VII. BIOGRAPHIES

Benjamin Carpio Flores. Obtained a bachelor's degree in Electrical Engineering and a Master's degree in Computer Sciences at Universidad Autónoma Metropolitana, he is at CENACE, Department of Economic Planning of Long Term, México D. F.

Ana Lilia Laureano Cruces was born in Mexico City. She obtained a bachelor's degree in Civil Engineering, a Master's degree in Computer Sciences, and a Doctoral degree in Science at UNAM. She has been full-time professor at Universidad Autónoma Metropolitana since 1989 and guest professor at Universidad Nacional Autonoma de Mexico since 2003. She has more than 70 publications in the field of artificial intelligence. Between 1998 and 2000, she was a guest researcher at the Departamento de Sistemas del Instituto de Automática Industrial del Consejo Superior de Ciencia y Tecnología de Madrid. Her principal research areas are: Multi-Agent Systems, Intelligent Systems applied to Education, Expert Systems and Behavioral Analysis and Affective Computing.



Rafael López Bracho is Titular Professor in the Department of Systems at the Universidad Autónoma Metropolitana-Azcapotzalco of Mexico City, Mexico. He holds a 3rd cycle doctorat in informatic from Université de Paris-Sud, France, since 1981. His research interests are in Graph Theory, Combinatorial Optimization and Operations Research.

Javier Ramírez Rodríguez is Full Professor in the Department of Systems at Universidad Autónoma Metropolitana in Mexico City. He is Doctor in Mathematics from Complutense University since 2001. His research paper have been published in refereed journal such as the European Journal of Operational Research, Lecture Notes in Artificial Intelligence, Omega and others. His research interests are in Heuristics Methods, Network Optimization and Soft Computing.