Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 3 (2012) 304 – 315

Procedia Technology

The 2012 Iberoamerican Conference on Electronics Engineering and Computer Science

A Graph-based Method to Solve the Economical Dispatch Problem Disregarding Slack Variables

Jaime Cerda Jacobo[a], Nancy P. Cira Perez[a,1], Juan J. Flores Romero[a]

[a]*Graduate Division, School of Electrical Engineering, University of Michoacan, Morelia, Michoacan, Mexico 58000*

Abstract

One of the greatest challenges to confront Nonlinear Programming Problems, it is the selection of the active and non active set of constraints of the system. For this reason many optimization applications prefer to use barrier or penalty methods with their related inefficiencies. This paper describes a graph-based solution for these models which facilitates the handling of such constraints and, therefore, the solution process for the model. To this end some parts of the graph are considered active or non active, depending on the actual model solution as well as the values of the Lagrange multipliers. At every solution step, there will probably be some changes on the graph topology to reflect the current conditions of the problem whose solution is in progress. These solutions besides being efficient, provide an optimal storage scheme as only the fundamental information of the problem is stored.

Keywords: Nonlinear programming, Economic dispatch, Dispersity.

Nomenclature

$\mathbb{G}$	The set of generators
Q	The inelastic compound load
λ	The energy price
z_g	NLP desicion variable
$\overline{\rho_g}$	Dual variable associted to z_g uppper bound
$\underline{\rho_g}$	Dual variable associted to z_g lower bound
$\lceil z_g \rceil$	The upper limit value of variable z_g
$\lfloor z_g \rfloor$	The lower limit value of variable z_g
Δz_g	The variable incrementz_g

1. Introduction

One of the greatest challenges to confront Nonlinear Programming Problems is the selection of the active and non active set of constraints of the system. For this reason many optimization applications prefer to use

Email addresses: jcerda@umich.mx (Jaime Cerda Jacobo), npaolacp@hotmail.com (Nancy P. Cira Perez), juanf@umich.mx (Juan J. Flores Romero)

[1]Supported by CONACYT

2212-0173 © 2012 Published by Elsevier Ltd.
doi:10.1016/j.protcy.2012.03.033

barrier or penalty methods with their related inefficiencies [1]. This paper describes a graph-based solution for these models which facilitates the handling of such constraints and, therefore, the solution process for the model. To this end some parts of the graph are considered active or non active, depending on the actual model solution as well as the values of the Lagrange multipliers. At every solution step, there will probably be some changes on the graph topology to reflect the current conditions of the problem whose solution is in progress. These solutions besides being efficient, provide an optimal storage scheme as only the fundamental information of the problem is stored.

Unfortunately the use of matrix methods with their tools and development environments for their handling make those graph-based methods in some sense more complex, discarding apriori the efficiency of them [2]. The objetive of this paper is twofold: The first one is to show the easiness of these methods presenting some of the advantages compared to other methods, established in [3] and [4],.and the second one is to propose a new elimination order based on the underlying graph topology. For this purpose, some parts of the graph are considered active or non active depending on the actual system solution and, only, the Lagrange Multipliers values as opposed to [4] where slack variables were used. Changes on those variables will modify the graph topology reflecting the actual conditions of the problem whose solution is in process, unlike the other methods that have an static topology [5]. Even more, some extensions to the model have been proposed which allow the application of decentralization techniques on the graph [6] as an alternative to some other matrix-based proposals [7, 8, 9, 10].

This document has been divided as follows: In section 2 the Economical Dispatch is introduced. Then, in section 3, the graph representation for this optimization problem is presented. After this, in section 4, Karush-Khun-Tucker conditions are introduced as they are the pillars over which all this mechanism is built. Immediately, in section 5, the algorithms to exploit such graphs are presented. Section 6 presents a study case to show the application of the technique proposed in this document. Section 7, as a sake of completeness, shows the convergence graphs. Finally, section 8 provides some conclusions about this methodology.

2. The Economical Dispatch

The Economical Dispatch can be taken as an initial solution to the DC optimal power flow problem, where the network effects are disregarded. Therefore, the system can be represented as a single node to which all of the generation units and the load are connected. Figure 1 shows an schematic representation of this model. There are three generators and one 850W load [11]:

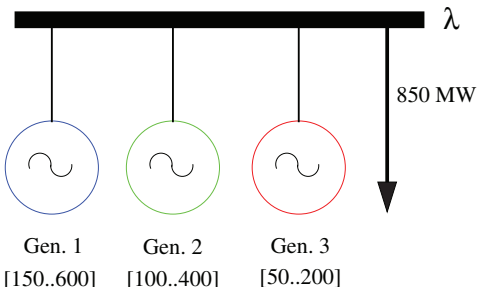

Fig. 1. Economical Dispatch Model

The Economical Dispatch model that describes this diagram is a Nonlinear Programming model, specifically a Quadratic Programming model, and is given by equations from 1 to 4.

$$\max_{p_g, q_l} \quad \sum_{l \in \mathbb{L}} B_l(q_l) - \sum_{g \in \mathbb{G}} C_g(p_g) \tag{1}$$

s.t.

$$\sum_{g \in \mathbb{G}} p_g - \sum_{l \in \mathbb{L}} q_l - Q = 0 \tag{2}$$

$$\mathcal{L}(z) = \sum_{g\in\mathbb{G}} (C_g(p_g) - \underline{\rho}_g(\lfloor P_g \rfloor - p_g) - \overline{\rho}_g(p_g - \lceil P_g \rceil)) - \lambda(\sum_{g\in\mathbb{G}_i} p_g - Q)$$

	$\nabla(\mathcal{L}(z))$	$H(\mathcal{L}(z))$			
z		p_g	λ	$\underline{\rho}_g$	$\overline{\rho}_g$
p_g	$\beta_g + 2\gamma_g p_g - \lambda - \underline{\rho}_g + \overline{\rho}_g$	$2\gamma_g$	-1	-1	1
λ	$\sum_{g\in\mathbb{G}} p_g - \sum_{l\in\mathbb{L}} q_l - Q$	-1			
$\underline{\rho}_g$	$\lfloor p_g \rfloor - p_g$	-1			
$\overline{\rho}_g$	$p_g - \lceil p_g \rceil$	1			

Table 1. The Newton step ingredients

$$\lfloor P_g \rfloor \leq p_g \leq \lceil P_g \rceil \quad \forall g \in \mathbb{G} \tag{3}$$

$$\lfloor Q_l \rfloor \leq q_l \leq \lceil Q_l \rceil \quad \forall l \in \mathbb{L} \tag{4}$$

In this problem the social welfare is meant to be maximized [12],. This is defined as the difference between the benefit of producing the energy minus the cost to produce it, i.e. $\sum_{l\in\mathbb{L}} B_l(q_l) - \sum_{g\in\mathbb{G}} C_g(p_g)$. There are two constraints: first the power balance constraint i.e. the energy produced has to be consumed, and the second one is about the physical limits of the generators. The grnerators production function is defined by quadratic characteristics as shown in equation 5:

$$C_g(p_g) = \alpha_g + \beta_g p_g + \gamma_g p_g^2 \tag{5}$$

In the same way, the economic benefit model for each variable load is provived by the quadratic concave function $B_l(q_l) = \beta_l q_l - \gamma_l q_l^2$. The economical dispatch purpose is to find an optimal allocation of energy production for each generator which satisfies the load. These allocations have to be within the limits defined for each of them. In this document it will be assumed that the loads does not respond to the prices i.e. they are captive loads, and will be ignored in the model. In general, a nonlinear optimization problem starts by building the Lagrangian. This is the base to implement the Newton step, which formulation is given by equation 6 [1]:

$$H(\mathcal{L}(z))\Delta z = -\nabla(\mathcal{L}(z)) \tag{6}$$

Table 1, shows the involved elements, in the economical dispatch, to compute the Newton step.

For the Lagrangian case, our system is defined by the equation showed on the head of the table. he second column represent the gradient of the Lagrangian with respect the variable in the first column. Finally, columns from 3 to the last one represent the Lagrangian Hessian matrix.

3. Economical Dispatch Graph

The Newton step graph representation economical dispatch system showed in Figure 1, is presented in Figure 2 .

In this graph each constraint is represented by a dual variable and a link set that represents the constraints lineal terms. These connect the primary variables with the dual variables. The only difference between equality constraints and inequality constraints is the link type use to build the interconnection structure. For the equality constraints, the links will be active along the complete solution process. On other hand, the interconnection structure for the inequality constraints only will be taken into account when the constraint is active.

To solve the economical dispatch problem, a graph-based representation will be used. The solution process will be to traverse the graph upwards and downwards using the algorithms that will be defined in

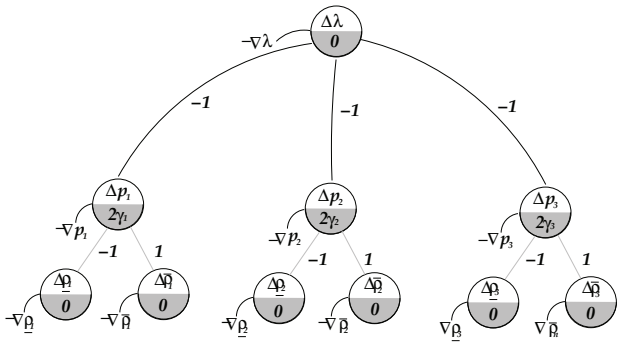

Fig. 2. Graph Corresponding to the basic node model

section 5. It is important to highlight that the graph is a tree therefore nodes in layer n will be only connected with node only one node in layer $n-1$. The only exception is in layer zero where there is only one node called *root* node, no more upper layers exist. Asume that there exist a connection between the variables i and j. If the Gaussian elimination is applied to the node j, then the only node that is affected by the node i, wich is a upper layer from de node j. The graph transition process is showed in Figure 3 when is applied the Gaussian elimination to the node j.

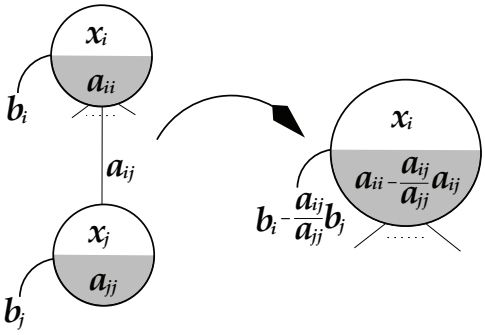

Fig. 3. Graph reduction when applying Gaussian elimination

Appling repeatedly this reduction will get a graph where the only node will be the *root* node. After the graph has been reduced, it will be used an algorithm that is based on the topology of the graph, to resolve it. This algorithm will use the graph topology like a guide to apply a back substitution. For this, suppose that the node i is in the layer n and the node k is in layer $n+1$. Then the equation for variable $x_i$will be defined by 7.

$$a_{ii}x_i + a_{ik}x_k = b_i \tag{7}$$

Where x_i can be expressed as equation 8 shows

$$x_i = \frac{b_i - a_{ik}x_k}{a_{ii}} \tag{8}$$

If the node i is the root node, then there are not more layers at the top, therefore the equation 8 turns into equation 9.

$$x_i = \frac{b_i}{a_{ii}} \tag{9}$$

Furthermore, when one of the constraints is active three nodes connected by two links define a path. In this case there exists a different elimination procedure. The node that represents the primary variable is the

node that will be reduced. Asume that there are connections between the variables i, j and k. If the Gaussian elimination is applied to the node k, this will affect the node i and node j. The graph reduction process is shown in Figure 4.

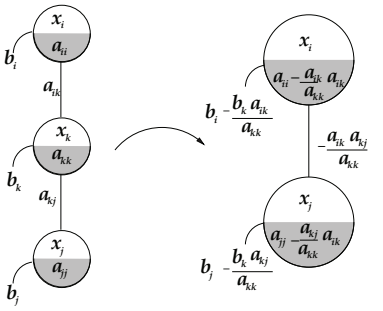

Fig. 4. Graph reduction when applying Gaussian elimination

After this reduction, Gaussian elimination will be applied to node j, as shown in Figure 3.

When the reduction ends, the same algorithm that was defined previously will be applied to perform the back substitution, but the node that is on the second layer will be solved at the end using equation 10.

$$x_k = \frac{b_k - a_{ik}x_i - a_{kj}x_j}{a_k} \tag{10}$$

4. Karush-Khun-Tucker Conditions

The Langrange multipliers method define optimality conditions for equality constraints. However, many problems are defined in terms of inequality constraints defined by 11.

$$\begin{aligned} &\min_z \ f(z) \\ \textbf{st.} \quad & g_j(z) = 0, \qquad j = 1, 2, ..., m \\ & h_k(z) \leq 0, \qquad k = 1, 2, ..., p \end{aligned} \tag{11}$$

To face this problem, Karush-Khun-Tucker conditions (KKT) generalize the Langrange multipliers method defining a minimum set of conditions which guarantee the optimality conditions for non lineal programming problems with inequality constraints. Furthermore, KKT conditions provide sufficient optimality conditions for convex programming problems, as the one we are dealing with. If z^* is the optimal solution for a non lineal problem with $n = |z|$ variables, m equality constraints and p inequality constraints, these conditions are [1]

$$\nabla f(z^*) + \sum_{j=1}^{m} \lambda_j \nabla g_j(z^*) + \sum_{k=1}^{p} \mu_k \nabla h_k(x^*) = 0 \tag{12}$$

$$g_j(z^*) = 0, \qquad j = 1, 2, .., m \tag{13}$$

$$h_k(z^*) \leq 0, \qquad k = 1, 2, .., p \tag{14}$$

$$\mu_k h_k(z^*) = 0, \qquad k = 1, 2, .., p \tag{15}$$

$$\mu_k \geq 0, \qquad k = 1, 2, .., p \tag{16}$$

Where equation 12 represents equilibrium between the gradients of the objective function and the active constraints. Equations 13 and 14 represent the feasibility of the solution in the optimum point z^*. Equation 15 represent the complementarity conditions (i.e. $\mu_k = 0$ or $h_k(z^*) = 0$). Finally, equation 16 represents dual feasibility.

Condition 16, establishes the Lagrange multipliers non negativity property. If the Lagrange multiplier was a negative, then z^* is within the feasible region, not at the boundary, and is even feasible to improve the objective function. Therefore, we can conclude that its corresponding constraint is not active anymore. This condition will be exploited to discriminate the active from the non active parts of the graph.

5. Algorithms

Let us define two boolean variables, $isLB_g$ and $isUB_g$ for each generator g. Equation 17 controls the lower bound represented by $isLB_g$

$$isLB_g \leftarrow \begin{cases} true, & \text{si } p_g < \lfloor p \rfloor \wedge \neg isLB_g \\ false, & \text{si } \underline{\rho_g} < 0 \wedge isLB_g \end{cases} \tag{17}$$

whereas Equation 18 controls the upper bound given $isLB_g$

$$isUB_g \leftarrow \begin{cases} true, & \text{si } p_g > \lceil p \rceil \wedge \neg isUB_g \\ false, & \text{si } \overline{\rho_g} < 0 \wedge isUB_g \end{cases} \tag{18}$$

Both variables are initialized to $false$. Two algorithms will be defined: the first one $ReduceIt(x_i, x_{i-1})$, algorithm 1, will reduce the graph applying the procedure described in Figures 3 and 4; and the second one $ResolveIt(x_i, x_{i-1})$, algorithm 2, will solve the graph applying equations 8, 10, and 9, depending on the actual node to be solved.

Algorithm 1 ReduceIt($\mathbb{G}, \lambda$)

```
for all g ∈ 𝔾 do
  if isLB_g or isUB_g then
    if isLB_g then
      reduce(p_g, ρ_g)
      reduce(p_g, λ̄)
      reduce(λ, ρ_g)
    else
      reduce(p_g, ρ̄_g)
      reduce(p_g, λ)
      reduce(λ, ρ̄_g)
    end if
  end if
end for
```

Finally, the algorithm 3, *main*, will apply iteratively the algorithms *reduceIt* and *solveIt* until a convergence criterion is achieved.

6. Study Case

In this section a three generators case will be analyzed. The Table 2 provides the data for this system.

In the search of clarity, the graph will only show the Δx values, in the lower semicircle of the node, as well as the values for the variables x in each iteration, next to the node. Starting with the graph shown in Figure 5, in this case all the constraints are assumed non actives.

Using algorithms 1 and 2, the graph values shown in Figure 6 are obtained.

In Figure 6 generator 1 power exceeds its maximum power whereas generator 3 power does not satisfy its minimum power. The upper constraint in generator 1 has to be activated as well as the lower constraint in generator 3. It is necessary a new iteration, but the graph will have new parts to include the constraints, as Figure 7 shows.

Algorithm 2 SolveIt($\mathbb{G}, \lambda$)

solve(λ)
for all $g \in \mathbb{G}$ **do**
  if $isLB_g$ o $isUB_g$ **then**
    if $isLB_g$ **then**
      solve($\underline{\rho_g}, \lambda$)
      $\underline{\rho_g} \leftarrow \underline{\rho_g} + \Delta\underline{\rho_g}$
      solve($\underline{p_g}, \underline{\rho_g}, \underline{\lambda}$)
      $p_g \leftarrow p_g + \Delta p_g$
      if $\underline{\rho_g} < 0$ **then**
        $\underline{isLB} \leftarrow F$
      end if
    else
      solve($\overline{\rho_g}, \lambda$)
      $\overline{\rho_g} \leftarrow \overline{\rho_g} + \Delta\overline{\rho_g}$
      solve($p_g, \overline{\rho_g}, \lambda$)
      $p_g \leftarrow p_g + \Delta p_g$
      if $\overline{\rho_g} < 0$ **then**
        $isUB \leftarrow F$
      end if
    end if
  else
    if $p_g < \lfloor p_g \rfloor$ **then**
      $isLB \leftarrow T$
    else
      if $p_g > \lceil p_g \rceil$ **then**
        $isUB \leftarrow T$
      end if
    end if
  end if
end for

Algorithm 3 main($\mathbb{G}, \lambda$)

Begin z
repeat
  Evaluate $\nabla(\mathcal{L}(z))$
  reduceIt($\mathbb{G}, \lambda$)
  solveIt($\mathbb{G}, \lambda$)
until Convergence achieved

generador	α	β	γ	$\lfloor p \rfloor$	$\lceil p \rceil$
1	459	6.48	0.00128	150	600
2	310	7.85	0.00194	100	400
3	78	7.97	0.00482	50	250

Table 2. System components' data

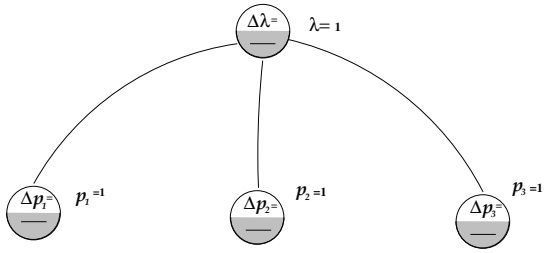

Fig. 5. Initial Graph

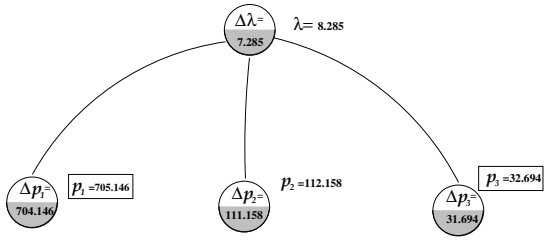

Fig. 6. Graph resolved first iteration

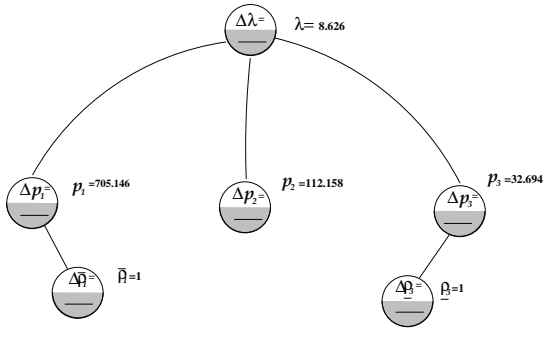

Fig. 7. Initial graph, second iteration

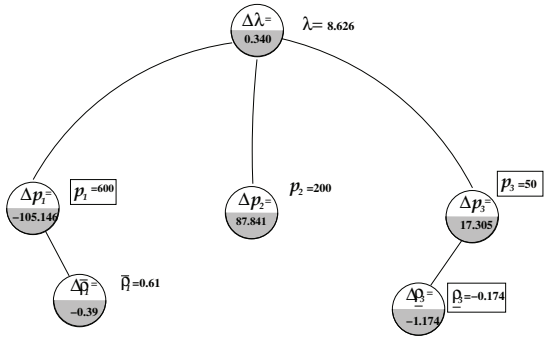

Fig. 8. Graph solved, second iteration

Using algorithms 1 and 2, the new graph values are showed in Figure 8.

Figure 8 shows generator 1 is at its maximum value whereas generator 3 at its minimum value and $\underline{\rho_3}$ is negative. Therefore, according to KKT condition 16, this constraint must not remain active. Another iteration with a new graph topology, as shown in Figure 9, has to be applied.

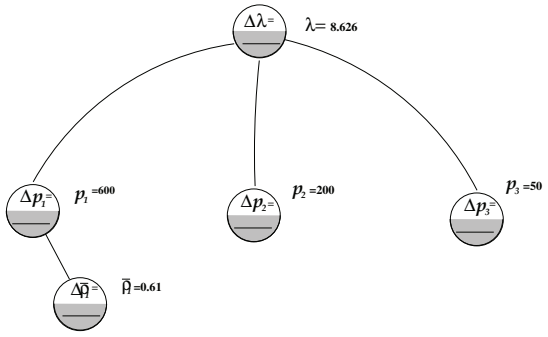

Fig. 9. Initial graph, third iteration

Applying algorithms 1 and 2, the new graph values are showed in Figure 10.

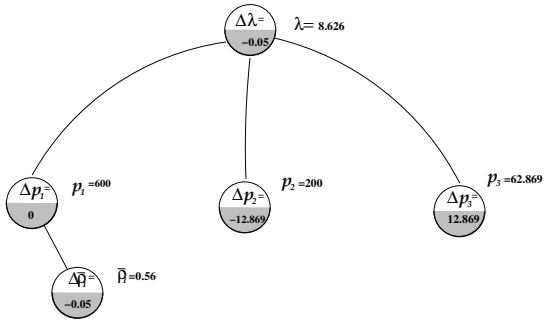

Fig. 10. Graph solved, third iteration

As it can be seen in Figure 10 the lower bound for generator 3 has been eliminated. Furthermore, as the Δ's are different than zero, it is necessary to apply a new iteration, with the same topology that is showed in figure 11.

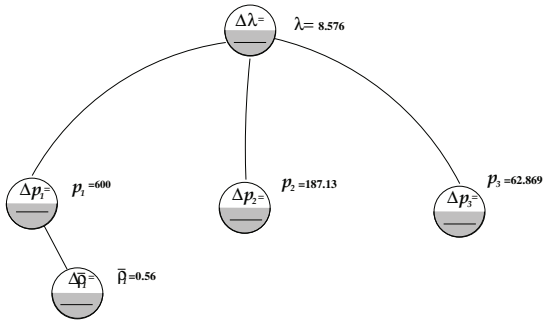

Fig. 11. Initial graph, fourth iteration

Again, applying the algorithms 1 and 2, the new graph values are showed in Figure 12.

As it can be seen in Figure 12 all the Δ's are zero, therefore it is not necesary to apply another iteration i.e. convergence has been reached.

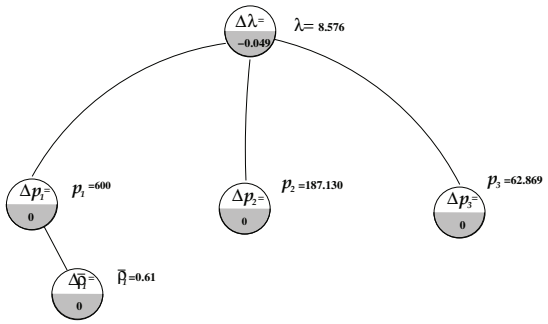

Fig. 12. Graph solved, fourth iteration

7. Results

For the sake of completeness, in this section the results previously discussed are shown via some convergence curves. Figure 13, shows the convergence for p_1, p_2, and p_3, Figure 14, shows the convergence for the dual variables i.e. λ, $\overline{\rho_1}$, and $\overline{\rho_3}$. Finally, Figure 15, shows the production functions behavior. It has to be highlighted the time a dual variable lives within the process, if ever. For instance, as shown in Figure 13, $\overline{\rho_1}$ lives from iteration 2 to the end of the process while $\overline{\rho_3}$ has a short period of life of only one iteration. Furthermore $\underline{\rho_1}$, $\underline{\rho_2}$, $\overline{\rho_2}$, and $\overline{\rho_3}$ did not appear in the solution process at all. This has implications of efficiency in time and memory which are not addressed by actual solvers at all.

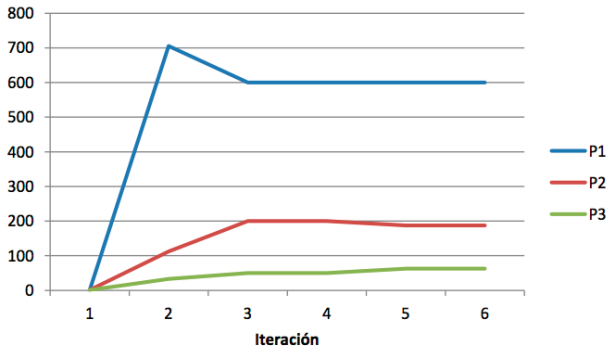

Fig. 13. p_1, p_2, and p_3 convergence.

8. Conclusions

In this document, a new elimination scheme has been presented. Under this scheme the graph reduction process is granted to be free of singularities as non zero pivots are always chosen. To this end, an example of how to use a graph reduction for the Economical Dispatch Problem has been presented. This has been done introducing a topological model for the Newton step using its graph-based model. This graph model is multilayer with only one kind of variables per layer: dual variables, decision variables, and dual bound variables for the variables. Starting with an initial graph topology, this topology evolves along the solution process in accordance with the actual solution of the problem. Even though there are methods that use the dispersity of the matrix, this is not done in a dynamic way, i.e. they only use the initial model but without the layer structure and the evolving characteristic. Therefore, it can be concluded that this representation and and its graph solution process exploits dispersity of the system at the maximum with its the respective computational savings.

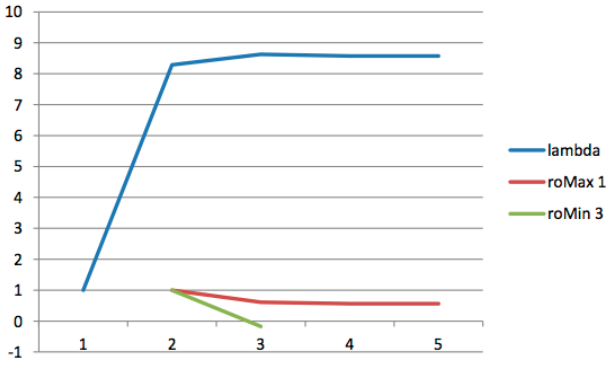

Fig. 14. $\lambda, \overline{\rho_1}$, and $\underline{\rho_3}$ convergence.

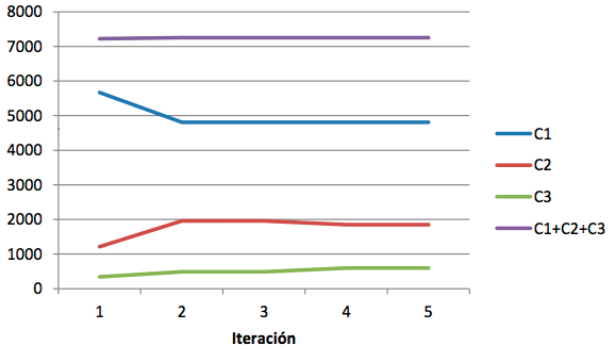

Fig. 15. $C_1(p_1), C_2(p_2)$, and $C_3(p_3)$ convergence.

References

[1] I. Griva, S. G. Nash, A. Soferr, Linear and nonlinear optimization, SIAM Society for Industrial and Applied Mathematics, 2009.
[2] J. Nocedal, S. Wright, Numerical Optimization, 2nd Edition, Springer Verlag, 2006.
[3] J. Cerda, D. De Roure, A graph-based economical dispatch model, in: H. R. Arabnia, G. A. Gravvanis, A. M. G. Solo (Eds.), Proceedings of WORLDCOMP 2010: Foundation of Computer Sciences, CSREA Press, Las Vegas, Nevada, USA, 2010, pp. 132–138.
[4] J. Cerda, C. P., F. J., Un enfoque grco para la solucin de problemas de programacin no lineal, in: Proceedings of Reunin de Otoo de Potencia, Electronica y Computacion: ROPEC 2011:, Morelia, Mich. MEX, 2011, pp. 132–138.
[5] J. R. Gilbert, C. Moler, R. Schreiber, Sparse matrices in matlab: design and implementation, SIAM J. Matrix Anal. Appl. 13 (1) (1992) 333–356. doi:http://dx.doi.org/10.1137/0613024.
[6] J. Cerda, D. De Roure, A decentralised dc optimal power flow, in: I. P. E. Society (Ed.), Proceedings of the Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, IEEE Power Engineering Society, Nanjing, China, 2008.
[7] A. G. Bakirtzis, P. N. Biskas, A decentralized solution of the dc-opf of interconnected power systems, IEEE Transactions on Power Systems 18 (3) (2003) 1007–1013.
[8] A. Bakirtzis, P. N. Biskas, N. Macheras, N. Pasialis, A decentralized implementation of dc optimal power flow on a network of computers, IEEE Transactions on Power Systems 20 (1) (2000) 25–33.
[9] P. Biskas, A. Bakirtzis, Decentralised opf of large multiarea power systems, IEE Power, Generation, Transmission and Distribution 153 (1).
[10] A. J. Conejo, J. A. Aguado, Multi-area coordinated decentralized dc optimal power flow, IEEE Transactions on Power Systems 13 (4) (1998) 1272–1278.
[11] B. Wollenberg, A. Wood, Power Generation, Operation and Control, IEEE Press, Wiley Inter-Science, 1996.
[12] A. Mas-Collel, A. Whinston, J. Green, Microeconomic Theory, Oxford Universiy Press, New York, 1995.