An Optimal Resource Assignment Problem in Smart Grid

Prakash Ranganathan
Department of Electrical Engineering
University of North Dakota
Grand Forks, ND, USA
e-mail:prakashranganathan@mail.und.edu

Kendall Nygard
Department of Computer Science
North Dakota State University
Fargo, ND, USA
e-mail:Kendall.Nygard@ndsu.edu

Abstract— **The paper describes a resource allocation problem in a smartgrid application formulated and solved as a binary integer programming model. For handling power outages from the main distribution circuit, the intelligent agents in the smart grid have to utilize and negotiate with DER (distributed energy resource) agents that act on behalf of the local generators in the grid, to negotiate power supply purchases to satisfy shortages. We develop a model that can assign these DERs optimally to available multiple regional utility areas or units (RUAs) that are experiencing power shortages. This is a resource assignment problem. The DERs in our model depict the behavior of power generated through windturbine, solar powered generation or other renewable power generation units and the region or area refers to a centralized distribution unit. The integer programming approach is called a capacity based Iterative Binary Integer Linear Programming (C-IBILP). All simulation results are carried out using the optimization tool box in MATLAB. Computation results exhibits very good performance for problem instances tested and validates the assumptions made.**

Keywords*— *C-IBILP; DER; RUA; BB.

I. INTRODUCTION

Dynamic real-time power systems often operate in continuously changing environments such as adverse weather conditions, sudden transformer failures, malfunctioning of a sub-system of a transmission or distribution network. These disruptions along with the complexity of our power network systems, cause the energy demand and loads of a power system to fluctuate, potentially resulting in widespread outages and huge price spikes. Data from the North American Electric Reliability Council (NERC) and analyses from the Electric Power Research Institute (EPRI) indicate that average outages from 1984 to the present time have affected nearly 700,000 customers per event annually [1]. Smaller outages occur much more frequently and affect tens to hundreds of thousands of customers every few weeks or months, while larger outages occur every two to nine years and affect millions. Although preventing these outages remain challenging, such changes (increases or decreases) in demand by consumers can often be offset by distributed energy resources (DERs), which are renewable resources like solar and wind based power to satisfy the shortages or reduce the outage levels. In our work, we consider the use of such DER-based standby mechanisms and formulate to support their issue. We apply an Iterative Binary Integer Linear Programming (IBILP) technique [2] to optimally assign DERs to a region based on criteria such as power levels, demands and preferences. A resource allocation for complex power system is robust with respect to variations in demand and fluctuations in power levels. The amount of additional power that DERs can generate and be effectively utilized in power network is a measure of robustness. Hence, we argue that a capacity based the Iterative Binary Integer Linear Programming (C-IBILP) model is inherently a robust resource allocation.

The structure of the remaining paper is as follows: In Section II, an overview and related work for the smart grid is discussed. In Section III, we present a general formulation of this DER assignment problem. In Section IV, we describe how to solve this problem optimally by using a branch-and-bound based (BB) algorithm with equality and inequality constraints. In Section V, we show the experimental results.

II. RELATED WORK

Mathematical programming has enjoyed a burgeoning presence in theoretical computer science, both as a framework for developing algorithms and, increasingly, as a bonafide model of computation whose limits are expressed in terms of sizes of formulations and integrality gaps of formulations [3, 4, 17]. Linear formulations are an appealing model of computation because both optimization and decision problems fit naturally into the framework, and both theoretically tractable and efficient practical algorithms exist for solving linear programs. For instance, state-of-the-art approaches to exactly solving large-scale instances of many NP hard problems rely on integer programming approaches that require the repeated solution of integer programs representing the problems [5]. The polynomial-time algorithms of [15] and other algorithms [13-16] cannot be applicable in this application due to high complexity and extensive run-times. Modification will be investigated in future work. We refer to a fundamental model for DER assignment as the Capacity based Iterative Binary Integer Linear Programming (C-IBILP) model. There has been little attention given to this type of approach in smart electrical grid analyses. To our knowledge, smart-grid problems of this type have not been solved for DER allocations using optimization models that perform optimal matching of

 ISBN: 978-1-61208-107-6

supply sources demand sites using prediction of generation and market-controlled consumption. Such optimization algorithms are comparable to hard unsolved problems in inference, optimization, and control [12].

III. DER-ASSIGNMENT PROBLEM

To illustrate our problem formulation, we assume that there are 7 areas (RUAs) and 6 DER units with the demand and preference levels shown in Fig 1. We define a Regional Utility Area (RUA) as the local distribution power utilities within the micro grid that distributes the power within their network for its loads [1]. For simplicity we name them Area 1, Area 2….Area 7 as illustrated in Figure 1. The power demand and the preferences in Fig.1 depict a demand driven DER assignment problem that also accommodates preference information. The parameters in the figure are for illustration purposes.

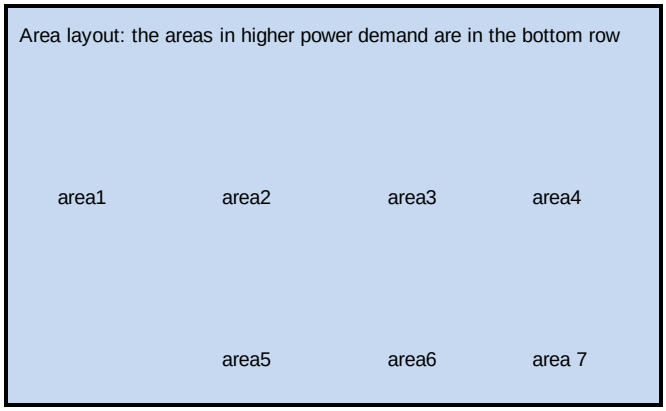

Figure 1. RUA layout

A simple allocation ‘text’ script on MATLAB would be as follows: text (0.1, .73, 'area1'); text (.35, .73, 'area2'); text (.60, .73, 'area3'); text (.82, .73, 'area4'); text (.35, .42, 'area5'); text (.60, .42, 'area6'); text (.82, .42, 'area 7').

For example, suppose our simulation study is charged with a need to optimally assign 6 DERs, DER1, DER 2, DER3, DER 4, DER 5, DER 6, to 7 regional utility areas (RUA) based on criteria such as capacity in power levels that these DER are able to generate and preferences in the area that these DER wish to operate. For simplicity in our optimization procedure, we also assume that each RUA can have no more than one DER, and each DER gets exactly one RUA. The DER can have a preference for the area that they wish to join, and their preferences are considered based on their capacity, i.e., the more power they have been able to generate the power (kW), the higher the capacity.

We weigh the preferences based on capacity power level of DER'S through a preference weight matrix (pwm), so that the more power that the DER's can generate, the more their preferences count.

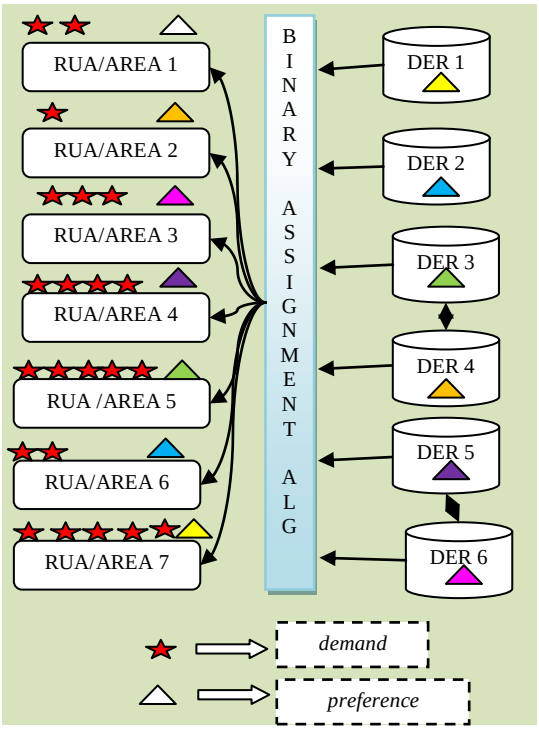

Figure 2. DER vs RUA assignment problem

Also, we impose multiple constraints such as some RUA have demand, some do not, and some demands are higher than others; DER 3 and DER 4 often work together, so we would like them to be no more than one RUA away, and DER 5 and DER 6 often work together so they also should be no more than one RUA away. Our approach to solve the assignment problem is to formulate it as a capacity based Iterative Binary Integer Linear Programming (C-IBILP) model and relax the integrality constraints. Our overall objective is to maximize the satisfaction of the preferences weighted by capacity which will allocate these DERs to their areas. This is done through a binary integer programming model by defining a linear objective function. Our algorithm uses a branch-and-bound procedure with linear programming bounds with 'minimum integer infeasibility' as the branch strategy and ‘depth first search for the node search strategy.

To develop our problem formulation, the first step is to choose what each element of our solution vector |x| represents. We use binary integer variables which represents the specific assignments of DERS to RUAs. If the DER is assigned to a RUA, the variable takes the value 1 and if not assigned, the variable takes the value 0. We consider the DER's in sequential order as DER 1, DER 2, DER 3, DER 4, DER 5, DER 6 and DER 7. The nth sequence of elements in vector |x| stores the assignment variables for DER n. In

 ISBN: 978-1-61208-107-6

our example |x (1)| to |x (7)| correspond to DER1 being assigned to Area 1, Area 2, etc., up to Area 7. In all, our vector |x| has 6 sequences of 7 elements each or 42 elements in all. Each sequence has a single binary variable set to 1, enforcing a multiple choice condition for each DER.

A. DER Capacities

We impose constraints based upon DER preference level in their area of operation driven by their capability to generate power. The concept is that the more power that a DER can generate, the higher preference level. For example consider the randomly set power levels given in kiloWatts (kW) below.

a. DER 1→9 kW
b. DER 2→10 kW
c. DER 3→5 kW
d. DER 4→3 kW
e. DER 5→1.5 kW and
f. DER 6→ 2 kW

We create a normalized weight vector based on capacity and also assume that certain DERs should be used in some preferred region or area, such as a DER with higher power generation capability being be used in large demand areas. This normalized weigh vector can be obtained on MATLAB as follows:

capacity = [9 10 5 3 1.5 2];
weight vector = capacity/sum (capacity);

```
>> capacity

capacity =

    9.0000   10.0000    5.0000    3.0000    1.5000    2.0000

>> weightvector

weightvector =

    0.2951    0.3279    0.1639    0.0984    0.0492    0.0656
```

B. RUA Preferences

We set up a preference weight matrix (pwm or prefmatrix) where the rows correspond to AREAS and the columns correspond to DERS. We assume that each DER will give values for each area so that the sum of all their choices, (i.e., their columns), sums to 100. A higher number means the DER prefers the area. We justify the use of the preference matrix by noting that limitations in algorithm scalability and data availability preclude a fully centralized solution to the problem of interest. Thus, decision making must be decentralized, and we accordingly divide the power network into many smaller RUAs, where the prefmatrix concept is applied to individual regions in the network.
An example of DER preferences is shown below:

DER1 = [0; 0; 0; 0; 10; 40; 50];
DER2 = [0; 0; 0; 0; 20; 40; 40];
DER3 = [0; 0; 0; 0; 30; 40; 30];
DER4 = [1; 3; 3; 3; 10; 40; 40];
DER5 = [3; 4; 1; 2; 10; 40; 40];
DER6 = [10; 10; 10; 10; 20; 20; 20];

The i^{th} element of a DER's preference vector is the value the i^{th} RUA. Thus, the combined preference matrix is expressed as 'prefmatrix':

prefmatrix = [DER1 DER2 DER3 DER4 DER5 DER6];

```
>> prefmatrix

prefmatrix =

     0     0     0     1     3    10
     0     0     0     3     4    10
     0     0     0     3     1    10
     0     0     0     3     2    10
    10    20    30    10    10    20
    40    40    40    40    40    20
    50    40    30    40    40    20
```

Case 1: We treat the above 'prefmatrix' arrangement as case 1 for analysis. We then weigh the preferences matrix by the |weightvector| to scale the columns by capacity. We also reshape this matrix as a vector in column-order so that it corresponds to our |x| vector. This is achieved in MATLAB script as follows:

PM = prefmatrix * diag (weightvector);

```
>> diag(weightvector)

ans =

    0.2951         0         0         0         0         0
         0    0.3279         0         0         0         0
         0         0    0.1639         0         0         0
         0         0         0    0.0984         0         0
         0         0         0         0    0.0492         0
         0         0         0         0         0    0.0656
```

c = PM (:);

```
>> PM

PM =

         0         0         0    0.0984    0.1475    0.6557
         0         0         0    0.2951    0.1967    0.6557
         0         0         0    0.2951    0.0492    0.6557
         0         0         0    0.2951    0.0984    0.6557
    2.9508    6.5574    4.9180    0.9836    0.4918    1.3115
   11.8033   13.1148    6.5574    3.9344    1.9672    1.3115
   14.7541   13.1148    4.9180    3.9344    1.9672    1.3115
```

 ISBN: 978-1-61208-107-6

IV. OBJECTIVE FUNCTION AND ADDED CONSTRAINTS

A. *Objective function*

Our objective is to maximize the total preferences measure weighted by capacity. This is a linear objective function max c'*x or equivalently min -c'*x with c being preferences of DER. We use the BINTPROG script [2] of MATLAB to run our model that is defined as:

$$\min_{x} f^{T}x : \begin{cases} A.x \le b, \\ Aeq.x = beq, \\ x : binary \end{cases}$$

where.,

f:Vector containing the coefficients of the linear objective function.
A: Matrix containing the coefficients of the linear inequality constraints $A \cdot x \le b$.
b: Vector corresponding to the right-hand side of the linear inequality constraints.
Aeq: Matrix containing the coefficients of the linear equality constraints $Aeq \cdot x = beq$.
beq: Vector containing the constants of the linear equality constraints.
*x0:*Initial point for the algorithm.
Options: Options structure containing options for the algorithm.
x: a binary integer solution
vector—that is, its entries can only take on the values 0 or 1.

B. *Constraints*

The first set of constraints requires that each DER is assigned to exactly one area. For example, since DER2 is the second DER, we enforce the condition that |sum(x (8:14)) =1|. We represent these linear constraints in an equality matrix Aeq and right hand side vector beq, where |Aeq*x = beq|, by building the appropriate matrices. The matrix |Aeq| consists of ones and zeros. For example, the second row of |Aeq| corresponds to DER2 getting exactly one RUA, so the row pattern is the following:

0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0

These conditions are implemented in MATLAB code as follows:

|Aeq (2, :)*x = 1| is equivalent to |sum(x (8:14)) = 1|.
numAREAS = 7;
numDERS = 6;
numDim = numAREAS * numDERS;
onesvector = ones (1, numAREAS);
Each row of Aeq corresponds to one DER.

Aeq =blkdiag (onesvector, onesvector, onesvector, onesvector, onesvector, onesvector);
beq = ones (numDERS, 1);
view the structure of Aeq, that is, where there are nonzeros (ones)
figure;

The second sets of constraints are inequalities. These constraints specify that each AREA has no more than one DER in it, i.e., each AREA has one DER in it, or is empty. We build the matrix |A| and the vector |b| such that |A*x <= b| to capture these constraints. Each row of |A| and |b| corresponds to a RUA and so row 1 corresponds to the DER assigned to RUA 1. In this case, the rows have the type of pattern shown below for row 1:

1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 ... 1 0 0 0 0 0 0

Each subsequent row is similar but is shifted (circularly) to the right by one spot by one position. For example, row 3 corresponds to RUA 3 and enforces that |A(3,:)*x <= 1|, so that AREA 3 cannot have more than one DER. Figures 3 and 4 illustrate equality and inequality constraints that are explained above.

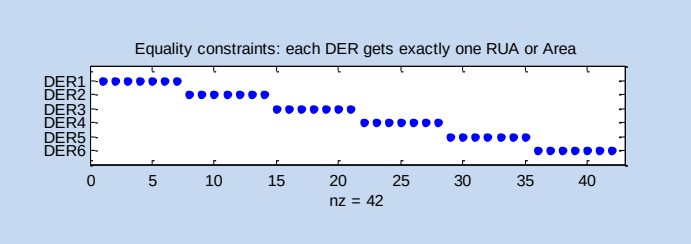

Figure 3. Equality Constraints

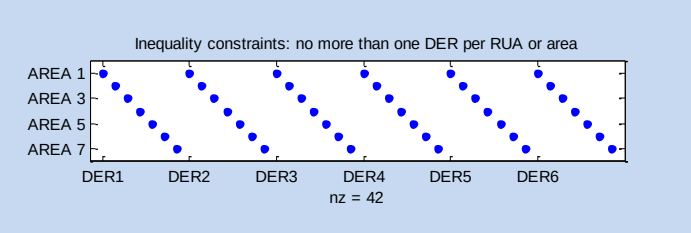

Figure 4. Inequality constraints

A = repmat(eye(numAREAS),1,numDERS);
b = ones(numAREAS,1);

where '*repmat*' represents the replicate and tile array. The elements of next set of constraints are also inequalities, so they are added to the matrix |A| and vector |b|, that already contain the inequalities from above. We wish to enforce that DER3 and DER4 are no more than one AREA (RUA) from each other, and similarly for DER5 and DER6. First the symmetric distance matrix for the RUAs is built using

 ISBN: 978-1-61208-107-6

physical locations and Manhattan (i.e., the "taxicab" metric).
D = zeros(numAREAS); // generates a 7 x 7 zero matrix
Setting up the top right half of the matrix
D(1,2:end) = [1 2 3 2 3 4];
D(2,3:end) = [1 2 1 2 3];
D(3,4:end) = [1 2 1 2];
D(4,5:end) = [3 2 1];
D(5,6:end) = [1 2];
D(6,end) = 1;
The lower left half is the same as the upper right D = triu(D)' + D; We find the RUA's that are more than one distance unit away.

```
>> D

D =

     0     1     2     3     2     3     4
     1     0     1     2     1     2     3
     2     1     0     1     2     1     2
     3     2     1     0     3     2     1
     2     1     2     3     0     1     2
     3     2     1     2     1     0     1
     4     3     2     1     2     1     0

>> D = triu(D)' + D;
>> D

D =

     0     1     2     3     2     3     4
     2     0     1     2     1     2     3
     4     2     0     1     2     1     2
     6     4     2     0     3     2     1
     4     2     4     6     0     1     2
     6     4     2     4     2     0     1
     8     6     4     2     4     2     0
```

[AREAA,AREAB] = find(D > 1);
numPairs = length(AREAA);
This finds |numPairs| pairs of AREAS. For example, if DER3 occupies one AREA in the pair, then DER4 cannot occupy the other AREA in the pair, else it would be more than one unit away in terms of AREA. The same condition holds for DER5 and DER6. This gives |2*numPairs| additional inequality constraints which we add to |A| and |b|. By adding rows to A, we accommodate these constraints as follows:
numrows = 2*numPairs + numAREAS;
A((numAREAS+1):numrows, 1:numDim) =
zeros(2*numPairs,numDim);
For each pair of AREAS in numPairs, for the |x(i)| that corresponds to DER 3 in |AREAA| and for the |x(j)| that corresponds to DER4 in |AREAB|, x(i) + x(j) <= 1 i.e., either DER3 or DER4 can occupy one of these AREAS, but not both.

C. Branch and Bound (BB) Strategy

The branch and bound algorithm is a well-known optimal solution method. Branch and bound (BB) algorithms are methods for solving non-convex global optimization problems [6-8]. They are exact (non-heuristic), in the sense that they calculate a provable upper and lower bounds on the globally optimal objective value and they terminate when all suboptimal feasible solutions have been eliminated. Branch and bound (BB) algorithms can be computationally slow. In the worst case they require effort that grows exponentially with problem size. We achieved fast convergence in our problems. We do note that due to total unimodularity of the basic A matrix, that a network-based customized linear programming solver could be used to provide the lower bounds very quickly in large problems. The BB algorithm is a well known algorithm in the research community [6-9]. An example run of the Branch and Bound algorithm is shown in Fig.5 followed by a snippet of MATLAB code showing the iterative output for each node displayed in the branch and bound algorithm. We let the BINTPROG choose the start point.

```
x0 = [];
f = -c;
options = optimset('Display','iter','NodeDisplayInterval',1);
[x,fval,exitflag,output] =
bintprog(f,A,b,Aeq,beq,x0,options);
fval
exitflag
output
```

To reduce the number of nodes explored, the time, or number of iterations taken, there are alternative options available. BINTPROG use the options to adjust the algorithm with differing node and branching variable strategies [2].

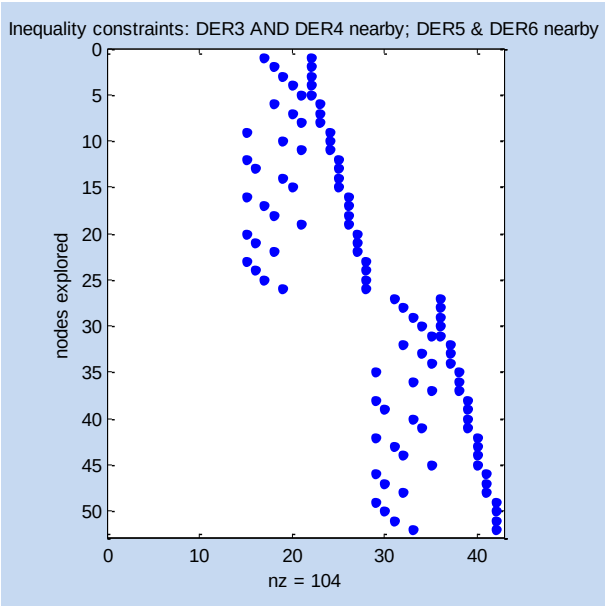

Figure 5. BB search algorithm with Inequality constraints

For example, the default branching strategy is |'maxinfeas'|, which chooses the variable with the maximum integer infeasibility for the next branch, that is, the variable whose value is closest to 0.5. Running the problem again with the branching strategy set to |'mininfeas'|, the variable the minimum integer infeasibility is chosen (that is, the variable whose value is closest to 0 or 1 but not equal to either).

For structuring the tree, depth-first and best-node search strategy are available. For example, in 'df', at each node in the search tree, if there is a child node one level down in the tree that has not already been explored, the algorithm chooses one such child to search. Otherwise, the algorithm moves to the node one level up in the tree and chooses a child node one level down from that node. In best-node (bn) strategy, the node with lowest bound on the objective function is the default. In our limited computational experience, convincing and acceptable results were quickly reached. For future work, we would plan to increase the scale of our test problems and investigate improved BB schemes.

V. RESULTS

The simulation is carried out in a MATLAB platform. The results show that the optimal value is reached after 163 iterations with 54 nodes participation in 1.22 seconds (case1) using the capacity based Iterative Binary Integer Linear Programming (C-IBILP) based branch and bound method which maximizes the satisfaction of the DER preferences weighted by its capacities.

The final output shown in Figure 6 presents the DER allocation with the RUA 1 or area 1 treated as empty for optimal assignment.

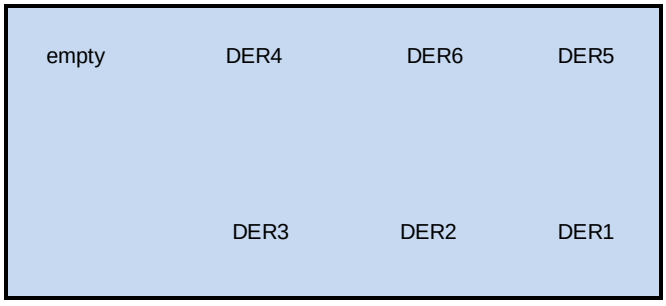

Figure 6. An optimal DER assignment solution for case 1

Case 2: If we change the preferences of DER's according to the matrix shown below, then the optimal solution is reached with 13 iterations, 1 node in 0.047 seconds with default node and branch strategies.

```
>> prefmatrix

prefmatrix =

     0     0     0     1     3    70
     0     0     0     3    40    10
     0     0     0     3     1    10
     0     0    70     3     2    10
    10    90    30    10    10    20
    40    40    40    60    40    20
    50    40    30    40    40    20
```

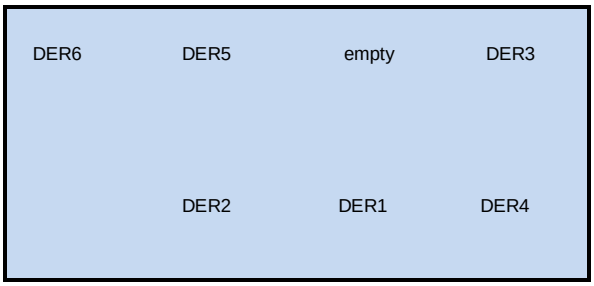

Figure 7. An optimal DER assignment solution to case 2

VI. CONCLUSION

The paper presents a resource assignment problem for smart grid application. The capacity based Iterative Binary Integer Linear Programming (C-IBILP) model is designed to specify an optimal allocation of distributed energy resources (DER's) during power outage periods to satisfy shortages. Computational results show that our C-IBILP algorithm exhibits very good performance for problem instances tested. A branch and bound algorithm for the smartgrid problem was described. It combines the extension results previously presented in the literature with new elements, such as a new lower bound that works by exploiting some properties connected with the ad- hoc branching rule we have developed. Computational results establish that the algorithm is very competitive. It greatly improves the results obtained by methods that have recently appeared in the literature. The limitation of our approach is that the method does not scale well for larger DERs. Our current efforts involve extending this assignment model to a more scalable assignment formulation for which larger numbers of DER's can to serve each RUA.

VII. REFERENCES

[1] M. Amin, "Balancing market priorities with security issues: interconnected system operations and control under the restructured electricity enterprise," IEEE Power and Energy Magazine, vol. 2, pp. 30-38, July/August 2004.

 ISBN: 978-1-61208-107-6

[2] Mathworks Inc, Binary Integer Programming (BINTPROG), http://www.mathworks.com, accessed 08.31.2010.
[3] S. Arora, B. Bollobas, L. Lorasz, and I. Tourlakis, "Proving integrality gaps without knowing the linear program," Theory of Computing, vol. 2, pp. 19-51, February, 2006.
[4] M. Alekhnovich, S. Arora, and I. Tourlakis, "Towards strong nonapproximability results in the Lovasz-Schrijver hierarchy." Proc. of the 37th Annual ACM Symp. on Theory of Computing, ACM, 2005, pp.294–303, doi:10.1145/1060590.1060634.
[5] C. Barnhart, E. Johnson and G. Nemhauser,1998. "Branch-and-Price: Column generation for solving huge integer programs," Operations Research vol. 46, 1998, pp.316–329.
[6] B. Balakrishnan, and S. Balemi. "Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems," Int. J. of Robust and Nonlinear Control, vol. 1, 1991, pp.295–317, October–December 1991.
[7] E. Lawler and D. Wood. "Branch-and-bound methods: A survey," Operations Research, vol 14, 1966, pp.699–719.
[8] R. Moore. Global optimization to prescribed accuracy. Computers and Mathematics with Applications, vol. 39, 1991, pp.25–39.
[9] L. Wolsey, Integer Programming, John Wiley & Sons, 1998.
[10] G. Nemhauser and L. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, 1988.
[11] F. Hillier, Frederick. and G. Lieberman., Introduction to Operations Research, McGraw-Hill, 2001.
[12] http://www.oe.energy.gov/smartgrid.htm, accessed 08.31.2010.
[13] P. Kouvelis and P. Yu. Robust Discrete Optimization and its Applications. Kluwer Academic Publishers, 1997.
[14] G. Kozina and V. Perepelista. "Interval spanning trees problem: solvability and computational complexity," Interval Computations, vol. 1, 1994, pp.42-50.
[15] J. Kruskal. "On the shortest spanning subtree of a graph and the travelling salesman problem," Proc. of the American Mathematical Society, 1956, pp.48-50.
[16] R. Montemanni and L. Gambardella. "A branch and bound algorithm for the robust spanning tree problem with interval data," European Journal of Operational Research, vol. 161, March, 2005, pp.771-779.
[17] S. Khot and N. Vishnoi, "The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics," Proc. of the 46th Symposium on Foundations of Computer Science, 2005, pp. 53-63, doi:10.1109/SFCS.2005.74.