Paper SAS1681-2015

Using SAS/OR® to Optimize the Layout of Wind Farm Turbines

Sherry Xu, Jinxin Yi, Steven Gardner, Joshua Griffin, and Baris Kacar, SAS Institute Inc.

ABSTRACT

A Chinese wind energy company designs several hundred wind farms each year. An important step in its design process is micrositing, in which it creates a layout of turbines for a wind farm. The amount of energy that a wind farm generates is affected by geographical factors (such as elevation of the farm), wind speed, and wind direction. The types of turbines and their positions relative to each other also play a critical role in energy production. Currently the company is using an open-source software package to help with its micrositing. As the size of wind farms increases and the pace of their construction speeds up, the open-source software is no longer able to support the design requirements. The company wants to work with a commercial software vendor that can help resolve scalability and performance issues. This paper describes the use of the OPTMODEL and OPTLSO procedures on the SAS® High-Performance Analytics infrastructure together with the FCMP procedure to model and solve this highly nonlinear optimization problem. Experimental results show that the proposed solution can meet the company's requirements for scalability and performance.

INTRODUCTION

The wind farm development department at a major Chinese wind energy company (referred to as "CEN" for confidentiality reasons) provides wind farm design solutions and services to its customers. It designs several hundred wind farms every year. As part of this process, designers need to find the optimal layout of wind turbines in order to maximize the total net power production. The layout of turbines on a wind farm directly affects profitability and return on investment.

Many factors impact the power production of a wind farm, including geographical factors (such as elevation of the farm) and wind distributions (such as wind speed distribution and wind direction distribution). The design of a wind farm must take these factors into account and find a turbine layout that can generate as much wind power as possible. Consider the typical wind farm, which has 22 turbines. At the current price of electricity, a 1% increase in power generation will yield around $1.5 million in additional revenue over the 20-year lifetime of the turbines.

PROBLEM STATEMENT

The wind energy industry has experienced a tremendous upsurge in recent years. CEN's requests for wind farm designs have increased dramatically, from 60 farms in 2012 to over 100 farms in 2013. The size of wind farms is also increasing, with more requests for larger farms and higher-volume turbines. The number of new middle-sized wind farms that contain 200–300 turbines has increased by over five times compared to previous years. The software currently supporting CEN's design requirement is OpenWind, which is an open-source and industry-standard software package for optimizing wind farm layout. However, OpenWind is no longer able to support CEN's design requirement, for two primary reasons. First, OpenWind's performance cannot keep pace with the increase in CEN's project size and volumes. It can take a considerable amount of time for OpenWind to produce its best solution, especially for large wind farms and high-volume turbines. Second, the heuristics that OpenWind uses to search for good solutions cannot provide a quantitative error bound on the quality of the returned solution (for more information, see Yilmaz 2012).

ASSUMPTIONS

The following assumptions are widely used in the research literature and in commercial software packages in the wind energy industry (Kusiak and Song 2010):

- The wind turbines that are considered for a wind farm are homogeneous; that is, they all have the same power curve function, $P = p(v)$, where v is the wind speed at the wind turbine rotor and P is the turbine power output.
- The number of wind turbines to be installed is fixed.
- The wind speed v follows a Weibull distribution, $p_v(v,k,c) = \frac{k}{c}(\frac{v}{c})^{k-1}e^{-(v/c)^k}$, where k is the shape parameter, c is the scale parameter, and $p_v(v,k,c)$ is the probability density function.

The wind can blow in any direction. The probability that it will blow in a particular direction and the parameters of Weibull distribution are provided at grid points on a wind farm. In Figure 1, the grid lines are 50 meters apart. Even though all input parameters are provided at these grid points, the turbines, which are depicted by blue dots, can be installed at any point on the farm.

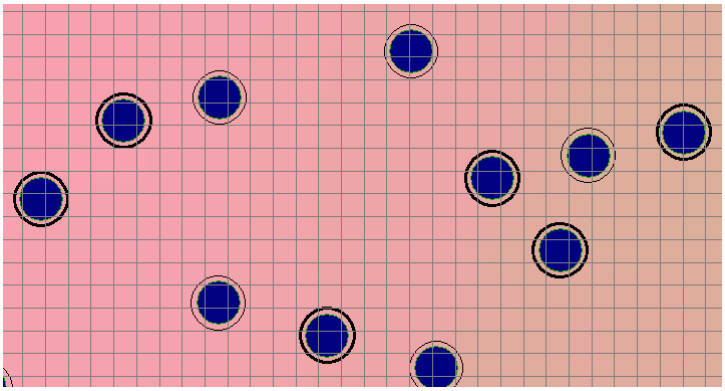

Figure 1. Grid Points on a Wind Farm

WAKE EFFECT BETWEEN TURBINES

The wake effect is a very important factor to consider in the design of wind farms. When two turbines line up in the same wind direction, the wind speed at the downstream turbine is reduced as wind passes through the upstream turbine, resulting in power loss. This is referred to as the wake effect or array loss. Figure 2 illustrates the basic concept of the wake effect.

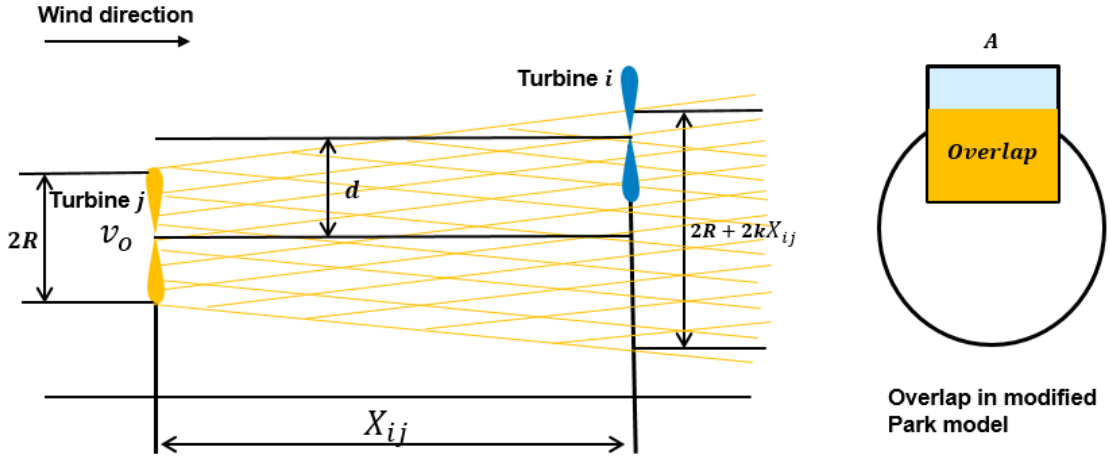

Figure 2. The Wake Effect: Turbine i Is in the Wake of Turbine j

Turbine j is upstream from turbine i with respect to the wind direction. The cone (depicted by the yellow grid) behind turbine j is called the wake. Because turbine i is in the wake of turbine j, the wind speed at i is reduced, and consequently power loss occurs at i.

This paper uses the modified Park model to calculate the wake effect. It assumes an immediate velocity deficit behind the rotor and the linear rate of expansion of the wake with distance downstream. The velocity deficit factor of turbine i due to turbine j's wake is calculated as

$$\tau_{ij} = \left(1 - \sqrt{1 - C_t}\right)\left(\frac{R}{R + kX_{ij}}\right)^2 \times \frac{overlap}{A}$$

where C_t is turbine j's thrust coefficient; X_{ij} is the distance between the two turbines, following the wind direction; R is the radius of the rotor; k is an empirically determined wake-decay constant; $overlap$ is the area of the yellow shaded area in Figure 2; and A is the area of the square, whose length of side is equal to the rotor diameter.

When a turbine is affected by the wakes of multiple turbines, its velocity deficit factor is calculated as the maximum deficit; that is,

$$\tau_i = \max_{j \in \Omega_i} \tau_{ij}$$

where Ω_i is the set of turbines that affect turbine i.

CONSTRAINT

Besides producing the wake effect, placing two wind turbines too close together also increases turbulence. Turbulence can decrease the performance of turbines downstream, because the fluctuations in wind speed and direction can be too fast for the downstream turbines to adapt to in order to maintain optimal status, causing greater wear and tear on turbine components. In this paper, a circular distance constraint is applied: the distance between two turbines should be at least three rotor diameters in any wind direction.

OBJECTIVE

The objective of wind farm layout optimization is to maximize the total net power generated by the turbines. The net power is equal to the gross power minus the power loss due to wake effects. If N is the number of turbines to be installed, then the expected total net power, E, is calculated as

$$E = \sum_{i=1}^{N} \int_0^{2\pi} \int_0^{\infty} \rho(\theta) p(v) \frac{k_i(\theta)}{c_i(\theta)} \left(\frac{v(\theta)}{c_i(\theta)}\right)^{k_i(\theta)-1} e^{-(v(\theta)/c_i(\theta))^{k_i(\theta)}} dv d\theta$$

where $\rho(\theta)$ is the probability density function of the wind direction and $p(v)$ is the power output when the speed of the wind that reaches the turbine rotor is equal to v.

Wind speed and direction are each discretized into small bins so that the integration can be approximated with discrete summations. The 360-degree wind direction is evenly divided into N_s sectors. The wind speed is also evenly divided into small bins, from v_{start} to v_{stop}, and each bin's width is v_{step}. Therefore, the preceding equation can be rewritten as

$$E = \sum_{i=1}^{N} \sum_{s=1}^{N_s} \sum_{v=v_{start}}^{v_{stop}} \rho_{s,i} \times p_i(v_{in}(v)) \times \int_{v-\frac{1}{2}v_{step}}^{v+\frac{1}{2}v_{step}} \frac{k_{i,s}}{c_{i,s}} \left(\frac{v}{c_{i,s}}\right)^{k_{i,s}-1} e^{-(v/c_{i,s})^{k_{i,s}}} dv$$

where $v_{in}(v)$ is the speed of the wind that reaches the rotor when the free-stream wind speed is equal to v, and $p_i(v_{in}(v))$ is the average power derived from the power curve over the interval $[v_{in}(v) - \frac{1}{2}v_{step}, v_{in}(v) + \frac{1}{2}v_{step}]$.

The preceding equation is coded in a function named EnergyCapture to approximate net power for any turbine layout. The following code segment defines the objective function EnergyCapture by using the FCMP procedure in Base SAS® 9.4 (for more information, see the *Base SAS 9.4 Procedures Guide*).

```
proc fcmp outlib=work.funcs.temp;

/*************************************************************************************************
 name: EnergyCapture
```

```
purpose: Calculate the total net power production given the turbine layout
argument:
1. layout – turbines layout
   layout[*,1] – x coordinate of turbine
   layout[*,2] – y coordinate of turbine
2. windrsc – 2-dimension array of wind resource data
   windrsc[*,1] – x coordinate
   windrsc[*,2] – y coordinate
   windrsc[*,3] to windrsc[*,5] – parameters of Weibull distribution and probability of sector 1
   windrsc[*,6] to windrsc[*,8] – parameters of Weibull distribution and probability of sector 2
   ...
   windrsc[*,36] to windrsc[*,38] – parameters of Weibull distribution and probability of sector 12
3. elevgrid – elevation at discrete grid points
4. ctcurve – array
   ctcurve[*,1] – wind speed
   ctcurve[*,2] – thrust coefficient at corresponding wind speed
5. powercurve – array
   powercurve[*,1] – wind speed
   powercurve[*,2] – power output at corresponding wind speed
6. vstart – energy capture starting from wind speed = vstart
7. vstop – energy capture ending at wind speed = vstop
8. hubheight – hub height
9. Z0 – approximate length
10.TtlNetPower – total net power in GWh
11.Netpower – net power of each turbine in MWh
*************************************************************************************************************/
    subroutine EnergyCapture(Layout[*,2],windrsc[*,38],elevgrid[*],ctcurve[*,2],powercurve[*,2],
                             Vstart,Vstop, hubheight,Z0,TtlNetPower,Netpower[*]);
        outargs TtlNetPower, Netpower; /*two output arguments*/
        nturbine=dim(layout);
        nSpeedStep = floor((Vstop-Vstart)/&speedstep.)+1;
        array order[1]/nosymbols;
        array ele[1]/nosymbols;
        array vv[1,1]/nosymbols;          /*incident wind speed*/
        array airdsty[1]/nosymbols;
        array tmpnet[1,1]/nosymbols;
        array olpratio[1,1]/nosymbols;
        array vlossratio[1]/nosymbols;
        call dynamic_array(order,nturbine);
        call dynamic_array(ele,nturbine);
        call dynamic_array(vv,nturbine,nSpeedStep);
        call dynamic_array(airdsty,nturbine);
        call dynamic_array(tmpnet,&sectors.,nturbine);
        call dynamic_array(olpratio,nturbine,nturbine);
        call dynamic_array(vlossratio,nturbine);
        TtlNetPower=0;
        /*************************************************************************************************
         Call predefined FCMP function GetElevation to calculate the elevation of each turbine
         rotor given each turbine's location
        *************************************************************************************************/
        do i=1 to nturbine;
            order[i]=i;
            ele[i]=GetElevation(layout[i,1],layout[i,2],elevgrid)+hubheight;
            Netpower[i]=0;
            airdsty[i]=&AirDensity.+ele[i]*(&DecayCoff.)/1000;
        end;
```

```
k=1/(2*log(hubheight/Z0));
/*************************************************************************************************
 For each wind sector, first rank all turbines by calling predefined subroutine Ordering.
 The turbines ranked lowest are in the wake of turbines ranked higher with respect to
 wind direction. For each turbine and each wind speed bin, calculate the velocity deficit
 and net power production.
*************************************************************************************************/
do s=1 to &Sectors.;
    call Ordering(s,layout,order);
    do n=1 to nturbine;
        do nspeed=1 to nSpeedStep;
        vv[n,nspeed]=vspeed;
        end;
    end;
    do i=1 to nturbine;
        xi=layout[order[i],1];
        yi=layout[order[i],2];
        eledown=ele[order[i]];
        diadown=&D.;
        do j=1 to i-1;
            xj=layout[order[j],1];
            yj=layout[order[j],2];
            eleup=ele[order[j]];
            diaup=&D.;
            lossratio=0;
            overlap=0;
            call CalSpeedLos(xj,yj,diaup,xi,yi,diadown,s,k,eleup,eledown,lossratio,overlap);
            olpratio[i,j]=overlap;
            vlossratio[j] = lossratio;
        end;
        WeibullA=0;WeibullK=0;Prob=0;
        call GetAKP(layout[order[i],1],layout[order[i],2],s,windrsc,WeibullA,WeibullK,Prob);
        do nspeed=1 to nSpeedStep;
            vspeed=Vstart+(nspeed-1)*&speedstep;
            tao=0;      /*speed loss*/
            do j=1 to i-1;
            Ctj=CT(ctcurve,vv[order[j],nspeed]);
            lossratio = (1-sqrt(1-max(0.02,min(0.999,Ctj))))*vlossratio[j];
            if lossratio>tao then tao=lossratio;
            end;
            vv[order[i],nspeed]=vspeed*(1-tao);
            avgnetpower=Poweroutput(powercurve,vv[order[i],nspeed],airdsty[order[i]]);
            vforward= vspeed + &speedstep.*0.5;   /*half step forward*/
            vbackward= vspeed - &speedstep.*0.5;  /*half step backward*/
            speedprob=cdf('WEIBULL',vforward,WeibullK,WeibullA)-
                        cdf('WEIBULL',vbackward,WeibullK,WeibullA);
            tmpnet[s,order[i]]=sum(tmpnet[s,order[i]],Prob*avgnetpower*speedprob*8.766);
        end;
    end;
end;
/*************************************************************************************************
 Sum up the total net power
*************************************************************************************************/
do i=1 to nturbine;
    do s=1 to &Sectors.;
         Netpower[i]=sum(Netpower[i],tmpnet[s,i]);
```

```
            end;
            TtlNetPower=sum(TtlNetPower,Netpower[i]);
        end;
        TtlNetPower=TtlNetPower/1000;     /*in GWH*/
    endsub;

run;
```

The approximation is compared to the energy capture result from using OpenWind software. The average difference is less than 0.5%.

SOLUTION APPROACH

The basic goal of wind farm layout optimization is to find the best layout for a wind farm in order to maximize the total net power of its wind turbines. This paper presents a hybrid approach to solving this problem by using Base SAS and SAS/OR® software.

The solution approach consists of four steps:

1. Identify potential sites on a grid with high gross power.
2. Use the MILP solver to determine the maximum number of turbines that can be installed.
3. Use the MILP solver to select 100 sites that generate the best net power.
4. Use the OPTLSO procedure to improve the solution in step 3.

The following sections provide details about these steps.

STEP 1: IDENTIFY THE SUBSET OF GRID POINTS WITH THE HIGHEST GROSS POWER

The wind farm under consideration is 12 × 21 kilometers in size. Figure 3 shows an image of the farm generated by OpenWind. The various colors and shades indicate the power density and elevation at a particular site on the farm. The redder (darker) the color, the higher the power density. It is easy to see that the upper part of the farm is an area with high power density.

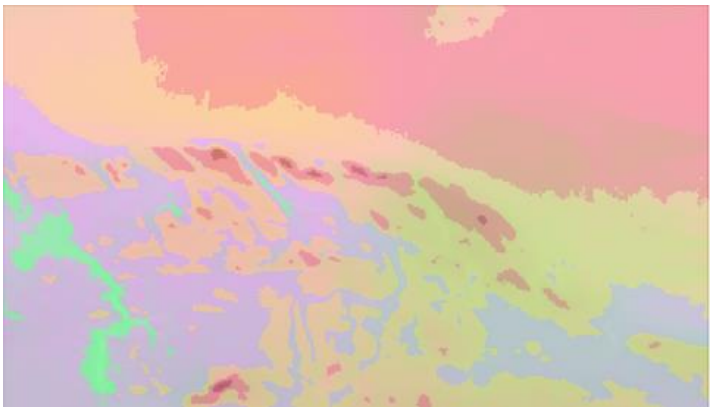

Figure 3. Wind Farm

The power density is provided at discrete points on a grid that covers the entire farm. Based on the power density and other geographical factors (such as wind speed and elevation), you can compute the gross power at each point. The gross power ranges from 3,600 to 5,200 megawatt-hours (MWh), with the 75th percentile at around 4,000 MWh. Because you need to consider only 100 sites on such a big wind farm, you can focus on sites whose gross power is at least 4,000 MWh. That gives you about 27,000 potential sites, which cover the high-power sites very well. (See the darker red areas in Figure 4.)

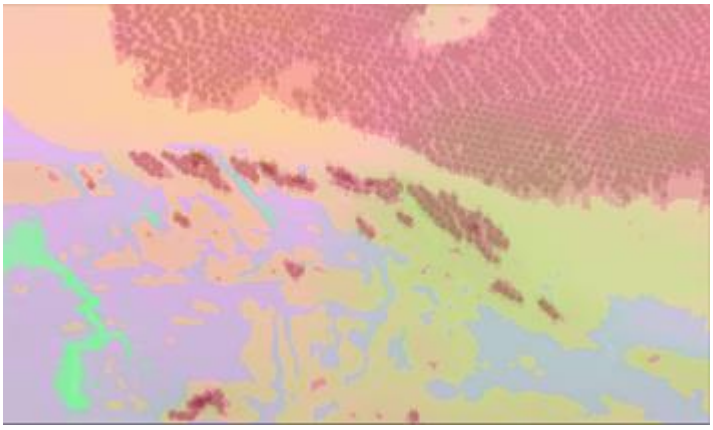

Figure 4. Potential Sites with High Gross Power

STEP 2: DETERMINE THE MAXIMUM NUMBER OF TURBINES THAT CAN BE INSTALLED

Given the 27,000 high-power sites, you can develop the following mathematical model to determine the largest number of turbines that can be installed at the wind farm while still maintaining the minimum separation constraint:

$$\max \sum_{i \in G} P_i x_i$$

Subject to

$$x_i + x_j \leq 1, \forall (i,j) \in A$$

$$x_i \in \{0,1\}, \forall i \in S$$

In this model, P_i is the gross power that can be generated at site i. Set S contains all 27,000 potential sites. Variable x_i is binary; it has a value of 1 when site i is selected, and 0 otherwise. Set A contains all pairs of sites that cannot be selected together because of minimum separation constraints; that is, these sites are too close to each other.

Because you have a huge number of potential sites, set A contains more than 850,000 pairs. To reduce the number of constraints, first you can find the maximal cliques from these pairs. A clique consists of two or more sites, only one of which can be selected because of the minimum separation constraints. In other words, all the sites in a clique are mutually too close to each other. For example, if you have three pairs,

$$x_1 + x_2 \leq 1$$

$$x_1 + x_3 \leq 1$$

$$x_2 + x_3 \leq 1$$

then sites 1, 2, and 3 form a clique and you can select only one of them. You can replace these three constraints with one constraint,

$$x_1 + x_2 + x_3 \leq 1$$

This single constraint is equivalent to the three constraints, and more importantly, it is a stronger constraint than all three of them combined. Stronger constraints are preferred because they speed up the optimization solver. The number of constraints is reduced by almost two-thirds after you replace set A with clique constraints. The following code segment defines and solves the preceding optimization problem by using the OPTMODEL procedure in SAS/OR 13.2 (for more information, see *SAS/OR 13.2 User's Guide: Mathematical Programming*).

```
proc optmodel;
   set node;
   number Power{node};
```

```
    num Xcord{node};
    num Ycord{node};
    /*************************************************************************
      Read gross power and coordinates of each potential site
    *************************************************************************/
    read data power into node=[node]
    Power=GrossPower Xcord=X_Coordinate Ycord=Y_Coordinate;
    /*************************************************************************
      Read pairs of sites that are too close to each other according to
      minimum separation constraints
    *************************************************************************/
    set <num,num> CONFLICTS;
    read data Conflict into CONFLICTS=[site_i site_j];
    /*************************************************************************
      Find maximal cliques by using NETWORK solver
    *************************************************************************/
    set <num,num> ID_NODE;
    solve with NETWORK / links=(include=CONFLICTS)
                         clique
                         out=(cliques=ID_NODE);
    set CLIQUES init {};
    set NODES_c {CLIQUES} init {};
    for {<c,i> in ID_NODE} do;
        CLIQUES    = CLIQUES    union {c};
        NODES_c[c] = NODES_c[c] union {i};
    end;
    /*************************************************************************
      Define the optimization model:
      X: decision variables
      obj: maximization of gross power
      constraints: clique constraints to satisfy the minimum separation
                   requirement
    *************************************************************************/
     var X{node} binary;
     max obj_totalGrossPower=sum{i in node} Power[i]*X[i];
     con Clique {c in CLIQUES}: sum {i in NODES_c[c]} X[i] <= 1;
    /*************************************************************************
      Solve the model with MILP solver
      Set relative gap at 1% and maximum time at 600 seconds to get
      a reasonably good solution
    *************************************************************************/
     solve with milp /relobjgap=0.01 maxtime=600;
     for{i in node} X[i]=round(X[i]);
    /*************************************************************************
      Write the solution to a data set
    *************************************************************************/
    create data feasible_sites from [node]={i in node:X[i]=1}
        x_coordinate=Xcord[i]  y_coordinate=Ycord[i]  GrossPower=Power[i];
quit;
```

The MILP solver finds a feasible solution within 600 seconds. The solution contains 1,040 sites.

STEP 3: SELECT 100 GRID POINTS FOR TURBINES THAT MAXIMIZE THE TOTAL NET POWER

Of the 1,040 sites that are selected in the previous step, 100 sites generate the maximum net power, that is, the gross power minus the power loss from the wake effect. To pick them out, you can build the following mathematical model:

$$\max \sum_{i \in S} P_i x_i - 2y_i$$

Subject to

$$y_i \geq D_{ij}(x_i + x_j - 1), \forall i \in S, \forall j \in S$$

$$\sum_{i \in S} x_i = 100$$

$$x_i \in \{0,1\}, \forall i \in S$$

$$y_i \geq 0, \forall i \in S$$

P_i, D_{ij}, and S are the input of this model; P_i is the gross power if a turbine is at site i; and D_{ij} denotes the power loss at site i that is caused by site j. Set S contains the 1,040 sites selected in the previous step.

The model contains two decision variables: x_i and y_i. The variable x_i indicates whether site i is selected or not: $x_i = 1$ if the site is selected, and $x_i = 0$ otherwise. The variable y_i is the maximum of all pairwise power losses that are generated by selected sites. The characteristic of variable y_i is guaranteed by the first constraint. Specifically, the constraint reduces to $y_i \geq D_{ij}$ when both x_i and x_j are equal to 1; otherwise it is equivalent to $y_i \geq 0$.

The original objective of the optimization is to maximize the total net power of the wind farm, which is a highly nonlinear function, as discussed in the "OBJECTIVE" section on page 3. In order to use the MILP solver, this step uses $\sum_i P_i x_i - 2y_i$ to approximate the total net power. The approximation is very accurate; its average error is around 1%.

It is important to point out that optimal sites are selected from set S in the model. Any sites that are selected from this set satisfy the minimum separation constraints. Therefore there's no need to apply the clique constraints defined in the previous step. The following code segment defines and solves the mathematical model by using the OPTMODEL procedure.

```
proc optmodel;
   set node;
   number Power{node};
   num Xcord{node};
   num Ycord{node};
   read data feasible_sites into node=[node]
     Power=GrossPower Xcord=x_coordinate Ycord=Y_coordinate;
   /*************************************************************************
     Read power loss at site i caused by site j
   *************************************************************************/
   set <num,num> PAIR ;
   num deficit{PAIR};
   read data deficit into PAIR=[site_i site_j] deficit=deficit;
   /*************************************************************************
     Define Optimization Model
     X: binary variable to indicate whether a site is selected or not
     MaxDeficit: maximum power loss at a site across all neighboring
                 sites that are selected
     totalPower: implicit variable, equal to the total gross power
     totalDeficit: implicit variable, equal to the total power loss
     con cover: 100 sites are to be selected
     con MaximumDeficit: when X[i]=X[j]=1, MaxDeficit[i]>=Deficit[i,j]
    *************************************************************************/
   var X{node} binary;
   var MaxDeficit{node}>=0;
   impvar totalPower = sum{i in node} Power[i]*X[i];
```

```
    impvar totalDeficit = sum{i in node} MaxDeficit[i];
    max obj_totalNetPower=totalPower-totalDeficit*2;
    con cover: sum{i in node} X[i]=&numTurbine;
    con MaximumDeficit {<i,j> in PAIR}: MaxDeficit[i]>=Deficit[i,j]*(X[i]+X[j]-1);
    /*****************************************************************
      Solve the model with MILP solver
    *****************************************************************/
    solve with milp / relobjgap=0.01 maxtime=3600;
    /*****************************************************************
      Write solution to an output data set
    *****************************************************************/
    for{i in node} X[i]=round(X[i]);
    create data layout from [node]={i in node:X[i]=1}
       x_coordinate=Xcord[i] Y_coordinate=Ycord[i] GrossPower=Power[i];
quit;
```

The MILP solver is able to find a solution that has a total net power of 425.676 gigawatt-hours (GWh). Figure 5 shows the 100 sites selected by the solver.

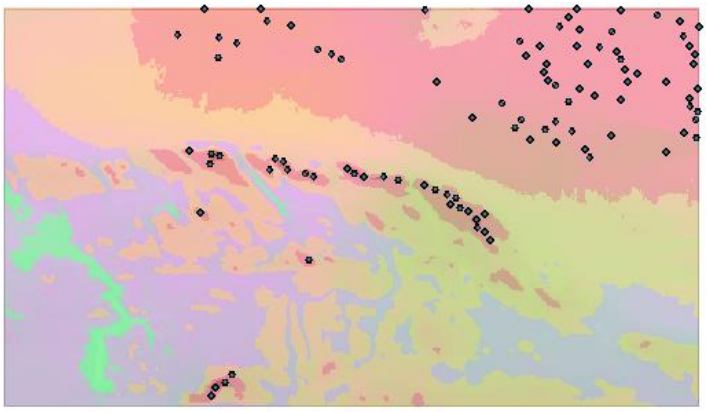

Figure 5. Best Turbine Layout Found by MILP Solver

STEP 4: USE THE OPTLSO PROCEDURE TO IMPROVE THE SOLUTION IN STEP 3

The OPTLSO procedure in SAS/OR performs optimization of general nonlinear functions that are defined by the FCMP procedure in Base SAS for both continuous and integer variables. These functions do not need to be expressed in analytic closed form, and they can be nonsmooth, discontinuous, and computationally expensive to evaluate. PROC OPTLSO uses projections to handle bound and linear constraints, and it uses a merit function to penalize nonlinear constraint violations. No attempt is made to exploit (or prescribe) structure in the treatment of objective and constraint functions.

PROC OPTLSO runs in either single-machine mode or distributed mode. All these features make PROC OPTLSO an obvious choice for refining the wind farm solution obtained by the MILP solver.

To enable PROC OPTLSO to refine the wind farm solution, the first step is to create the FCMP function that would calculate the net power generated from a particular turbine layout. This paper defines an FCMP function—EnergyCapture to compute the net power—in the "OBJECTIVE" section on page 4. PROC OPTLSO uses this FCMP function to define the objective $f(z)$ to help find a more optimal wind farm layout. As PROC OPTLSO generates new trial layout configurations, it passes the (x_i, y_i) location for wind turbine i, $i = 1,2,\ldots,k$, encoded as the vector $z = (x_1, x_2, \ldots, x_k, y_1, \ldots, y_k)$ to the FCMP function $f(z)$ for evaluation. In this case the number of variables is given by $n = 2k$. The FCMP function is evaluated, and the result that is returned to PROC OPTLSO is the net power generated by the specified wind turbine locations. The search algorithms in PROC OPTLSO then use this result to guide their search for future turbine layouts to consider for evaluation. The stated objective of total net power output is used as the measure in determining whether one layout is preferred to others that have been considered.

As in the objective case, the nonlinear constraint function $c(z)$ is realized as an FCMP function that returns a nonzero value if the circular boundary for any wind turbine is violated. Performance can be improved if the constraint violation measure is quantitative, in that given two infeasible layouts, one can be preferred because it has a smaller constraint violation. The quantitative violation measure is in contrast to a constraint depiction that simply returns 0 for feasible and 1 for infeasible, for example. To quantify the constraint violation, the sum of the distance between points of intersection between circles is returned as a single constraint. Computationally this $c(z)$ is much cheaper to evaluate than $f(z)$, so in practice $f(z)$ is not evaluated if an unacceptable constraint violation is detected. In practice, elliptical boundaries might be desired. The constraint FCMP function (for circular and optionally elliptical boundaries) for a case involving 100 wind turbines is as follows.

```
%macro ListXvars(Num);
  x1 %do i = 2 %to &Num; ,x&i  %end;
%mend;
%macro ListYvars(Num);
  y1 %do i = 2 %to &Num; ,y&i  %end;
%mend;
%let NW=100;
%let diam=80;
%let minorAxisDist=3.05;
%let majorAxisDist=3.05;
proc fcmp outlib=work.funcs.temp;
    function nlcElli(%ListXvars(&NW.),%ListYvars(&NW.));
        array x[&NW.] x1-x&NW.;
        array y[&NW.] y1-y&NW.;
        penalty = 0.0;
        num = 0;
        pi=arcos(0)*2;
        theta=135*pi/180;
        ct=cos(theta);
        st=sin(theta);
        a=&minorAxisDist.*&diam./2;
        b=&minorAxisDist.*&diam./2;
        do i=1 to &NW.;
            xi = x[i];
            yi = y[i];
            do j=1 to &NW.;
                xj = x[j];
                yj = y[j];
                d1=xj-xi;
                d2=yj-yi;
                c1=ct*d1  + st*d2;
                c2=-st*d1 + ct*d2;
                dist = (c1/(2.0*a))**2 + (c2/(2.0*b))**2;
                pi = max(1.01-dist,0);
                if ((i ne j) && (dist < 1.01)) then do;
                    num = num + 1;
                    penalty = penalty + pi*pi;
                    end;
                end;
            end;
          return (penalty);
      endsub;
run;
```

The wind farm problem can then be presented to PROC OPTLSO as the following optimization problem:

$$\min_{z} \quad f(z)$$

Subject to

$$z_\ell \leq z \leq z_u$$

$$c(z) = 0$$

PROC OPTLSO solves general nonlinear problems by simultaneously applying multiple instances of global and local search algorithms in parallel. This streamlines the process of needing to first apply a global algorithm in order to determine a good starting point to initialize a local algorithm. For example, if the problem is convex, a local algorithm should be sufficient, and the application of the global algorithm would create unnecessary overhead. If the problem instead has many local minima, then failing to run a global search algorithm first could result in an inferior solution. Rather than attempting to guess which paradigm is best, PROC OPTLSO simultaneously performs global and local searches while continuously sharing computational resources and function evaluations. The resulting run time and solution quality should be similar to what you get when you automatically select the best global and local search combination, given a suitable number of threads and processors. Moreover, because information is shared, the robustness of the hybrid approach can be increased over that of hybrid combinations that simply use the output of one algorithm to hot-start the second algorithm.

The default version of PROC OPTLSO uses genetic algorithms (GA) to perform the global search and generating set search (GSS) to perform local searches. PROC OPTLSO can also warm-start if good initial guesses are available. In wind farm layout optimization, the solution that is returned from a MILP approximation of the nonlinear optimization problem is used to warm-start. When you initialize with the MILP solution, GSS instances have the most impact in increasing the objective value. Thus the MILP solution provides an excellent guess for the region where the global solution is most likely to be found and makes a more in-depth global search unnecessary. Nonetheless, the GA does marginally improve solve time and solution quality, and thus a nontrivial population size is still specified for GA.

Because of PROC OPTLSO's parallel capabilities, you can use the procedure to evaluate a large number of trial wind turbine locations when searching for a better wind turbine layout. Experiments are conducted using a 12-node grid of computers, with up to 32 threads of execution per node. This gives PROC OPTLSO a total of 384 threads of execution for performing its parallel computations. In step 3, the MILP solver already found a solution with a total net power of 425.676 GWh. In the current step, the MILP solution is used to warm-start, and the maximum run time of PROC OPTLSO is set to 10 minutes. The benefit of running in a parallel environment is clearly demonstrated by Table 1, which shows the higher evaluation count when you have the same time limit.

Nodes/Threads/Total	$f(z_{final})$	Evaluations	Run Time
1/1/1	434.56 GWh	1,052	10 mins
1/32/32	435.68 GWh	6,176	10 mins
4/32/128	436.12 GWh	13,578	10 mins
8/32/256	436.38 GWh	21,204	10 mins
12/32/384	436.63 GWh	29,184	10 mins

Table 1. Performance Comparison in Parallel Environment

In general, exploiting structure when present is always warranted. The trade-off is the additional developer time needed to create a special-purpose solver that with each assumption narrows the probability that the solver can be applied in other scenarios. A benefit of using PROC OPTLSO is that with minimal developer time, you can create an effective prototype end-to-end solver for wind farm layout optimization problems, clearly demonstrating the benefits of warm-starting the process with a MILP presolve.

CONCLUSION

This paper presents a hybrid approach that uses Base SAS and SAS/OR software to address a wind farm layout optimization problem. Wake effects and circular constraints are considered in the optimization. Because the objective function is highly nonlinear, two MILP models are developed to maximize the total net power approximately. The first model considers the minimum distance constraint and finds both the maximum number of turbines to be installed and the corresponding turbine sites on the wind farm. The second model optimizes the total net power based on the sites found in the first MILP model. In the final step, the OPTLSO procedure is used to improve the solution returned from MILP models, and fully demonstrates the benefit of running in a parallel environment. With more developer time, you can streamline the process by replacing the general-purpose solvers used by PROC OPTLSO with equivalent local solvers designed specifically for wind farm layout optimization that could exploit the structure of the problem and underlying equations.

The wake effect model in this paper is a simple one, but you can consider and easily implement more complex wake effect models by using the optimization approach that the paper presents.

REFERENCES

AWS Truepower. (2010). “OpenWind.” http://awsopenwind.org/.

Kusiak, A., and Song, Z. (2010). “Design of Wind Farm Layout for Maximum Wind Energy Capture.” *Renewable Energy* 35:685–694.

SAS Institute Inc. (2014). *SAS/OR 13.2 User’s Guide: Local Search Optimization*. Cary, NC: SAS Institute Inc.

Yilmaz, E. (2012). “Benchmarking of Optimization Modules for Two Wind Farm Design Software Tools.” Department of Wind Energy, Gotland University, Sweden.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Sherry Xu
SAS Research and Development (Beijing) Co., Ltd.
Motorola Plaza, No.1 Wang Jing East Road
Chao Yang District, Beijing, China 100102
Phone: 86-10-8319-3465
Sherry.Xu@sas.com
www.sas.com