Computational Optimization and Applications, 33, 7–49, 2006

DOI: 10.1007/s10589-005-5957-4

A Local Relaxation Approach for the Siting of Electrical Substations

WALTER MURRAY* walter@stanford.edu
Department of Management Science and Engineering, Stanford University, Stanford, CA 94305-4026

UDAY V. SHANBHAG udaybag@uiuc.ed
Department of Mechanical and Industrial Engineering, Urbana, IL 61801

Received March 10, 2005; Revised June 27, 2005; Accepted July 6, 2005

Abstract. The siting and sizing of electrical substations on a rectangular electrical grid can be formulated as an integer programming problem with a quadratic objective and linear constraints. We propose a novel approach that is based on solving a sequence of **local** relaxations of the problem for a given number of substations. Two methods are discussed for determining a new location from the solution of the relaxed problem. Each leads to a sequence of strictly improving feasible integer solutions. The number of substations is then modified to seek a further reduction in cost. Lower bounds for the solution are also provided by solving a sequence of mixed-integer linear programs. Results are provided for a variety of uniform and Gaussian load distributions as well as some real examples from an electric utility. The results of GAMS/DICOPT, GAMS/SBB, GAMS/BARON and CPLEX applied to these problems are also reported. Our algorithm shows slow growth in computational effort with the number of integer variables.

1. Introduction

An important planning problem faced by electric utilities concerns where to place new substations to meet future electrical load. The planning objective is to minimize the cost of new substations and the electrical losses when operating the system. The constraints include Kirchhoff's law equations, voltage bounds and capacity bounds on the current at substation nodes. The electrical load supported by the grid and the linkage impedance are known. Kirchhoff's law leads to constraints (called the load-flow equations) that relate the voltages and the currents at the nodes through the **admittance** matrix. The problem may be formulated as a mixed-integer programming problem with a quadratic objective and linear constraints. By introducing a novel relaxation, we attempt to solve such a problem by solving a sequence of quadratic programs with continuous variables.

The paper is organized into five sections. Section 2 discusses some of the early research in this area. Section 3 gives a formulation of the problem, while in Section 4 our new algorithm is described. Section 5 describes different relaxations of the original problem that provide a means of obtaining an initial feasible solution and deriving lower bounds on cost. Results of applying our new algorithm on a variety of test and real world problems are given in section 6. To allow a comparison, Section 6 also gives the results of applying GAMS/DICOPT, GAMS/SBB and CPLEX to the smaller problems.

*To whom correspondence should be addressed.

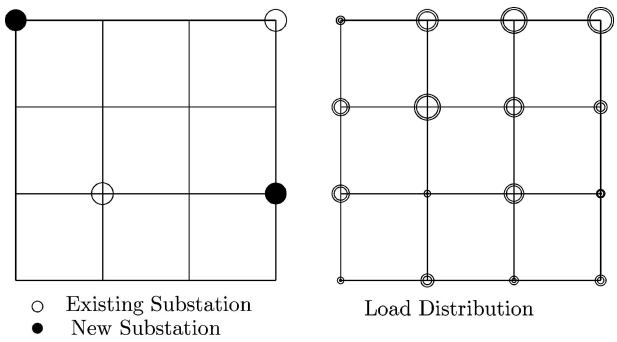

Figure 1. The substation siting optimization problem: The figure on the left shows the placement of existing or fixed substations and new substations. On the right, the load or current distribution on the grid is shown.

The last section lists the contributions and conclusions of this study and proposes some future work.

2. Past research

A useful introduction to the problem is provided in Willis et al. [29]. This reference discusses the problem briefly and focuses on some specific aspects such as feeder layout, cable sizes and load distribution across the feeders and substations (figure 1). A second paper by the same authors [30] discusses the the variety of methods that can be used to solve the problem. For example, transshipment algorithms convert the electrical network problem into one for which network flow algorithms can be applied.

2.1. Transshipment methods

One of the earliest references of an optimal planning approach toward the design of electrical networks is by Knight [15]. Based on a given set of geographical locations, a minimum cost network design was obtained using a network flow algorithm (Ahuja et al. [3] provide an excellent review of network flow algorithms). The objective function accounted for circuit length and apparent power and the constraints included security and switchgear limitations. Similar approaches are also seen in the following references [6, 8, 13].

In [28], Wall et al. use a transshipment algorithm to determine decisions conditional on a choice of integer variables. A branch-and-bound algorithm selects the optimal solution from the candidate solutions generated by the transshipment code. Capital costs are included for potential substations. Note that the essential problem formulation for the transshipment code is similar to that used by Adams and Laughton [2]. The same authors present more details on the embedded transshipment algorithm in [27].

Sharif et al. [25] give an algorithm that used both mixed integer linear programming (MIP) and spanning tree methods to estimate future expansion paths for a radial

distribution network. The general methodology comprises two steps: The first, the generation of spanning trees to connect the source to the demand nodes, and second, a MIP approach to ascertaining which spanning tree to use as the optimal solution.

2.2. *Non-transshipment algorithms*

El-Kady [8] modeled the planning of distribution substations and associated feeders over time. Marshall et al. [17] solved a distribution planning problem by a slight modification in the objective function. The costs in this particular methodology arose from cabling, switchgear and the incremental costs of building an extra circuit to a load node. This ensures that every load point has some security of supply; if one circuit goes down, another supplies it with power. The cost of losses was not included in this study. Provisions were made to ensure the security of the network. The solution methodology adopted is that of maximizing the Lagrangian dual using the NETOPT program.

Yahav and Oron [32] accounted for the nonlinear costs of losses and construction through the solution of a nonlinear program using the off-the-shelf solver GINO (it uses the generalized reduced gradient solver GRG2 by Lasdon and Waren [19]. New feeder routes and substation locations are the optimal decisions of the mathematical program. Amongst the more restrictive assumptions made are the linear relationship between losses and distance as well as the ignoring of voltage constraints.

2.3. *Mixed-integer nonlinear programs*

It is shown in Section 3 that the SSO problem may be formulated as the following mixed-integer quadratic program:

$$
\begin{array}{lll}
\text{MIQP} & \underset{x}{\text{minimize}} & \tfrac{1}{2}x^T Q x + e^T y \\
 & & Ax + By = b \\
 & \text{subject to} & x \geq 0 \\
 & & y_i \in \{0,1\} \text{ for all } i,
\end{array}
$$

where Q is a positive semidefinite matrix. Part of matrix A problem has network structure in that network is represented by arectangular grid. The MIQP is a special case of a more general class of problems called mixed-integer nonlinear programs or MINLPs. These problems fall under a larger class of hard optimization problems called NP-hard problems [20]. No convenient optimality conditions exist for such optimization problems and one has to find an optimal solution and then verify that the solution is indeed optimal. In fact, such verification may require an exponential number of iterations even if one were to start from a globally optimal solution.

One of the more common approaches to solving such MINLPs is to apply branch-and-bound methods. Such methods solve continuous relaxations of the MINLPs and

partition the feasible region to avoid fractional integer solutions. Leyffer's results [16] conclude that the computation times grow exponentially with the number of integer variables. The GAMS/SBB solver on the GAMS platform represents one implementation of a branch-and-bound method.

This approach contrasts sharply with a method that alternates between solving a mixed-integer linear programming problem and a nonlinear programming problem. The master problem may be defined by outer-approximation [7] or by using the ideas of generalized Benders decomposition [9]. Either case results in a mixed-integer linear program for the master problem. The solution to the master problem provides a new estimate of integer variables y, which are subsequently used in an NLP (nonlinear program) subproblem. The solution to the NLP subproblem provides a new estimate of the continuous variables x. The master problem provides a lower bound to the optimal solution and these bounds are used to test for termination. Leyffer [16] discusses an example that demonstrates that the outer-approximation algorithm may take an exponential number of iterations. Duran and Grossmann [7] forms the basis for the GAMS/DICOPT solver in GAMS.

Leyffer's thesis provides a useful literature review into alternative approaches to solving MINLPs, such as the use of linearizations [4], rounding [23], or even extensions of sequential quadratic programming (SQP) in the discrete domain [5]. A recent doctoral thesis by Ng [22] focused on homotopy methods to solve certain classes of MINLPs.

The GAMS/BARON solver is an efficient method for obtaining global optima of nonlinear integer programs. In [26], Tawarmalani and Sahinidis present a detailed development of the steps of this method and a commercial implementation is also available [24]. Given that the algorithm relies on branch-and-bound technology, one can expect difficulties when the problem has a large number of discrete variables. In [21], the authors compare the performance of over 1000 test problems with sizes as large as a thousand variables. Based on these tests, they find that GAMS/BARON is the most robust with GAMS/OQNLP close behind. We also report the performance of GAMS/BARON on some at the smaller test problems. The other solvers GAMS/OQNLP [1] and LINGO were not available to us and have not been compared.

Our interest is in large-scale MIQPs; the SSO problem even for the most modest problems sizes has at least a 100 (10×10) integers and may go as high as 10,000 (100×100) integers. As we show in the sections to follow, the problem has a distinct structure that forms the basis for our algorithm. We also provide a polynomial bound on the work required for a major iteration of the algorithm. Such a bound is useful for ensuring that there is slow growth in computational time with the number of integer variables.

3. Formulation of the problem

The core of the constraint system of the SSO problem is the load-flow system. While this shall be developed in greater detail in Section 5, the admittance matrix and some of its properties are described next. This leads into a description of some nonlinear integer programming formulations of the problem.

3.1. *Variables and parameters*

$m \times n$	Dimension of rectangular electrical grid
$\overline{I}, V, \ell$	Vector of nodal currents, voltages and loads[1]
Y, l_y	Admittance matrix and linkage admittance (same for all linkages)
$C_{\text{cap}}, C_{\text{loss}}$	Cost of a new substation and network losses
S_{cap}	Capacity of a new substation
N_{ss}, N_L	Set of nodes occupied (not occupied) by substations
V^{l}	Lower bound on voltage
n_{load}	Number of load-bearing nodes on electrical grid
$\min_{\text{ss}}, \max_{\text{ss}}$	Minimum and maxmimum number of substations
n^*_{ss}	Optimal number of substations
$load_{\text{S}}$	Total load borne by the electrical grid
ρ	Threshold parameter in CG (center of gravity) method
$x_{\text{cg}}, y_{\text{cg}}$	x and y coordinates of the center of gravity in the CG method
$N(i)$	Set of nodes neighboring node i. For instance in a 4×4 grid, the neighbors of node 1 are nodes 2 and 5 viz. $N(1) = \{2, 5\}$.
e, e_k	Column of ones and kth unit vector
Z_{ss}	The "null space" matrix associated with the positions of substations, both new and old. The matrix comprises rows of the identity matrix corresponding to the nodes at which no substations are located. The numbering is done row-wise. For example, if there were one substation in the center of a 3×3 grid, then Z_{ss} would be the identity matrix $I_9 \in \Re^{9\times 9}$ with row 5 missing
Y_{ss}	The "range space" matrix associated with the positions of substations, both new and old. The matrix comprises rows of the identity matrix corresponding to the nodes at which substations are located. For the 3×3 grid with one substation in the center, $Y_{ss} = e_5^T$

3.2. *The admittance matrix*

Assuming a DC load-flow model, the current vector $\overline{I}$ is linked to the voltage vector V through the linear equation

$$\overline{I} = YV, \tag{3.1}$$

where Y is the admittance matrix. Each node interior to the grid has linkages with nodes on four sides while nodes on the boundaries have correspondingly fewer connections. The admittance matrix captures the relationship of a node with its neighbors. Therefore, each row of (3.1) can be thought of as a current-balance relationship in which the current at a particular node is equal to the sum of the current flows along the links connected to it. It follows that the admittance matrix Y can be represented as a symmetric block tridiagonal matrix

$$Y = \begin{pmatrix} A_1 & B_1 & & & \\ B_1 & A_2 & B_2 & & \\ & \ddots & \ddots & \ddots & \\ & & B_{n-2} & A_{n-1} & B_{n-1} \\ & & & B_{n-1} & A_n \end{pmatrix},$$

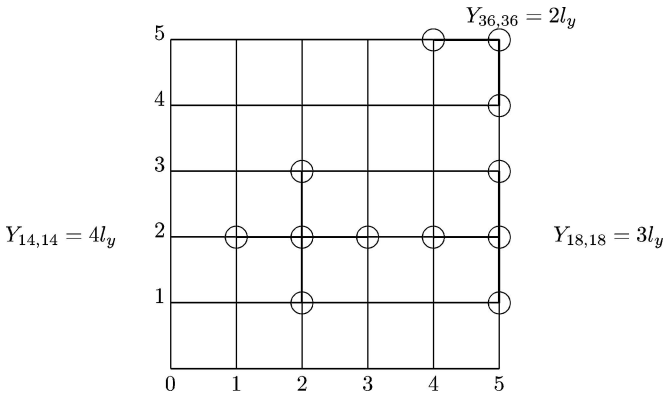

Figure 2. Constructing the admittance matrix *Y*. The captions show how the diagonal elements of the *Y* matrix are dependent on the number of linkages to the corresponding nodes.

where the blocks A_j, $j = 1, \ldots, n$ and B_j, $j = 1, \ldots, n-1$ are tridiagonal and diagonal in structure, respectively. The diagonal element Y_{ii} of any row is equal to the negative sum of the off-diagonal elements. In particular

$$Y_{ii} = -\sum_{j \in N(i)} Y_{ij},$$

where $N(i)$ represents the coordinates of the neighbors of i. Therefore, the row elements of the admittance matrix sum up to 0. If l_y represents the admittance of any link of a regular electrical grid, the admittance matrix blocks are defined as follows:

$$A_1 \text{ and } A_n = l_y \begin{pmatrix} 2 & -1 & & & & \\ -1 & 3 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 3 & -1 & \\ & & & - & 2 & \end{pmatrix},$$

$$A_2, \ldots, A_{n-1} = l_y \begin{pmatrix} 3 & -1 & & & & \\ -1 & 4 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 4 & -1 & \\ & & & -1 & 3 & \end{pmatrix}$$

$$\text{and } B_1, \ldots, B_{n-1} = l_y \begin{pmatrix} -1 & & & & & \\ & -1 & & & & \\ & & \ddots & & & \\ & & & -1 & & \\ & & & & -1 & \end{pmatrix}.$$

Lemma 3.1. *The admittance matrix Y associated with an $m \times n$ electrical grid is symmetric and positive semidefinite.*

Proof: By Gershgorin's theorem, $\lambda(Y)$ lie in at least one of the discs with centers Y_{ii} and radii $\sum_{j \neq i} |Y_{ij}|$. Since $Y_{ii} = \sum_{j \neq i} |Y_{ij}|, \forall i$, none of the discs extend into the negative real axis which implies that the eigenvalues are nonnegative. Thus Y is positive semidefinite. Symmetry of Y follows trivially from its definition. □

One of the important properties of the Y matrix is its rank deficiency as the following lemma shows.

Lemma 3.2. *The column vector of ones is an eigenvector associated with the zero eigenvalue. In other words, we have $Ye = 0$.*

Proof: This follows directly from the construction of the Y matrix. □

The existence of a zero eigenvalue allows one the voltages to be modified by multiples of e without impacting the load-flow solution. This can be mathematically stated as

$$\overline{I} = YV = Y(V + ke) = Y\tilde{V},$$

where k is some scalar.

We are now ready to prove the main result that the admittance matrix of an $m \times n$ grid, $Y_{m \times n}$, has a rank deficiency of 1 and shall use a constructive argument to do so. We first prove that $Y_{1 \times n}$ has a rank deficiency of 1. We then show that an augmentation of the grid with nodes and linkages does have not alter the number of zero eigenvalues.

Lemma 3.3. *The admittance matrix of a grid of dimension $1 \times n$ has a rank deficiency of 1.*

Proof: The admittance matrix of a $1 \times n$ grid is a $n \times n$ matrix given by:

$$\begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}.$$

The LDL^{T} factorization of this matrix is given by:

$$\begin{pmatrix} 1 & & & & \\ -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ & & & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0 \end{pmatrix} \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \\ & & & & 1 \end{pmatrix}.$$

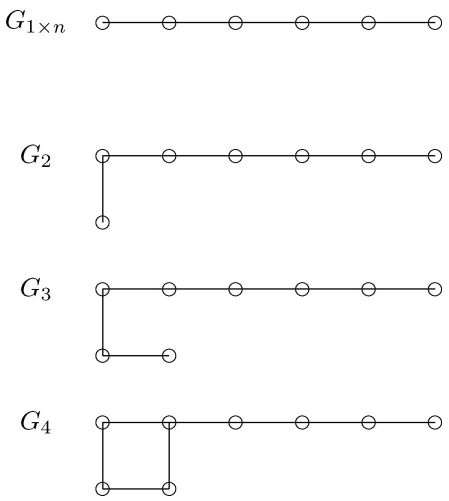

Figure 3. This figure shows the addition of 3 linkages and 2 nodes to the $1 \times n$ grid. The addition occurs in three steps with the corresponding admittance matrices being called $Y_{1\times n}$, Y_2, Y_3 and Y_4. We prove that each step maintains the rank deficiency as 1.

Therefore, the admittance matrix has exactly one zero eigenvalue. □

Lemma 3.4. *Let $Y_{1\times n}$ represent the admittance matrix associated with a grid of dimension $1 \times n$. This grid may be modified as shown in figure 3. Based on figure 3, the corresponding admittance matrices $Y_{1\times n}$, Y_2, Y_3 and Y_4 have one zero eigenvalue.*

Proof:

1. $Y_{1\times n} \to Y_2$: The addition of a linkage and a node to the $1 \times n$ grid results in a change in the admittance matrix. The new admittance matrix may be written as

$$Y_2 = \bar{Y}_{1\times n} + D,$$

where

$$\bar{Y}_{1\times n} = \begin{pmatrix} Y_{1\times n} & 0_{n,1} \\ 0_{1,n} & 0 \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} e_1 e_1^T & -e_1 \\ -e_1^T & 1 \end{pmatrix}.$$

Define

$$L_n \equiv \begin{pmatrix} I_n & \\ e_1^T & 1 \end{pmatrix}$$

We have

$$\begin{aligned} L_n^T Y_2 L_n &= \begin{pmatrix} I_n & e_1 \\ & 1 \end{pmatrix} \begin{pmatrix} Y_{1\times n} + e_1 e_1^T & -e_1 \\ -e_1^T & 1 \end{pmatrix} \begin{pmatrix} I_n & \\ e_1^T & 1 \end{pmatrix} \\ &= \begin{pmatrix} Y_{1\times n} + e_1 e_1^T - e_1 e_1^T & \\ & 1 \end{pmatrix} \\ &= \begin{pmatrix} Y_{1\times n} & \\ & 1 \end{pmatrix}. \end{aligned}$$

The nonsingularity of L_n implies that Y_2 has one zero eigenvalue.

2. $Y_2 \to Y_3$: The admittance matrix Y_3 may be written as

$$Y_3 = \bar{Y}_2 + D,$$

where

$$\bar{Y}_2 = \begin{pmatrix} Y_2 & \\ & 0 \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} e_{n+1} e_{n+1}^T & -e_{n+1} \\ -e_{n+1}^T & 1 \end{pmatrix}.$$

Define

$$L_{n+1} \equiv \begin{pmatrix} I_{n+1} & \\ e_{n+1}^T & 1 \end{pmatrix}.$$

We have

$$\begin{aligned} L_{n+1}^T Y_3 L_{n+1} &= \begin{pmatrix} I_{n+1} & e_{n+1} \\ & 1 \end{pmatrix} \begin{pmatrix} Y_2 + e_{n+1} e_{n+1}^T & -e_{n+1} \\ -e_{n+1}^T & 1 \end{pmatrix} \begin{pmatrix} I_{n+1} & \\ e_{n+1}^T & 1 \end{pmatrix} \\ &= \begin{pmatrix} Y_2 + e_{n+1} e_{n+1}^T - e_{n+1} e_{n+1}^T & \\ & 1 \end{pmatrix} \\ &= \begin{pmatrix} Y_2 & \\ & 1 \end{pmatrix}, \end{aligned}$$

which implies that Y_3 has one zero eigenvalue.

3. $Y_3 \to Y_4$: Unlike the previous two steps, the grid merely requires an addition of a linkage without adding any further nodes. The admittance matrix Y_4 is then given by

$$Y_4 = Y_3 + D,$$

where

$$D = \begin{pmatrix} e_2 e_2^T & -e_2 \\ -e_2^T & 1 \end{pmatrix} = \begin{pmatrix} e_2 \\ -1 \end{pmatrix} \begin{pmatrix} e_2 \\ -1 \end{pmatrix}^T = w w^T.$$

The eigenvalues of

$$\lambda_i(Y_4) \in [\lambda_i(Y_3), \lambda_{i+1}(Y_3)], \quad i = 1, \cdots, n+1,$$

where the eigenvalues of Y_3 are ordered such that

$$0 = \lambda_1(Y_3) < \lambda_2(Y_3) \leq \cdots \leq \lambda_{n+2}(Y_3).$$

This follows from Theorem 8.1.8 in [12]. But, we know that Y_4 has at least one zero eigenvalue from Lemma 3.2. It follows that Y_4 has exactly one zero eigenvalue.

□

We now state the main result of this section:

Theorem 3.1. *The admittance matrix Y of an $m \times n$ grid has exactly one zero eigenvalue, with corresponding eigenvector e.*

Proof: The admittance matrix associated with the $1 \times n$ grid is shown to have exactly one zero eigenvalue (Lemma 3.3). By using Lemma 3.4 repeatedly, we may construct an admittance matrix associated with an $m \times n$ grid. The number of zero eigenvalues is maintained through these transitions giving us the required result. □

A consequence of this theorem is that we can replace the system

$$\overline{I} - YV = 0$$

by one in which the last row is omitted. The resulting specification of the load-flow system ignores the current in the *mn*th node. This is specified by the conservation constraint proved in Lemma 3.5:

$$\sum_{i=1}^{mn} \overline{I}_i = 0.$$

For purposes of clarity, our formulation shall maintain the form of the load-flow system as $\overline{I} = YV$.

Lemma 3.5. *The sum of nodal currents $\sum_{i=1}^{mn} \overline{I}_i = 0$.*

Proof: Consider the load-flow system:

$$(I \quad -Y)\begin{pmatrix} \overline{I} \\ V \end{pmatrix} = 0.$$

Multiplying both sides by e^T and using the results from the previous two lemma, we obtain the result as follows:

$$
\begin{aligned}
e^T(\,I \quad -Y\,)\begin{pmatrix}\overline{I}\\ V\end{pmatrix} &= 0\\
(\,e^T \quad -e^TY\,)\begin{pmatrix}\overline{I}\\ V\end{pmatrix} &= 0\\
(\,e^T \quad -e^TY^T\,)\begin{pmatrix}\overline{I}\\ V\end{pmatrix} &= 0\\
(\,e^T \quad 0\,)\begin{pmatrix}\overline{I}\\ V\end{pmatrix} &= 0\\
e^T\overline{I} &= 0.
\end{aligned}
$$

□

3.3. Nonlinear integer programming formulation

This section shall formulate the nonlinear integer program of interest. The objective function is the sum of the capital costs of new substations and the cost of electrical losses. The latter is given by $C_{\text{loss}}V^TYV$, where V^TYV represents the electrical losses in the system and C_{loss} represents the cost per unit of losses. The constraints on voltage and current at a particular node are contingent on whether the node has a substation placed on it or not.

We define two sets of nodes: N_{ss} is the set of nodes housing new and old substations while N_{L} contains all other nodes. These sets are modified through the algorithm. Moreover, the dependency of the constraints on whether $i \in N_{\text{ss}}$ or not can be stated in terms of binary decision variables. Let y_i be such that if $i \in N_{\text{ss}}$ then $y_{\text{i}} = 1$, otherwise $y_{\text{i}} = 0$. Clearly the dimension of y is equal to the number of nodes. Such a specification allows us to specify the cost of new substations as $C_{\text{cap}}e^Ty$ where C_{cap} and e^Ty represent the unit cost of a new substation and the number of new substations, respectively. The constraint system comprises the following:

1. The load-flow system $\overline{I} = YV$.
2. The bounds on nodal voltages.
3. The specification that a nodal current may be either a variable or a fixed quantity based on whether the node houses a substation or not.

It follows that the problem may be stated as

SSO_1	$\underset{V,\overline{I},y}{\text{minimize}}$	$C_{\text{cap}}e^Ty + C_{\text{loss}}V^TYV$
		$\overline{I} - YV = 0$
		$V^l \le V_i \le 1$
	subject to	$\overline{I_i} \le S_{\text{cap}}, y_i = 1$
		$\overline{I_i} = \ell_i, y_i = 0,$

where S_{cap} is the capacity of a new substation. The bounds on V and $\overline{I}$ can be stated in terms of the substation installation decisions y. Such a formulation may be rewritten as follows:

$$
\begin{array}{lll}
\text{SSO} & \underset{V,\overline{I},y}{\text{minimize}} & C_{\text{cap}} e^T y + C_{\text{loss}} V^T Y V \\
 & & \overline{I} - YV = 0 \\
 & \text{subject to} & V^l \le V \le e \\
 & & (1-y_i)\ell \le \overline{I}_i \le (1-y_i)\ell_i + y_i S_{\text{cap}}, \quad \text{for all } i, \\
 & & y_i \in \{0,1\}, \quad \text{for all } i.
\end{array}
$$

(SSO) is a mixed-integer programming problem with a quadratic objective. Nonlinear integer programming solvers such as GAMS/SBB, GAMS/DICOPT, GAMS/BARON and CPLEX/MIQP may be used to solve (SSO). However, these solvers slow down significantly when the number of binary variables exceeds 100 (10×10) as we shall describe in section 6.

Our new algorithm can solve problems of the order of (50×50) resulting in 2500 binary variables with a MATLAB implementation.

4. The substation siting optimization algorithm

The approach adopted in our new method is to mimic linesearch and trust-region methods for solving smooth problems in continuous variables. Such algorithms use a local approximation to the problem to determine an improved estimate of the solution. In the case of the substation location problem, the search direction or step translates into a move to a neighboring node (or possibly remaining at the current node) for **all** the substations.

Although this is a reduced task from that of considering all possible moves, there are still 9^K possible moves, given K new substations, where K is typically 50–150. Limiting the number of substations that can move is clearly suboptimal since their relative positions are important and may need to be maintained. For the possible number of choices to be tractable, the limit on the number of substations allowed to move would need to be very small. In some discrete problems, once the discrete variables have been specified, the evaluation of the objective is trivial. Here, it is necessary to solve a large scale quadratic program (QP) to evaluate a trial placement. This clearly limits the number of trial moves. Simply moving one or a small number of substations is also unlikely to succeed.

In our new algorithm, we describe how a simultaneous move for all substations is obtained from a **local** relaxation of the problem. The following is a brief summary of the steps given together with their correspondence to the steps of nonlinear programs for continuous variables.

- *Determining an initial feasible solution*. Optimization algorithms for continuous variables with linear constraints typically start from an initial feasible point and maintain feasibility in all iterates. An advantage of such an approach is that the objective is

then a measure of merit and the iterates provide a strictly monotonically improving sequence of iterates. In general, determining an initial feasible point to a discrete problem is hard. However, we provide a method to find such a solution for this problem.

- *Determining a descent step*. For general nonlinear programs, the direction of descent is obtained by solving a system of equations or a quadratic program (QP). Our algorithm requires the solution of a local relaxation of the problem that results in a QP.
- *Determining the step length*. The linesearch algorithm is replaced by one of two possible methods. Each method uses the information from the local relaxation to determine a new set of locations with lower objective.
- *Termination criteria*. While the optimality conditions could be one test of termination for a nonlinear program, no such conditions exist for a nonlinear program with integer variables. Thus the termination is based not on having reached an "optimum" but on not being able to find a better position.

4.1. An initial feasible solution

Determining an initial feasible solution is not difficult since we could saturate the grid with substations. However, since the number of nodes is substantially larger than the optimal number of substations, such an approach would result in a very poor initial point.

An initial feasible solution is obtained by first solving a global relaxation of the integer variables in the problem, coupled with a modified objective. The relaxed discrete variables are then rounded up to give an integer solution. Note that by rounding up, we are increasing the resources and hence will not violate constraints on voltage or capacity. The nonzero components of y specify where to place new substations while the number of nonzero components specifies the number of new substations. Unless steps are taken to prevent it, there is a danger in this approach that the number of substations in the initial feasible solution may be very large.

The modified objective function replaces the capital cost by a smooth function $C(y; \mu)$ which is defined as follows

$$C(y;\mu) = \sum_i f(y_i, \mu) = \sum_i C_{\text{cap}}(1 - e^{-\mu y_i})$$

The variation of f with y_i is shown in figure 4. It can be seen that if the capital cost associated with a new substation forms a step function, the function f is a lower bound to this step function. In fact, as

$$\text{as } \mu \to \infty,\ f(y_i, \mu) \to \begin{cases} C_{\text{cap}}, & y_i > 0 \\ 0, & y_i = 0 \end{cases}.$$

It may be noted that such a formulation also admits the convex relaxation in which $f(y_i) = C_{\text{cap}} y_i$, $y_i \in [0, 1]$. This may be achieved by allowing μ to be a function of y_i

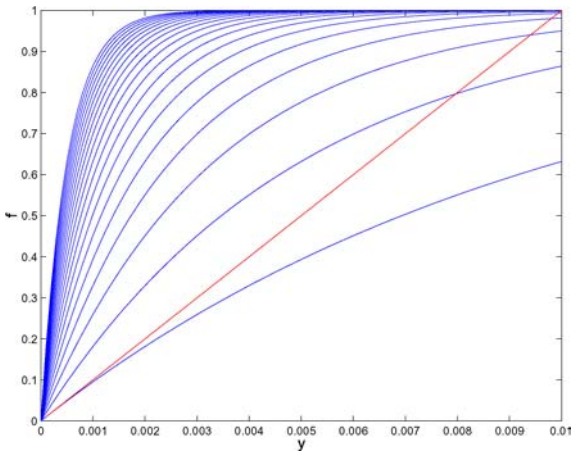

Figure 4. This figure shows how $f(y_i, \mu)$ tends to a step function as μ increases. We have shown μ going from 1 to 20. Also shown is the 45 degree line represented by the convex relaxation $f_C(y_i)$.

and representing the function f as:

$$f(y_i, \mu(y_i)) = C_{\text{cap}}(1 - e^{-\mu(y_i)y_i})) = C_{\text{cap}}(1 - e^{\frac{1}{y_i}(log_e(1-y_i))y_i} = C_{\text{cap}}y_i.$$

This function is denoted by $f_c(y_i)$ and is shown as the diagonal line in figure 4. The new formulation of the problem can then be stated as:

$$\begin{array}{lll} \text{SSO}_{\text{cont}} & \underset{V,\overline{I},y}{\text{minimize}} & V^TYV + \sum_i C_{\text{mult}}(1 - e^{-\mu y_i}) \\ & & \overline{I} - YV = 0 \\ & \text{subject to} & V^l \le V \le e \\ & & (1 - y_i)\ell \le \overline{I_i} \le (1 - y_i)\ell_i + y_i S_{\text{cap}}, \qquad \text{for all } i \\ & & 0 \le y_i \le 1, \qquad \text{for all } i, \end{array}$$

where $C_{\text{mult}} = \frac{C_{\text{cap}}}{C_{\text{loss}}}$. Since C_{cap} and C_{loss} are given, C_{mult} is fixed. However, for the feasibility phase it is often beneficial to deviate from the prescribed value. Note that for a small enough C_{mult}, a substation would be placed at every load node. This would be represented by the specification: $y_i = 1, \forall i \in \{j : \ell_j < 0\}$.

For large enough values of C_{mult}, the choice of a single nonzero valued component of y would be preferred to two or more nonzero components of y, provided both choices are feasible. Consequently, large values of C_{mult} force variables onto their lower or upper bounds and no rounding is needed for such variables. Moreover, it reduces the number of nonzero elements and hence the number of substations in the initial feasible solution.

The resulting solution need not have all $y_i \in \{0,1\}$. However, any nonzero $y_i < 1$ is rounded to 1. We shall illustrate the impact of changing C_{mult} on the solution by the following example.

Example 4.1. Consider an electrical grid of dimension 6×6 with a non-uniform load distribution (this is described further in Section 6.1). We set $\mu = 10$ and consider three levels of capital cost: $C_{\text{cap}} = 0.001, 0.0005$ and 0.0003. The cost of losses is set at unity. The supply of power for case 1 is as follows:

Supply: Case 1					
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0.9353	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

implying that a substation will be placed at position (4, 3).

Case 2. on the other hand has a capacity cost of 0.0005. The power supply distribution is

Supply: Case 2					
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0.3991	0	0	0
0	0	1.0000	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Case 3. specifies a capital cost of 0.00025 resulting in the following distribution of supply:

Supply: Case 3					
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0.4347	0	0	0
0	0	0.3528	0	0	0
0	0	0	0	0.0555	0
0	0	0	0	0	0

Table 1. Comparison of cost and losses.

	Case 1	Case 2	Case 3
Number of substations	1	2	3
C_{mult}	0.00100	0.00050	0.00025
Cost of losses	0.00170	0.00096	0.00080
Cost of capital	0.00100	0.00100	0.00075
Total cost	0.00270	0.00196	0.00155

and requires the installation of 3 substations with positions (3, 3), (4, 3) and (5, 5). The cost statistics are provided in Table 1. Note that despite the capital cost falling by nearly a factor of 2 across each case, the number of substations only doubled from case 1 to 2. This corresponds well with the drop in losses. For instance, the losses nearly get halved from case 1 to 2 but do not drop as significantly when another substation is added. This is because the third substation has an outflow of only 0.0055 as opposed to 0.4347 and 0.3598 from the first two substations.

In practice, this method of obtaining an initial feasible solution has been highly effective. Typically, the number of substations in the solution is smaller than the optimal number, which is beneficial to our search procedure (for the optimal number of substations).

4.2. A local relaxation: "Relax" phase

Given an initial feasible point, we keep the **number** of substations fixed and attempt to determine their optimal location. As we shall discuss in Section 4.4, the number of substations is adjusted in a second phase. The basic idea is that we wish to determine a new assignment with a lower cost from all possible neighbors of the current assignment. Since each node has eight neighbors, this implies that for k substations, there are $(9^k - 1)$ possible new assignments. In general, for even small values of k (say 5), this represents a large amount of work in terms of solving the resulting load-flow equations.

To get some guidance on which neighbors are attractive, we introduce new substations at all neighboring locations. We call this a 9-point stencil (see figure 5). Each stencil can, at most, bear a load equal to that of the capacity of a single substation. We assume that the current locations of the substations are sufficiently separated so that the stencils do not overlap. It remains to be determined as to what each substation node of the 9-point stencil contributes in terms of power. If the outcome is that one substation bears the whole load (unlikely), then this is clearly an improvement since it is the solution of a relaxed problem. However, in the more likely outcome that each substation in the 9-point stencil bears some load, we deduce an attractive location of the substation based on the distribution of the supply across a stencil. We can determine this distribution of capacity by solving a QP in continuous variables. Given a placement of new substations on the electrical grid, this implies that we can construct a "null space" matrix Z_{ss} and a "range space" matrix Y_{ss}, associated with the current set of substation positions (see Section 3.1). The subproblem to be solved can be stated as

$$
\begin{array}{lll}
\text{SSO}_{\text{sub}} & \underset{V,\overline{I}}{\text{minimize}} & V^T Y V \\
& \text{subject to} & \begin{array}{c} \overline{I} - YV = 0 \\ Z_{\text{ss}}\overline{I} + Z_{\text{ss}}\ell = 0 \\ \sum_{j(i)\in N_9(i)} \overline{I}_{j(i)} \leq S_{\text{cap}} + \sum_{j(i)\in N_9(i)} \ell_{j(i)}, i \in N_{\text{ss}} \\ V^l \leq V \leq e. \end{array}
\end{array}
$$

The objective function represents the total losses across the network. The first two constraints represent the load-flow equations. The third constraint specifies that the sum of the currents from the 9 points of a stencil cannot exceed the prescribed substation current capacity plus the total load over the stencil (note that load by convention is assumed to be negative). The last constraint shows the bounds on nodal voltages. Figure 5 shows three grids that explain the aforementioned relaxation procedure.

4.2.1. The integer subproblem. The integer subproblem is the solution of the original problem with the integer variables fixed at some pre-specified levels. This immediately converts the original mixed-integer QP into a QP in continuous variables. The solution to this subproblem, along with the current values of the integer variables, represent a feasible solution to the original MIQP. In fact, the cost of the original MIQP is the cost of the integer subproblem plus the installed cost of new substations. The solution of the problem ($\text{SSO}_{\text{subint}}$) gives the voltages at non-substation nodes and the current flows at all substation nodes . The only difference between (SSO_{sub}) and ($\text{SSO}_{\text{subint}}$) is that the latter problem has a neighborhood defined by N_1 or a 1-point stencil instead of a 9-point stencil, implying no relaxation of substation capacity.

$$
\begin{array}{lll}
\text{SSO}_{\text{subint}} & \underset{V,\overline{I}}{\text{minimize}} & V^T Y V \\
& \text{subject to} & \begin{array}{c} \overline{I} - YV = 0 \\ Z_{ss}\overline{I} + Z_{ss}\ell = 0 \\ \sum_{j(i)\in \mathcal{N}_1(i)} \overline{I}_{j(i)} \leq S_{\text{cap}} + \sum_{j(i)\in \mathcal{N}_1(i)} \ell_{j(i)}, i \in N_{ss} \\ V^l \leq V \leq e. \end{array}
\end{array}
$$

The solution of the local relaxed problem is used to infer a good choice of node to which to move a specific substation. One possibility is to choose the node that is the most important (has the largest current outflow). In Section 4.3 we describe two alternative strategies. In both cases, either no move is made or a better feasible solution is found. We conclude this discussion by noting that solvability of SSO_{sub} and $\text{SSO}_{\text{subint}}$ is guaranteed from the continuity of their objectives and the compactness of their feasible regions.

Result 4.1. *There exists an optimal solution to the subproblems* SSO_{subint} *and* SSO_{sub}.

Proof: By virtue of the compactness of the feasible regions of $\text{SSO}_{\text{subint}}$ and SSO_{sub} and the continuity of their objectives, an optimal solution to both problems exists based on Weierstrass's theorem. □

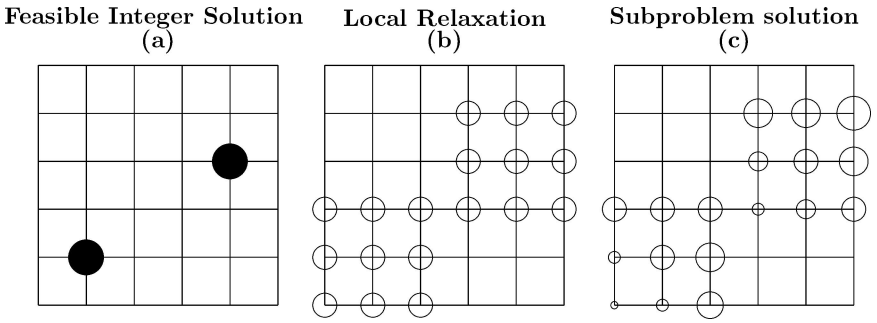

Figure 5. Figure (a) shows a feasible integer solution. The next step demonstrates how the relaxation places "substations" around the current positions. The variables are then given by the flows emerging from these 9-point stencils. Figure (c) shows the solution in which the circle size represents the importance of the node.

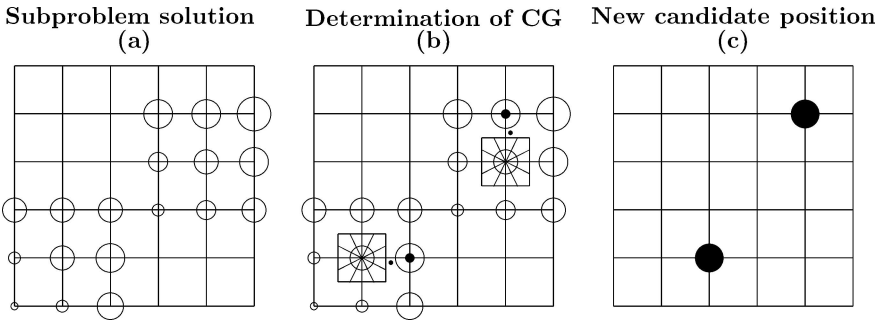

Figure 6. The CG method: Figure (a) shows the subproblem solution. (b) shows the orthants associated with each stencil and the small darkened circles outside the inner stencils represent the centers-of-gravity of the stencil. The placement of the darkened circle in the larger circles represents a candidate position of the substations. Finally (c) shows the new positions of the substations.

4.3. Finding an improved set of locations: "Shift" phase

The "relax" phase replaces each substation by a 9-point stencil and solves the resulting continuous quadratic program. The "shift" phase determines a new set of substation positions with lower system cost. If such a set cannot be found by the method, the algorithm terminates with the current integer solution. Two methods shall be presented for this purpose:

1. The Center-of-Gravity or the CG method.
2. The Assignment method.

4.3.1. The CG method. The coordinates, say x_{cg} and y_{cg}, of the center of gravity associated with a stencil placed at (i, j) are defined by:

$$x_{cg} := \frac{\sum_{k=j-1}^{j+1} (\bar{I}_{(i+1)k} - \bar{I}_{(i-1)k})}{\sum_{p=i-1}^{i+1} \sum_{k=j-1}^{j+1} \bar{I}_{pk}}, \tag{4.1}$$

$$y_{cg} := \frac{\sum_{k=i-1}^{i+1}(\bar{I}_{k(j+1)} - \bar{I}_{k(j-1)})}{\sum_{p=i-1}^{i+1}\sum_{k=j-1}^{j+1}\bar{I}_{pk}}. \tag{4.2}$$

If (x_{cg}, y_{cg}) is "close" to (i, j) the substation is not moved from (i, j). There are a number of measures to define "close". We use a square of size ρ (where $\rho \in [0, 1]$) centered at (i, j). When $\rho = 1$, the square is the size of the stencil. For the results reported here, an initial value of $\rho = 0.1$ was used. This is somewhat conservative since one might expect that $\rho = 0.5$ represents the threshold of when it is worthwhile to move a substation. However, the efficiency of the algorithm is such that we can afford to be conservative.

If the center of gravity is outside the square, we identify an orthant in which it lies. The stencil is divided into eight orthants, which are specified as follows. Suppose the line joining the center-of-gravity to the center of stencil k makes an angle ϕ_k with the positive $x-$axis, where

$$\phi_k = \tan^{-1}\left(\frac{x_k}{y_k}\right).$$

The angle associated with this CG is given by:

$$\phi_k = \begin{cases} \tan^{-1}\left(\dfrac{|x_k|}{|y_k|}\right), & x_k, y_k > 0, \\ \tan^{-1}\left(\dfrac{|x_k|}{|y_k|}\right) + \dfrac{\pi}{2}, & x_k > 0 > y_k, \\ \tan^{-1}\left(\dfrac{|x_k|}{|y_k|}\right) + \pi, & 0 > x_k, y_k, \\ \tan^{-1}\left(\dfrac{|x_k|}{|y_k|}\right) + \dfrac{3\pi}{2}, & y_k > 0 > x_k. \end{cases} \tag{4.3}$$

The candidate substation nodes are defined by:

$$\texttt{cand-node} = \begin{cases} 0 & d_k < \rho, \\ \left\lfloor \frac{\frac{\pi}{16} + \phi_k}{\frac{\pi}{8}} \right\rfloor & d_k \geq \rho, \end{cases} \tag{4.4}$$

where the central node is numbered as 0 and the bordering nodes are numbered clockwise, from 1 to 8, starting from the one immediately above the central node. The orthant in which the center of gravity lies identifies a unique node. We restrict our interest to this node. In so doing, we have reduced our combinatorial problem from $(9^k - 1)$ to $(2^k - 1)$. While the latter is substantially smaller than the former, it will for most cases still be too large to carry out an exhaustive search.

Assuming there is at least one instance of a stencil for which the center of gravity is outside the square, we assign the substations accordingly. Then the subproblem $\text{SSO}_{\text{subint}}$ is solved to yield the voltages and the cost of the new location is determined. If the objective is lower, an improved discrete solution is obtained. Otherwise, ρ is increased to a point that reduces by one the number of stencils for which the center of gravity lies outside the square. This is repeated until there are no alternatives to the

current set of nodes. Note that at the final step, we attempt to move only one substation (and this is for the stencil for which the center of gravity lies furthest from the central node).

As in the choice of the initial ρ, this is a conservative strategy for adjusting the parameter. An alternative is to limit the maximum number of adjustments, say to 10, which implies that at each adjustment, approximately $\frac{\bar{K}}{10}$ substations are fixed, where $\bar{K}$ is the number of substations for which the center of gravity is outside the initial square. The computational cost of finding a new set of positions is comprised of two sets of operations:

1. Number of solves of subproblem ($\mathrm{SSO}_{\mathrm{subint}}$).
2. Number of operations required to specify candidate positions.

Result 4.2. *Assume that there are K new substations at a beginning of a major iteration.*
(a) The CG method requires $O(\frac{K(K+1)}{2})$ *operations to find an improved set of substation positions. This is termed as a major iteration.*
(b) Every major iteration of the CG method requires a maximum of K solves of (SSO_{subint}).

Proof:

(a) Suppose we are given a feasible configuration of substation positions. The CG method requires $O(K)$ operations to determine the centers of gravity and thus a new set of positions, given that there are K new substations. If this set of positions has an improved cost, we terminate; otherwise we fix the substation with the smallest distance from the center of gravity and repeat (note that this is equivalent to increasing ρ). Proceeding in this way, the computational complexity of each iteration of the CG method is $O(K) + O(K-1) + \ldots + O(1) = O(\frac{K(K+1)}{2})$.
(b) The subproblem ($\mathrm{SSO}_{\mathrm{subint}}$) is solved every time a candidate set of positions is made available. If the candidate set fails to provide an improvement, then one of the substations is fixed. A new candidate set is now determined and the process repeats. It follows that this process can be repeated a number of times that is no more than the number of new substations. Therefore the subproblem ($\mathrm{SSO}_{\mathrm{subint}}$) is solved a maximum of K times.

□

4.3.2. The assignment method. While the CG method moves from one position to the next by determining the location of the centers of gravity of the stencils associated with new substations in the relaxed solution, the Assignment method has a different philosophy. This method moves the substations sequentially in a "greedy" fashion. Given a set of `unassigned` substations, we determine an assignment of exactly one of the substations to one of the nodes in its 9-point stencil. This assignment corresponds to the `estimated` minimal increase in cost. Once we move this particular substation, the number of `unassigned` substations reduces by one. We repeat this process till no unassigned or free substations remain. Before proceeding, we define a k-integer solution:

Definition 4.1. A k-integer solution represents a system in which exactly k of the substations have been assigned and the remaining $(K-k)$ substations are relaxed in the

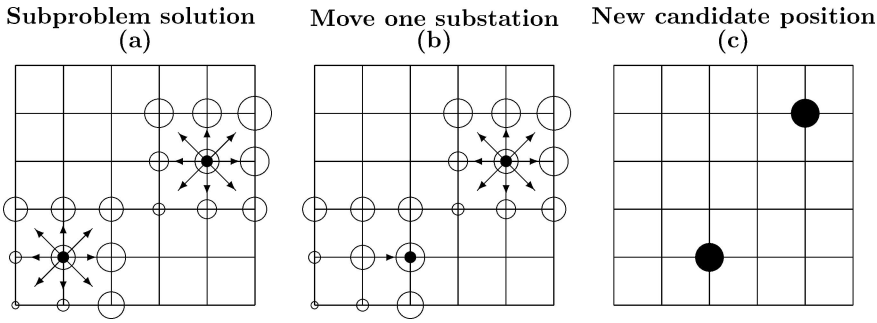

Figure 7. The assignment method: Figure (a) shows the subproblem solution with possible moves for each substations. (b) Moving the substation on the left to its right produces the smallest increase in cost. The next step is to determine where to move the next substation. (c) The candidate positions of the new substations are shown.

form of their 9-point stencils. We may also refer to the 0-integer solution as a relaxed solution.

In keeping with Definition 4.1, we start with a 0-integer solution and assign K substations sequentially till we reach a K-integer solution. Such a sequential assignment requires a decision of the substation to be moved and its destination node. We make such a decision by constructing continuous cost trajectories between discrete solutions. These trajectories are continuous functions that connect a k-integer solution and a $(k+1)$-integer solution.

Continuous trajectories between discrete solutions. Let the flows at each node in the kth 9-point stencil associated be labeled as $i_{k,\,j}$, where $j = 0, \ldots, 8$. The labeling starts from the central node and moves clockwise from the top left hand corner. We construct a continuous mapping that modifies the flows on such a stencil so as to concentrate the sum of the flows at one of the 9 nodes on the stencil, which we shall denote as $c(k)$.

We define a vector function f_k, where f_k scales the distribution of current on stencil k using a scaling parameter α_k. We have

$$f_k(\alpha_k, i_{k,j}) = \begin{cases} (1-\alpha_k) i_{k,j}, & j \neq c(k) \\ \sum\limits_{j \neq c(k)} \alpha_k i_{k,j} + (1-\alpha_k) i_{k,\,c(k)}, & j = c(k). \end{cases} \tag{4.5}$$

It follows that

$$f_k(0, i_{k,j}) = \begin{cases} i_{j,k}, & j \neq c(k) \\ i_{j,k}, & j = c(k) \end{cases} \quad \text{and} \quad f_k(1, i_{k,j}) = \begin{cases} 0, & j \neq c(k) \\ \sum\limits_{j} i_{j,\,k}, & j = c(k) \end{cases} \tag{4.6}$$

respectively. Two distinct mappings are required to construct trajectories to two different points on the stencil. Note that moving from α_k from 0 to 1 results in an increase in the losses. This follows immediately from the fact that eight fewer nodes are now occupied by substations. It should be noted here that no claim is made about the feasibility of the intermediate k-integer solutions. Since we are only using these estimates to determine

Algorithm 1: The CG method at the k^{th} major iteration

Input: $\overline{I}$, `oldpos`
Output: `newpos`
[**Initialize parameters**]
  $n_{\text{fixed}} \leftarrow 0$;
  $n_{\text{new}} \leftarrow K$;
  $z, z_0 \leftarrow z^{k-1}$;
  $\rho \leftarrow 0.1$;
[**Routine returns improved position or terminates**]
while $n_{new} > 0$ **or** $z > z_0$ **do**
  [**Determine centers of gravity and distances for each stencil**]
  for $i = 1$ **to** $n_{\textbf{new}}$ **do**
    calculate CG of stencil $i \rightarrow (x_i, y_i)$;
    calculate distance of $(CG_i) \rightarrow d_i$;
    if $d_i > \rho$ **then**
      Update candidate position of substation i ;
  solve integer problem ($\text{SSO}_{\text{subint}}$) with optimal objective z_{int} ;
  $z_0 \leftarrow z$;
  $z \leftarrow z_{\text{int}}$;
  [**If cost does not increase, fix substation else update**]
  if $z > z_0$ **then**
    fix substation j where $j = \arg\min_i(d_i)$;
    $n_{\text{fixed}} \leftarrow n_{\text{fixed}} + 1$;
    $n_{\text{new}} \leftarrow n_{\text{new}} - 1$;
  else
    update `newpos` with candidate position ;
[**If all substations fixed, no improved position found**]
if $n_{\textbf{new}} = 0$ **then**
  No better integer solution can be found ;
return `newpos` ;

the move, we are not concerned with whether the intermediate steps are feasible with respect to voltage bounds.

Estimating the "Best" move. Given the relaxed solution, the sequence of substations to move and their destimation nodes is to be determined. Each substation can either stay in its current position or can move to node $c(k)$ on stencil k. The specification of $c(k)$ is based on the node that has the greatest percentage of the current (not including the center node). An alternate specification of $c(k)$ could be made based on the octant in which the center of gravity lies.

Suppose the losses of the system are denoted by $F(V)$, where V is the vector of nodal voltages. By assigning a substation, the resulting change in voltage may be specified as Δv. The corresponding change in cost is given by

$$\begin{aligned} \Delta F(V, \Delta v) &= (V + \Delta v)^T Y (V + \Delta v) - V^T Y V \\ &= \Delta v^T Y (2V + \Delta v). \end{aligned} \tag{4.7}$$

We wish to determine the index that has the least cost increase:

$$i = \arg\min_{\Delta v_j}\{F(V + \Delta v_j) - F(V)\} = \arg\min_{\Delta v_j}\{\Delta v_j^T Y(2V + \Delta v_j)\}. \tag{4.8}$$

Note that these changes in voltage are not arbitrary and correspond to the solution of $Y\Delta v_j = \Delta i v_j$, where $\Delta\ i_{vj}$ is the change in current as a result of moving all the current to a particular node. The solution to (4.8) requires a ranking of $2K$ possible voltage trajectories, based on $2K$ possible moves (each substation can stay in its current position or move to candidate node $c(k)$). The cost increment would require solving the system $Y\Delta v = \Delta i_v$. Recall that Y is a banded positive semidefinite matrix and has banded Cholesky factors given by

$$Y = R^T R.$$

The system

$$Y\Delta v = \Delta i_v,$$

has rank $mn-1$, where $\Delta i_v \in \Re^{mn}$. Therefore, it suffices to use only the first $mn-1$ rows of Y. The resulting system is then given by:

$$\begin{pmatrix} Y_1 & w \end{pmatrix} \begin{pmatrix} \Delta v_1 \\ \vdots \\ \Delta v_{mn} \end{pmatrix} = \begin{pmatrix} \Delta i_{v,1} \\ \vdots \\ \Delta i_{v,mn-1} \end{pmatrix}.$$

Moreover, if one specifies $\Delta v_{mn} = 0$, the system becomes:

$$\begin{pmatrix} Y_1 \end{pmatrix} \begin{pmatrix} \Delta v_1 \\ \vdots \\ \Delta v_{mn-1} \end{pmatrix} = \begin{pmatrix} \Delta i_{v_1} \\ \vdots \\ \Delta i_{v_{mn-1}} \end{pmatrix},$$

which may be solved by first finding the Cholesky factor of Y_1, namely R_1. The following lemma exploits the fact that the Cholesky factors are banded to get a tighter bound on the complexity of solving the load-flow equations.

Lemma 4.1. *The solution to $Y_1\Delta v' = \Delta i_v{}'$ can be obtained in $O(mnb^2)$ operations, where b is the size of the band and $Y_1 \in \Re^{mn\times mn}$.*

Proof: The solution of this system would require a banded Cholesky factorization ($O(mnb^2)$) and two triangular factorizations ($O(mnb)$ each) implying a total of $O(mnb^2)$ operations (see [12]). □

Since $2K$ different moves have to be compared when deciding between K substations, it follows that the above system has to be solved $2K$ times to determine the location of the first substation. However, one has to repeat this process for each substation with successively fewer substations to consider. This leads to the following result:

Lemma 4.2. *The system $Y_1 \Delta v' = \Delta\, i_v'$ has to be solved K(K+1) times to move from the relaxed system to a K−integer solution.*

Proof: This follows immediately from the identity that $\sum_{k=1}^{K} 2k = K(K+1)$. □

Let us now restate the problem at each step of the assignment method.

$$\Delta v^* = \arg\min_{\Delta v}\{F(V + \Delta v) - F(V)\} = \arg\min_{\Delta v}\{\Delta v^T Y(2V + \Delta v)\}. \tag{4.9}$$

The following theorem gives a polynomial bound on the number of operations required to either determine an improved integer solution or terminate.

Theorem 4.1. *The assignment method requires $O((K(K+1))^2 mnb^2)$ operations to determine a better integer solution or terminate.*

Proof: The kth step of the assignment method requires $2k$ calculations of the product $\Delta v^T Y(2V + \Delta v)$. The calculation $\Delta v^T Y(2V + \Delta v)$ requires $O(mnb^2)$ operations, where b is the size of the band of Y. Moreover, each step also requires the solution of the load-flow equations $\Delta i_v = Y \Delta v$, which were shown to have a complexity of $O(mnb)$. The total complexity of the assignment method can then be stated as $2.O(mnb^2) + \cdots + 2K.O(mnb^2) = O(K(K+1)mnb^2)$. However, the assignment method may be repeated $\frac{K(K+1)}{2}$ times to either obtain an improved integer solution or terminate. Therefore the total number of operations is $O((K(K+1))^2 mnb^2)$. □

The steps of a single major iteration of the assignment methodare shown in Algorithm 2.

4.4. Adjusting the number of substations

Once a solution has been obtained for a given number of substations, we need to adjust the number to search for a lower total cost. This may be achieved by either adding or removing one substation. The initial feasible solution obtained by a global relaxation methods specifies the number of new substations. However, the initial number of substations may not have been correct and in this section, we show how by a local search (by adding or removing a substation), we can determine an "optimal" number of new substations.

The objective function of the (SSO) problem is the sum of the cost of capital and the cost of losses. The "optimal" number of new substations is based on a trade-off between the cost of losses and the cost of capital. It shall be shown that the cost of losses is a monotonically decreasing function with the number of substations. The cost of capital is linearly related to the number of new substations.

We shall first discuss how losses change with the addition of new substations. Let L_k^* denote the minimal losses based on k substations, where $\min_{ss} \le k \le \max_{ss}$.

Lemma 4.3. *L_k^* decreases strictly monotonically with increasing k.*

Algorithm 2: The Assignment method at the k^{th} major iteration

Input: $\overline{I}$, V, `oldpos`
Output: `newpos`
[**Initialize parameters**]
  $n_{\text{fixed}} \leftarrow 0$;
  $n^1_{\text{new}}, n^2_{\text{new}} \leftarrow K$;
  $z, z_0 \leftarrow z^{k-1}$;
[**Routine returns improved position or terminates**]
while $n^1_{\textbf{new}} > 0$ **or** $z > z_0$ **do**
  for $i = n^2_{\textbf{new}}$ **to** 1 **do**
    [**Loop to estimate incremental costs**]
    for $j = 1$ **to** i **do**
      Determine candidate node $c(j)$ for stencil j ;
      [**Determine voltage change based on current change**]
      Solve $Y\Delta v_{5,j} = \Delta i_{5,j}$;
      Solve $Y\Delta v_{c(j),j} = \Delta i_{c(j),j}$;
    $m = \arg\min_{\Delta v}\{\Delta v^T Y(2V + \Delta v)\}$;
    update candidate position of substation m ;
  solve integer problem ($\text{SSO}_{\text{subint}}$) with optimal objective z_{int} ;
  $z_0 \leftarrow z$;
  $z \leftarrow z_{\text{int}}$;
  [**Routine returns improved position or terminates**]
  if $z > z_0$ **then**
    fix substation j where $j = \arg\min_i(d_i)$;
    $n_{\text{fixed}} \leftarrow n_{\text{fixed}} + 1$;
    $n^1_{\text{new}} \leftarrow n^1_{\text{new}} - 1$;
  else
    update `newpos` with candidate position ;
[**If all substations fixed, no improved position found**]
if $n^1_{\textbf{new}} = 0$ **then**
  No better integer solution can be found ;
return `newss` ;

Proof: Given that the system has k substations and the minimal losses are given by L^*_k. Suppose the addition of a new substation at an unoccupied load node results in losses L_{k+1}. Note that if no unoccupied load node exists, then the losses are already zero since all the loads nodes have substations placed at them. Then the addition of this substation implies that at least some of the load at that node gets served by the new substation with zero losses. In the past, this load required dispatch from some neighboring substation. Thus there is a strict decrease in losses by the addition of a new substation or $L_{k+1} < L^*_k$. But this is merely a feasible set of locations and any optimal solution will have losses $L^*_{k+1} \leq L_{k+1}$. Therefore, L^*_k monotonically decreases with increasing k. □

The objective function of (SSO) is given by: $C_{\text{loss}} V^T Y V + C_{\text{cap}} e^T y$, where the number of new substations is given by $e^T y$. Suppose we denote the number of new substations as

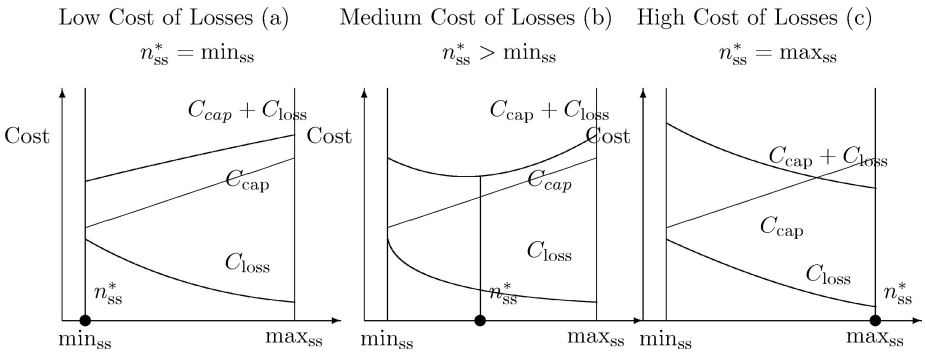

Figure 8. Optimal number of new substations: Figure (a) shows that the optimal number of substations n^*_{ss} is equal to min_{ss} when the cost of losses is sufficiently low. Similarly, when the cost of losses is sufficiently high, $n^*_{ss} = max_{ss}$ as shown in (c). Figure (b) shows an intermediate case. If the cost of losses is very small compared to the cost of capital, the optimal number of substations will always be the minimum number of substations required to physically managed the load subject to voltage constraints. In general, if the impedances are not excessively high, voltage constraints will not be binding. This is seen in the graph on the left in figure 8. If the cost of losses is very large compared to the cost of capital, the optimal number of substations will always be the maximum number of substations required to physically managed the load subject to voltage constraints. This is shown in the rightmost graph in figure 8. This amounts to having as many substations as there are nodes with loads. We discuss an upper bound on the number of substations in Section 5. If the cost of losses is at an intermediate value, the optimal number of substations needs to be determined. We shall be working within this domain. The middle graph in figure 8 shows this profile. Note that for real systems, the optimal number of substations is a tiny fraction of the number of nodes. Typically, the optimal number is close to the minimal possible number of substations.

n, then the cost of losses will be denoted by $f_L(n)$ and the cost of capital will be denoted by $f_C(n)$, The total cost is given by $f_L(n) + f_C(n)$, where $f_L(\cdot)$ is a strictly decreasing function while $f_C(\cdot)$ is a strictly increasing function. However, the functions are not differentiable since n can only take integer values.

The total capital cost is a linear function of the number of new substations and is denoted by kC_{cap}, where C_{cap} is the cost of a substation of size S_{cap} and k is the number of new substations. Thus the maximum capital cost will be mnC_{cap}. If the load at each node is bounded from above by S_{cap}, then the minimal losses are zero when a substation is placed at each node. The optimal number of substations is obviously related to the cost of capital relative to the cost of losses. Figure 8 shows the profile of system cost based on different ratios.

The `add-ss` and `remove-ss` routines are specified as follows:

1. `remove-ss`: This routine removes a substation that has the lowest utilization. If the system stays feasible, we use `relax-and-shift` to obtain a final set of locations. If the total system cost reduces, we invoke `remove-ss` . If there is no further improvement, we exit `remove-ss` .
2. `add-ss`: This routine adds a substation at a point of minimal voltage. Note such an addition will always result in a feasible solution. `relax-and-shift` is then invoked and if the total system cost is an improvement, we continue to keep adding substations till no further improvement can be obtained.

Finally the termination criteria in each of the routines are specified as follows:

1. `remove-ss`: If the removal of a substation results in an infeasible solution, the removal of substation phase is terminated. If the total cost after one major iteration of the `remove-ss` routine is higher than before, this phase terminates.
2. `add-ss`: If the addition of a substation results in no improvement in total cost after a major iteration, then this phase terminates.

Algorithm 3: The `relax-and-shift` subroutine

Input: `oldpos`
Output: `newpos`
[**Initialize parameters**]
  $n_{\text{fixed}} \leftarrow 0$;
  $n_{\text{new}} \leftarrow K$;
  majiter $\leftarrow 0$;
  $z_0 \leftarrow$ cost of initial feasible solution ;
[**Do while at least one substation is not fixed**]
while $n_{\mathbf{new}} > 0$ **do**
  majiter $\leftarrow$ majiter $+ 1$;
  Solve subproblem SSO_{sub} ;
  [**Use either the Assignment method or the `CG` method**]
  if Assignment method is employed then
    `newpos` = assignment(`oldpos`) ;
  else
    `newpos` = CG(`oldpos`) ;
  [**If no improvement and** $n_{\mathbf{new}} > 0$**, then fix a substation**]
  if $z_{\mathbf{int}} < z_0$ **then**
    $z_0 \leftarrow z$;
    $z \leftarrow z_{\text{int}}$;
  else
    `oldpos` $\leftarrow$ `newpos` ;
    fix closest substation;
    $n_{\text{new}} = n_{\text{new}} - 1$;
return `newpos` ;

5. Theoretical properties

This section discusses some of the theoretical properties of the algorithm. The first subsection focuses on deriving some simple bounds on the number of new substations. This is followed by a discussion of the complexity bounds on the major iterations of the SSO algorithm. Third, some lower bounds on the optimal solution are described.

5.1. Bounds on the number of substations

Suppose the total load imposed on the electrical grid is given by $load_S$, where

$$\text{load}_S = \sum_{i=1}^{mn} \ell_i.$$

The total possible capacity that can be introduced on the grid is by the placement of a substation at every node that does not already have a fixed substation. Therefore, the total installable capacity on the electrical grid is given by the number of remaining nodes times the rated capacity of each substation S_{cap}. Moreover, the total load has to be bounded by the total installable capacity plus the total fixed substation capacity.

We have

$$load_S \leq (mn - n_f)S_{\text{cap}} + \sum_{i=1}^{n_f} S_i^{\text{cap}}, \tag{5.1}$$

where the electrical grid has mn nodes and n_f fixed substations with capacities indexed by S_i^{cap}. One can derive a weak lower bound for the number of new substations from this relationship.

Result 5.1. *The maximum number of new substations is given by* n_{ss}^U*, where*

$$n_{ss}^U \leq mn - n_f,$$

and the minimum number of new substations is given by n_{ss}^L*, where*

$$n_{ss}^L \geq \frac{load_S - \sum_{i=1}^{n_f} S_i^{\text{cap}}}{S_{\text{cap}}}.$$

Proof: This follows from the inequality 5.1. □

A **dynamic** upper bound on the number of substations is also immediately available by taking the ratio of the current losses to the cost of a new substation. For instance, if the current losses cost \$10 and the each new substation costs \$1 then a maximum of 10 substations may be added. Obviously, it may turn out that a much smaller number proves optimal. In particular, the dynamic upper bound at iteration k is denoted by $n_{ss}^{U,k}$ is given by:

$$n_{\text{ss}}^{U,k} = n_{\text{ss}} + \left\lceil \frac{C_{\text{losses}}^k}{C_{\text{cap}}} \right\rceil,$$

where n_{ss} represents the current number of substations, C_{losses}^k the current expenditure on losses and C_{cap} the cost of new capacity.

We may claim a polynomial bound on the major iteration of the SSO algorithm through the following algorithm (stated without proof).

Theorem 5.1. *If the problems* SSO_{sub} *and* SSO_{subint} *are solved by a primal-dual interior point method [31], then every major iteration of the SSO algorithm requires polynomial effort.*

Proof: This follows immediately from Result 4.2 and Theorem 4.1 and from the polynomial effort to solve convex QPs [31]. □

5.2. Obtaining a lower bound

While our algorithm provides a feasible integer solution, it would be useful to determine a lower bound on the optimal value of the (SSO). A weak lower bound is immediately available from a continuous relaxation of the SSO problem. If a reformulation of the original problem (SSO) is written as a general mixed-integer quadratic program:

$$\begin{array}{lll} \text{MIQP} & \underset{x,y}{\text{minimize}} & \frac{1}{2}x^T Qx + e^T y \\ & & Ax = b \\ & \text{subject to} & Bx + Dy \leq q \\ & & x \in \Re^n, y \in \mathbf{Z}^m, \end{array}$$

then a continuous relaxation of the MIQP is available by replacing the binary constraints on y by bound constraints, namely

$$\begin{array}{lll} \text{CQP} & \underset{x,y}{\text{minimize}} & \frac{1}{2}x^T Qx + e^T y \\ & & Ax = b \\ & \text{subject to} & Bx + Dy \leq q \\ & & x \in \Re^n, 0 \leq y \leq e. \end{array}$$

The problem (CQP) is a convex quadratic program since Q is a positive semidefinite matrix. The resulting cost z^{CQP} of CQP satisfies $z^{CQP} \leq z^{MIQP}$. This bound is often quite weak.

Duran and Grossmann [5] discuss an outer-approximation algorithm for solving mixed-integer nonlinear programs, with a convex objective and convex constraints. We shall use similar ideas in prescribing a lower bound to (MIQP). It may be noted that the SSO algorithm provides a finite sequence of upper bounds to the optimal solution.

Theorem 5.3. *Let z be the solution to MIQP. The SSO algorithm provides a monotonically decreasing finite sequence of upper bounds z^i to the optimal objective z such that:*

$$z \leq z^{\mathcal{I}} \leq z^{\mathcal{I}-1} \leq \cdots \leq z^1.$$

Recall that for a convex function $f(x)$, the following result holds from convex analysis:

Result 5.2. *Let S be a nonempty open convex set and $f: S \to S$ is differentiable. Then f is convex if and only if it satisfies the gradient inequality:*

$$f(v) \geq f(u) + \nabla f(u)^T (v - u), \quad \forall u, v \in S.$$

This allows us to formulate a mixed-integer program or MIP based on the linearization of the quadratic objective of MIQP at x_f. The resulting MIP may be stated as:

M	$\underset{x,y,\mu}{\text{minimize}}$	$z_l := e^T y + \mu$
		$\mu \geq \frac{1}{2} x_f^T Q x_f + x_k^T Q (x - x_f)$
	subject to	$Ax = b,$
		$Bx + Dy \leq q,$
		$y \in Z^m, x \in \Re^n, \mu \in \Re.$

Assumption 5.1
(a) $\frac{1}{2} x^T Q x$ *is bounded from above by* f_U.
(b) *(Slater's constraint qualification): There exists a point* $x \in X$ *such that* $Bx + Dy < 0$, *for each* $y \in Z^m \cap V$, *where* $V = \{y : Bx + Dy \leq 0$ *for some* $x \in X\}$.

Lemma 5.1. *Given that Assumption 5.1 holds, the optimal objective* z_l *of M is a lower bound to the optimal objective of MIQP.*

Proof: We provide an analogous proof to that in [7]. The optimal objective of MIQP is

$$z(x^*, y^*) := e^T y^* + \frac{1}{2}(x^*)^T Q x^*$$
$$= \min \left\{ e^T y + \frac{1}{2} x^T Q x : Ax = b, Bx + Dy \leq q, x \in \Re^n, y \in Z^m \right\}.$$

This problem may be reformulated as:

MIQP2	$\underset{x,y}{\text{minimize}}$	$\mu + e^T y$
		$\frac{1}{2} x^T Q x - \mu \leq 0$
	subject to	$Ax = b$
		$Bx + Dy \leq q$
		$x \in \Re^n, y \in \mathbf{Z}^m, \mu \in [0, f_U].$

However, it may be seen that $0 \geq \frac{1}{2} x^T Q x - \mu \geq \frac{1}{2} x_f^T Q x_f + x_k^T Q (x - x_f)$ implying that the feasible region of M, $\mathcal{F}_{MIQP2} \mathcal{F}_M$. Therefore $z_l \leq z$. □

We term (M) as a relaxed master problem. A sequence of relaxed master problem $M(k)$ may be defined as follows:

M(k)	$\underset{x,y,\mu}{\text{minimize}}$	$z_l := e^T y + \mu$
		$\mu \geq \frac{1}{2} x_1^T Q x_1 + x_2^T Q (x - x_1)$
		$\mu \geq \frac{1}{2} x_2^T Q x_2 + x_2^T Q (x - x_2)$
		$\vdots$
	subject to	$\mu \geq \frac{1}{2} x_k^T Q x_k + x_2^T Q (x - x_k)$
		$Ax = b$
		$Bx + Dy \leq q$
		$y \in Z^m, x \in \Re^n, \mu \in \Re.$

The optimal value of $M(k)$ increases monotonically with k since the feasible region of $M(k)$ gets smaller due to the addition of cuts. Obviously, if at the kth iterate, the solution obtained by the SSO algorithm is within ϵ of the largest lower bound (as provided by the optimal value of $M(k)$), then we may claim to have an ϵ-optimal solution.

Lemma 5.2. *The optimal objective of M(k) is denoted by z_k. The sequence z_k is monotone nondecreasing and $z_k \leq z$ for all $k \in \{1, \ldots, \mathcal{I}\}$, where $\mathcal{I}$ is the number of major iterations of the SSO algorithm.*

Proof: This follows immediately from the fact that $\mathcal{F}_{M(k)} \subseteq \mathcal{F}_{M(k-1)} \subseteq \cdots \subseteq \mathcal{F}_{M(1)}$. □

Theorem 5.4. *If some member of the upper bound sequence z^i is less than some member of the lower bound sequence z_k for some $k, i \in \{1, \ldots, \mathcal{I}\}$, then the solution obtained by the SSO algorithm is optimal.*

Proof: This is obvious. □

If the SSO algorithm terminates at a sub-optimal solution, then there will be a finite gap between $z^{\mathcal{I}}$ and $z_{\mathcal{I}}$. At this point, neither then `CG` method nor the assignment method is able to provide an improved integer solution. It may be possible to get further improvement by using the solution of $M(\mathcal{I})$ to obtain an improved solution. If one does adopt such a strategy, the method becomes similar to the outer-approximation method as described in [7]. However, we shall assume that such a strategy would prove useless for problems in which the number of integers is of the order of a few thousand.

6. Results

This section discusses some computational results. Section 6.1 specifies the platforms, operating systems and solvers used in the implementation and testing. We tested the algorithm on three different load distributions and provide a description of each. Section 6.2 compares the performance of our algorithm with that of some commercial MINLP solvers such as DICOPT, SBB, and CPLEX. Section 6.3 gives a description of the behavior of the SSO algorithm on some large scale problems while Section 6.4 shows the impact of changing C_{mult} on the optimal solution. Section 6.5 discusses the variation of the system cost with the number of newly installed substations. Finally, we discuss the quality of the solution in Section 6.6

6.1. Implementation details

The SSO algorithm was implemented in MATLAB 6.5 and tested on two systems:

1. A Pentium P-4 processor running WINDOWS-XP with 512 MB of RAM.
2. A SunEnterprise 5500 running Solaris 8 with 4 GB of RAM.

The Tomlab [14] interface was required for using the `SNOPT` [10] and `SQOPT` [11] subroutines in MATLAB. We calibrated the results by running the problems against four

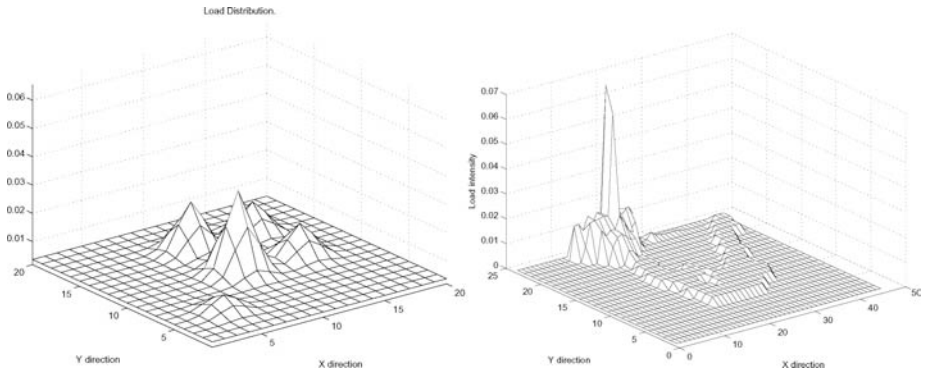

Figure 9. Gaussian load distribution: 20 × 20 (left) and sample snohomish PUD load distribution: 24 × 46 (right).

commercial solvers: CPLEX, GAMS/DICOPT, GAMS/SBB and GAMS/BARON. The algorithms were tested on three different load distributions:

1. Uniform load distribution.
2. Gaussian load distribution as shown at the left of figure 9.
3. Sample load distribution from the Snohomish PUD at the right of figure 9.

6.2. *Computational results from commercial solvers*

This subsection shall compare the performance of three MINLP solvers with our algorithm. We considered the following solvers for our calibration tests: GAMS/DICOPT, GAMS/SBB and CPLEX. We restricted our load distribution to the gaussian load distribution as described in the earlier section. Results for grid sizes varying from 6 × 6 to 20 × 20 are given in Table 2. GAMS/DICOPT did not converge for even the smallest size and its performance shall not be reported. Table 2 reports the final costs (scaled by $1e^3$) and final number of substations for the `CG` and `Assignment` methods as obtained for the non-uniform distribution. We report the performance of SBB and CPLEX. It should be noted that both these solvers fail to solve problems to optimality for sizes larger than 9 × 9. We get suboptimal results for SBB upto 19 × 19. However, the results obtained by them in some cases are 10% worse than the CG algorithm.

We also studied the performance of a global optimization solver GAMS/BARON for a set of problems (see Table 3). We find that except for the 6× 6 and 7 × 7 cases, GAMS/BARON never finds the optimal solution, given the limitation of 3,600 s of CPU time.

An interesting observation that may be made with such a solver is that the performance is affected by the underlying combinatorial problem. For instance, in the 6 × 6 case, there was exactly one substation that was finally installed and the resulting computational time with GAMS/BARON missing was 28 s. However, when the load was increased to a level requiring a minimum of two substations was necessary, the computational time went up to 6 h and 34 s.

Table 2. Comparison with Commercial MINLP Solvers: SBB: Note that z_0 and n_0 represents the cost and the number of substations in the initial feasible solution. C: CPLEX, A: Assignment Method, G: CG Method, s: SBB. z has been scaled by $1e^3$. Note that the results from SBB are early termination results at 1000 branch-and-bound nodes. * implies failure and z* represents optimal values obtained by SBB when there is no upper bound on branch-and-bound nodes.

	Initial		CG		Assign.		SBB			CPLEX	
Size	n_0	z_0	n_G	z_G	n_A	z_A	n_s	z_s	z_s^*	n_C	z_C
6 × 6	1	2.2	1	2.2	1	2.2	1	2.1	2.1	1	2.1
7 × 7	1	2.0	1	2.0	1	2.0	1	2.1	2.0	1	2.0
8 × 8	2	4.9	2	4.2	2	4.1	2	4.3	4.1	2	4.1
9 × 9	2	5.0	2	4.3	2	4.2	2	4.5	*	2	4.2
10 × 10	3	7.4	4	6.5	3	6.6	3	6.4	*	*	*
11 × 11	2	9.6	4	6.6	3	6.7	3	6.8	*	*	*
12 × 12	3	11.8	5	8.7	4	8.9	4	9.2	*	*	*
13 × 13	3	17.8	5	10.7	6	11.1	4	11.8	*	*	*
14 × 14	3	17.0	6	11.0	6	11.3	4	12.4	*	*	*
15 × 15	4	22.2	6	13.0	8	13.5	5	15.0	*	*	*
16 × 16	5	20.8	8	15.5	8	15.8	5	15.4	*	*	*
17 × 17	5	28.3	9	17.6	8	18.5	7	16.9	*	*	*
18 × 18	6	26.4	10	19.6	11	20.3	8	22.2	*	*	*
19 × 19	6	34.5	11	21.9	10	23.1	9	23.9	*	*	*
20 × 20	6	38.7	13	24.2	12	25.5	*	*	*	*	*

Table 3. Comparison with GAMS/BARON: In several cases, the solver terminates with a solution found in the pre-processing phase and finds no better solution till the computational time limit expires.

Size	z_{CG}	z_{BARON}	$\frac{z_{BARON}}{z_{CG}}$	CPU_{BARON}
6 × 6	0.002	0.002	1.00	28
7 × 7	0.002	0.002	1.00	79
8 × 8	0.0042	0.011	2.53	3600*
9 × 9	0.0043	0.008	1.79	475
10 × 10	0.0065	0.015	2.29	387
11 × 11	0.0066	0.009	1.41	1376
12 × 12	0.0087	0.032	3.65	3600*
13 × 13	0.0107	0.020	1.84	3109
14 × 14	0.011	0.021	1.95	319
15 × 15	0.013	0.040	3.11	3600*
16 × 16	0.0155	0.029	1.89	387
17 × 17	0.0176	0.031	1.76	678
18 × 18	0.0196	0.364	18.56	3600*

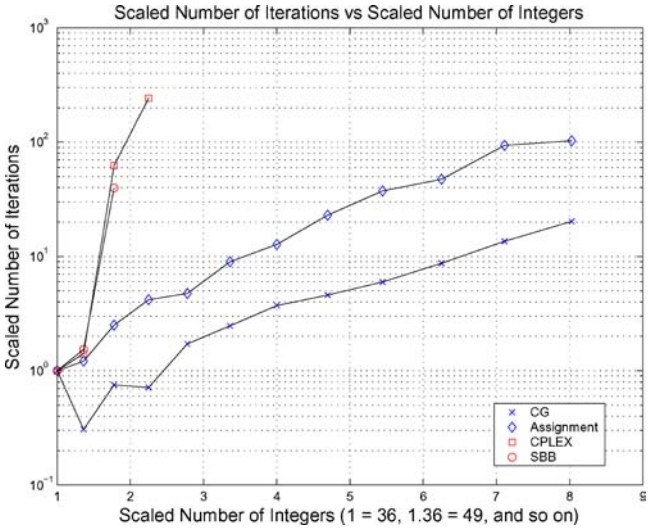

Figure 10. Comparison of scaled computational effort Cost vs. scaled number of integers. This ratio is set as 1000 when the solver fails to converge or terminate gracefully. The *y*-axis has been drawn with a logarithmic scale.

It is observed that the SSO algorithm produces comparable optimal costs to those produced by commercial solvers in the few cases that we can compare. While it is difficult to compare the computational effort of our algorithm with commercial solvers on the basis of iterations, it is possible to construct ratios of every algorithm's performance with the effort taken to solve the 6 × 6 case. We show the scaled computational effort with the scaled number of integers in figure 9.[2] It can be seen that our algorithm shows modest growth in computational effort with the number of integers. However, the two commercial solvers show exponential growth in computational effort. In fact, for the 8 × 8 case, the computational effort of SBB and CPLEX grows by factors of 39 and 62 respectively. To exemplify the differences in time taken, the SSO algorithm takes less than a minute to solve the 15 × 15 case while CPLEX takes over 10 h (on Solaris 8). Note that CPLEX took 3.47 CPU seconds to solve the 6 × 6 problem.

6.3. Computational results on large problems

This section shall show the workings of the algorithm on sizes as large as 50 × 50. In terms of integer variables, this would be more than 25 times the largest problem that CPLEX can handle within a reasonable period of time. We vary the grid size and use two load distributions: a uniform (Table 5) and a gaussian (Table 4) load distributions with sizes ranging from 10 × 10 to 50 × 50. The CG method consistently produced better results efficiently. It may also be noticed that the two methods do not always terminate with the same number of substations. Tables 6 and 7 show the lower bounds obtained for both forms of the SSO algorithm. As a comparison, the cost obtained by solving the convex quadratic program has also been shown. Since, the lower bounds require the

Table 4. Non-uniform load distribution. Note that z_0 and n_0 represents the cost and the number of substations in the initial feasible solution.

	Initial		CG		Assign.		
Size	n_0	z_0	$n_{\mathtt{G}}$	$z_{\mathtt{G}}$	$n_{\mathtt{A}}$	$z_{\mathtt{A}}$	$z_{\mathtt{convex}}$
10 × 10	3	0.0074	4	0.0065	3	0.0066	0.0031
15 × 15	4	0.0222	6	0.013	8	0.0135	0.0031
20 × 20	6	0.0387	13	0.0242	12	0.0255	0.0056
25 × 25	8	0.1085	20	0.0375	18	0.0388	0.0087
30 × 30	13	0.1377	26	0.0565	24	0.0585	0.0127
35 × 35	18	0.2254	37	0.0762	30	0.0824	0.0173
40 × 40	20	0.2696	46	0.0993	41	0.106	0.0224
45 × 45	26	0.3532	56	0.1278	51	0.1356	0.0285
50 × 50	34	0.4364	65	0.1605	54	0.1766	0.0350

Table 5. Uniform load distribution. Note that z_0 and n_0 represents the cost and the number of substations in the initial feasible solution.

	Initial		CG		Assign.		
Size	n_0	z_0	$n_{\mathtt{G}}$	$z_{\mathtt{G}}$	$n_{\mathtt{A}}$	$z_{\mathtt{A}}$	$z_{\mathtt{convex}}$
10 × 10	3	0.0092	4	0.0066	4	0.0067	0.0031
15 × 15	4	0.0302	8	0.0136	8	0.0136	0.0030
20 × 20	6	0.0498	13	0.0247	14	0.0248	0.0056
25 × 25	8	0.1374	20	0.0382	20	0.0383	0.0086
30 × 30	13	0.1377	27	0.0569	26	0.0572	0.0127
35 × 35	18	0.2696	38	0.0767	36	0.0774	0.0172
40 × 40	20	0.282	46	0.1002	38	0.1068	0.0223
45 × 45	26	0.3659	57	0.1285	48	0.1371	0.0284
50 × 50	34	0.4562	74	0.1571	45	0.1979	0.0350

solution of a large mixed-integer linear program, computation of such bounds becomes difficult for large problems.

6.4. *Examining the impact of* C_{mult}

This section examines the impact of changing C_{mult} on the optimal solution. Table 8 considers a 15 × 15 and a 20 × 20 grid with a non-uniform load distribution. We solve the problem with three choices of C_{mult}: 0.005, 0.003 and 0.001. It may be noticed that all three settings result in very similar solutions albeit through very different paths. The first starts with only 4 substations and goes on to add two more. By reducing C_{mult} to 0.003 results in an initial setting at 5 substations with termination at 7 substations. Finally, if we set C_{mult} at 0.001, the initial feasible solution starts at 7 substations and

Table 6. Lower bounds: Non-uniform load distribution.

	CG		Assign.		
Size	$z_{\texttt{CG}}$	$z^L_{\texttt{CG}}$	$z_{\texttt{Assign}}$	$z^L_{\texttt{Assign}}$	$z_{\texttt{convex}}$
6 × 6	0.0022	0.0010	0.0022	0.0010	0.0005
7 × 7	0.0020	0.0010	0.0020	0.0010	0.0005
8 × 8	0.0041	0.0022	0.0044	0.0031	0.0010
9 × 9	0.0043	0.0015	0.0044	0.0017	0.0010
10 × 10	0.0064	0.0020	0.0066	0.0039	0.0015
11 × 11	0.0066	0.0035	0.0067	0.0037	0.0015
12 × 12	0.0090	0.0042	0.0089	0.0051	0.0020
13 × 13	0.0107	0.0052	0.0111	0.0060	0.0026
14 × 14	0.0111	0.0072	0.0114	0.0081	0.0025
15 × 15	0.0129	0.0092	0.0138	0.0084	0.0031

Table 7. Lower bounds: Uniform load distribution.

	CG		Assign.		
Size	$z_{\texttt{CG}}$	$z^L_{\texttt{CG}}$	$z_{\texttt{Assign}}$	$z^L_{\texttt{Assign}}$	$z_{\texttt{convex}}$
6 × 6	0.0023	0.0014	0.0023	0.0010	0.0005
7 × 7	0.0024	0.0010	0.0024	0.0010	0.0005
8 × 8	0.0047	0.0012	0.0046	0.0033	0.0010
9 × 9	0.0049	0.0010	0.0047	0.0025	0.0010
10 × 10	0.0067	0.0032	0.0066	0.0037	0.0015
11 × 11	0.0068	0.0029	0.0068	0.0035	0.0015
12 × 12	0.0091	0.0030	0.0091	0.0059	0.0020
13 × 13	0.0113	0.0058	0.0113	0.0060	0.0025
14 × 14	0.0115	0.0074	0.0114	0.0072	0.0025
15 × 15	0.0136	0.0080	0.0135	0.0075	0.0030

adds no further substations. It may be seen that the `add-ss` routine ensures that a poor initial choice does not affect the final solution significantly.

6.5. *Total cost vs. number of substations*

An estimate of the optimal number of substations is made when one obtains an initial feasible solution. This is modified later in the algorithm by the addition or removal of a substation. Figure 11 shows the relationship of the total cost with the number of substations in the system for a 20 × 20 grid. Both the CG and Assignment methods have similar relationships though the former settles on 10 substations while the latter terminates at 11 substations.

Table 8. Impact of changing C_{mult} on solution for a 15 × 15 and a (20 × 20) grid.

15 × 15						20 × 20					
0.005		0.003		0.001		0.005		0.003		0.001	
n	*z*	*n*	*z*	*n*	*z*	*n*	*z*	*n*	*z*	*n*	*z*
4	2.22	5	1.84	7	1.46	6	3.87	7	3.76	8	3.65
4	1.81	5	1.58	7	1.39	6	3.60	7	3.38	8	3.20
4	1.72	5	1.55	7	1.39	7	3.31	7	3.34	8	3.15
5	1.63	5	1.50	7	1.38	7	3.13	7	3.29	8	3.05
5	1.61	5	1.49	7	1.36	7	3.08	7	3.26	8	3.04
5	1.55	6	1.47	7	1.36	7	3.04	7	3.25	8	2.95
5	1.53	6	1.41	7	1.36	7	2.99	8	3.18	8	2.94
5	1.49	6	1.40	7	1.35	8	2.94	8	3.11	8	2.91
5	1.47	7	1.38	7	1.34	8	2.82	8	3.08	8	2.88
5	1.44	7	1.34	7	1.33	8	2.81	8	3.07	8	2.88
5	1.43	7	1.30			9	2.83	8	3.03	8	2.83
5	1.40	7	1.30			9	2.66	8	3.00	9	2.79
5	1.40					10	2.62	8	3.00	9	2.71
6	1.37					10	2.60	8	2.99	9	2.68
6	1.33					10	2.55	8	2.96	9	2.64
6	1.33					10	2.52	8	2.96	10	2.68
6	1.30					11	2.50	8	2.93	10	2.60
						11	2.49	8	2.92	10	2.56
						11	2.49	8	2.90	10	2.54
						12	2.55	9	2.78	10	2.53
						12	2.46	9	2.72	11	2.56
						12	2.45	9	2.70	11	2.53
						13	2.53	9	2.69	11	2.53
						13	2.47	10	2.66	11	2.52
						13	2.42	10	2.64	11	2.52
						13	2.42	10	2.62	11	2.50
								10	2.61	12	2.50
								10	2.61	12	2.45
								11	2.61	12	2.45
								11	2.49		
								11	2.49		
								12	2.56		
								12	2.46		
								13	2.51		

Continued on next page.

Table 8. (Continued).

15 × 15						20 × 20					
0.005		0.003		0.001		0.005		0.003		0.001	
n	*z*	*n*	*z*	*n*	*z*	*n*	*z*	*n*	*z*	*n*	*z*
								13	2.47		
								13	2.47		
								13	2.46		
								14	2.54		
								14	2.51		
								14	2.44		

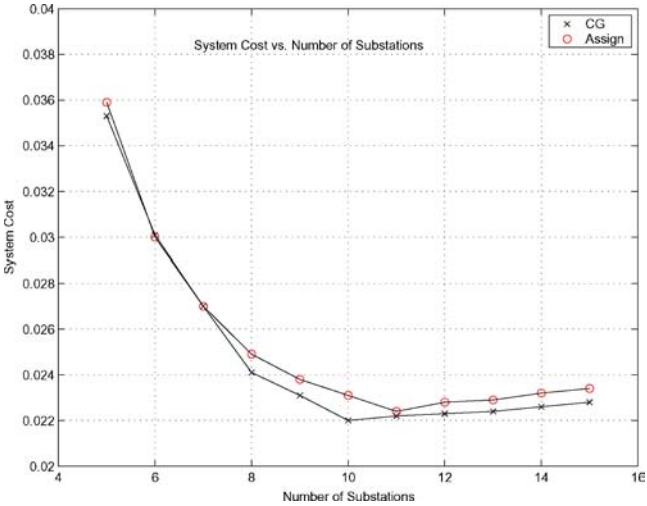

Figure 11. Comparison of total cost vs. number of substations: This cost is in per units (pu) with $C_{\text{mult}} = 0.001$. Note that in general, costs of 28 MVA substations are of the order of $4,000,000.

6.6. *Quality of solution*

There is no assurance that the point at which the algorithm terminates is the global minimizer or even a local minimizer of the problem. Indeed to confirm a given point is a local minimizer is an NP-hard problem. Consequently, any algorithm that makes assurances of a point being a minimizer would be intractable on the size of problem of interest. However, there are ways to judge how well the algorithm has performed.

We first note that the behavior of the optimal number of substations is not of concern. All the cases that we have examined are unimodal with the minimizer being well defined. Moreover, as the number of substations increases, the slope of system cost tends to C_{cap} since the losses tend to zero (see figure 11). Our main concern is whether a better distribution exists for the substations, given the number. On small problems the solution obtained can be compared to CPLEX, which when it works does obtain the

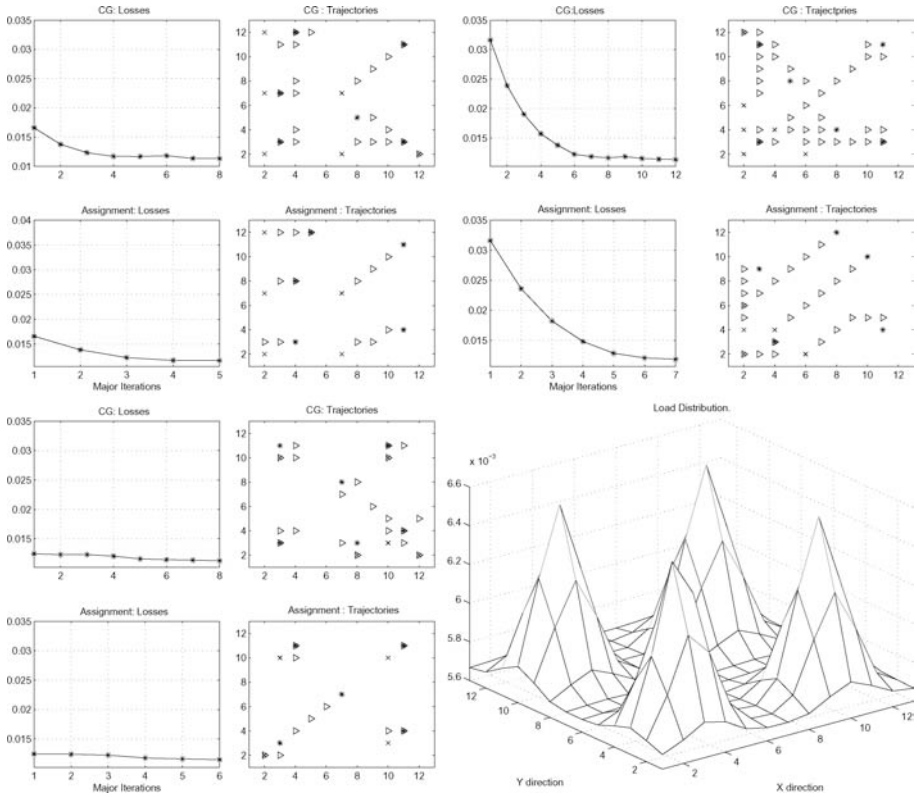

Figure 12. Effect of starting points: Clockwise from top left: (i) Symmetric initial placement with cost = 0.0166 and final costs 0.0113 (CG) and 0.0117 (Assignment) (ii) Poor initial placement with cost = 0.0316 and final costs 0.0113 (CG) and 0.0118 (Assignment) (iii) Good placement as specified by the solution on SSO_{sont} with cost = 0.0125 and final costs given by 0.0113 (CG) and 0.0115 (Assignment) (iv) Load distribution. Note that "x" and "*" refer to the initial and final positions while the triangles denote the intermediate positions.

correct solution. However, in the case of small problems a substation being displaced by one node results in a significant relative error in cost since there are only one or two substations. In practice a displacement of one node is of little significance. Recall that we are identifying which blocks to place the substation and where in the block is decided later. It could be at a boundary and hence be very similar to a solution in which the substation is placed in the adjacent block. In all the tests the biggest difference in displacement was one substation being out by one node. The relative difference in cost was around 1%.

By choosing special load distributions, the solution can be inferred from geometric arguments. For example, given a uniform load and say four substations we would expect the placement of the substations to symmetric. Again in all the tests on such problems the termination point was symmetric or at most one node displaced with the relative cost being very close to the minimum cost. Figure 13 shows the load distribution, when a four gaussian loads are placed in the centers of the four quadrants.

We can also check whether or not we converge to the same or similar termination points if we start from different initial points. Since we have a procedure to determine

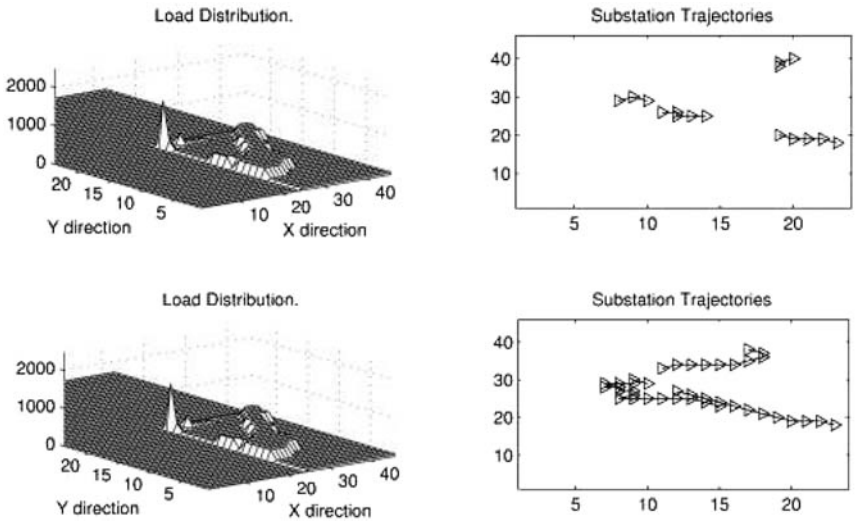

Figure 13. Trajectories of substations—Snohomish load distribution: Good starting point (above) and poor starting point (below).

the initial feasible point, we obtain different feasible points by altering the parameters C_{mult} and μ of that procedure. In all cases there was little difference between the solutions obtained although on occasion the solutions would not have the same number of substations the savings in the losses were almost in balance. Some results from varying C_{mult} are given in Table 8. Some reassurance may also be obtained from the smooth behavior of the objective both in how it reduces as we search for the optimum distribution and the smooth behavior as the number of substations is adjusted.

1. *A symmetric initial placement*. If we start with a symmetric initial point for the substations, it results in an initial cost of 0.0166. The final placement results in a cost given by 0.0113 (CG) and 0.0117 (Assignment).
2. *A poor initial placement*. Similarly, if we place all the substations in the bottom left hand quadrant, the resulting initial cost is quite high (0.0316). However, the final costs of both methods are comparable and are given by 0.0113 (CG) and 0.0118 (Assignment).
3. *A good initial placement*. Finally, if we solve the nonconvex nonlinear program (SSO_{cont}), then the initial cost is 0.0125 which is not significantly greater than what we consider to be an optimal cost. The final costs when we use such a starting point are 0.0113 (CG) and 0.0115 (Assignment).

We also considered a more realistic load distribution as given by the Snohomish PUD on a 24×46 grid. We ran the SSO algorithm using both good and poor initial positions. It can be seen that the substations drift over significant distances to overcome a poor initial placement of substations (as shown in figure 13). We find that the final cost differs only by 1%.

7. Conclusions and future work

We consider a grid-based location problem with quadratic costs and linear constraints. We have proposed a new method for locating substations in such a grid. The proposed algorithm solves a quadratic program to obtain a direction of descent. Two different methods are proposed for determining an appropriate step. Both methods despite being markedly different, produced similar results. Of the two methods, the CG method consistently produced better results efficiently. The number of substations is also optimized by adding and removing substations until no further improvement can be obtained.

Bounds have been provided on the computational effort for a major iteration. Both sets of bounds depend on the square of the total number of substations. In practice, however such a growth has not been observed. It is also seen that the size of the continuous quadratic subproblem is not affected by the number of substations. Results have been provided for cases up to 2500 integers (50×50) and larger cases have also been solved. Comparable results on general purpose nonlinear integer programming solvers have also been provided. For cases larger than 81 integer variables, these solvers fail. The growth in effort with the number of integers is slow with the proposed algorithm. However, for the commercial solvers, this growth is observed to be exponential.

The efficiency of the CG method could be improved by taking a less conservative strategy in the choice of ρ. As we have noted earlier, we can fix more than one substation at a time when we are unable to find an improved solution immediately. We could also warm-start the QP subproblem or use a more sophisticated search for the number of substations. To date, none of these steps have proved necessary.

There is little, if at all, in theory to deal with more complex nonlinear grid-based facility location problems. Our method may be extended to general nonlinear cost functions by using a general NLP solver (such as `SNOPT`) instead of a QP solver. Alternately, we may also use a local QP approximation of the nonlinear problem.

The problem as posed may be embellished in a number of ways. For example, the size of substations can be varied among a discrete set or the transmission lines may be upgraded to specified levels. Such enhancements add to the size and complexity of the model but do not alter its form. To be able to solve such problems clearly requires that we solve the current problem quickly. This has been achieved by our new algorithm.

Acknowledgments

We extend our sincere thanks to Robert H. Fletcher, Public Utility District No. 1 of Snohomish County, Everett, Washington and Patrick Gaffney, Bergen Software Services International (BSSI), for their guidance and support. This research was supported by Office of Naval Research grant N00014-02-1-0076 and National Science Foundation.

Note

1. The current (voltage) is unknown at substation (non-substation) locations
2. We do not include GAMS/BARON in this comparison since there are only two points in which the optimal solution is obtained.

References

1.OQNP and MSDLP, User's Manual, Optimal Methods Inc. and OptTek System Inc.

2.R.N. Adams and M.A. Laughton, Static and time phased network synthesis, Proc. IEE, vol. 121, pp. 139–147, 1974.

3.R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall: Englewood Cliffs, NJ, 1993.

4.H.M. Amir and T. Hasegawa, "Nonlinear mixed-discrete structural optimization," Journal of Structural Engineering, vol. 115, pp. 626–646, 1989.

5.J.Z. Cha and R.W. Mayne, "Optimization with discrete variables via recursive quadratic programming: Part 2 - algorithms and results," Transactions of the ASME, Journal of Mechanisms, Transmissions and Automation in Design, vol. 111, pp. 130–136, 1989.

6.D.M. Crawford and S.B. Holt, "A mathematical optimization technique for locating and sizing distribution substations and deriving their optimal service areas," IEEE Transactions on PAS, pp. 230–235, 1975.

7.M. Duran and I. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," Mathematical Programming, vol. 36, pp. 307–339, 1986.

8.M.A. El-Kady, "Computer-aided planning of distribution substation and primary feeders," IEEE Transactions on PAS, vol. PAS-103, pp. 1183–1192, 1984.

9.A.M. Geoffrion, "Generalized benders decomposition," J. Optim. Theory Appl,, vol. 10, pp. 237–260, 1972.

10.P.E. Gill, W. Murray, and M.A. Saunders, SNOPT: An SQP algorithm for large-scale constrained optimization, Tech. Report NA–97–2, Systems Optimization Laboratory, Stanford University, San Diego, CA, 1997.

11.User's guide for SQOPT 5.3: A Fortan package for large-scale linear and quadratic programming, tech. report, Systems Optimization Laboratory, Department of Operations Research, Stanford University - 94305-4022, 1997.

12.G.H. Golub and C.F. Vanloan, "Unsymmetric positive definite linear systems," Linear Algebra and Its Applications, vol. 28 pp. 85–98, 1979.

13.K.S. Hindi and A. Brameller, "Design of low-voltage distribution networks: a mathematical programming method," Proc. IEE, vol. 124, pp. 54–58. 1977.

14.K. Holmstrom, "The TOMLAB Optimization Environment in Matlab," Advanced Modeling and Optimization, vol. 1, pp. 47–69. 1999.

15.U.G.W. Knight, "The logical design of electricity networks using linear programming methods," Proc. IEE, vol. 117, pp. 2117–2127. 1970.

16.S. Leyffer, Deterministic Methods for Mixed Integer Nonlinear Programming, PhD thesis, University of Dundee, Dundee, Scotland, UK, 1993.

17.A.C. Marshall, T.B. Boffy, J.R. Green, and H. Hague, "Optimal design of electricity networks," IEEE Proceedings-C, vol. 138, pp. 69–77. 1991.

18.E. Masud, "An interactive procedure for sizing and timing distribution substations using optimization techniques," IEEE Trans. on PAS, vol. 93, pp. 1281–1286. 1974.

19.J.J. More and S.J. Wright, Optimization software guide, part 2, pp 87–89, SIAM Philadelphia, 1993.

20.G.L. Nemhauser and L.A. Wolsey, "Integer and Combinatorial Optimization," John Wiley, New York, 1988.

21.A. Neumaier, O. Shcherbina, W. Huyer, and T. Vinko, "A comparison of complete global optimization solvers," Mathematical Programming, Series B., vol. 103, pp. 333–356. 2004.

22.K.-M. Ng, A Continuation Approach to Solving Continuous Problems with Discrete Variables, PhD thesis, Stanford University, Stanford, CA 94305, June, 2002.

23.G.R. Olsen and G.N. Vanderplaats, "Methods for nonlinear optimization with discrete design variables," AIAA Journal, vol. 27, pp. 1584–1589, 1989.

24.N.V. Sahinidis and M. Tawarmalani, "Baron 7.2: Global Optimization of Mixed-Integer Nonlinear Programs," User's Manual, 2004.

25.S. Sharif, M. Salamat, and A. Vanelli, "Model for future expansion of radial distribution networks using mixed integer programming," IEEE, pp. 152–155, 1996.

26.M. Tawarmalani and N.V. Sahinidis, "Global optimization of mixed-integer nonlinear programs: A theoretical and computational study," Mathematical Programming, vol. 99, pp. 563–591. 2004.

27.G.L. Thompson and D.L. Wall, "A branch and bound model for choosing optimal substation locations,"

IEEE Transactions on PAS,, PAS-100 pp. 2683–2687, 1981.
28.D.L. Wall, G.L. Thompson, and J.E.D. Northcote-Green, "An optimization model for planning radial distribution networks," IEEE Transactions on PAS, PAS-98 pp. 1061–1066. 1979.
29.H.L. Willis, H. Tram, M.V. Engel, and L. Finley, "Optimization applications to distribution," IEEE Computer Applications in Power, vol. 10, pp. 12–17, 1995.
30.H.L. Willis, H. Tram, M.V. Engel and L. Finley, "Selecting and applying distribution optimization methods," IEEE Computer Applications in Power, vol. 1, pp. 12–17, 1996.
31.S.J. Wright, Primal-Dual Interior-Point Methods, SIAM, 1996.
32.K. Yahav and G. Oron, "Optimal locations of electrical substations in regional energy supply systems," IEEE, pp. 307–310, 1996.