First online: 24 November 2016

Development and Application of the DIRECT Algorithm for Leak Detection in Water Distribution Systems

M. N. Jasper[1], E. D. Brill[1], R. Ranjithan[1], G. Mahinthakumar[*1]

[1]Dept. of Civil, Construction, and Environmental Engineering,
North Carolina State Univ., Campus Box 7908, Raleigh, NC 27695, USA

[*1]gmkumar@ncsu.edu

Abstract-**The Dividing Rectangles (DIRECT) search is a deterministic, derivative-free, global search algorithm. The algorithm searches for the global minimum by recursive space partitioning, essentially grouping similar regions within the decision space and selecting a sample from each group. The DIRECT algorithm was initially designed for continuous problems, but has since been modified to allow for integer variable types. Though DIRECT has been previously used for discrete numbers, the algorithm has not been extended to other discrete variable types, such as graph nodes or multi-dimensional points. This research further extends the DIRECT algorithm to use a mix of continuous and discrete variables, including connected graph nodes. In this paper, the algorithm is applied to leak detection problems in water distribution systems (WDSs), which involve both discrete network nodes and continuous leak magnitudes. In addition, the DIRECT algorithm is parallelized using a master-worker paradigm and tested using cluster resources for a moderate number of processors. The generalization and abstraction of the DIRECT algorithm presented in this research will enable the application of DIRECT to a wider class of problems than previously possible.**

Keywords- DIRECT; Dividing Rectangles Search; Water Distribution Systems; Leak Detection

I. INTRODUCTION

The Dividing Rectangles (DIRECT) search is a deterministic, derivative-free, global search algorithm originally developed for solving bound constrained optimization problems [1]. This method is able to locate areas around local optima in relatively few function evaluations, but it takes many more function evaluations to converge around a global optimum [2]. The DIRECT method was originally designed for continuous variables, but it has since been modified to allow for integer variables [3]. DIRECT has been used in many engineering applications, such as ship design [4], battery design [5], astrophysics problems [6], pancreatic modeling [7], and yeast modeling [8], among many other problems. However, the DIRECT method has not previously been applied to discrete variable types other than integers, such as graph nodes or coordinate points.

The primary purpose of this research is to generalize the DIRECT algorithm to allow for other variable types as long as the decision space can be intelligently grouped or divided, and as long as each group or section can be representatively sampled. The contributions in this research allow the DIRECT method to be used for many other variable types, not just numerical variables. Furthermore, the DIRECT method has not been previously applied to problems involving a water distribution system (WDS). Another goal of this research is to evaluate the feasibility of using DIRECT for solving leak detection problems in a water network.

II. THE DIRECT ALGORITHM

The DIRECT algorithm was designed by Jones et al. [1] to balance global and local searching without the need for computing derivatives. The DIRECT algorithm is deterministic, not stochastic, and there is therefore no need for multiple runs. Furthermore, DIRECT was designed to have few algorithmic parameters, so that fine tuning is minimized [1]. In the original DIRECT algorithm, there is only one parameter to manipulate, though later modifications have allowed other options and parameters. More details about the DIRECT algorithm, subsequent modifications, and their applications to this research can be found in the thesis from which this paper is based [9].

The DIRECT method has been typically used for problems with only bound constraints, meaning that each of the decision variables has a minimum and a maximum value, but no other constraints. However, modifications of the DIRECT algorithm have been developed to handle other types of constraints [2, 3].

Furthermore, the DIRECT method is considered to be a sampling algorithm, or a derivative-free algorithm, because it does not use the derivative of the objective function. This is particularly useful in engineering applications and other difficult problems where information about the function, such as the smoothness, continuity, and differentiability, is not available. Often times in engineering problems, the objective function may be the result of a simulation or some other iterative method. Therefore, it is preferable in many engineering problems to use not only a derivative free algorithm, but one that requires few

DOI: 10.5963/JAO0401002

function evaluations and is easily parallelizable. The DIRECT algorithm takes relatively few function evaluations when compared to other methods [1], and can be parallelized [2, 10-13]. Because of these advantages, the DIRECT algorithm is ideal for use in engineering problems.

The DIRECT algorithm involves a number of key considerations, including the division of the search space and choice of optimal rectangles. A simplified version of the original DIRECT algorithm is described in the next sub sections; see [3] for more details.

A. Dividing the Search Space

The DIRECT algorithm begins with a hypercube that represents the entire search space and samples the center point (Fig. 1a). Then, the algorithm samples the objective function at points around the center point, plus and minus one-third of the cube's side length in every dimension (Fig. 1b). Formally stated, it samples points $\mathbf{c} \pm \delta \mathbf{u}_i$ for all $i = 1, 2, \ldots, n$, where $\mathbf{c}$ is the center point of the hypercube, δ is one-third the side length of the hypercube, n is the number of dimensions, and $\mathbf{u}_i$ is a unit vector in the i^{th} dimension (i.e. a vector with all zeros, except for a one in the i^{th} dimension).

Once these samples have been evaluated, DIRECT divides the hypercube into thirds along the dimension with the best objective function value first (Fig. 1c), and then it recursively divides the center rectangle along the subsequent dimensions (Fig. 1d) in order of their best function value. This means that the dimension with the best objective function value will have the largest sub rectangles in order to allow the area with the best objective function value to be searched more.

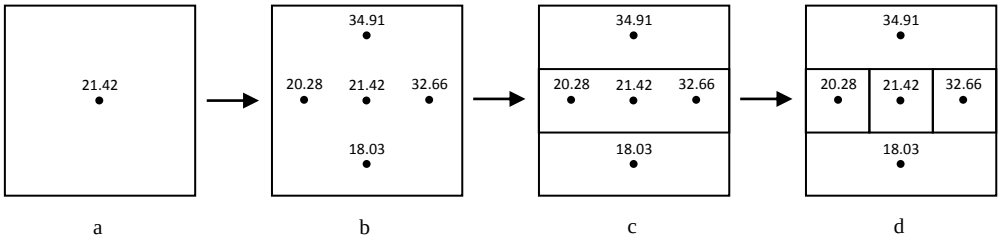

Fig. 1 DIRECT sampling and division in two dimensions with the objective function value shown above each point

Once the initial hypercube is divided, there will be many hyperrectangles as well as smaller hypercubes. The hyperrectangles are divided in a similar manner to the hypercube, except that only the longest sides of the rectangle are sampled, sorted, and divided.

B. Selection of Potentially Optimal Rectangles

When choosing the rectangles to further divide, the DIRECT algorithm does not simply pick the rectangles with the most optimal objective function evaluations. The DIRECT method chooses multiple "potentially optimal" rectangles in each iteration based on both the objective function value and the size of the rectangles.

The rectangle size, called the rectangle radius, is determined based on the number of times the rectangle has been divided T and the total number of dimensions N. Every time that a rectangle is trisected, the length of the longest side ℓ is divided by 3. It is important to note that for any given rectangle used in the DIRECT method, there are only two side lengths: a longer side length ℓ (Eq. 1) and a shorter side length ℓ_s (Eq. 2). This is because the DIRECT method always divides the longer sides of the potentially optimal rectangles. Level k is defined as the number of times the longest side has been trisected (Eq. 3), and stage j is the count of shorter sides for that particular rectangle (Eq. 4). Eq. 5 relates the total number of divisions T to the level k and the stage j. Finally, the rectangle radius r is determined by Eq. 6.

$$\ell = 3^{-k} \tag{1}$$

$$\ell_s = 3^{-(k+1)} \tag{2}$$

$$k = \lfloor T / N \rfloor \tag{3}$$

$$j = \operatorname{mod}(T, N) \tag{4}$$

$$T = k \cdot N + j \tag{5}$$

$$r = \frac{3^{-k}}{2}\left(\frac{j}{9} + N - j\right)^{0.5} = \frac{3^{-k}}{2}\sqrt{N - \frac{8j}{9}} \tag{6}$$

Fig. 2 illustrates how the DIRECT algorithm identifies potentially optimal rectangles. The rectangle sizes (i.e., the rectangle radiuses) are plotted along the x-axis, and the objective function value is plotted along the y-axis. In this plot, the rectangles are represented by points. The points along the lower right of the delineated convex hull are chosen as "potentially optimal". This approach maximizes the size of the rectangles divided (for global search) and minimizes the objective function value (for local search), similar to an approximated Pareto Front.

DOI: 10.5963/JAO0401002

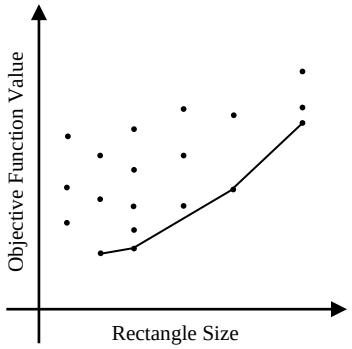

Fig. 2 Graphical interpretation of "potentially optimal" rectangle selection, with rectangles represented as points, and the "potentially optimal" rectangles along the delineated convex hull

C. DIRECT Algorithm Steps

Below is a simplified step by step description of the DIRECT algorithm for multiple dimensions.

0. Initialize.

Normalize all variables to a unit hypercube.

Sample the midpoint of the initial rectangle representing the entire search space.

1. Select Rectangles.

Determine the set of potentially optimal rectangles.

The potentially optimal rectangles are chosen along the lower right convex hull of the Rectangle Size vs. Objective Function Value plot (Fig. 2).

2. Divide Rectangles.

For each rectangle in the set of potentially optimal rectangles:

Evaluate the points at one-third of the longest side length δ away from the center point $\mathbf{c}$ for all dimensions $j = 1, 2, \ldots n$ with the longest side length. $\mathbf{c} \pm \delta\mathbf{u}_j$ where $\mathbf{u}_j$ is the unit vector in dimension j.

Divide each rectangle into thirds along every dimension with the longest side in order of the minimum objective function value in each dimension.

The points evaluated previously are now the centers of these new rectangles.

3. Terminate or Iterate.

Stop when the number of evaluations is greater than the maximum number of function evaluations, or

Stop when the number of iterations is greater than the maximum number of iterations.

Otherwise, continue to iterate and go back to Step 1.

III. THE DIRECT ALGORITHM FOR THE GENERALIZED VARIABLE TYPE

The motivation for an abstract variable type is to be able to use other discrete variable types with the DIRECT method, specifically to represent nodes in a water distribution system (WDS), which are essentially graph nodes. To be able to use DIRECT with graph nodes, more abstraction and generalization is needed, in the same manner that Jones abstracted the DIRECT method for integer variables and mixed real and integer variables [3].

A. Generalized Variable Type

Each variable in the decision space represents a dimension of the rectangle, and when a rectangle is divided or sampled, the individual variables that constitute the rectangle must be divided or sampled. When applying the DIRECT method to general variable types, the two main considerations are how to handle the division and how to select a midpoint of the variable's range.

1) Division

To generalize the division process, every variable type must have its own division process that returns three new child variables that are subdivisions of the original variable. If the variable range does not contain three points (as can be the case with discrete variable types), then it can return two new variables and leave the third empty, as is done in the integer division described in [3]. This makes the rectangle division process independent of the variable type, leading to a more modular implementation. table 1 gives examples of divisions for different variable types.

DOI: 10.5963/JAO0401002

2) Midpoint

Each variable must also have a method of selecting the midpoint of the variable's range. If the variable is an integer type, it will return the integer midpoint, or if the variable type is a graph node, it can return the midpoint node's index. This midpoint selection process is important so that the representative objective function value for the rectangle can be evaluated.

3) Singularity

When dealing with discrete variables, care must be taken when the size of the variable's range reduces to below three. For instance, if there is only one item in the variable's range, then it should not be divided any further. The variable is considered "singular" if the variable only contains one item. When a variable is trisected with only two items, one of the child variables will be empty.

TABLE 1 POSSIBLE DIVISION PROCEDURES FOR VARIOUS VARIABLE TYPES

Variable Type	Possible Division Procedures
Real	A real variable range $[a, b]$ can be divided into three child variables $[a, a + \Delta]$, $[a + \Delta, b - \Delta]$, $[b - \Delta, b]$, where $\Delta = (b - a)/3$
Integer	An integer variable range $[a, b]$ can be divided into three child variables $[a, a + \Delta - 1]$, $[a + \Delta, b - \Delta]$, $[b - \Delta + 1, b]$, where $\Delta = \lfloor (b - a + 1)/3 \rfloor$
Discrete Set	A discrete set of sorted integer or real numbers can be divided into subsets where each subset has approximately the same number of members per subset For example, {1, 2, 4, 8, 16, 32, 64} can be divided into three subsets {1, 2}, {4, 8, 16}, {32, 64}.
Binary	A binary variable can be trisected, so that the third child is NULL. {0, 1} → {0}, {1}, NULL
Cartesian Points	A set of Cartesian points can be divided into three subsets based on either the x-coordinate or the y-coordinate, or it can alternate between the dimension it divides each time the variable is divided.
Graph Nodes	A set of graph nodes can be partitioned based on adjacency information, such as using recursive minimum cuts, or using clustering algorithms.
Water Distribution System (WDS) Nodes	A set of WDS nodes can be recursively skeletonized using network skeletonization algorithms, starting from a very coarse skeletonization and then refining the skeletonization as the set is further divided.

B. Generalized Rectangle

A rectangle is a subdivision of the decision space, and is composed of variable objects (as described in the previous section) of different types that represent the dimensions of the rectangle. The rectangle division process, midpoint selection process, and determination of the rectangle radius are very important in the Dividing Rectangles search.

1) Division

A selected rectangle is divided along the nth dimension and returns three child rectangles. This means that the nth variable will be trisected into three new variables called the individual variable's division method, which depends on the variable type, and the other variables will be copied in the child rectangles.

2) Midpoint

The midpoint method returns the midpoint of the rectangle. This midpoint can be thought of as a "central" or representative sample of the decision space contained within the rectangle. This midpoint is not necessarily geometrically central because some of the variable types solved for in the problem may be discrete. The midpoint of the rectangle is considered to be the midpoint of all of the variable objects contained within the rectangle, and therefore considers each individual variable's midpoint procedure in dimensional order. The midpoint of the rectangle is the point in the decision space that is evaluated in the objective function, and represents this rectangle when determining the potentially optimal rectangles.

3) Singularity

The rectangle is considered singular if all of its variables are singular. If the rectangle is singular, it means that there is only one solution contained within the rectangle, and that the rectangle will not be considered for division any longer. However, the objective function value of the singular rectangle will still be considered when determining the best solution found.

4) Radius

The rectangle radius r is calculated by Eq. 6, which is directly dependent on the dimensionality of the problem N and the number of times the rectangle has been divided T. The rectangle radius is the measurement of the rectangle size that is used when determining the potentially optimal rectangles. It is the x-axis in the objective function value vs. rectangle size plot. It is

DOI: 10.5963/JAO0401002

important that this size criteria does not depend on the individual variable types and only on the number of times the rectangle is divided.

IV. Leak Detection in Water Distribution Systems

Water distribution systems are a vital part of modern infrastructure, bringing safe clean drinking water to the public. However, these systems are susceptible to leaks and contaminant intrusion. High pressures, freezing water, corrosion, and aging can cause cracks in the distribution pipes. It has been estimated that anywhere between 3 and 50 percent of water is lost to leaks in a WDS: three percent in well-maintained systems, and fifty percent in aging systems and in developing countries [14]. Large leaks are usually easy to locate because they can cause significant property damage and flooding, but small leaks gradually lose water into the soil and can be difficult to locate. Leaks cause the pressure to drop in the system, which requires more pumping to maintain the required pressure. If there is negative pressure in the pipe, a leak can become a contaminant intrusion point by leaching chemicals from soil into the water.

A. Current Approaches

Utilities typically monitor locations that are prone to leaks, based on a history of previous leakage or the age of the pipes. A leak can be detected, for example, by using acoustic listening devices that "hear" the sound of water escaping from the pipe. There are other field methods, such as thermal imaging and tracer gas analysis. There are also simulation-based approaches, the most common of which is called Inverse Transient Analysis (ITA). However, ITA requires the use of induced pulses (e.g. opening and closing a fire hydrant) and backward calculation to determine the leak location. Also, ITA is usually used for locating leaks along a long, straight pipe, not in a water network.

These current methods are usually expensive and time consuming. This research seeks to use measurements that can be routinely collected, such as the pressure and chlorine concentration at sensor nodes and the flow through certain metered pipes. In some more progressive utilities, these measurements can be obtained in real time. These measurements can carry a signature that will help identify the leak location and magnitude by using an inverse-modeling approach, potentially reducing the time and expense of leak detection.

B. Water Distribution Simulator

EPANET is an open source water distribution system (WDS) simulator that is widely used and freely available from the US EPA [15]. It uses the network topology, physical properties of the links and nodes, and network demands to determine many of the hydraulic and quality values at each timestep, including the pressure heads and water quality (usually chlorine concentration) at each node, and the flow through each pipe. EPANET divides the total duration time of the simulation into hydraulic timesteps and quality timesteps. At each timestep, it solves many nonlinear equations simultaneously using an iterative method called the "Gradient Method" [16]. EPANET uses properties, like the current tank level, as an initial condition for the next timestep. If a leak configuration is known, it can be modeled as an input to the water distribution simulator, and it will affect the pressure, quality, and flow values, as illustrated in Fig. 3.

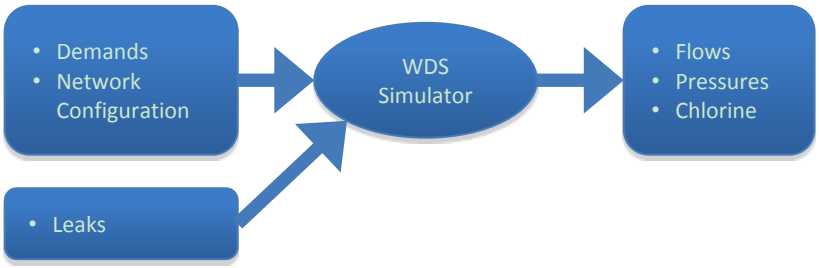

Fig. 3 The WDS Simulator

C. Leak Representation

Leaks are modeled in EPANET as emitters, such as sprinklers or fire hydrants. The distinction between an emitter and a demand is that a demand is a flow out of a node that is known and fixed at each timestep, but an emitter is a flow that depends on the pressure at that node at that time. The equation for the emitter flow is given below in Eq. 7 [16].

$$f_{n,t} = c_n \cdot p_{n,t}^{\gamma} \tag{7}$$

where n is the emitter node (i.e. leak node), t is the current timestep, γ is the pressure exponent, which is fixed at $\gamma = 0.5$, p is the pressure at node n at time t, f is the emitter flow (i.e. flow through the leak) for node n at time t, and c is the emitter coefficient for node n.

When modeling leaks, the variables in this equation are the leak location (the leak node n) and the leak magnitude (the emitter coefficient c_n).

DOI: 10.5963/JAO0401002

D. Inverse Modeling Approach

It is not a straightforward operation to invert the complex models in EPANET to obtain the inputs, such as leak location and magnitude, from the outputted pressure, flow, and water quality values. However, this problem can be formulated as a parameter estimation problem, and an inverse modeling approach (specifically, a simulation-optimization approach) can be used to determine the leak locations and leak magnitudes. In this approach, the leak parameters can be iteratively estimated using an optimization method, such as DIRECT, with a WDS simulator, such as EPANET, to minimize the difference (or error) between the simulated and the measured values (Fig. 4).

E. Objective Function

The goal of this inverse modeling approach is to find the leak parameter values (location and magnitudes) that minimize the error between the simulated values and the measured values. Since there are three different type of measurements with different units and magnitudes (for pressure [ft], quality [mg/L], and flow [gpm]), these are normalized to be incorporated into a single objective function. Eq. 8 below gives the normalized objective function, which is minimized for this leak detection problem.

$$\frac{\sum_n \sum_t (p_{nt} - p_{nt}^0)^2}{\sum_n \sum_t (p_{nt}^0)^2} + \frac{\sum_n \sum_t (q_{nt} - q_{nt}^0)^2}{\sum_n \sum_t (q_{nt}^0)^2} + \frac{\sum_l \sum_t (f_{lt} - f_{lt}^0)^2}{\sum_l \sum_t (f_{lt}^0)^2} \quad (8)$$

where p represents the pressure values, q the quality values, f the flow values, t the current timestep, n the sensor node, l the sensor link (pipe), and 0 the observed or measured values (as opposed to simulated values).

Fig. 4 below shows the objective function as used in this simulation-optimization approach, along with the decision variables.

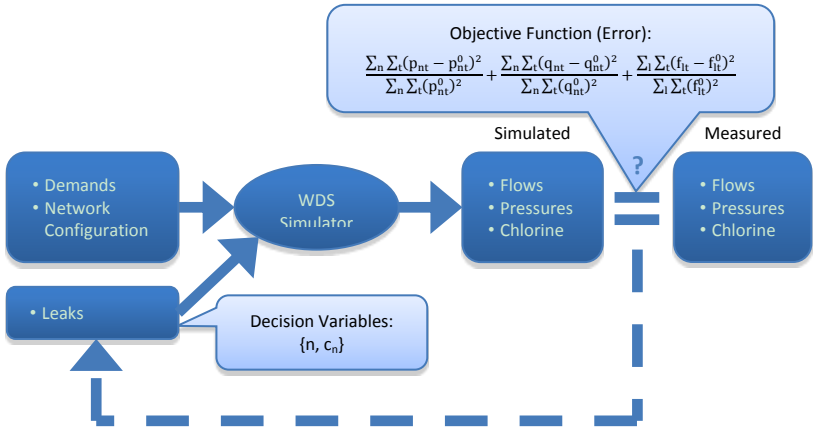

Fig. 4 The Simulation Optimization Approach with Decision Variables and Objective Function

F. Dividing the Discrete Network Nodes

In this research, the WDS nodes were considered to be a set of Cartesian points. When the DIRECT algorithm searches among these points for the leak node, the points are first trisected along the x-coordinate. The second time the points are divided, they are trisected along the y-coordinate. The algorithm continues to alternate the coordinate dimension that is used to divide the set of points. This process is similar to how a k-d tree is divided [17], and will yield an approximately equal number of nodes in each child rectangle. Fig. 5 shows WDS nodes recursively trisected four times. The first trisection occurs along the x-axis, then each of the three child rectangles are trisected along on the y-axis. The midpoint is selected as the node closest to the spatial center of a rectangle, tightly enclosing the points contained in the variable.

DOI: 10.5963/JAO0401002

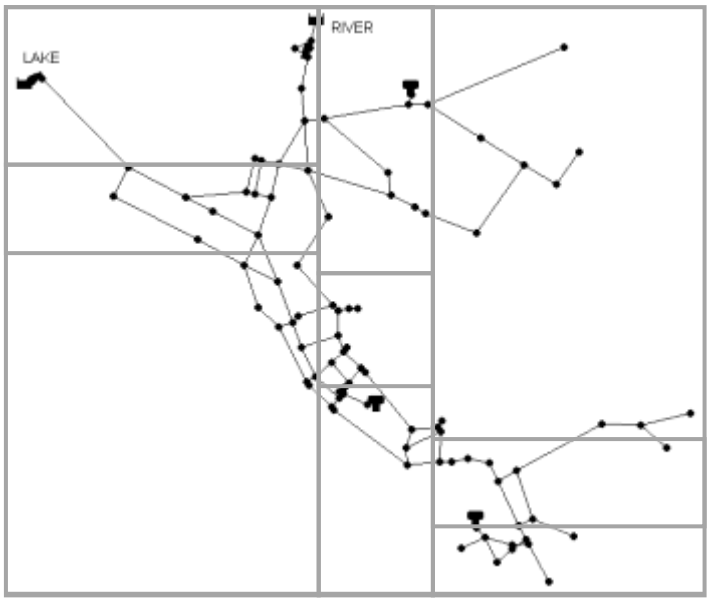

Fig. 5 Recursively Trisected WDS Nodes

G. Problem Representation

The leak locations (leak nodes, n) and leak magnitudes (emitter coefficients, c_n, as shown in Eq. 7) can be found in two practical ways:

Method 1 For a fixed number of leaks N, find the location n (discrete) and magnitude c (continuous) for each leak. This means that there will be a mix of discrete and continuous variables. The decision vector will have the following form:

$(n_1, c_1), (n_2, c_2), \ldots (n_N, c_N)$

with two decision variables per leak. This formulation does not have the flexibility to solve for an arbitrary number of leaks. If the number of leaks is unknown, the number of leaks could be incrementally increased until the solutions converge.

Method 2 For a fixed subset of "candidate" nodes, find the magnitude c (continuous) for each node. The decision vector will be:

$c_1, c_2, c_3, c_4, \ldots c_N$

where N is the total number of candidate nodes. This means that there are only continuous variables, and there is only one decision variable per candidate node. When a candidate node n has no leak, the corresponding magnitude c_n should be zero. This approach has the flexibility to solve for any number of leaks and can be useful if there are already suspected leak areas or nodes. However, to incorporate all of the nodes in the WDS, it would take as many decision variables as there are nodes. This could mean tens of thousands of decision variables for large networks.

V. LEAK DETECTION RESULTS

The DIRECT algorithm was used to solve for leak locations and magnitudes on two test networks, one large and one small. The dimensional scalability of the DIRECT algorithm is tested for both problem formulations (i.e., Methods 1 and 2) as described previously. The dimensional scalability is also tested for the large network for both problem formations. Finally, the dimensional scalability of the DIRECT algorithm is compared to a genetic algorithm.

In all of these test cases, the measured values are generated by introducing artificial leaks in the network and then using these simulated values as the measured values. The actual leak magnitudes are all 1.0. Also, the leak magnitude (modeled as an emitter coefficient), which is continuous, was discretized at increments of 0.1 between 0 and 10 when solving for the variable in DIRECT.

A. Small Network

The small network used for the following tests is the Net3 network provided as an example network with EPANET. It contains 92 junctions (nodes), 2 reservoirs, 3 tanks, 117 pipes, and 2 pumps. The total flow through the network is 13,158 gpm, with demands ranging from 5 to 200 gpm.

In Fig. 6 below, the small network is shown with the actual leak locations used when tested. The sensor nodes where pressure and quality are measured are enclosed in blue squares. The sensor links where the flow is measured are immediately

DOI: 10.5963/JAO0401002

next to the reservoirs and tanks. The nodes where actual leaks are placed are enclosed in red circles and numbered. The ordering of the leak nodes indicates the location of the actual leak nodes. For example, when there is one actual leak, it is at node 1, when there are two actual leaks, they are at nodes 1 and 2, and so on. When using Method 2 to solve for the leak magnitudes at selected candidate nodes, the numbering also indicates which nodes are the candidate nodes. For example, if there are 4 candidate nodes and 2 actual leaks, the candidate nodes will be at nodes 1, 2, 3, and 4 and the actual leaks will be at nodes 1 and 2.

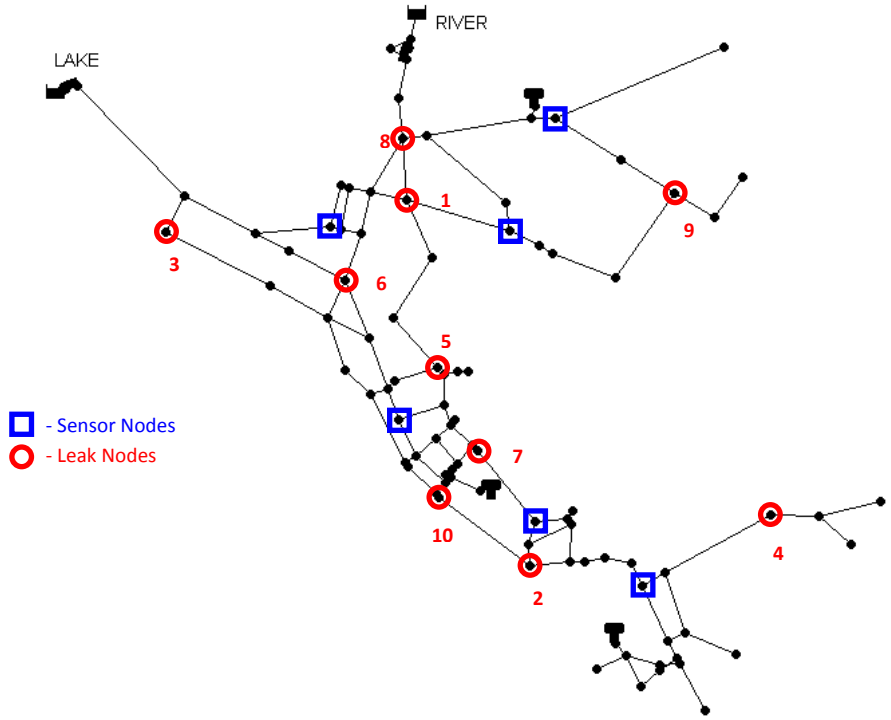

Fig. 6 Actual and Candidate Leak Locations for the Small Network (Net3)

1) Location and Magnitude for Small Network (Method 1)

Fig. 7 below shows the final objective function value after a maximum of 100,000 function evaluations when using the DIRECT method to solve for the leak location and magnitude. The number of leaks that are solved for is gradually increased, and the number of actual leaks is gradually increased. The leak detection problem becomes harder to solve as the number of leaks being solved for increases. This is because the dimensionality of the problem is increasing linearly and the hypervolume of the search domain is increasing exponentially. Therefore, it is expected that the algorithm will take many more rectangle divisions (and thus more function evaluations) to converge on the global optimum.

The leak magnitudes were discretized, and therefore an objective value of zero indicates that the algorithm found the true leak case. Ideally, when the error is close to zero, the leak solution found is close to the actual leak scenario.

The horizontal axis for Fig. 7 indicates the number of leaks searched for as well as the number of actual leaks. For example, the label "2 1" means that two leaks were searched for (4 total decision variables), but there was only one actual leak. In this case, ideally, one leak will have a non-zero magnitude at the actual leak node, and the other leak will have a zero magnitude at an arbitrary node.

The plot in Fig. 7 shows that it does indeed become more difficult to find the actual leak scenario as the number of leaks being solved for increases. However, the graph also shows that it becomes increasingly difficult as the number of actual leaks increases. This is because as the number of actual leaks increases, the number of nodes with a non-zero magnitude increases. When the magnitude is zero, it does not matter which node the algorithm has chosen. Therefore, increasing the number of actual leaks essentially increases the number of "active" decision variables.

Fig. 8 and Fig. 9 below show some selected examples of the solutions returned by the DIRECT algorithm when solving for the leak location and magnitude (Method 1) at the end of the maximum function evaluations.

DOI: 10.5963/JAO0401002

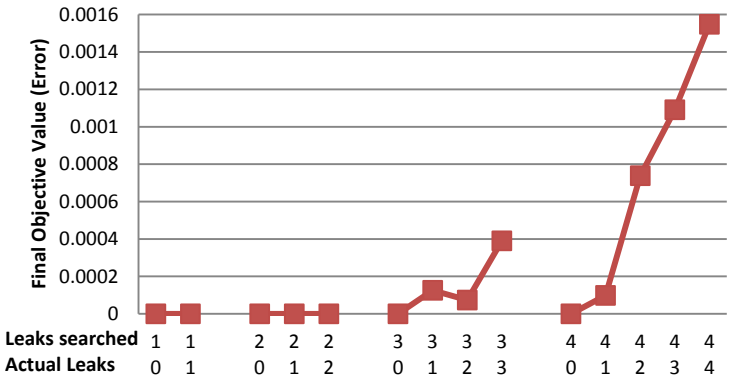

Fig. 7 Final Objective Function Value for Various Leak Configurations for the Small Network using Method 1

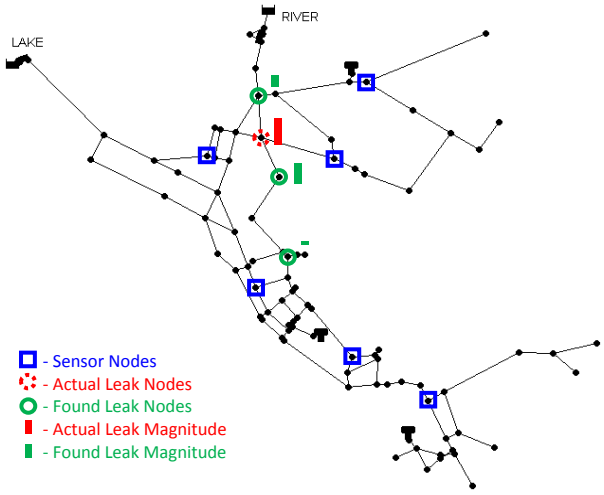

Fig. 8 DIRECT Solution for 3 Leaks Searched For and 1 Actual Leak "3 1"

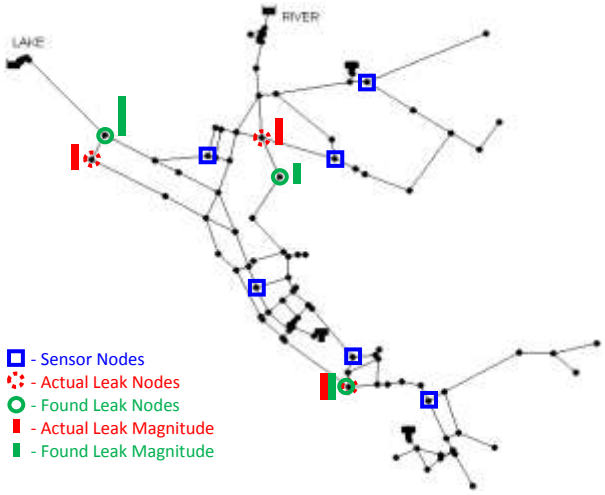

Fig. 9 DIRECT Solution for 3 Leaks Searched For and 3 Actual Leaks "3 3"

In Fig. 8, the DIRECT solution has one larger leak and one smaller leak at adjacent nodes to the actual leak node, and one very small leak farther away from the actual leak node. This example shows that multiple smaller leaks can have similar objective function values as a single leak. In various tests performed in this research, it was found that solutions that have similar total leak magnitudes (related to the total water loss) yield very similar objective function values.

The DIRECT solution shown in Fig. 9 is not very different from the actual leak scenario. The leak nodes found were adjacent to the actual leak nodes, and the magnitudes are very similar. Ironically, the solution in Fig. 9, which appears closer to the actual leak scenario, has a greater error (i.e. objective function value) than the solution in Fig. 8.

2) Magnitude at Candidate Nodes for Small Network (Method 2)

Fig. 10 below shows the final objective function value after a maximum of 100,000 function evaluations when solving for the magnitude at candidate locations. The number of candidate leak nodes is gradually increased, and the number of actual leaks is gradually increased. As stated earlier, the leak detection problem becomes harder to solve as the dimensionality increases. Therefore, it is expected that the algorithm will take an increasing amount of function evaluations to find the actual leak scenario as the number of candidate nodes increases.

The horizontal axis for Fig. 10 has a similar notation to Fig. 7. The number of candidate nodes is first, and the number of actual leaks is second. For example, the label "4 2" means that there are four candidate nodes, but only two actual leaks. Ideally, the two magnitudes corresponding to the actual leak nodes would be non-zero, and the other two magnitudes would be zero.

DOI: 10.5963/JAO0401002

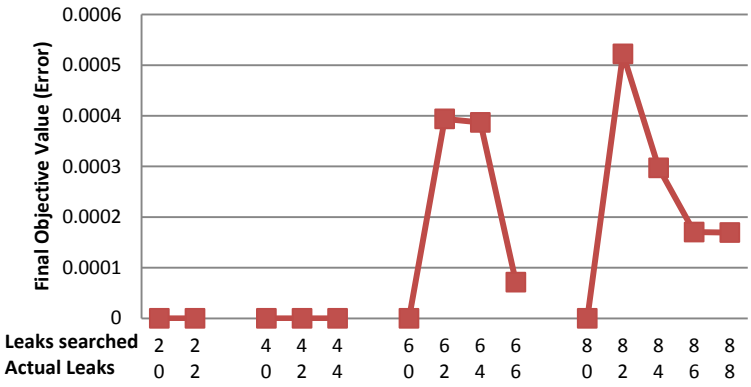

Fig. 10 Final Objective Function Value Comparison
for the Small Network using Method 2

The graph in Fig. 10 shows that it becomes more difficult to find the actual leak scenario as the number of candidate leaks increases. However, the graph also shows a curved pattern when comparing final objective values for leak scenarios with the same number of candidate leaks. The algorithm converges to a solution closer to the true optimum solution when the number of actual leaks is close to zero or close to the number of candidate nodes. The algorithm diverges with leak scenarios between those cases. This is because the leak magnitude has been discretized, and the number of combinations of solutions that yield similar objective function values (the solutions that have similar total magnitudes) is lower when the number of actual leaks is close to zero or close to the number of candidate nodes. Also, the magnitude of all the actual leaks is 1.0. If the magnitudes had been different for each leak, then the trend of increasing difficulty with an increasing number of actual leaks would have been seen, as in Fig. 7.

Comparing the magnitudes of the errors in Fig. 10 and Fig. 7, it can be seen that Method 2 yields lower error values. This is because Method 2 reduces some of the complexity of the leak detection problem by preselecting nodes.

Fig. 11 and Fig. 12 show some selected examples of the solutions returned by the DIRECT algorithm at the end of the maximum function evaluations when solving for the magnitudes at candidate nodes (i.e. Method 2).

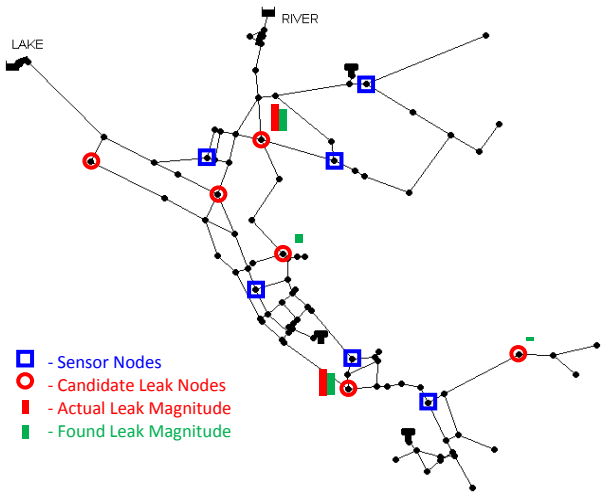

Fig. 11 DIRECT Solution for 6 Candidate Nodes and 2 Actual Leaks "6 2"

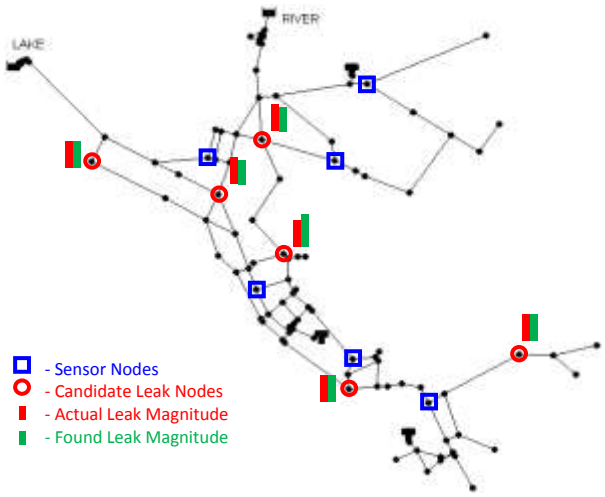

Fig. 12 DIRECT Solution for 6 Candidate Nodes and 6 Actual Leaks "6 6"

In Fig. 11, the solution returned is very similar to the actual leak scenario. There are only small magnitudes at nodes that do not actually have leaks. The DIRECT solution in Fig. 12 is also very similar to the actual leak scenario. There are only very small differences in magnitudes. The solution shown in Fig. 12 has a smaller error than the solution in Fig. 11, as shown in the graph in Fig. 10.

B. Large Network

The large network tested in this research is the Network_2 network given in the Battle of the Water Sensor Network (BWSN) competition [18]. It contains 12,523 junctions (nodes), 2 reservoirs, 2 tanks, 14822 pipes, 4 pumps, and 5 valves. The total flow through the network is 11,061 gpm, with almost all demands being under 25 gpm.

In Fig. 13, the large network is shown with the actual leak locations used when tested. The notation is the same as in Fig. 6. The sensor nodes are enclosed in blue squares. The sensor links where the flow is measured are immediately after the

DOI: 10.5963/JAO0401002

reservoirs and tanks. The nodes where actual leaks are placed are enclosed in red circles and numbered. As in Fig. 6, the ordering of the leak nodes indicates the location of the actual leak nodes. For example, when there is one actual leak, it is at node 1, when there are two actual leaks, they are at nodes 1 and 2, and so on. When using Method 2 to solve for the leak magnitude at selected candidate nodes, the numbering also indicates which nodes are the candidate nodes.

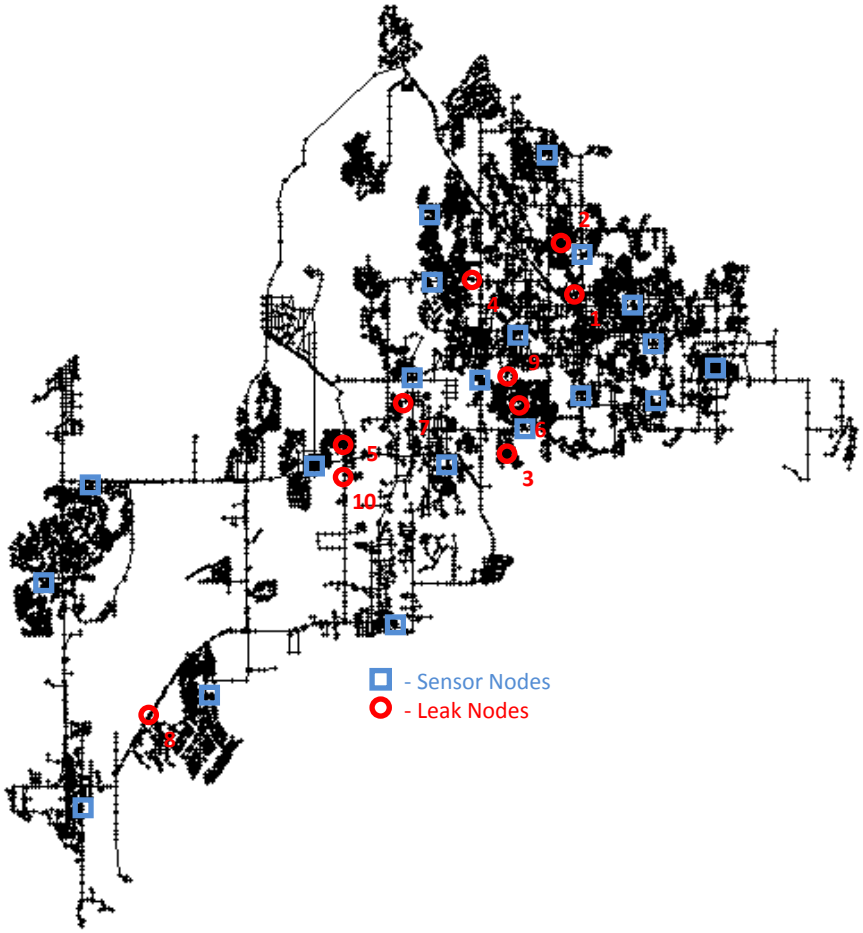

Fig. 13 Actual and Candidate Leak Locations for the Large Network (Network_2)

1) Location and Magnitude for Large Network (Method 1)

Fig. 14 below shows the final objective function value after a maximum of 10,000 function evaluations when using the DIRECT method to solve for the leak locations and magnitudes in the large network. The maximum function evaluations limit is set lower for the large network because the simulations took much longer to run for the large network. The graph shows that the dimensional scalability trends for the large network are essentially the same as those for the smaller network. Increasing the number of leaks searched for also increases the difficulty of converging on the actual leak scenario. Increasing the number of actual leaks increases the difficulty as well.

DOI: 10.5963/JAO0401002

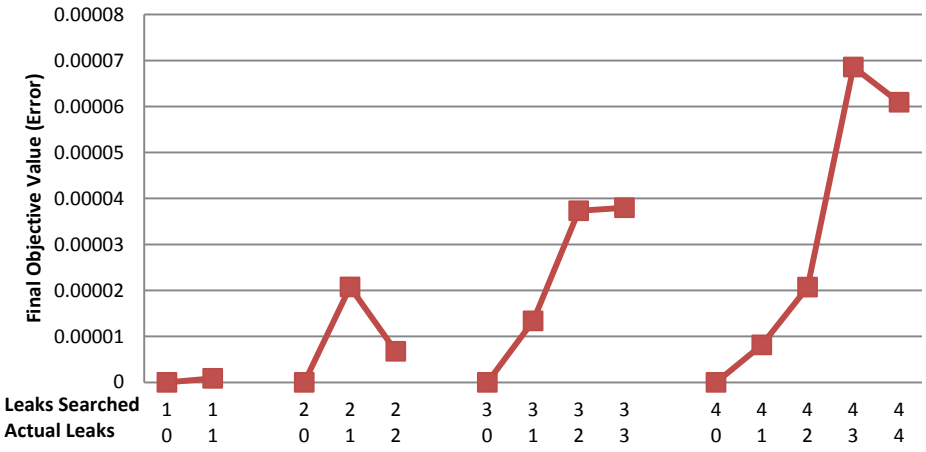

Fig. 14 Final Objective Function Value Comparison
for the Large Network using Method 1

Fig. 15 shows the leak scenario where three leaks are searched for and there are three actual leaks. The leak nodes found are far away from the actual leak nodes. This is due to multiple factors. First, many more than 10,000 function evaluations would be needed to converge near the actual leak nodes. Second, because the network is so vast, the 20 sensors do not give the resolution to easily distinguish between nodes. Leaks at different points in the network can have similar sensor readings because the sensors are sparse in very dense and vast network.

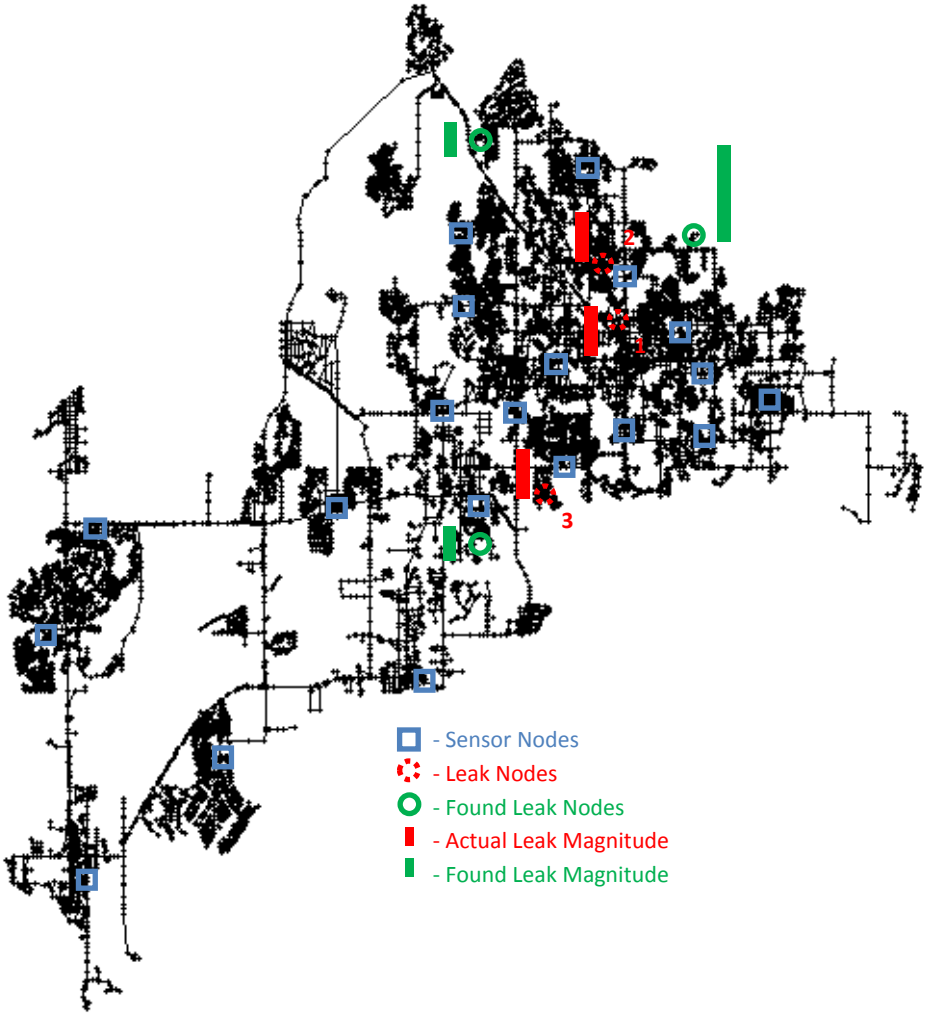

Fig. 15 DIRECT Solution when Searching for 3 Leaks
with 3 Actual Leaks “3 3”

DOI: 10.5963/JAO0401002

2) Magnitude at Candidate Nodes for the Large Network (Method 2)

Fig. 16 below shows the final objective function value after a maximum of 10,000 function evaluations when using the DIRECT method to solve for the magnitude at candidate locations within the large network. Again, the dimensional scalability trends for the large network for Method 2 are essentially the same as those in the smaller network. Increasing the number of leaks searched for increases the difficultly of converging on the actual leak scenario. Also, it more quickly converges to the actual leak scenario when the actual number of leaks is close to zero or close to the number of candidate nodes.

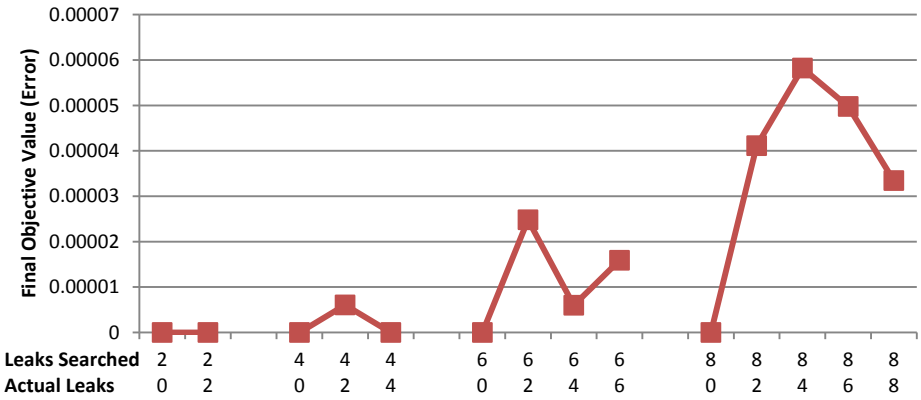

Fig. 16 Final Objective Function Value Comparison
for the Large Network using Method 2

Comparing the magnitudes of Fig. 16 with Fig. 14, Fig. 10, and Fig. 7, it should be noted that the magnitude of the errors using Method 2 are smaller than the errors that resulted from using Method 1. Again, this is because the complexity of the problem is reduced by preselecting candidate nodes. Also, it is important to note that the errors for the large network are much smaller than the errors for the small network. This means that leaks of the same magnitude have less of an effect on the large network measurement values than on the small network. This does not mean that the solutions returned by the algorithm are any closer to the actual leak scenario. In fact, the solutions shown in Fig. 14 and Fig. 16 are farther from the actual leak scenario than the solutions shown for the small network.

Fig. 17 below shows the leak scenario where there are six candidate nodes and six actual leaks. The magnitudes at the candidate nodes are very similar to the actual leak scenario for four out of six of the candidate nodes.

DOI: 10.5963/JAO0401002

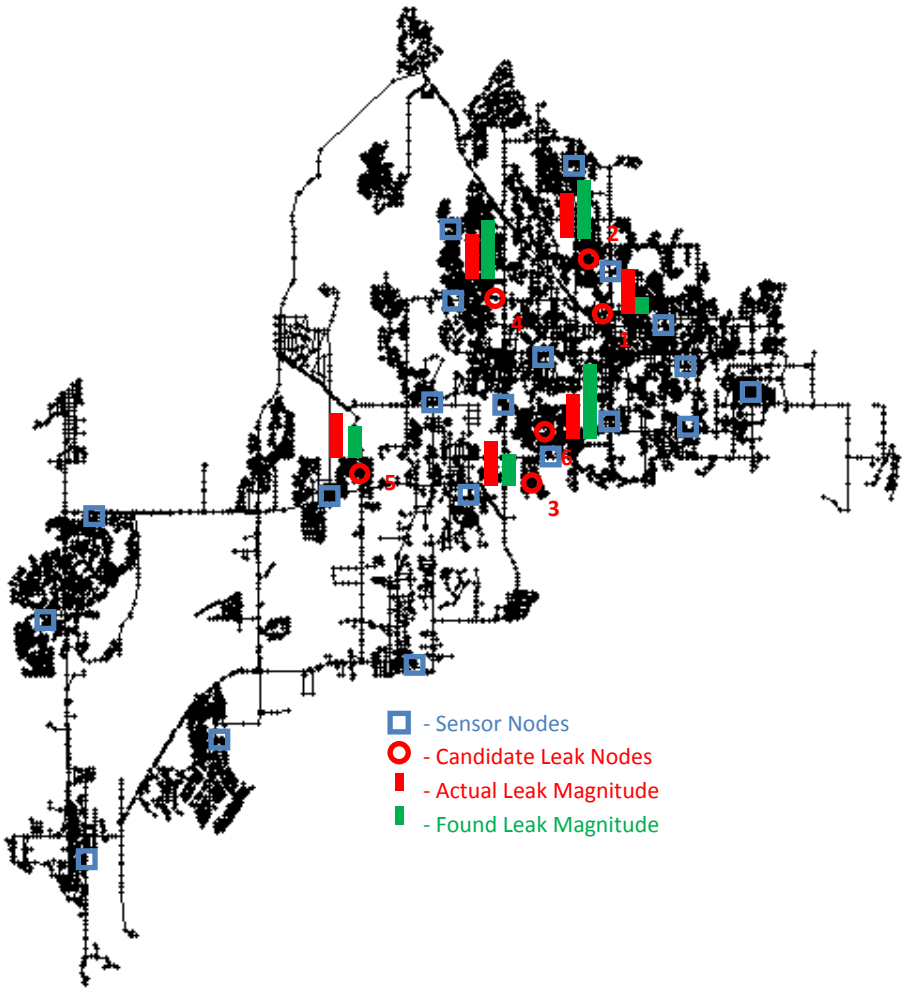

Fig. 17 DIRECT Solution with 6 Candidate Nodes and 6 Actual Leaks in the Large Network "6 6"

C. DIRECT Parameterization, Test Functions, and Comparison with GA

Another part of this research involved testing different parameters for DIRECT, comparing this version of DIRECT to previous versions for common test functions, and finally, comparing the DIRECT method to a genetic algorithm (GA) for solving the water distribution problem. The DIRECT algorithm performed similarly to 10 trials of a GA with the same number of function evaluations in each trial. However, since the DIRECT is deterministic and only runs once, it used fewer total function evaluations and computational resources. More about the comparison to GA, evaluating test functions, and testing DIRECT parameters can be found in [9].

VI. PARALLEL PERFORMANCE

A secondary goal of this research was to provide a high-performance computing (HPC) framework for the DIRECT algorithm to solve the leak detection problem. The DIRECT algorithm has been parallelized before [2, 10-13]. There are two main places that parallel functionality can be added: parallelizing the independent function evaluations, or parallelizing the sampling and evaluating of each of the rectangles. This research, however, parallelizes only the independent function evaluations (i.e., the simulation component) since it takes up over 90% of the computational time for most problems. Also, by keeping the optimization and simulation components separate, another optimization algorithm can easily be interchanged to solve the leak detection problem, and this implementation of the DIRECT algorithm can easily be used to solve other problems.

A. Parallel Implementation

In this research, parallelism is incorporated using a master-worker paradigm where the master performs the DIRECT optimization, distributes the EPANET simulations to the workers, and collects the results. The worker processors perform the independent objective function evaluations (i.e., the EPANET simulations). A generalized middleware was developed so that the DIRECT algorithm could be used with any objective function, including those involving simulators such as EPANET. Fig. 18 illustrates the master-worker parallel implementation with EPANET as the simulator. Instead of employing a single master, a hierarchical master-worker paradigm employing multiple masters could be used to improve scalability as in the pDIRECT_II

DOI: 10.5963/JAO0401002

algorithm implemented by He et al. (2008). A single master-worker paradigm is sufficient for this study, since the problems are typically low dimensional (< 20) and the target architectures are moderately sized (< 200 processor cores).

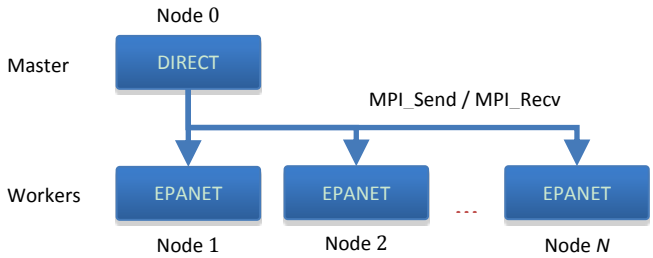

Fig. 18 Schematic of parallel implementation of DIRECT for the water distribution problem

The Message Passing Interface version 2 (MPI) [19] is used to perform the communication between the master and worker processors. The master node handles only DIRECT optimization computations and does not execute any WDS simulations. The master processor distributes the different leak scenarios generated by the DIRECT algorithm to the worker processors using MPI_Send and MPI_Recv. All of the processors are initially sent one simulation, and then the remaining simulations are sent to the processors as they finish their current simulations. Thus, as the processors finish a simulation, they are sent a new simulation. The MPI_ANY_SOURCE keyword is used by the Master process to receive results from the slaves that finish first. This more effectively balances the work load among the worker processors by minimizing synchronization overhead and idle time. Thus, if one simulation or processor is running slower than the others, it will not hold up the other simulations from being completed.

B. Computer Architecture

The parallel performance tests were run on a local Linux cluster in the Civil Engineering Department at NCSU. It is a six node cluster, with each node having four Opteron processors with eight cores each, giving a total of 192 cores. The hardware specifications are:

- 4 processors per node
  AMD “Magny-Cours” Opteron 6128MS 2.0 GHz, 8 Core CPU
- 64 GB total memory at 1333Mhz
- 4 TB HD (Primary node)
- 500 GB HD (Secondary nodes)
- 6 nodes connected with a Gigabit switch

C. Speedup Curves for Different Versions of DIRECT

The DIRECT algorithm is not expected to yield very good parallel performance because the number of function evaluations can change from iteration to iteration. As stated earlier, the DIRECT algorithm chooses multiple “potentially optimal” rectangles to divide per iteration, and the number of rectangles chosen changes per iteration. Therefore, it is difficult to equally balance the load per processor and amortize the communication and optimization costs.

The speedup curves shown below in Fig. 19 are the time to run 100,000 function evaluations (WDS simulations) while optimizing the three leak problems for the small network, where there are two actual leaks. There is a limit of 100,000 function evaluations.

DOI: 10.5963/JAO0401002

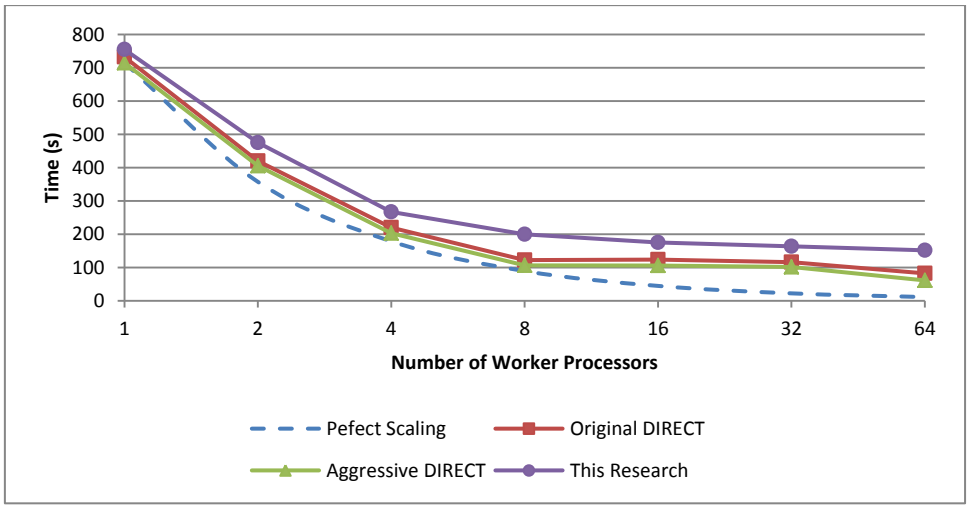

Fig. 19 Speedup Curves for Various Versions of DIRECT

The scalability tests show that the original DIRECT algorithm scales better than the version of DIRECT used in this research. This is because it performs many more function evaluations per iteration and has a lower optimization computation overhead per iteration. However, the original DIRECT algorithm will typically take more function evaluations to optimize a problem than other versions of DIRECT, so care must be taken in determining which options work best when using DIRECT in parallel mode, especially if the objective function is very time consuming to compute.

The original DIRECT algorithm plateaued from 8 to 32 worker processes, as savings in computation time in the function evaluations were offset by the serial overhead in the optimization computations and the communication time. Also, at 8 processes, the communication switches from intra-processor (communication among the 8 cores within a processor) to inter-processor (communication across processors), resulting in additional communication overhead.

The version of DIRECT used in this research to solve the leak detection problem did not scale as well as the original DIRECT algorithm. This is because it performs significantly fewer function evaluations per iteration, resulting in higher parallel overheads. In each iteration, the original DIRECT algorithm divides all of the dimensions of a rectangle, whereas this implementation of DIRECT only divides one dimension of a rectangle.

For applications on massively parallel architectures, the original DIRECT algorithm is a better algorithm to use than the algorithm used in this research. The original DIRECT algorithm performs many more objective function evaluations per iteration, and can therefore better utilize the processors.

Finally, there is another modification of the DIRECT algorithm that is designed for massively parallel applications, called Aggressive DIRECT [20]. Instead of using the convex hull strategy to determine potentially optimal rectangles, Aggressive DIRECT divides all of the rectangles that have the best objective function value for every size group. The purpose of Aggressive DIRECT is to increase the number of function evaluations per iteration, and thus better amortize the costs, yielding improved parallel performance. Because of the large number of function evaluations per iteration, Aggressive DIRECT can be used with larger number processors, which leads to improved turnaround time, though perhaps at the expense of increased computation. Aggressive DIRECT was implemented and compared to other versions of DIRECT in Fig. 19. The same plateau is seen as in the original DIRECT because of the same hurdles of intra- to inter-processor communication. While the improvement over original DIRECT is hardly noticeable, in this plot, this would improve further at a larger number of processors than tested in this study.

VII. CONCLUSIONS

In this paper, the original DIRECT algorithm has been extended to allow for generic variable types, including discrete types. This was done by abstracting the properties and division process for rectangles and different variable types. This extension allowed the DIRECT algorithm to be applied to finding the leak locations and magnitudes in a water distribution system, a mixed discrete and continuous variable problem with network nodes.

It was also shown that the DIRECT method can be used to solve for a small number of leaks in a small or large WDS. Two methods of formulating the leak detection problem were introduced, and both methods scaled similarly. As the number of leaks increases, the dimensionality of the problem increases, and thus the decision space increases exponentially. Because of the high dimensionality of the decision space in the leak detection problem, the DIRECT algorithm requires many more partitions to get close to the global optimum, and thus requires many function evaluations.

Finally, the DIRECT method used in this research was parallelized and the scalability was compared to the original DIRECT algorithm. The version of DIRECT used in this research does not scale as well as the original DIRECT method, but

DOI: 10.5963/JAO0401002

uses fewer function evaluations to converge. When using massively parallel architectures, the original DIRECT or, preferably, the Aggressive DIRECT algorithm should be used.

The introduction of a method to generalize the DIRECT algorithm for different variable types, as presented here, provides flexibility for this algorithm to be applied to a wider class of problems. Future work could target further improvements of the DIRECT algorithm for water distribution and other network class problems. For example, in this research, the WDS nodes were considered to be Cartesian points and were divided along their X-Y coordinates. However, other division schemes can be used, such as k-means clustering. The WDS nodes can also be considered as graph nodes and divided using their adjacency information. The weight of each of the graph links can be the length of the pipe, the volume of the pipe, or the average hydraulic residence time. Considering the WDS nodes as graph nodes could mean using minimum cuts, k-way partitioning, or various other clustering and partitioning algorithms as the division scheme. The WDS nodes can also be recursively skeletonized, starting with a coarse skeleton and refining the skeleton of the region to be divided. WDS nodes can also be divided based on pressures by identifying and dividing pressure zones. Testing of these ideas with WDS problems is needed before any conclusions can be drawn regarding the viability of DIRECT for WDS and other network class problems.

In conclusion, the generalization and abstraction of DIRECT presented in this paper will provide a basis for the DIRECT algorithm to be applied to a wider class of problems and to be used in new ways that were previously impossible.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1100458. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

DEFINITIONS AND ABBREVIATIONS

DIRECT - The Dividing Rectangles Search Algorithm.
WDS - Water Distribution System.
Leak detection - detecting and identifying the location and magnitude of leaks in a WDS.
Rectangle size - measured by the rectangle radius.
Rectangle radius - an abstraction of the center-to-vertex distance, which is a function of the number of times the rectangle has been divided and the total number of dimensions.
Rectangle dimension - a dimension of the rectangle that is represented by a variable of a certain type (e.g. real, integer, graph node, etc.).
Rectangle variable - a dimension in search space that represents a range of possible values. Can be one of many types: real, integer, graph node, etc.
Rectangle side - the range of possible values that define this rectangle in this dimension.
Trisect - to divide the rectangle into three partitions or sub-rectangles.
Rectangle level - the number of times a rectangle has been divided.
Rectangle stage - the number of short sides for the hyperrectangle.
Rectangle midpoint - a central, representative point in decision space that represents this rectangle.
Variable midpoint - a central value within the range of possible values for this rectangle side.
Rectangle singularity - having only one point contained within the rectangle (if using all discrete variable types).
Variable singularity - having only one value contained within the range of possible values for this dimension of the rectangle.
Simulation error - a measure of the difference between the simulated values and the measured values.

REFERENCES

[1] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the lipschitz constant," *Journal of Optimization Theory and Application*, vol. 79, no. 1, pp. 157-181, Oct. 1993.

[2] J. M. Gablonsky, "Modifications of the DIRECT Algorithm," North Carolina State University, 2001.

[3] D. R. Jones, "Direct Global Optimization Algorithm," in *The Encyclopedia or Optimization*, Kluwer Academic, 1999, pp. 725-735.

[4] A. Serani, G. Fasano, G. Liuzzi, S. Lucidi, U. Iemma, E. F. Campana, F. Stern, and M. Diez, "Ship hydrodynamic optimization by local hybridization of deterministic derivative-free global algorithms," *Applied Ocean Research*, vol. 59, pp. 115-128, Sept. 2016.

[5] J. Shen, S. Dusmez, and A. Khaligh, "Optimization of Sizing and Battery Cycle Life in Battery/Ultracapacitor Hybrid Energy Storage Systems for Electric Vehicle Applications," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2112-2121, Nov. 2014.

[6] D. di Serafino, G. Liuzzi, V. Piccialli, F. Riccio, and G. Toraldo, "A Modified DIviding RECTangles Algorithm for a Problem in Astrophysics," *Journal of Optimization Theory and Applications*, vol. 151, no. 1, pp. 175-190, Oct. 2011.

[7] D. Lv and B. Goodwine, "Pancreas Modeling from IVGTT Data Using A Deterministic Optimal Search Method," in *2009 IEEE International Conference on Bioinformatics and Biomedicine*, Washington, DC, 2009.

[8] T. D. Panning, L. T. Watson, N. A. Allen, K. C. Chen, C. A. Shaffer, and J. J. Tyson, "Deterministic parallel global parameter estimation for a model of the budding yeast cell cycle," *Journal of Global Optimization*, vol. 40, no. 4, pp. 719-738, Apr. 2008.

DOI: 10.5963/JAO0401002

[9] M. N. Jasper, "Development and Application of the DIRECT Algorithm for Leak Detection in Water Distribution Systems," North Carolina State University, 2014.

[10] J. D. Griffen and T. G. Kolda, "Asynchronous parallel hybrid optimization combining DIRECT and GSS," *Optimization Methods and Software*, vol. 25, no. 5, pp. 797-817, 13 Aug., 2009.

[11] J. He and L. T. Watson, "Dynamic data structures for a direct search algorithm," *Computational Optimzation and Applications*, vol. 23, no. 1, pp. 5-23, Oct. 2002.

[12] J. He, A. Verstak, L. T. Watson, and M. Sosonkina, "Design and implementation of a massively parallel version of DIRECT," *Computational Optimization and Applications*, vol. 40, no. 2, pp. 217-245, June 2008.

[13] J. He, L. T. Watson, and M. Sosonkina, "Algorithm 897: VTDIRECT95: Serial and parallel codes for the global optimization algorithm direct," *ACM Transactions on Mathematical Software (TOMS)*, vol. 36, no. 3, pp. 1-24, July 2009.

[14] R. Puust, Z. Kapelan, D. A. Savic, and T. Koppel, "A review of methods for leakage management in pipe networks," *Urban Water Journal*, vol. 7, no. 1, pp. 25-45, Feb. 2010.

[15] L. A. Rossman, "The EPANET Programmer's Toolkit for Analysis of Water Distribution Systems," in *29th Annual Water Resources Planning and Management Conference*, 2004.

[16] L. A. Rossman, "EPANET 2 Users Manual," US EPA, Cincinatti, OH, 2000.

[17] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, Sept. 1975.

[18] A. Ostfeld and J. G. Uber, "The Battle of the Water Sensor Networks (BWSN): A Design Challenge for Engineers and Algorithms," *Journal of Water Resources Planning and Management*, vol. 134, pp. 556-568, Nov. 2008.

[19] W. Gropp, E. Lusk, and R. Thakur, Using MPI-2: Advanced Features of the Message-Passing Interface, MIT Press, 1999.

[20] C. A. Baker, L. T. Watson, B. Grossman, W. H. Mason, and R. T. Haftka, "Parallel Global Aircraft Configuration Design Space Exploration," *International Journal of Computer Research*, vol. 10, no. 4, pp. 501-515, 28 June, 2001.

DOI: 10.5963/JAO0401002