Parallel Multi-Swarm Optimization Framework for Search Problems in Water Distribution Systems

Sarat Sreepathi[1], Downey Brill[2], Ranji Ranjithan[2],
Gnanamanikam (Kumar) Mahinthakumar[2]
{sarat_s,brill,ranji,gmkumar}@ncsu.edu
[1]CSC Dept., North Carolina State University,
[2]CCEE Dept., North Carolina State University
Raleigh, NC, USA - 27695.

Abstract

Population based heuristic search methods such as evolutionary algorithms (EA) and particle swarm optimization (PSO) methods are widely used for solving optimization problems especially when classical techniques are inadequate. A parallel optimization framework using multiple concurrent particle swarms is developed and applied to water distribution problems. Details of the enabling framework that couples the optimization methods with a parallel simulator built around EPANET will be discussed. In addition, algorithmic and computational performance results using ORNL's and ANL's leadership class parallel architectures will be presented for leakage detection and contaminant source characterization problems for two water distribution networks with 1,834 and 12,457 nodes respectively.

1 Motivation

Urban water distribution systems (WDSs) form a critical infrastructure that is vulnerable to accidental and intentional contamination incidents that could result in adverse human health and safety impacts. When a contamination event is detected via the first line of defense, e.g., data from a water quality surveillance sensor network and reports from consumers, the municipal authorities need to rapidly predict the source characteristics (such as location, and time of release and mass loading history), estimate the extent of the contamination in the network through simulation and confirmatory sampling, and prescribe control actions (such as flow controls using valves, hydrants and storage units, or injection of decontaminants) to mitigate the event. These interrelated issues could be posed as optimization problems in a simulation-optimization framework. This work is focused on development of a parallel optimization framework to aid in solving problems in water distribution networks. The rest of the paper is organized as follows: Section 2 introduces the optimization framework

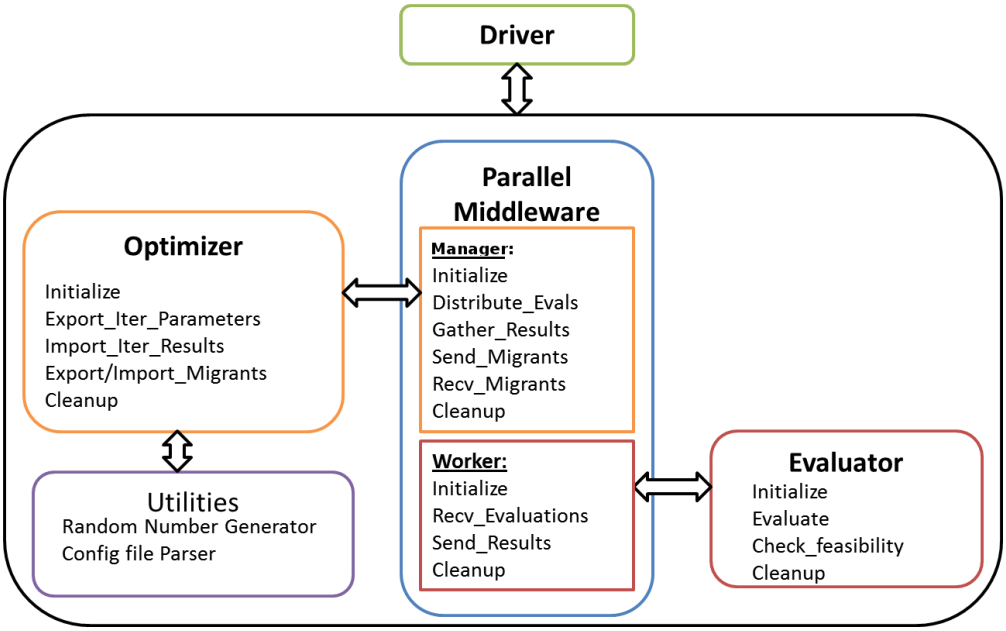

Figure 1: Basic architecture of the parallel optimization framework

and its major components. It also provides a brief overview of particle swarm optimization algorithms. Section 3 discusses specific problems in water distribution systems and the parallel EPANET(PEPANET) simulator used in this work.

2 Parallel Optimization Framework

This section describes the design of the parallel optimization framework that aids in coupling a heuristic optimizer with a target application. It provides the communication middleware between the optimizer and simulation components. This parallel framework facilitates seamless execution of the coupled optimization problem on high performance computing (HPC) systems. This framework is designed with support for multiple islands or swarms as required by the optimization algorithm and allows further experimentation with communication and migration patterns among the islands.

The framework facilitates the manager-worker paradigm for parallel execution. In the basic configuration, there would be one manager process and several associated workers. In the multi-swarm optimization scenario, there would be several manager processes and their corresponding workers. The manager processes execute the optimization component and the workers would execute the simulation component for the target application. The framework would provide support for periodic migration between swarms that can be handled by (a) blocking(synchronous) or (b) non-blocking(asynchronous) communication mechanisms.

2.1 Design

The optimization framework comprises of the following major components as illustrated in Figure 1:

- **Optimizer Module**: This encapsulates a supported optimization algorithm that interacts with the driver program and parallel middleware.
- **Evaluator Module**: This provides a streamlined interface to the optimization problem. The functions within this module are invoked by the driver program to compute the objective function value of the target application with given set of parameters.
- **Parallel Middleware**: The parallel middleware distributes the candidate solutions across all available processing elements (PEs) and collects the results once the problem evaluations are complete. It also provides additional communication operations in the case of multi-population optimization algorithms. This utilizes the Message Passing Interface(MPI)[Gropp et al., 1999] to provide support for parallelism.
- **Driver Module**: The driver module integrates the rest of the components together and drives program execution while retaining overall control.

2.2 Evaluator Module

The Evaluator module provides an interface to the target application including methods for initialization, termination and computing an evaluation of the test problem. The target optimization problem could be either a single or multi-objective problem taking an arbitrary number of real-valued and/or integer parameters and returning one or more real or integer valued objective values for each problem evaluation. A problem-specific evaluator, WDS (Water Distribution System) evaluator is used to model the leak detection and contamination source characterization problems in municipal water distribution systems.

2.3 Optimizer Module

This is an interface provided to an arbitrary population-based optimization algorithm. It provides function entry points to perform algorithm initialization, generate a set of evaluations, import results for a set of evaluations and termination at the end. This module interacts with the parallel middleware to distribute the objective function (usually simulation) evaluations and later retrieve the results of such evaluations. It then performs post-processing tasks specific to the optimization method and generates the population of candidate solutions for the next iteration. The Optimizer is also responsible to signal completion based on termination criteria either when a set number of iterations are completed or when the objective function value matches a desired threshold.

2.3.1 Swarm optimizer

As part of this research, we developed a particle swarm optimizer and coupled it with the Water Distribution Systems(WDS) simulator, EPANET. Presently, this optimizer implements a standard PSO algorithm [Kennedy et al., 2001b]. Additional efforts are underway to incorporate problem-specific knowledge into this optimizer.

2.3.2 Canonical PSO

This section provides a brief introduction to Particle swarm optimization (PSO). PSO is a swarm intelligence method wherein particles (representing candidate solutions) are 'flown' across parametric space to find optimal values [Kennedy et al., 2001a, Clerc, 2006]. It is a global optimization technique that belongs to the class of direct search methods. This method emulates the flocking or swarming behavior observed in nature among groups of birds or fish [Kennedy and Eberhart, 1995]. Each candidate solution is represented as a particle in the n-dimensional problem space and has an associated position and velocity attributes where n is the number of parameters being optimized. During every iteration, the velocity and position of a particle are updated using the information gleaned from that iteration to guide the search process. Each particle's position is influenced by the best position it has found so far (cognitive component) as well as the best position found by other particles in the swarm (social component). This method strives to find the optimal solution by learning and sharing such information among particles in the problem space.

Algorithm 1 Pseudocode for a Particle Swarm Optimizer.

```
1: GenerateInitialPopulation(pop)
2: for particle ← 1 to numParticles do
3:    Evaluate(particle) // Compute Objective function value
4: end for
5: while !StopCondition() do
6:    GetGlobalBest(pop)
7:    for particle ← 1 to numParticles do
8:       neighbors ← CalculateNeighborhood(topology, Position(particle))
9:       optional GetNeighborhoodBest(neighbors)
10:      UpdateVelocity(particle)
11:      UpdatePosition(particle)
12:   end for
13:   for particle ← 1 to numParticles do
14:      Evaluate(particle) // Compute Objective function value
15:   end for
16: end while
```

Algorithm 1 illustrates the steps in a standard particle swarm optimizer. The PSO algorithm starts with swarm of particles S whose positions are initialized to random locations

in the search space. During each iteration, the objective function value is computed for each particle.

$$\overrightarrow{\mathbf{v}}_i^t = w^t \overrightarrow{\mathbf{v}}_i^{t-1} + c_1\psi_1(\overrightarrow{\mathbf{p}}_i - \overrightarrow{\mathbf{X}}_i^{t-1}) + c_2\psi_2(\overrightarrow{\mathbf{p}}_g - \overrightarrow{\mathbf{X}}_i^{t-1}) \tag{1a}$$

$$\overrightarrow{\mathbf{X}}_i^t = \overrightarrow{\mathbf{X}}_i^{t-1} + \underbrace{\Delta\overrightarrow{\mathbf{X}}_i^t}_{\chi\overrightarrow{\mathbf{v}}_i^t} \tag{1b}$$

$\overrightarrow{\mathbf{X}}_i^t$ and $\overrightarrow{\mathbf{v}}_i^t$ represent the position and velocity vectors respectively of particle i during iteration t. Equation 1a illustrates the *UpdateVelocity* operation that is performed every iteration. $\overrightarrow{\mathbf{p}}_i$ denotes the personal best position found so far by a particle and $\overrightarrow{\mathbf{p}}_g$ represents the neighborhood best position. ψ_1 and ψ_2 are random variables drawn from a uniform random distribution $U(0,1)$. Equation 1b in turn shows how *UpdatePosition* operation is performed. w^t is known as the inertial weight responsible for the inertial component of a particle's velocity. It limits the rate at which particles congregate in problem space promoting better exploration. Larger values of w^t facilitate global exploration at the expense of convergence time, while smaller values of w^t result in shorter convergence time, but could result in convergence to a local optimum. w^t is gradually reduced over iterations to reduce the search space that a particle explores. To prevent scattering in problem space, Clerc [Clerc, 2006] introduced constriction coefficient (χ) to constrain the buildup of a particle's velocity. c_1 represents the cognitive coefficient that influences the pace at which a particle learns from its own experience so far. c_2 denotes the social efficient that determines how quickly information from other particles are incorporated in a particle's velocity.

3 Water Distribution Systems Applications

This section describes the contaminant source characterization and leak detection problems in water distribution systems and elaborates upon the problem formulation.

3.1 Water Distribution Systems Simulator

Our simulation component is based on EPANET [EPA, 2011], a freely available and widely used water distribution network hydraulic and water quality modeling tool from EPA. This model uses known pipe network topology, link/node physical characteristics (significantly, the water consumption rates over time), and network boundary and initial conditions, to simulate the space-time variation of flows, pressures, and water quality concentrations. The EPANET engine is available as a C language library with a well-defined API [Rossman, 1999].

Recently we have incorporated the multi-species version of EPANET (EPANET-MSX) into our parallel simulation-optimization toolkit [Sreepathi et al., 2007]. In addition to the functionalities offered by regular EPANET, EPANET-MSX can model complex reactions and transport of common contaminants and reagents in a water distribution system. In

this project, EPANET-MSX will be used as it can be used in the standard mode (regular EPANET) and multi-species mode. A typical EPANET simulation involves hydraulic steps (solve for nodal pressures and link flows) and water quality steps (solve for nodal concentrations). Each hydraulic step is typically followed by many water quality steps as the water quality time step is generally smaller than the hydraulic time step. Each hydraulic step entails a solution of nonlinear algebraic equations the size of nodes in the system.

As part of the first author's thesis work [Sreepathi, 2006, Sreepathi et al., 2007], he developed PEPANET to enable parallel execution of a large number of EPANET evaluations in an efficient manner in a high performance computing environment. PEPANET aggregates the EPANET (or MSX-EPANET) simulations into a single parallel execution for multiple sets of source characteristics to amortize the startup costs and minimize redundant computation. The MPI library [Gropp et al., 1999] was used for parallelizing PEPANET.

3.1.1 Problem formulation

The contaminant source characterization problem can be posed as parameter estimation for an inverse problem. The goal is to use the time-series of sensor data to recover the likely locations of the contaminant sources and their release histories (temporal mass loading history) that minimize the error between the predictions and the observations.

$$\text{Minimize} \qquad \left\{ \|\mathbf{C}_{obs} - C(\vec{\mathbf{m}})\|^2 \right\} \tag{2}$$

$$\text{Minimize} \qquad \left\{ w_1\|\mathbf{C}_{obs} - C(\vec{\mathbf{m}})\|^2 + w_2\|\mathbf{q}_{obs} - q(\vec{\mathbf{m}})\|^2 + w_3\|\mathbf{p}_{obs} - p(\vec{\mathbf{m}})\|^2 \right\} \tag{3}$$

Equations 2 and 3 describe the objective function formulations for the contaminant source characterization and leak detection problems respectively. Where $\mathbf{C}_{obs}$, $\mathbf{q}_{obs}$ and $\mathbf{p}_{obs}$ are vectors of concentration, flow and pressure measurements at the sensors obtained at various sampling times, respectively and $\mathbf{m}$ is the parameter vector being optimized. The weighting factors w_1, w_2, w_3 can be used to configure the relative importance of the various factors.

We built the functionality to solve the contaminant source characterization and leak detection problems using the PEPANET wrapper. The PEPANET module is linked to the optimization module either using (a) intermediate input/output files or (b) MPI-2 functions or (c) an Evaluator interface. The PEPANET program takes the decision variable sets (or parameter sets) from the optimization module and returns the objective values in an iterative fashion by running the simulations on parallel architectures.

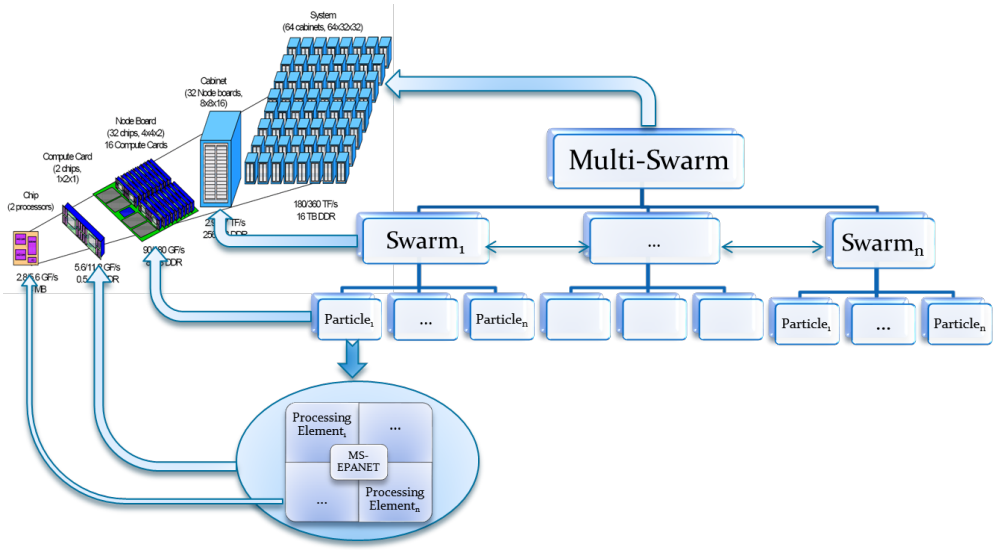

Figure 2: Multi-swarm optimizer framework displaying algorithmic and system architectural mapping

3.2 Multi-Swarm Optimization framework

Figure 2 illustrates the proposed multiswarm optimizer framework coupled with EPANET simulator. Periodically, swarms communicate information regarding the best parameter set found so far to each other. To the left, the diagram shows the system architecture of an IBM BlueGene supercomputer and the corresponding mapping of optimization algorithm components is shown on the right. The architecture of any modern supercomputer displays

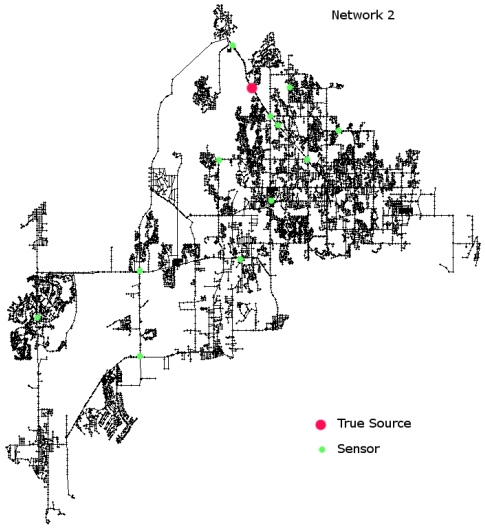

Figure 3: Water Distribution System network (Network2) used in this study

similar multi-level parallelism as illustrated here in the case of IBM BlueGene. Hence this kind of algorithm-system mapping would be feasible on other supercomputing architectures too.

4 Results

Performance results for the parallel multi-swarm optimization framework coupled with the water distribution systems simulator on the leadership class supercomputers at Oak Ridge National Laboratory will be presented at the conference. Figure 3 shows one of the water networks used in this study.

References

M. Clerc. *Particle Swarm Optimization*. Wiley-ISTE, 2006.

EPA. EPANET. http://www.epa.gov/nrmrl/wswrd/dw/epanet.html, 2011.

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

J Kennedy and R Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks Proceedings*, pages 1942–1948, 1995.

J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001a.

James Kennedy, Russell C.Eberhart, and Yuhui Shi. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001b.

L.A. Rossman. The EPANET programmer's toolkit. *In proceedings of Water Resources Planning and Management Division Annual Specialty Conference*, 1999.

Sarat Sreepathi. Cyberinfrastructure for Contamination Source Characterization in Water Distribution Systems. Master's thesis, North Carolina State University, December 2006. URL http://www.lib.ncsu.edu/resolver/1840.16/1446.

Sarat Sreepathi, Kumar Mahinthakumar, Emily Zechman, Ranji Ranjithan, Downey Brill, Xiaosong Ma, and Gregor von Laszewski. Cyberinfrastructure for contamination source characterization in water distribution systems. In *7th International Conference on Computational Science (ICCS 2007) Proceedings*, volume 4487 of *Lecture Notes in Computer Science*, pages 1058–1065, 2007.