

Evolutionary High-dimensional QoS Optimization for Safety-Critical Utility Communication Networks

Paskorn Champrasert*, Junichi Suzuki* and Tetsuo Otani**

*Department of Computer Science

University of Massachusetts, Boston, USA

{paskorn, jxs}@cs.umb.edu

**Central Research Institute of Electric Power Industry, Japan

ohtani@criepi.denken.or.jp

Abstract

This paper proposes and evaluates an evolutionary multiobjective optimization algorithm, called *EVOLT*, which heuristically optimizes QoS (quality of service) parameters in communication networks. *EVOLT* uses a population of individuals, each of which represents a set of QoS parameters, and evolves the individuals via genetic operators such as crossover, mutation and selection for satisfying given QoS requirements. For evaluating *EVOLT* in real-world settings that have high-dimensional parameter and optimization objective spaces, this paper focuses on QoS optimization in safety-critical communication networks for electric power utilities. Simulation results show that *EVOLT* outperforms a well-known existing evolutionary algorithm for multiobjective optimization and efficiently obtains quality QoS parameters with acceptable computational costs. Moreover, *EVOLT* visualizes obtained QoS parameters in a Self-Organizing Map in order to aid network administrators to intuitively understand the QoS parameters and the tradeoffs among optimization objectives.

Keywords: Quality of Service, Evolutionary Multiobjective Optimization, Self-Organizing Map, Power Utility Communication Networks

1 Introduction

This paper focuses on quality of service (QoS) optimization for safety-critical communication networks for electric power utilities. Power utilities leverage communication networks to monitor and control power delivery from power stations to consumers. Recently, the communication networks are increasingly required to adapt their QoS to frequent changes in their configuration and deployment, which are caused by organizational restructuring, deregulation, new power generation/delivery policies (e.g., distributed generation) and new software/hardware technologies.

QoS optimization is a combinatorial optimization problem to search the optimal configurations for network applications to satisfy given QoS requirements (e.g., data transmission latency) in a best-effort network whose capacity is limited. A network application's configuration is specified as a set of QoS parameters such as the parameters for flow control and data transmission reliability.

There exist four key research issues to solve the QoS optimization problem in real-world settings. First, this problem is known NP-hard [1], which can take a significant amount of time, labor and costs to find the optimal set(s) of QoS parameters from a huge number of parameter combinations. Second, it is also known a nonlinear problem [2], which linear optimization algorithms (e.g., [3–6]) do not work well for. Third, it often faces tradeoffs among conflicting QoS optimization objectives such as success rate, latency and jitter of data transmission. For example, in order to increase the success rate of data transmission, a network application may duplicate data and transmit duplicated data to a destination through multiple network routes. However, this can degrade the latency of data transmission due to increased network traffic and congestion. Thus, there exist multiple optimal sets (more precisely, the Pareto-optimal sets) of QoS parameters that satisfy QoS requirements. In practical decision-making by network administrators, it is important to reveal the optimal tradeoffs among conflicting optimization objectives by searching the Pareto-optimal parameter sets, instead of searching a single optimal parameter set as traditional QoS optimization algorithms do (e.g., [3–11]). The fourth issue is that, when the QoS optimization problem considers realistic network environments, it often possesses a number of QoS parameters and QoS optimization objectives. In general, optimization speed and quality degrade significantly when optimization problems have high-dimensional parameter and objective spaces [12–14]. High dimensionality in the parameter space leads to a combinatorial explosion of parameters, which greatly slows optimization

speed. High dimensionality in the objective space often leads to premature convergence, which fails to improve the optimization quality (or optimality) of QoS parameter sets. Traditional QoS optimization algorithms tend to deal with a limited number of parameters and optimization objectives; for example, less than 20 QoS parameters and three or less optimization objectives¹ [3–11, 15–22]. (An exception is [18], which considers 64 QoS parameters.)

In order to address these four research issues, this paper investigates an evolutionary QoS optimization algorithm, called *EVOLT*. It employs an evolutionary and heuristic optimization method because, in general, the method is robust in nonlinear NP-hard problems [23, 24]. *EVOLT* is designed as a multiobjective genetic algorithm (MOGA) that balances the tradeoffs among conflicting QoS optimization objectives and seeks the Pareto-optimal sets of QoS parameters that satisfy given QoS requirements. *EVOLT* uses a population of individuals, each of which represents a set of QoS parameters, and evolves the individuals through generations via genetic operators such as crossover, mutation and selection. Through the evolutionary process, *EVOLT* seeks the Pareto-optimal QoS parameters. In *EVOLT*, QoS requirements are considered as optimization constraints. Each constraint is defined as a tolerable QoS bound; for example, the upper bound in allowable latency of data transmission.

EVOLT implements new genetic operators specialized to handle high-dimensional parameter and objective spaces. Its aging operator identifies feasible and infeasible portions of each individual, which satisfy and violate given QoS requirements respectively, and preserves feasible portions across generations. This operator is intended to improve convergence speed in optimization when the parameter space is high-dimensional. *EVOLT*'s offspring size adjustment operator dynamically changes the number of offspring produced in each generation according to the current selection pressure. This operator is intended to avoid premature convergence and improve the optimality of QoS parameter sets by maintaining selection pressure in a high-dimensional objective space.

Moreover, *EVOLT* visualizes QoS parameter sets, which is high-dimensional data, in a low-dimensional (two dimensional) space with a self-organizing map (SOM). This aids network administrators to intuitively understand the tradeoffs among optimization objectives and the similarity among QoS parameter sets. It allows them to make well-informed decisions for choosing one of

¹In the research field of multiobjective optimization, an objective space is considered high-dimensional when it has more than three objectives [14].

QoS parameter sets and deploying that in their applications.

Another key feature of *EVOLT* is that it minimizes the number of constants that need to be manually configured. The performance of existing MOGAs is sensitive to constant values [25–27], and it is often very hard, if not impossible, for domain experts (e.g., network administrators) to tailor constant values for their problems. *EVOLT* has virtually no constants to configure manually. This feature makes *EVOLT* practical and usable for network administrators who do not know its algorithmic details.

This paper describes the design of *EVOLT* and evaluates its performance with simulated power utility communication networks that possess six optimization objectives and more than 850 QoS parameters. Simulation results show that individual operators in *EVOLT* work properly and they complement with each other well to deal with high-dimensional parameter and objective spaces. *EVOLT* outperforms a well-known existing MOGA, called NSGA-II [28], and efficiently obtains quality QoS parameters with acceptable computational costs.

2 Power Utility Communication Networks

This section overviews power utility communication networks (Section 2.1) and describes a set of configurations that *EVOLT* assumes for network structure and applications (Section 2.2), QoS parameters (Section 2.3) and QoS optimization objectives (Section 2.4). These configurations are obtained based on the experience that one of the authors has gained in the power utility industry.

2.1 Background

A power delivery system transmits generated electricity from power stations to consumers (Figure 1) [29, 30]. Electricity is transmitted in high voltage (e.g., 110 KV) from a power station in order to reduce energy loss in transmission, and distributed toward consumers through a chain of substations. Each substation is responsible for a certain physical region; for example, one for a state, one for a city in the state, and one for a town in the city. It reduces incoming voltage with a transformer(s) based on the electricity load in a region that it is responsible for. (The load in a region is determined by, for example, the number of consumers in the region. The higher load is required, the higher voltage a substation uses to deliver electricity.) This way, the volt-

age of generated power gradually reduces (e.g., to 1 KV) through a number of substations toward consumers.

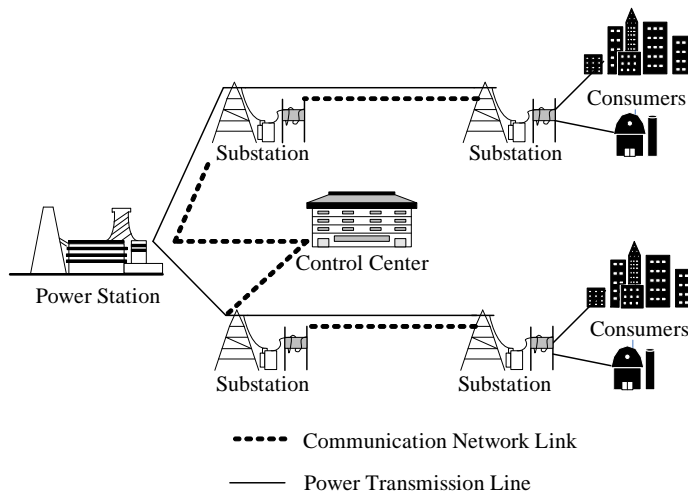


Figure 1: An Example Power Delivery System

Since most of substations and some of power stations are unmanned, power utilities remotely monitor and control them with communication networks [31]. Figure 1 shows an example network that consists of a control center, substations and a power station. Each substation and power station periodically monitors its operations and equipment (e.g., every few seconds), and transmits the monitored status information to a control center. A control center receives periodic updates on the status of substations and power stations and allows human operators to control their operations according to their current status.

2.2 Network Structure and Applications

A power utility communication network is often configured as a tree structure where a control center serves as its root (Figures 1 and 2). This paper considers two types of IP networks: a smaller-scale network of 34 nodes (a control center, 30 substations and 3 power stations) and a larger-scale network of 67 nodes (a control center, 60 substations and 6 power stations). In each network, there are two types of data communication routes between nodes: the primary and secondary routes (Figure 2). The primary routes are normally used for data transmission. The secondary routes are used when data is duplicated and transmitted redundantly through two different routes. (This is called a two route data transmission in this paper.) The Rapid Spanning Tree Protocol (IEEE

802.1w) [32] is used to avoid loops in network topology and reduce the time for spanning tree reconstruction. When a source node sends data to a destination node, the source node uses the topologically shortest path.

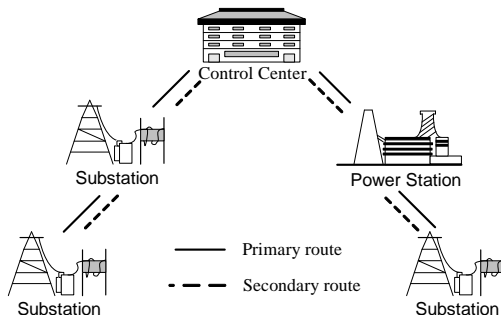


Figure 2: An Example Power Utility Communication Network

This paper also assumes two types of applications: a SCADA (Supervisory Control and Data Access) application and a maintenance application. Both applications are deployed on each node. In a SCADA application, each substation and power station periodically collects data on its operational status (e.g., a status of a substation’s circuit switching operation) and transmits it to a control center. In exchange, the control center periodically transmits data to substations and power stations for controlling their operations (e.g., on/off control for a substation’s circuit switching operation). In a maintenance application, each substation and power station periodically collects data on its equipment status and transmits the data to a control center. A SCADA application has more stringent QoS requirements than a maintenance application. See Section 4 for more configuration details of these two applications.

Each node operates two queues (or buffers) for data transmission: a policing queue and a shaping queue. A policing queue is used to receive incoming packets from remote nodes, queue them and pass them to the local node. A shaping queue is used to receive outgoing packets from the local node, queue them and transmit them toward their destinations.

2.3 QoS Parameters

EVOLT considers 13 QoS parameters described below. It optimizes them on each node so that both SCADA and maintenance applications satisfy given QoS requirements. This means *EVOLT* has 442 QoS parameters in total (13 parameters \times 34 nodes) in a smaller-scale network and 871

QoS parameters in total (13 parameters \times 67 nodes) in a larger-scale network.

Maximum size of a shaping queue (MSQ): The maximum number of packets that can be queued in a shaping queue. Its value range is $[0, 10]$ as an integer. If it is 0, traffic shaping is disabled. A shaping queue overflows if the number of queued packets exceeds this number.

Flush interval of a shaping queue (FSQ): The interval to flush packets from a shaping queue and transmit them to their destinations. Its value range is $[0, 100]$ as an integer. (Its unit is millisecond.) If it is 0, traffic shaping is disabled.

Maximum size of a policing queue (MPQ): The maximum number of packets that can be queued in a policing queue. Its value range is $[0, 10]$ as an integer. If $MPQ=0$, traffic policing is disabled. A policing queue overflows if the number of queued packets exceeds MPQ.

Flush interval of a policing queue (FPQ): The interval to flush packets from a policing queue and pass them to the local node. Its value range is $[0, 100]$ as an integer. (Its unit is millisecond.) If it is 0, traffic policing is disabled.

Aggregation size (AS): The number of packets that can be aggregated at a time in a shaping/policing queue. This number is used for both policing and shaping queues in the same node. When a queue contains more packets than this number, it aggregates those packets and transmits an aggregated packet to its destination even if its aggregation interval (AI; see below) has not expired yet. Packets are aggregated only when their application types (SCADA or maintenance) are same and their destinations are same. The range of this value is $[0, 10]$ as an integer. When this value is 0, packet aggregation is not performed.

Aggregation interval (AI): The time interval to aggregate packets in a shaping/policing queue. This number is used for both policing and shaping queues in the same node. When this number expires, queued packets are aggregated if their application types and their destinations are same. Its value range is $[0, 100]$ as an integer. (Its unit is millisecond.) If it is 0, packet aggregation is disabled.

Packet ordering (PO): When a node generates multiple packets at a time, it orders them in its shaping queue based on their size. The smaller the size of a packet is, the earlier it is dequeued and transmitted to its destination. This value is 0 or 1. 0 indicates packet ordering is disabled, and 1 indicates it is enabled. If it is disabled, a node injects generated packets to its shaping queue in a random order.

SCADA data duplication (SD): The number of duplicated SCADA data that a node transmits with the same route to their destination. Its value range is $[1, 5]$ as an integer. If it is 1, data duplication is disabled.

SCADA data duplication interval (SDI): The time interval to transmit duplicated SCADA data one by one with the same route to their destination. Its value range is $[0, 100]$ as an integer. (Its unit is millisecond.) When it is 0, data duplication is disabled.

SCADA multiple routes (SMR): The number of routes used to transmit duplicated SCADA data. Its value range is $[1, 2]$ as an integer. If it is 1, the primary route is used. If it is 2, both the primary and secondary routes are used.

Maintenance data duplication (MD): The number of duplicated maintenance data that a node transmits with the same route to their destination. Its value range is $[1, 5]$ as an integer. If this number is 1, data duplication is disabled.

Maintenance data duplication interval (MDI): The time interval to transmit duplicated maintenance data one by one with the same route to their destination. Its value range is $[0, 100]$ as an integer. (Its unit is millisecond.) When this value is 0, data duplication is not performed.

Maintenance multiple routes (MMR): The number of routes used to transmit duplicated maintenance data. Its value is 1 or 2. If it is 1, the primary route is used. If it is 2, both the primary and secondary routes are used.

2.4 Optimization Objectives

EVOLT considers the following three objectives in QoS optimization. Each of two (SCADA and maintenance) applications has these three objectives; *EVOLT* optimizes QoS parameters with respect to six objectives in total. A QoS requirement is assigned to each of these objectives. It serves as a constraint in the optimization process in *EVOLT*.

Success Rate (F_S): The average success rate of data transmissions from a source node to a destination node. This objective is to be maximized:

$$F_S = \frac{R}{T} \quad (1)$$

R denotes the number of data received at a destination node. T denotes the number of data

transmitted from a source node. The expected arrival time of each transmitted data is calculated by dividing the data's size by network bandwidth. If a destination node does not receive the data within a tolerable time bound after the expected arrival time, the data is assumed to be lost. (The data is not counted to calculate success rate, even if it arrives at the destination after this tolerable time bound.) If a destination node receives more than one duplicated data, only one of them is counted for the numerator (R) of Equation 1.

Latency (F_L): The average latency in packet transmissions from a source node to a destination node. This objective is to be minimized:

$$F_L = \frac{\sum_{p=1}^{N_p} L_p}{N_p} \quad (2)$$

L_p denotes the transmission latency of packet p , which is the interval between the time when a source node sends out p and the time when a destination node receives it. N_p denotes the total number of packets transmitted between a source node and a destination node.

Jitter (F_J): The average jitter in data transmissions from a source node to a destination node. It indicates the timeliness variation of data arrivals at a destination node. This objective is to be minimized:

$$F_J = \frac{\sum_{d=1}^{N_d} J_d}{N_d} \quad (3)$$

N_d denotes the total number of data types transmitted between a source node and a destination node. (See Table 1 for more details on the data types.) J_d denotes the temporal average jitter in transmitting data d . It is calculated as the exponentially weighted moving average (EWMA) of the current and past jitter values:

$$J_d = \text{EWMA}_{jitter}(t) = \alpha * |A_d - E_d| + (1 - \alpha) * \text{EWMA}_{jitter}(t - 1) \quad (4)$$

A_d and E_d denote the actual and estimated arrival times of data d .

2.5 Safety-Critical Applications in Power Utility Communication Networks

In order to ensure stable and safe power delivery, it is critical to operate power utility applications, particularly SCADA applications, in a real-time manner by satisfying given QoS requirements. For example, data loss, delay and high jitter in data transmission in these applications can damage electric devices or cause fire at consumer sites due to power current fluctuations. They can also cause a substation failure(s) and in turn a cascading failure of substations. In 2003, a large-scale cascading blackout in Northeast America was caused partially by defects in operating SCADA applications [33]. The blackout triggered malfunction of other infrastructures such as water supply, transportation, telephone lines and the Internet. It affected 10 million people in Canada and 45 million people in eight states in the U.S.

3 *EVOLT*: The Proposed Evolutionary Optimization Algorithm

This section describes *EVOLT*'s algorithmic structure and its operators.

3.1 Individuals

In *EVOLT*, each individual represents a set of QoS parameters. It consists of multiple *segments*, each of which represents a node in the network. Therefore, the number of segments in each individual is equal to the total number of nodes in the network. Figure 3 shows the structure of each individual. SS1 to SS n represent the first to the n -th substations. PS1 to PS m represent the first to the m -th power stations. CC represents the control center. Figure 3 shows 13 QoS parameters for the second substation.

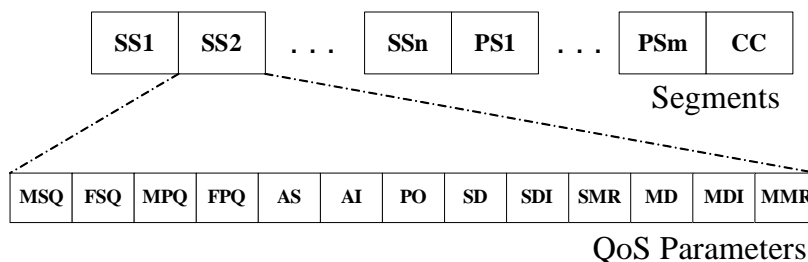


Figure 3: The Structure of an Individual

3.2 Evolutionary Optimization Process

EVOLT runs on the control center. Every time a packet arrives at its destination node, the node measures QoS on a packet by packet basis. It periodically reports historical QoS measures to the control center. Using the reported QoS measures as objective values, *EVOLT* performs its evolutionary optimization process to adjust QoS parameters. The adjusted QoS parameters are transmitted from the control center to individual nodes so that the nodes use them in their subsequent data transmissions.

Figure 4 shows the algorithmic structure of evolutionary optimization in *EVOLT*. The initial population (P^0) consists of μ individuals that contain randomly-generated QoS parameters. In each generation (g), a pair of individuals, called parents (p_1 and p_2), are chosen from the current population P^g using the binary tournament operator (`BTournament()`) [34]. A binary tournament randomly takes two individuals from P^g , compares them based on their fitness values, and chooses a superior one (i.e., the one whose fitness is higher) as a parent.

Each segment of an individual maintains its *age*. The age value of a segment indicates how many generations a node represented by the segment has satisfied optimization constraints (i.e., QoS requirements). If this aging method is enabled, two parents (p_1 and p_2) update the ages of their segments (`Aging()`). Then, they reproduce two offspring and mutate the offspring's QoS parameters (`AgingCrossoverMutation()`).

If aging is not enabled, two parents (p_1 and p_2) reproduce two offspring (q_c^1 and q_c^2) with the fitness-based crossover (`FitnessCrossover()`). Each offspring is mutated with a regular mutation operator (`Mutation()`), which randomly alters each QoS parameter in the offspring at the probability of $1/n$ where n denotes the total number of QoS parameters in each individual.

Binary tournament, crossover and mutation operators are performed repeatedly until the number of offspring reaches the offspring size ($|Q^g| = \lambda$). Once λ offspring are reproduced, they are combined with the parent population P^g . Then, a selection process occurs to sort $\mu + \lambda$ individuals in $P^g \cup Q^g$ and choose the top μ individuals as the next generation's population (P^{g+1}). This sorting is driven by the fitness of individuals (`FitnessSelection()`) or by their diversity as well as their fitness (`DiversitySelection()`).

The number of reproduced offspring (λ) can be dynamically adjusted on a generation by gen-

eration basis (`OffspringSizeAdjustment()`). If this adjustment is enabled, the offspring size in the next generation is re-computed to adjust the selection pressure in evolution and the density of individuals in the objective space. *EVOLT* terminates its evolutionary optimization process when the number of generations reaches a given limit ($g = g_{max}$).

```

main
   $g \leftarrow 0$ 
   $P^0 \leftarrow$  Randomly generated  $\mu$  individuals
  repeat
     $Q^0 \leftarrow \emptyset$ 
    repeat
       $p_1 \leftarrow$  BTournament( $P^g$ )
       $p_2 \leftarrow$  BTournament( $P^g$ )
      if Aging is enabled
        then  $\begin{cases} \text{AGING}(p_1) \\ \text{AGING}(p_2) \\ q_m^1, q_m^2 \leftarrow \text{AGINGCrossoverMutation}(p_1, p_2) \end{cases}$ 
      else  $\begin{cases} q_c^1, q_c^2 \leftarrow \text{FitnessCrossover}(p_1, p_2) \\ q_m^1 \leftarrow \text{Mutation}(q_c^1) \\ q_m^2 \leftarrow \text{Mutation}(q_c^2) \end{cases}$ 
      if  $q_m^1 \notin Q^g$ 
        then  $Q^g \leftarrow Q^g \cup q_m^1$ 
      if  $q_m^2 \notin Q^g$ 
        then  $Q^g \leftarrow Q^g \cup q_m^2$ 
      until  $|Q^g| = \lambda$ 
      if Diversity-aware ranking is enabled
        then  $P^{g+1} \leftarrow \text{DiversitySelection}(P^g \cup Q^g)$ 
        else  $P^{g+1} \leftarrow \text{FitnessSelection}(P^g \cup Q^g)$ 
      if Offspring size adjustment is enabled
        then  $\lambda \leftarrow \text{OffspringSizeAdjustment}()$ 
       $g \leftarrow g + 1$ 
    until  $g = g_{max}$ 

```

Figure 4: Evolutionary Optimization Process in *EVOLT*

3.3 Fitness Calculation based on Constraint-based Dominance Relationship

As described in Section 3.2, the notion of fitness is used in several operators in *EVOLT*. It quantifies how an individual is superior or inferior to the others. It is determined with *constraint-based dominance relationships* among individuals. The relationships rank individuals based on the objective values (i.e., QoS measures) and constraint violation (i.e., QoS requirement violation) that they yield. Individual X_i is said to *constraint-dominate* X_j if:

- X_i does not violate any constraints but X_j does,
- both X_i and X_j violate at least one constraints, and X_i dominates X_j with respect to their constraint violation, or
- both X_i and X_j do not violate any constraints, and X_i dominates X_j with respect to their objective values.

X_i is said to *dominate* X_j with respect to their constraint violation if:

- $V_k(X_i) \leq V_k(X_j)$ for all $k = 1, 2, \dots, m$, and
- $V_k(X_i) < V_k(X_j)$ for at least one $k \in 1, 2, \dots, m$

$V_k(X_i)$ denotes the violation that X_i yields in the k -th constraint. A constraint violation is the difference between a constraint value (i.e., QoS requirement) and an objective value (i.e., QoS measure).

X_i is said to *dominate* X_j with respect to their objective values if:

- $F_k(X_i) \leq F_k(X_j)$ for all $k = 1, 2, \dots, m$, and
- $F_k(X_i) < F_k(X_j)$ for at least one $k \in 1, 2, \dots, m$

$F_k(X_i)$ denotes the objective value that X_i yields in the k -th objective. It is assumed here that all objectives are to be minimized.

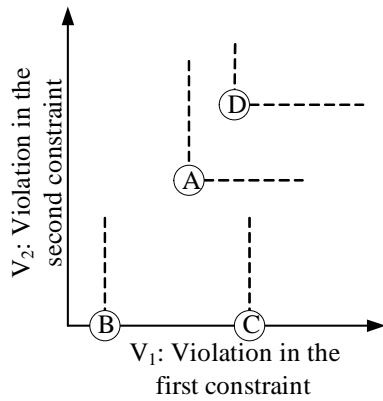


Figure 5: An Example Constraint Space

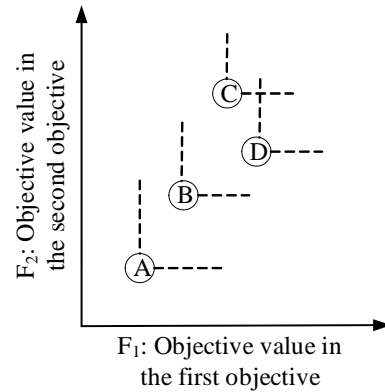


Figure 6: An Example Objective Space

Figure 5 shows an example two dimensional constraint space that illustrates the constraint violation by four individuals (A to D). A and D violate two constraints. B and C violate the first constraint. B dominates A , C and D with respect to their constraint violation because B violates less in both constraints than A , C and D . A and C do not dominate each other with respect to their constraint violation. D is dominated by A and C because D violates more in both constraints than A and C .

Figure 6 shows an example two-dimensional objective space that illustrates the objective values of four individuals (A to D). Both objectives are assumed to be minimized. A dominates B , C and D with respect to their objective values because A outperforms the other three individuals in both objectives. For the same reason, B dominates C and D . C and D do not dominate each other because one of them does not outperform the other in both objectives, and vice versa.

Considering Figures 5 and 6 simultaneously, B constraint-dominates A , C and D . A and C do not constraint-dominate each other, but do constraint-dominate D .

Fitness is calculated for each individual (X_i) as follows.

$$Fitness(X_i) = \mu - d_{X_i} \quad (5)$$

μ denotes the population size, and d_{X_i} denotes the number of individuals that constraint-dominate X_i . In an example of Figures 5 and 6, B 's fitness is four ($4 - 0$) because no individuals dominate it and the population size is four. The fitness of A and C is three ($4 - 1$) because they are dominated by an individual: B . D 's fitness is one.

3.4 Aging, Age-based Crossover and Age-based Mutation Operators

The proposed aging operator (`Aging()` in Figure 4) assigns age to every segment of each individual. The age value of a segment indicates how many generations a node represented by the segment has satisfied constraints (i.e., QoS requirements). It is incremented on a generation by generation basis if the node continues to satisfy constraints. It is reset to zero when the node violates at least one constraints. This way, the proposed aging operator distinguishes feasible and infeasible segments in each individual, which satisfy and violate constraints, respectively.

Figure 7 shows how age-based crossover and mutation are designed in `AgingCrossoverMutation()`.

(See also Figure 4.) p_1 and p_2 denote two parent individuals. s indicates the number of segments in an individual. $AGE_{p_1[i]}$ denotes the age of p_1 's i -th segment. Age-based crossover and mutation occur on a segment by segment basis. Crossover occurs on the i -th segment if two parents have the same age for the segment ($AGE_{p_1[i]} = AGE_{p_2[i]}$). **Crossover()** performs one-point crossover [34] among $p_1[i]$ and $p_2[i]$. If two parents have different ages for the i -th segment ($AGE_{p_1[i]} \neq AGE_{p_2[i]}$), a segment with a higher age is copied to two offspring segments. In this case, one-point crossover does not occur. As depicted in Figure 7, the proposed age-based crossover operator is designed to prioritize and preserve feasible segments in the population across generations, thereby improving optimization/convergence speed.

```

procedure AGINGCROSSOVERMUTATION( $p_1, p_2$ )
  for  $i \leftarrow 1$  to  $s$ 
    do {
      if ( $AGE_{p_1[i]} = AGE_{p_2[i]}$ )
        then  $\{q_c^1[i], q_c^2[i] \leftarrow \text{CROSSOVER}(p_1[i], p_2[i])$ 
      else if ( $AGE_{p_1[i]} > AGE_{p_2[i]}$ )
        then  $\{q_c^1[i] \leftarrow p_1[i]$ 
               $q_c^2[i] \leftarrow p_1[i]$ 
      else if ( $AGE_{p_2[i]} > AGE_{p_1[i]}$ )
        then  $\{q_c^1[i] \leftarrow p_2[i]$ 
               $q_c^2[i] \leftarrow p_2[i]$ 
       $q_m^1[i] \leftarrow \text{AGINGMUTATION}(q_c^1[i])$ 
       $q_m^2[i] \leftarrow \text{AGINGMUTATION}(q_c^2[i])$ 
    }
  return ( $q_m^1, q_m^2$ )

```

Figure 7: Age-based Crossover and Mutation

Figure 8 shows how the proposed age-based mutation operator (**AgingMutation()** in Figure 7) is designed to alter QoS parameters in a given segment of an individual (q_c). n denotes the number of QoS parameters in a segment. At the probability of $1/n$, each QoS parameter is mutated. The l -th QoS parameter ($q_c[l]$) is randomly altered to $q_m[l]$ with a normal distribution that has $q_c[l]$ as its mean and σ_i as its standard deviation.

```

procedure AGINGMUTATION( $q_c$ )
  for  $l \leftarrow 1$  to  $n$ 
    do {
      if  $\text{random}(0, 1) \leq 1/n$ 
        then  $q_m[l] \leftarrow N(q_c[l], \sigma_i)$ 
    }
  return ( $q_m$ )

```

Figure 8: Age-based Mutation

σ_i is called *mutation strength* for a segment in question (the i -th segment). Note that mutation strength is adjusted to each segment. σ_i controls the degree of mutation on QoS parameters; a higher mutation strength produces a probabilistically bigger difference between $q_c[l]$ and $q_m[l]$. With Equation 6, the proposed age-based mutation operator dynamically adjusts σ_i on a generation by generation basis.

$$\sigma_i^{g+1} = \frac{1}{|AGE_{p_1[i]} - AGE_{p_2[i]}| + 1} \sigma_i^g \quad (6)$$

σ_i at the g -th generation (σ_i^g) is adjusted by considering the age difference of the i -th segment between two parent individuals (p_1 and p_2). When the difference is large, the proposed operator decreases σ_i^g for exploiting $q_c[l]$ to favor a local search. When the difference is small, the proposed operator increases σ_i^g to favor exploration rather than exploitation. As far as $p_1[i]$ and $p_2[i]$ satisfy all given constraints, σ_i continues to be adjusted. ($AGE_{p_1[i]}$ and $AGE_{p_2[i]}$ continue to grow as well.) Once either $p_1[i]$ or $p_2[i]$ violates at least one constraints, the proposed operator resets σ_i to its initial value in order to perform the default global search. ($AGE_{p_1[i]}$ and $AGE_{p_2[i]}$ are reset to zero as well.) *EVOLT* is designed to exploit feasible segments in its mutation process, thereby improving optimization/convergence speed.

3.5 Fitness-based Crossover Operator

The proposed fitness-based crossover operator (`FitnessCrossover()` in Figure 4) assigns QoS parameters to offspring based on their parents' fitness values. A superior parent (a parent that has a higher fitness value) makes a higher contribution than the other to determine offspring's QoS parameters. As a result, offspring are reproduced more similar to a superior parent. The proposed operator is designed to increase optimization/convergence speed by taking advantage of parents' fitness values.

Figure 9 shows how the proposed crossover operator takes two parents (p_1 and p_2), examines their QoS parameters ($p_1[i]$ and $p_2[i]$) and fitness values ($fitness(p_1)$ and $fitness(p_2)$), and calculates offspring's QoS parameters ($q_c^1[i]$ and $q_c^2[i]$). n denotes the number of QoS parameters in an individual. Offspring's i -th QoS parameters ($q_c^1[i]$ and $q_c^2[i]$) are determined with the Euclidean distances ($d_1[i]$ and $d_2[i]$) from the average of two parents' QoS parameters ($center[i]$). The

distances are calculated based on parents' fitness values. For example, $d_1[i]$ is calculated in proportion to the ratio of p_1 's fitness ($fitness(p_1)$) to the summation of fitness values of two parents ($fitness(p_1) + fitness(p_2)$).

```

procedure FITNESSCROSSOVER( $p_1, p_2$ )
  for  $i \leftarrow 1$  to  $n$ 
     $center[i] \leftarrow (p_1[i] + p_2[i])/2$ 
     $d_1[i] \leftarrow \frac{fitness(p_1)}{fitness(p_1)+fitness(p_2)} * \frac{|p_2[i]-p_1[i]|}{2}$ 
     $d_2[i] \leftarrow \frac{fitness(p_2)}{fitness(p_1)+fitness(p_2)} * \frac{|p_2[i]-p_1[i]|}{2}$ 
  do  $\left\{ \begin{array}{l} \text{if } random(0, 1) > 0.5 \\ \text{then } \begin{cases} q_c^1[i] \leftarrow center[i] - d_1[i] \\ q_c^2[i] \leftarrow center[i] + d_2[i] \end{cases} \\ \text{else } \begin{cases} q_c^1[i] \leftarrow (p_1[i] - \frac{|p_2[i]-p_1[i]|}{2}) + d_1[i] \\ q_c^2[i] \leftarrow (p_2[i] + \frac{|p_2[i]-p_1[i]|}{2}) - d_2[i] \end{cases} \end{array} \right.$ 
  return ( $q_c^1, q_c^2$ )

```

Figure 9: Fitness-based Crossover Operator

Figure 10 shows an example on how to determine offspring's QoS parameters. This example assumes that the second parent (p_2) has a higher fitness value than the first parent (p_1): $fitness(p_2) > fitness(p_1)$. Thus, $d_2[i]$ is longer than $d_1[i]$; offspring are placed closer to p_2 than p_1 .

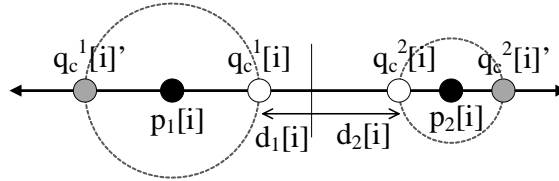


Figure 10: An Example Fitness-based Crossover

The proposed crossover operator is designed following a property in Holland's schema theorem [34,35], which proves that one-point crossover contributes to improve the average fitness values of individuals through generations. Given this property, Holland's schema theorem assumes that offspring are placed either inside or outside the region bounded by their parents. The proposed crossover operator emulates this property as shown in Figure 10; offspring's QoS parameters are placed as either $p_1[i] < q_c^1[i] < q_c^2[i] < p_2[i]$ or $q_c^1[i]' < p_1[i] < p_2[i] < q_c^2[i]'$ at the probability of 50%. This way, the proposed operator is intended to improve the average fitness values of individuals

through generations.

3.6 Diversity-aware Ranking Operator

Once λ offspring are reproduced via crossover and mutation, *EVOLT* ranks $\mu + \lambda$ (i.e., $|P^g \cup Q^g|$) individuals and selects the top μ of them as the individuals in the next generation (P^{g+1}). As described in Figure 4, *EVOLT* performs a fitness-based selection operator (`FitnessSelection()`) or a diversity-aware selection operator (`DiversitySelection()`). The fitness-based selection operator computes each individual’s fitness with the notion of constraint-based dominance (Section 3.3), ranks individuals based on their fitness values and selects the top μ of them.

The diversity-aware selection operator in *EVOLT* ranks individuals based on their diversity in the objective space as well as their fitness values. It computes each individual’s fitness with the notion of constraint-based dominance (Section 3.3), and computes each individual’s diversity with the notion of crowding distance [28]. A crowding distance indicates how an individual is distant from its nearest neighbors in the objective space. Thus, an individual with a higher crowding distance exists in a less crowded region in the objective space. The proposed diversity-aware selection operator plots individuals in a two dimensional space whose axes represent their fitness and diversity. Then, it determines the dominance relationships among individuals with respect to the two axes and ranks them from the ones with higher fitness and diversity to the ones with lower fitness and diversity. Finally, it selects the top μ individuals as the next generation’s individuals. The proposed diversity-aware selection operator is designed to maintain the diversity of individuals in order to reveal the trade-offs among conflicting objectives.

3.7 Offspring Size Adjustment Operator

This operator dynamically changes the number of offspring reproduced in a generation (λ in Figure 4) in order to adjust the density of individuals in the objective space as well as the selection pressure of individuals. In this paper, selection pressure (ψ) is measured as follows:

$$\psi = \frac{\mu + \lambda}{\mu} \tag{7}$$

μ denotes the population size. Selection pressure indicates how hard individuals can survive

to the next generation; a higher selection pressure means that individuals have lower chances to survive to the next generation. It is known that a low selection pressure significantly degrades optimization/convergence speed [34]. The proposed offspring size adjustment operator is designed to maintain a reasonably high selection pressure by adjusting λ in Equation 7.

The density of individuals in the objective space (η) is measured as follows:

$$\eta = \frac{\mu + \lambda}{\gamma} \quad (8)$$

γ denotes the volume of the objective space. In a higher-dimensional objective space, it is harder to determine dominance relationships among individuals because individuals have higher chances to be non-dominated with each other [14]. This often leads to premature convergence, which fails to improve the optimization quality of individuals. The proposed offspring size adjustment operator is designed to alleviate this problem by increasing λ in Equation 8 and in turn maintaining the density of individuals in the high-dimensional objective space.

The size of offspring is adjusted as follows based on those in the current (the g -th) and previous (the $(g - 1)$ -th):

$$\lambda_{g+1} = \lambda_g + \left(\frac{\lambda'_{g-1}}{\lambda_{g-1}} - \frac{\lambda'_g}{\lambda_g} \right) \lambda_g \quad (9)$$

λ_g denotes the number of offspring reproduced at the g -th generation, and λ'_g denotes the number of offspring that survive to the next generation through a selection process (Section 3.6). Thus, $\frac{\lambda'_g}{\lambda_g}$ indicates the survival ratio of offspring. If it is lower than the survival ratio at the previous generation ($\frac{\lambda'_{g-1}}{\lambda_{g-1}}$), the proposed operator considers that convergence/evolution does not proceed well due to a lack of enough selection pressure and/or individual density in the objective space. Therefore, the operator increases the number of offspring reproduced in the next generation (λ_{g+1}). Conversely, if $\frac{\lambda'_g}{\lambda_g} > \frac{\lambda'_{g-1}}{\lambda_{g-1}}$, the operator decreases λ_{g+1} .

3.8 Constants in *EVOLT*

Each operator in *EVOLT* is carefully designed to minimize the number of constants to be manually configured. For example, the proposed fitness-based crossover operator eliminates the constants for crossover rate and the number of cross sections. The proposed age-based mutation operator

eliminates the constants for mutation range and strength.

EVOLT has three constants: mutation rate, the initial mutation strength and population size. However, they all are trivial to configure. *EVOLT* uses $1/n$ as mutation rate. (n denotes the number of QoS parameters in a segment. See Figure 8.) This is a widely accepted design for mutation rate in the field of evolutionary algorithms. The initial mutation strength is set to one, and mutation strength is dynamically adjusted at runtime (Section 3.4). The population size is fixed as a constant; however, the size of reproduced offspring is dynamically adjusted (Section 3.7).

4 Simulation Evaluation

This section shows a set of simulation results to evaluate the performance of *EVOLT*.

4.1 Simulation Configurations

All simulations were carried out with a modified Java Network Simulator (JNS)², a Java implementation of the ns-2 simulator³. Two types of utility communication networks are simulated. A smaller-scale network consists of 34 nodes: a control center, 30 substations and 3 hydro power stations. A larger-scale network consists of 67 nodes: a control center, 60 substations and 6 hydro power stations. Both networks are constructed with a tree topology, as shown in Figure 2, and a bandwidth of 10 Mbps. The packet loss rate is 10^{-10} on each link between two nodes.

A SCADA application and a maintenance application are deployed on each node. Table 1 shows a set of data types used in the two applications. There are 16 data types: 8 SCADA data types (S1 to S8) and 8 maintenance data types (M1 to M8). All of these 16 types of data are periodically transmitted. The tolerable time bounds in arrival delay of SCADA data are 1 second for S1 to S4 data types and 0.25 second for S5 and S8 data types. No tolerable time bound is given maintenance application data.

Table 2 shows a set of QoS requirements (or optimization constraints) for SCADA and maintenance data transmissions in a smaller-scale and a larger-scale networks. Table 3 shows a set of algorithmic configurations for *EVOLT* and NSGA-II, which is a well-known evolutionary multi-objective optimization algorithm [28]. Note that the maximum number of generations in a single

²<http://jns.sourceforge.net/>

³<http://www.isi.edu/nsnam/ns/>

Table 1: Simulated Data Types

Data Type	Source	Destination	Data Size
S1	Substation	Control center	1 bytes
S2	Substation	Control center	6 bytes
S3	Power station	Control center	1 bytes
S4	Power station	Control center	6 bytes
S5	Control center	Substation	2 bytes
S6	Control center	Substation	6 bytes
S7	Control center	Power station	2 bytes
S8	Control center	Power station	6 bytes
M1	Substation	Control center	450 bytes
M2	Substation	Control center	3,600 bytes
M3	Substation	Control center	200 bytes
M4	Substation	Control center	400 bytes
M5	Substation	Control center	50 bytes
M6	Substation	Control center	50 bytes
M7	Substation	Control center	200 bytes
M8	Substation	Control center	300 bytes

simulation (g_{max}) is 100 and 300 in a smaller-scale network and a larger-scale network, respectively. Every experimental result is the average of 10 independent simulation results. All experiments are carried out with a smaller-scale network except in Section 4.4.

Table 2: QoS Requirements (optimization constraints)

		Latency (sec)	Jitter (sec)	Success Rate (%)
Smaller-scale network	SCADA	0.8	0.4	99
	Maintenance	2.0	0.9	95
Larger-scale network	SCADA	1.6	0.8	99
	Maintenance	4.0	1.8	95

Table 3: Algorithmic Configurations in *EVOLT* and NSGA-II

Configuration	<i>EVOLT</i>	NSGA-II
EWMA coefficient (α in Equation 4)	0.8	0.8
g_{max}	100 or 300	100
μ	100	100
Mutation rate	$1/n$	$1/n$
Crossover rate	N/A	0.9
Degree of SBX crossover	N/A	15
Degree of polynomial mutation	N/A	20

4.2 Comparison of *EVOLT* and NSGA-II

Table 4 compares *EVOLT* and NSGA-II with three metrics: distribution, span and QoS violation. Distribution is a diversity metric that measures the degree of uniform distribution of individuals in the objective space. It is computed as the standard deviation of Euclidean distances among

individuals:

$$D = \sqrt{\frac{\sum_{i=1}^{N-1} (d_i - \bar{d})^2}{N - 1}} \quad (10)$$

d_i denotes the Euclidean distance between an individual (the i -th individual) and its closest neighbor (the $(i + 1)$ -th individual) in the objective space. \bar{d} denotes the mean of d_i . N denotes the number of individuals in the population. The objective space is normalized to compute the distribution metric. Lower distribution means that individuals are more uniformly distributed.

Span is another diversity metric that measures how widely individuals explore and cover the objective space. It is calculated as the maximum Euclidean distance between two individuals:

$$S = \max_{i,j \in \mu} \left(\sqrt{\sum_{k=1}^n (x_i[k] - x_j[k])^2} \right) \quad (11)$$

μ denotes the population. i and j denote individuals in the population. n is the number of objectives. $x_i[k]$ denotes i 's objective value in the k -th objective. Higher span means that individuals spread more widely.

The QoS violation metric indicates how many individuals violate at least one QoS requirements.

Table 4 shows the average and standard deviation of 10 independent simulation results. A bold font face indicates which algorithm outperforms in a metric in question. The symbols * and ** are placed when the average results of *EVOLT* and NSGA-II are statistically different (via t-test) with the 95% and 99% significance levels, respectively. As Table 4 demonstrates, *EVOLT* outperforms NSGA-II in all three metrics. Particularly, in the span and QoS violation metrics, *EVOLT* outperforms NSGA-II at the significance level of 99%.

Table 4: Comparison of *EVOLT* and NSGA-II with the Distribution, Span and QoS Violation Metrics

Algorithm		Distribution	Span	QoS Violation
<i>EVOLT</i>	Average	0.021	0.28**	0**
	SD	0.003	0.113	0
NSGA-II	Average	0.03	0.061	23.7
	SD	0.004	0.075	41.67

Tables 5 and 6 compare *EVOLT* and NSGA-II with respect to QoS optimization objectives in SCADA and maintenance data transmissions. In both the average and standard deviation results,

EVOLT outperforms NSGA-II in all objectives except jitter in maintenance data transmission. On the average basis, *EVOLT* satisfies all QoS requirements in both SCADA and maintenance data transmissions. In fact, as Table 4 shows, all *EVOLT* individuals satisfy all QoS requirements in 10 simulations. In contrast, on the average basis, NSGA-II violates the jitter requirement in SCADA data transmission. As Table 4 shows, more than 20% of individuals in the NSGA-II population violate at least one QoS requirements.

Table 5: QoS Comparison in SCADA Data Transmission

Algorithm		Latency (sec)	Jitter (sec)	Success Rate (%)
<i>EVOLT</i>	Average	0.715	0.340	100
	SD	0.036	0.023	0
NSGA-II	Average	0.749	0.401	100
	SD	0.282	0.216	0
QoS requirement		0.8	0.4	99

Table 6: QoS Comparison in Maintenance Data Transmission

Algorithm		Latency (sec)	Jitter (sec)	Success Rate (%)
<i>EVOLT</i>	Average	1.54	0.871	100
	SD	0.49	0.46	0
NSGA-II	Average	1.80	0.671	100
	SD	0.604	0.342	0
QoS requirement		2.0	0.9	95

Table 7 compares *EVOLT* and NSGA-II with the \mathcal{C} -metric [36]. The metric is defined as follows in order to compare two algorithms: A and B .

$$\mathcal{C}(A, B) = \frac{|\{b \in B \mid \exists a \in A : a \succ b\}|}{|B|} \quad (12)$$

$a \succ b$ means that a constraint-dominates b . $\mathcal{C}(A, B)$ is calculated as the fraction of B 's individuals that at least one individual of A constraint-dominates. Thus, if $\mathcal{C}(A, B) = 1$, all of B 's individuals are constraint-dominated by at least one of A 's individuals. As shown in Table 7, $\mathcal{C}(EVOLT, NSGA-II)$ is significantly greater than $\mathcal{C}(NSGA-II, EVOLT)$. This complements the results of Tables 5 and 6 and demonstrates that *EVOLT* outperforms NSGA-II with respect to QoS optimization objectives.

Tables 4 to 7 consistently illustrate that the operators in *EVOLT* work properly to find quality QoS parameters and *EVOLT* outperforms NSGA-II.

Table 7: Comparison of *EVOLT* and NSGA-II with the \mathcal{C} -metric

\mathcal{C} -metric	\mathcal{C} -metric Value (%)
$\mathcal{C}(EVOLT, NSGA-II)$	97
$\mathcal{C}(NSGA-II, EVOLT)$	0

4.3 Variations of *EVOLT*

This section analyzes the impacts of *EVOLT*'s genetic operators on its performance. For this analysis, five variations of *EVOLT* are configured and evaluated:

1. *Baseline*: enables *EVOLT*'s fitness-based crossover operator and disables aging, diversity-aware ranking and offspring size adjustment. (The baseline configuration differs from NSGA-II in that it performs its fitness-based crossover operator rather than SBX.)
2. *Baseline+A*: enables aging and disables fitness-based crossover on top of the baseline configuration.
3. *Baseline+D*: enables diversity-aware ranking on top of the baseline configuration.
4. *Baseline+O*: enables offspring size adjustment on top of the baseline configuration.
5. *EVOLT*: enables aging, diversity-aware ranking and offspring size adjustment and disables fitness-based crossover on top of the baseline configuration.

Table 8 compares these *EVOLT* variations with respect to the distribution, spread and QoS violation metrics. *Baseline+D* yields the best distribution result it considers the distribution of individuals in its selection process. *Baseline+O* is the worst in distribution. *EVOLT* yields the best spread performance. This means that the spread of individuals improves by combining the genetic operators in *EVOLT*. No individuals violate QoS requirements in all variations.

Table 8: Comparison of *EVOLT* Variations with the Distribution, Spread and QoS Violation Metrics

Variation	Distribution	Spread	QoS Violation
<i>EVOLT</i>	0.021	0.28	0
<i>Baseline</i>	0.028**	0.26*	0
<i>Baseline+A</i>	0.027**	0.23*	0
<i>Baseline+D</i>	0.018	0.25	0
<i>Baseline+O</i>	0.032**	0.27	0

Table 8 also shows whether the result of *EVOLT* is statistically different (via t-test) from those of the other four variations. With respect to distribution, *EVOLT* statistically outperforms *Baseline*, *Baseline+A* and *Baseline+O* at the significance level of 99%. The results of *EVOLT* and *Baseline+D* are statistically at the same level. With respect to spread, *EVOLT* statistically outperforms *Baseline* and *Baseline+A* at the significance level of 95%. These results demonstrate that genetic operators complement with each other well in *EVOLT* to balance distribution and spread while satisfying QoS requirements.

Table 9 compares *EVOLT*'s variations and NSGA-II with the \mathcal{C} -metric. $\mathcal{C}(\text{EVOLT}, \text{NSGA-II})$ and $\mathcal{C}(\text{NSGA-II}, \text{EVOLT})$ are 97% and 0%, respectively, as discussed in Section 4.2 (Table 7). All *EVOLT*'s variations outperform NSGA-II. For example, $\mathcal{C}(\text{Baseline}, \text{NSGA-II}) > \mathcal{C}(\text{NSGA-II}, \text{Baseline})$ ($46 > 11$). *Baseline+A* and *Baseline+O* tend to outperform the other variations; aging and offspring size adjustment successfully improve the optimality of individuals.

Table 9: Comparison of *EVOLT* Variations with the \mathcal{C} Metric (%)

\mathcal{C}	NSGA-II	EVOLT	Baseline	Baseline+A	Baseline+D	Baseline+O
NSGA-II	–	0	11	0	20	0
EVOLT	97	–	47	23	34	13
Baseline	46	32	–	13	27	3
Baseline+A	78	46	26	–	82	47
Baseline+D	39	29	37	3	–	8
Baseline+O	84	64	54	16	79	–

Figure 11 compares the optimality of *EVOLT*'s variations with the generation distance (GD) metric. This metric indicates the minimum distance between non-constraint-dominated individuals and the Utopian point in the normalized objective space:

$$GD = \min_{i \in \mu} \sqrt{\sum_{k=1}^n (x_i[k])^2} \quad (13)$$

μ denotes the population. n is the number of objectives. $x_i[k]$ denotes the i -th non-constraint-dominated individual's objective value in the k -th objective. Objective values of success rate are computed as $(1 - \text{success rate})$ so that the Utopian point is placed at $(0, 0, 0)$. A lower generation distance means that individuals converge better and closer to the Utopian point.

As Figure 11 illustrates, *EVOLT* variations converge individuals generation by generation. *Baseline+A* yields the fastest convergence; aging successfully improves convergence speed by pre-

serving feasible segments in individuals across generations. *EVOLT*'s convergence speed is the second fastest.

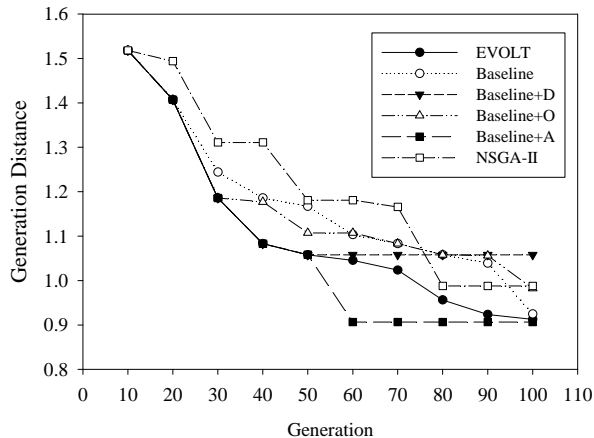


Figure 11: Comparison of *EVOLT* Variations with the Generation Distance Metric

Table 10 shows the number of generations that *EVOLT* and NSGA-II requires to achieve a given GD. *EVOLT* always reach a given GD faster than NSGA-II. The average speedup is 1.98. This means that *EVOLT* converges approximately two times faster than NSGA-II.

Table 10: Comparison of *EVOLT* and NSGA-II in Convergence Speed

GD	<i>EVOLT</i> (# of generations)	NSGA-II (# of generations)	Speedup
1.8	1	3	3
1.6	3	5	1.67
1.4	5	5	1
1.2	5	8	1.6
1.05	18	54	3
1.03	52	85	1.63

Table 8, Table 9 and Figure 11 demonstrate that *EVOLT*'s operators complement with each other well and their combination successfully balances the optimality, diversity (i.e., distribution and spread) and convergence speed of individuals while satisfying given QoS requirements.

4.4 Scalability Analysis

This section discusses the scalability of *EVOLT* by comparing simulation results in a smaller-scale and larger-scale networks. Tables 11 and 12 show QoS objective values and QoS violation in a smaller-scale network and a larger-scale network, respectively. In both networks, *EVOLT* satisfies

all QoS requirements in SCADA and maintenance data transmissions. No individuals violate QoS requirements. Tables 11 and 12 demonstrate that *EVOLT* scales to the network size (i.e., the number of nodes in the network).

Table 11: QoS in SCADA Data Transmission

		Latency (sec)	Jitter (sec)	Success Rate (%)	QoS Violation
Smaller-scale	Agerage	0.715	0.340	100	0
	SD	0.036	0.023	0	0
Larger-scale	Agerage	1.52	0.754	100	0
	SD	0.069	0.039	0	0

Table 12: QoS in Maintenance Data Transmission

		Latency (sec)	Jitter (sec)	Success Rate (%)	QoS Violation
Smaller-scale	Agerage	1.54	0.871	100	0
	SD	0.49	0.46	0	0
Larger-scale	Agerage	3.07	1.76	100	0
	SD	1.02	0.71	0	0

5 Visualization of QoS Parameters with Self-Organizing Maps

When *EVOLT* finishes its evolution process at the last generation, it provides non-constraint-dominated individuals, as its solutions, to network administrators. They choose one of the solutions to adapt their network applications' QoS performance. However, it is not always straightforward and obvious for network administrators to decide which one to choose because different solutions can yield very different QoS results even though all of them are feasible and non-constraint-dominated. For example, a solution may yield low latency and high jitter while another solution may yield high latency and low jitter. Moreover, high dimensionality in the objective space make it harder for network administrators to choose a solution that can yield their desired/preferred QoS performance.

EVOLT visualizes its solutions with a self-organizing map (SOM)⁴ in order to aid network administrators to intuitively understand the tradeoffs among QoS optimization objectives and the similarity among solutions. It maps each solution, which is high-dimensional (six dimensional) data, on a low-dimensional (two dimensional) space. For example, Figure 12 shows a 25×25 SOM that maps 100 solutions obtained from a particular simulation with a smaller-scale network. (All

⁴SOM is an unsupervised classifier that classifies high-dimensional data in a low-dimensional space [37].

individuals are constraint-dominated at the last generation.) The number in a cell indicates the number of solutions mapped to the cell. The SOM illustrates that 43 solutions in the right bottom cell (i.e., the (25, 25) cell) yield similar QoS results. Their QoS results are very different from those of the individuals at distant cells; for example, five individuals in the (1, 2) cell and 13 individuals in the (1, 4) cell.

Figures 13 to 16 shade Figure 12’s cells in four different ways based on the objective values of solutions. For example, Figures 13 and 14 indicate the average latency and jitter in SCADA data transmissions. These two figures illustrates that 43 solutions in the (25,25) cell yield low latency (lower than 0.6 second) and high jitter (higher than 0.35 second). 6 solutions in the (11, 12) cell yield high latency (higher than 0.7 second) and low jitter (lower than 0.3 second). 6 individuals in the (25, 15) cell yield mid-range latency (between 0.6 and 0.7 second) and mid-range jitter (between 0.3 and 0.35 second). Therefore, if a network administrator places a higher priority to latency than jitter, it is reasonable for him/her to choose a solution from the (25,25) cell rather than the (11, 12) cell. If he/she places the same level of priority to latency and jitter, a solution in the (25, 15) cell is a reasonable choice for him/her. This way, *EVOLT* allows network administrators to make well-informed decisions for choosing one of given solutions and deploying that in their applications.

6 Related Work

This paper describes a set of extensions to the authors’ prior work [38]. This paper investigates a new operators, diversity-aware ranking and offspring size adjustment operators, which the prior work did not focus on. It also studies a SOM-based visualization method for non-dominated individuals; it was beyond the scope of the prior work. Moreover, this paper carries out more extensive simulation studies than the prior work.

There are several research efforts that apply genetic algorithms (GAs) to optimize operational parameters in power systems [39–42]. [39–41] study parameter optimization for controllers (e.g., programmable logic controllers; PLC) in substations and control centers in order to, for example, stabilize the power current in circuit switching devices. [42] leverages a GA to find the optimal size of equipment (e.g. turbine size) in a power station and the optimal locations of power stations to be constructed in a power delivery system. Unlike these existing work, this paper studies QoS

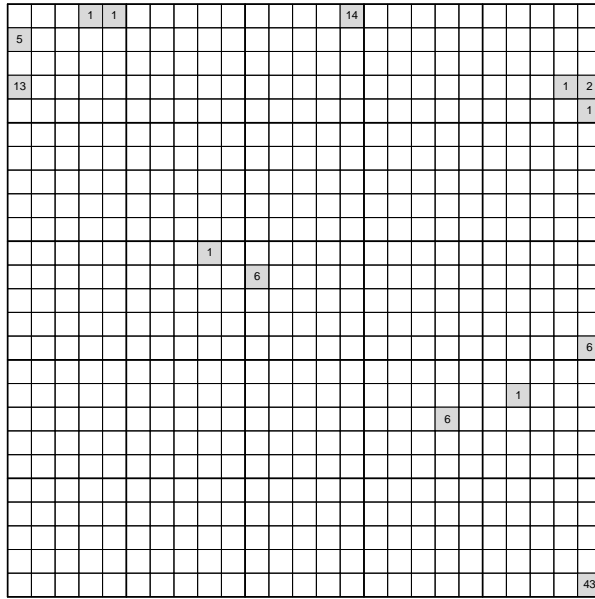


Figure 12: Solutions mapped on a 25×25 SOM

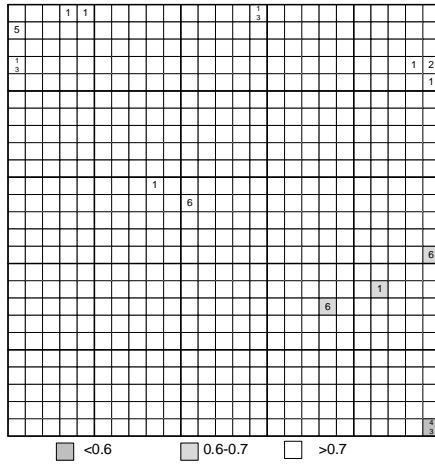


Figure 13: SCADA Latency

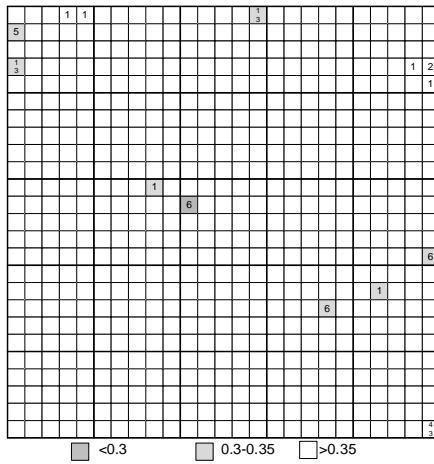


Figure 14: SCADA Jitter

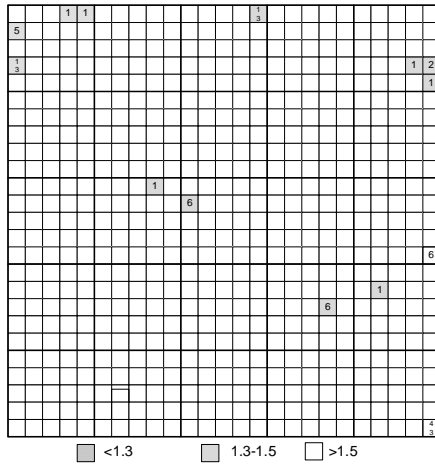


Figure 15: Maintenance Latency

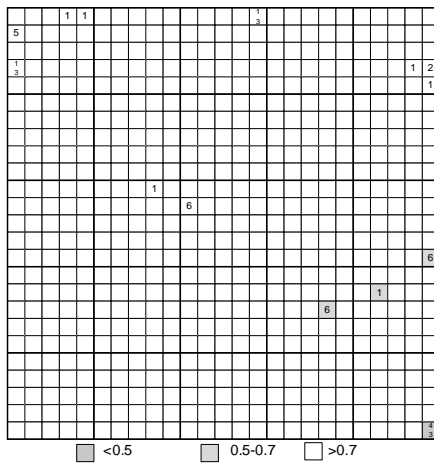


Figure 16: Maintenance Jitter

optimization in a power utility communication network that connects a control center, substations and power stations.

As discussed in Section 1, existing QoS optimization algorithms have considered a limited number of QoS parameters and optimization objectives [3–11, 15–22]. None of them consider high-dimensional parameter and objective spaces as *EVOLT* does.

Several linear optimization algorithms have been proposed for the QoS optimization problem [3–6]. However, they are not designed to seek the optimal tradeoffs among conflicting optimization objectives. They also have a scalability issue; their computational costs increase exponentially as their parameter space grows.

In order to solve large-scale QoS optimization problems, it is required to use heuristic algorithms such as GAs [7–11]. In general, GAs scale better than linear optimization algorithms. However, it is not always straightforward to manually tune weight values in the fitness functions of classical GAs. (A classical GA has a fitness function as a weighted sum of objective values.) Also, classical GAs do not seek the optimal tradeoffs among conflicting objectives.

Multiobjective GAs (MOGAs) avoid these issues in classical GAs. They seek the optimal tradeoff (or Pareto-optimal) solutions and have no weight parameters to manually configure in their fitness functions [15–18]. Unlike existing MOGAs, *EVOLT* is designed to handle high-dimensional parameter and objective spaces well, minimize the number of manually-configured constants in genetic operators and visualize non-dominated individuals in a low-dimensional (two dimensional) SOM space.

Parameter-less GA [43] and Meta-GA [44] intend to reduce the number of manually-configured constants. Parameter-less GA eliminates a constant for the population size and adjusts it dynamically. However, it introduces a new constant to determine how often the population size is adjusted. Meta-GA introduces an extra (or meta-level) GA to optimize the constants in a regular GA. However, the meta-level GA has a set of constants to be configured manually. Unlike these existing GAs, *EVOLT* has no constants that are hard to configure, as discussed in Section 3.

7 Conclusion

This paper proposes and evaluates a multiobjective GA (MOGA), called *EVOLT*, which optimizes QoS parameters in power utility communication networks that have high-dimensional parameter and objective spaces. Simulation results show that individual operators in *EVOLT* work properly and complement with each other to handle high-dimensional parameter and objective spaces well. *EVOLT* outperforms a well-known existing MOGA, called NSGA-II, and efficiently obtains quality QoS parameters that satisfy given QoS requirements. Moreover, *EVOLT* visualizes QoS parameter sets in a two dimensional SOM space in order to aid network administrators to intuitively understand the similarity among QoS parameter sets and the tradeoffs among optimization objectives.

Several extensions are planned as future work. A key extension is dimensionality reduction in the parameter and objective spaces. Several dimensionality reduction algorithms will be investigated to reduce the dimensions of the parameter and objective spaces before and/or during an optimization process in *EVOLT*.

References

- [1] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, 1996.
- [2] Y. Diao, J. L. Hellerstein, and S. Parekh. Optimizing Quality of Service Using Fuzzy Control. In *Proc. of IFIP/IEEE Int'l Workshop on Distributed Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications*, pages 42–53, 2002.
- [3] Y. Diao, N. Gandhi, J.L. Hellerstein, S. Parekh, and D.M. Tilbury. Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics with Application to the Apache Web Server. In *Proc. of IEEE/IFIP Network Operations and Management Symposium*, pages 219–234, 2002.
- [4] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley. Performance Specifications and Metrics for Adaptive Real-time Systems. In *Proc. of IEEE Real-Time Systems Symposium*, pages 13–23, 2000.
- [5] D. A. Menasce, D. Barbara, and R. Dodge. Preserving QoS of E-commerce Sites through Self-tuning: A Performance Model Approach. In *Proc. of ACM Conference on Electronic Commerce*, pages 224–234, 2001.

- [6] Z. Liu, M. S. Squillante, and J. L. Wolf. On Maximizing Service Level Agreement Profits. In *Proc. of ACM Conference on Electronic Commerce*, pages 213–223, 2001.
- [7] A. T. Haghghat, K. Faez, M. Dehghan, A. Mowlaei, and Y. Ghahremani. GA-based Heuristic Algorithms for QoS Based Multicast Routing. *Knowledge-Based Systems*, 16(5-6):305–312, 2003.
- [8] L. Barolli, A. Koyama, and N. Shiratori. A QoS Routing Method for Ad-hoc Networks Based on Genetic Algorithm. In *Proc. of Int'l Workshop on Database and Expert Systems Applications*, pages 175–179, 2003.
- [9] A. Riedl. A Hybrid Genetic Algorithm for Routing Optimization in IP Networks Utilizing Bandwidth and Delay Metrics. In *Proc. of IEEE Workshop on IP Operations and Management*, pages 166–170, 2002.
- [10] F. Xiang, L. Junzhou, W. Jieyi, and G. Guanqun. QoS Routing Based on Genetic Algorithm. *Computer Communications*, 22(15):1392–1399, 1999.
- [11] D. Montana, T. Hussain, and T. Saxena. Adaptive Reconfiguration of Data Networks Using Genetic Algorithms. In *Proc. of ACM Conference on Genetic and Evolutionary Computation*, pages 1141–1149, 2002.
- [12] V. Khare, X. Yao, and K. Deb. Performance Scaling of Multi-Objective Evolutionary Algorithms. In *Proc. of Int'l Conference on Evolutionary Multi-Criterion Optimization*, pages 376–390, 2003.
- [13] R. C. Purshouse and P. J. Fleming. Evolutionary Many-Objective Optimization: An Exploratory Analysis. In *Proc. of IEEE Congress on Evolutionary Computation*, pages 2066–2073, 2003.
- [14] H. Ishibuchi, N. Tsukamoto, Y. Hitotsuyanagi, and Y. Nojima. Effectiveness of Scalability Improvement Attempts on the Performance of NSGA-II for Many-Objective Problems. In *Proc. of ACM Conference on Genetic and Evolutionary Computation*, pages 649–656, 2008.
- [15] A. Roy and S. K. Das. QM²RP: a QoS-based Mobile Multicast Routing Protocol Using Multi-objective Genetic Algorithm. *Wireless Networks*, 10(3):271–286, 2004.
- [16] B. Sun and L. Li. Optimizing on Multiple Constrained QoS Multicast Routing Algorithms Based on GA. *Journal of Systems Engineering and Electronics*, 15(4):677–683, 2004.
- [17] H. Meunier, E. G. Talbi, and P. Reininger. A Multiobjective Genetic Algorithm for Radio Network Optimization. In *Proc. of IEEE Congress on Evolutionary Computation*, pages 317–324, 2000.
- [18] A. Koyama, L. Beralli, K. Matsumoto, and B. O. Apduhan. A GA-based Multi-purpose Optimization Algorithms for QoS Routing. In *Proc. of IEEE Int'l Conference on Advanced Information Networking and Applications*, pages 23–28, 2004.
- [19] T. Ye and S. Kalyanaraman. A Recursive Random Search Algorithm for Large-scale Network Parameter Configuration. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):196–205, 2003.

- [20] A. Orda. Routing with End-to-end QoS Guarantees in Broadband Networks. *IEEE/ACM Transactions on Networking*, 7(3):365–374, 1999.
- [21] T. F. Abdelzaher and K. G. Shin. End-host Architecture for QoS-adaptive Communication. In *Proc. of IEEE Symposium on Real-Time Technology and Applications*, pages 121–130, 1998.
- [22] S. Chen and K. Nahrstedt. An Overview of Quality of Service Routing for Next-generation High-speed Networks: Problems and Solutions. *IEEE Network*, 12(6):64–79, 1998.
- [23] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Son, 2001.
- [24] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.
- [25] F. G. Lobo, C. F. Lima, and Z. Michalewicz. *Parameter Setting in Evolutionary Algorithms*. Springer, 2007.
- [26] A.E. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith. Parameter Control in Evolutionary Algorithms. In *Studies in Computational Intelligence*, volume 54, pages 19–46. Springer, 2007.
- [27] F. G. Lobo and D. E. Goldberg. The Parameter-less Genetic Algorithm in Practice. *Information Sciences: An International Journal*, 167(1-4):217–232, 2004.
- [28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
- [29] J. Northcote-Green and R. Wilson. *Control and Automation of Electrical Power Distribution Systems*. CRC Press, 2006.
- [30] H. W. Beaty. *Electric Power Distribution Systems: A Nontechnical Guide*. PennWell Books, 1998.
- [31] M. Shahidehpour and Y. Wang. *Communication and Control in Electric Power Systems: Applications of Parallel and Distributed Processing*. Wiley-IEEE, 2003.
- [32] IEEE Computer Society. 802.1D: IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges. June 2004.
- [33] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez-Gasca, et al. Causes of the 2003 Major Grid Blackouts in North America and Europe, and Recommended Means to Improve System Dynamic Performance. *IEEE Transactions on Power Systems*, 20(4):1922–1928, 2005.
- [34] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley., 1989.
- [35] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.
- [36] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.
- [37] T. Kohonen. The Self-organizing Map. *Neurocomputing*, 21(1-3):1–6, 1998.

- [38] P. Champrasert, J. Suzuki, and T. Otani. Constraint-based Evolutionary QoS Adaptation for Power Utility Communication Networks. In *Proc. of IEEE Int'l Conference on Tools with Artificial Intelligence*, pages 395–403, 2009.
- [39] J. W. Finch and M. R. Besmi. Genetic Algorithms applied to a Power System Stabilizer. In *Proc. IEEE Int'l Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 100–105, 1995.
- [40] I. J. Ramírez-Rosado and J. L. Bernal-Agustín. Genetic Algorithms applied to the Design of Large Power Distribution Systems. *IEEE Transactions on Power Systems*, 13(2):696–703, 1998.
- [41] Y. L. Abdel-Magid and M. A. Abido. Optimal Multiobjective Design of Robust Power System Stabilizers using GA. *IEEE Transactions on Power Systems*, 18(3):1125–1132, 2003.
- [42] J. E. Lansberry and L. Wozniak. Adaptive Hydrogenerator Governor Tuning with a Genetic Algorithm. *IEEE Transactions on Energy Conversion*, 9(1):179–185, 1994.
- [43] G. R. Harik and F. G. Lobo. A Parameter-less Genetic Algorithm. In *Proc. of ACM Conference on Genetic and Evolutionary Computation*, pages 258–265, 1999.
- [44] J. Clune, S. Goings, B. Punch, and E. Goodman. Investigations in Meta-GAs: Panaceas or Pipe Dreams? In *Proc. of ACM Conference on Genetic and Evolutionary Computation*, pages 235–241, 2005.