Constraint-based Evolutionary QoS Adaptation
for Power 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 studies an evolutionary multiobjective optimization algorithm, called EVOLT, which heuristically optimizes QoS (quality of service) in communication networks for electric power utilities. EVOLT uses a population of individuals, each of which represents a set of QoS parameters, and evolves them via genetic operators such as crossover and mutation for satisfying given QoS requirements. 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.

I. Introduction

This paper focuses on QoS (Quality of Service) adaptation in communication networks for electric power utilities. Power utilities use communication networks to control and monitor power delivery from power stations to consumers through a number of substations. Recently, the communication networks are increasingly required to adapt their QoS to frequent changes in their configuration and deployment, which can be caused by organizational restructuring, deregulation, new power delivery policies (e.g., distributed generation) and new software/hardware technologies.

This requirement brings a challenge in tuning QoS parameters. (They include the parameters for packet priority control, flow control and data transmission reliability.) QoS parameter tuning is a process to find a right set of QoS parameters for each power utility application to satisfy its QoS requirements such as a requirement for packet transmission latency. The challenge in QoS parameter tuning is attributed mainly to its NP-hard complexity [1]. The granularity of QoS parameters tends to be fine so that applications can be highly configurable and adaptive for various QoS requirements. This increases the number of QoS parameters and the number of parameter combinations. Moreover, different applications have different QoS requirements. For example, a SCADA (Supervisory Control and Data Access) application and other applications often have very different QoS requirements. It is often time-consuming, error-prone and even chaotic to manually tune QoS parameters for multiple applications.

In order to address the above challenge, this paper proposes and evaluates a constraint-based evolutionary multiobjective optimization algorithm, called *EVOLT*, which heuristically seeks the Pareto-optimal QoS parameters for each application to satisfy given QoS requirements. *EVOLT* uses a population of individuals, each of which represents a set of QoS parameters, and evolves them via genetic operators (e.g., crossover and mutation) for optimizing them. In *EVOLT*'s evolutionary optimization process, QoS metrics (e.g. transmission latency) and QoS requirements are considered as optimization objectives and constraints, respectively. A QoS requirement is defined as a tolerable QoS bound; for example, the highest allowable latency.

This paper describes the design of *EVOLT* and evaluates its performance through simulations. 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.

II. Power Utility Communication Networks

A power delivery system is intended to transmit generated electricity from power stations to consumers [2] (Figure 1). 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 voltage of generated power gradually reduces (e.g., to 1 KV) through a number of substations toward consumers.
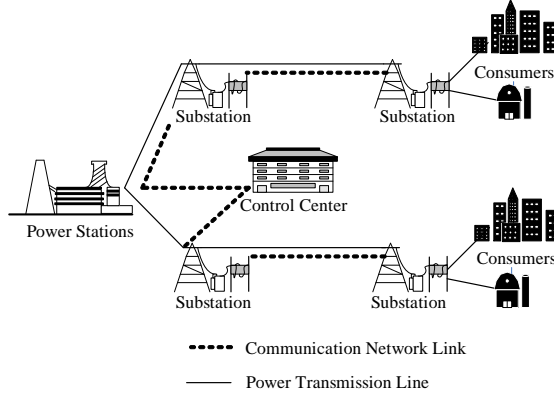
Fig. 1: An Example Power Utility Comm. Network

Since most of substations and some of power stations are unmanned, power utilities remotely monitor and control them with communication networks [3]. Figure 1 shows an example network that consists of a control center, a power station and substations. Each of substations and power stations periodically monitors its operations and equipment (e.g., every few seconds), and transmits the monitored state information to a control center. A control center receives periodic updates on the states of substations and power stations and allows human operators to control their operations according to their current states.

A power utility communication network is often configured as a tree structure in which a control center serves as the root node (Figure 1). This paper assumes an IP network of 34 nodes: a control center, 30 substations and 3 power stations. There are two types of data communication routes between nodes: the primary and secondary routes 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 two route data transmission in this paper.) When a source node sends data to a destination node, the source node uses the topologically shortest path.

This paper also assumes two types of applications: SCADA (Supervisory Control and Data Access) application and 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 state and transmits it to a control center. A SCADA application has higher QoS requirements than a maintenance application.

Each node has two queues 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 to their destinations.

III. *EVOLT*: The Proposed Algorithm

This section presents QoS parameters (Section III-A) and QoS optimization objectives (Section III-B) that *EVOLT* considers. Sections III-C to III-H describe *EVOLT*'s algorithm structure and its operators.

A. 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.

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 a 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.

B. Optimization Objectives

EVOLT considers the following three objectives in QoS optimization. Each of two (SCADA and maintenance) applications has those 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 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:

$$F_S = \frac{R}{T} \tag{1}$$

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 ignored 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 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:

$$F_L = \frac{\sum_{p=1}^{N_p} L_p}{N_p} \tag{2}$$

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 that arrive at a destination node.

Jitter (F_J)**:** The average jitter in data transmissions from a source node to a destination node. This objective is to be minimized:

$$F_J = \frac{\sum_{p=1}^{N_d} J_d}{N_d} \tag{3}$$

N_d denotes the total number of data types. J_d denotes the temporal average jitter of transmitting the same type of data d. It is calculated as the exponentially weighted moving average (EWMA) of the current and past jitter values:

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

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

C. Individuals

In *EVOLT*, each individual represents a set of QoS parameters. It consists of multiple segments, each of which represent 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 2 visualizes the structure of an individual. SS1 to SSn represent the first to n-th substations. PS1 to PSm represent the first to m-th power stations. CC represents a control center. Figure 2 shows 13 QoS parameters for the second substation.
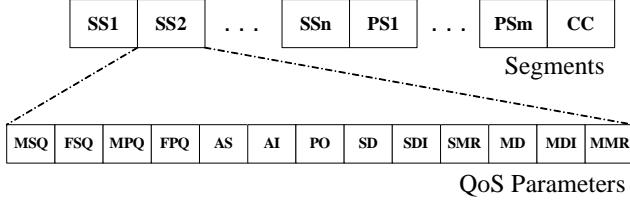
| SS1 | SS2 | . . . | SSn | PS1 | . . . | PSm | CC |

Segments

| MSQ | FSQ | MPQ | FPQ | AS | AI | PO | SD | SDI | SMR | MD | MDI | MMR |

QoS Parameters

Fig. 2: The Structure of an Individual

D. Evolutionary Optimization Process

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

Figure 3 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 a binary tournament operator (`BTounament()`). 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.

Two parents reproduce two offspring (q_c^1 and q_c^2) with a crossover operator (`crossover()`). Each offspring is mutated with a mutation operator (`mutation()`); its QoS parameters are altered. The binary tournament, crossover and mutation operators are performed repeatedly until the number of offspring ($|Q^g|$) reaches the population size (μ).

Once μ offspring are reproduced, they are combined with the parent population P^g. Then, a selection operator (`selection()`) selects the top μ individuals from 2μ individuals in $P^g \cup Q^g$ as the next generation's population (P^{g+1}). This selection is driven based on fitness values of individuals. For all segments of each individual in P^{g+1}, an aging operator (`aging()`) determines their age. The notion of age is used to indicate which segments are superior to the others in an individual. (The higher it is, the more superior its corresponding segment is to the others.)

EVOLT terminates its evolutionary optimization process when the number of the generations (g) reaches its maximum limit (g_{max}).

E. Dominance-based Fitness Calculation

As described in Section III-D, 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 relation-*

```
main
  g ← 0
  P^0 ← Randomly generated μ individuals
  Q^0 ← ∅
  repeat
    repeat
      p_1 ← BTounament(P^g)
      p_2 ← BTounament(P^g)
      q_c^1, q_c^2 ← Crossover(p_1, p_2)
      q_m^1 ← Mutation(q_c^1)
      q_m^2 ← Mutation(q_c^2)
      if q_m^1 ∉ Q^g
        then Q^g ← Q^g ∪ q_m^1
      if q_m^2 ∉ Q^g
        then Q^g ← Q^g ∪ q_m^2
    until |Q^g| = μ
    P^{g+1} ← Selection(P^g ∪ Q^g)
    Aging(P^{g+1})
    g ← g + 1
  until g = g_max
```

Fig. 3: Evolutionary Optimization Process in *EVOLT*

ships among individuals. The relationships rank individuals based on the QoS measures (or objective values) and constraint 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, but X_i dominates X_j with respect to constraint violation, or
- both X_i and X_j do not violate any constraints, but X_i dominates X_j with respect to objective values.

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

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

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

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

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

$F_k(X_i)$ denotes the objective value that X_i yields in the k-th objective.

Figure 4 shows an example two-dimensional constraint space that illustrates constraint violation relationships among four individuals (A to D). A and D violate two constraints. B and C violate the first constraint. Figure 5 shows an example objective space that illustrates dominance relationships among individuals with two objectives to be minimized. A is the best in both objectives; it is *non-dominated*. 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 vise versa. Given Figures 4 and 5, B is *non-constraint-dominated* because its constraint violation is minimum in both of two constraints. A and C do not constraint-dominate each other because one of
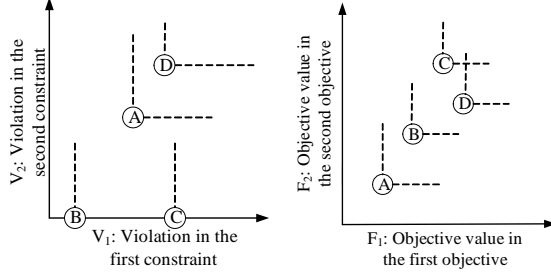
Fig. 4: An Example Constraint Violation Relationship

Fig. 5: An Example Dominance Relationship
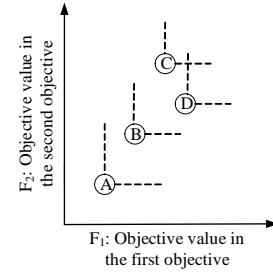
them does not yield lower violation than the other in both constraints, and vice versa. D is constraint-dominated by A and C.

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

$$Fitness(X_i) = \mu - d' \qquad (5)$$

μ denotes the population size, and d' denotes the number of individuals that constraint-dominate X_i. Thus, non-constraint-dominated individuals have the highest fitness. In an example of Figures 4 and 5, B's fitness is four because it is non-constraint-dominated and the population size is four. Fitness of A and C is three because they are dominated by a single individual: B. D's fitness is one.

F. Aging Operator

Since there are multiple nodes in power utility communication networks and nodes share the same links, the fluctuation happens when a node changes its QoS parameters. This makes evolutionary algorithms hard to find a good set of QoS parameters. In order to reduce the degree of fluctuation and improve the speed of QoS parameters finding process, the aging operator is proposed.

In *EVOLT*, each segment (i.e., a set of the QoS parameters for a node) maintains *age* value which is the number of generations that this node meets the constraints. *Age* is incremented when packets that are transmitted from the node meet the QoS constraints. *Age* is reset to 0 when packets violate at least one constraint. *Age* is used to preserve the segments that satisfy the QoS constraints.

Age value is used when genetic operators (crossover and mutation operators) are performed. s denotes the number of segments (i.e., the number of node). When two parents perform crossover, the age values of the parents' segments are compared. The fitness-based crossover operator (Section III-G) only performs when the *ages* of two individual segments are the same. When the *age values* are different, the parent's segment that has higher *age* is copied to the

two offspring. Then, the two offspring are mutated using aged-based mutation operator (Section III-H).

G. Fitness-based Crossover Operator

Fitness-based adaptive crossover is designed to eliminate the crossover operator parameters (e.g., crossover rate, the number of cross sections) and improve convergence speed. The offspring's gene values are assigned in proportion of two parents' fitness values. As the result, the two offspring are generated close to the better parent (the parent that has higher fitness value).

Figure 6 shows the pseudo code of the fitness-based crossover operator. Two parents (p_1 and p_2) perform crossover. n denotes the number of gene elements (the number of QoS parameters). Assume that the second parent's gene value i is higher than that of the first parent($p_2[i] > p_1[i]$). The offspring's gene values ($c_1[i], c_2[i]$) are calculated by using the Euclidean distances ($d_1[i]$, $d_2[i]$) from the average (*center*[i]) of two parents' gene values. The distance is calculated in proportion of two parents' fitness values. For instance the distance d_1 is calculated in proportion of the first parent's fitness value to the summation of two parents' fitness values. The first offspring's gene value is generated toward the first parent's gene value using the distance ($d_1[i]$) from the center. Figure 7 shows an example how the offspring gene value are calculated. This example assumes that the second parent fitness value is higher than that of parent 1 ($fitness(p_2) > fitness(p_1)$). Thus, the average distances from two offspring's gene values to the second parent gene value is shorter than to parent 1's gene value. As the results, these two offspring are placed closer to the better parent.

procedure CrossOver(p_1, p_2)
$\quad$**for** $i \leftarrow 1$ **to** n
$\quad$**do** $\begin{cases} center[i] \leftarrow (p_1[i] + p_2[i])/2 \\ d_1[i] \leftarrow \frac{(\text{Fitness}(p_1))}{\text{Fitness}(p_1) + \text{Fitness}(p_2)} * \frac{|p_2[i] - p_1[i]|}{2} \\ d_2[i] \leftarrow \frac{(\text{Fitness}(p_2))}{\text{Fitness}(p_1) + \text{Fitness}(p_2)} * \frac{|p_2[i] - p_1[i]|}{2} \\ \textbf{if } U(0,1) > 0.5 \\ \quad \textbf{then } \begin{cases} q_c^1[i] \leftarrow center[i] - d_1[i] \\ q_c^2[i] \leftarrow center[i] + d_2[i] \end{cases} \\ \quad \textbf{else } \begin{cases} q_c^1[i] \leftarrow (p_1[i] - \frac{|p_2[i] - p_1[i]|}{2}) + d_1[i] \\ q_c^2[i] \leftarrow (p_2[i] + \frac{|p_2[i] - p_1[i]|}{2}) - d_2[i] \end{cases} \end{cases}$
$\quad$**return** (q_c^1, q_c^2)

Fig. 6: Fitness-based Adaptive Crossover

The fitness-based adaptive crossover operator is designed following a property in Holland's schema theorem [4], [6], which proves that single-point crossover on binary strings contributes to improve the average fitness values of individuals through generations. The property is that offspring's gene values are placed either inside or

outside the region bounded by the parents' gene values. The fitness-based adaptive crossover operator simulates this property as shown in Fig. 7; the values of the offspring's behavior policy parameters are placed as either $p_1[i] < q_c^1[i] < q_c^2[i] < p_2[i]$ or $q_c^1[i]' < p_1[i] < p_2[i] < q_c^2[i]'$.). In the pseudo code (figure 6), $random(0,1)$ generates a uniform random number between 0 and 1. With a probability of 0.5, the offspring's gene values are enclosed by the parents' gene values. Otherwise, the offspring's gene values enclose the parent's gene values. Therefore, this operator is guaranteed to improve the average fitness values of the agents through generations.
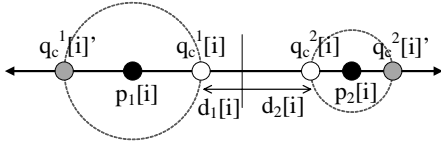

Fig. 7: Fitness-based Adaptive Crossover

H. Age-based Mutation Operator

The proposed age-based mutation operator is designed to avoid local optimal results by dynamically adjusting the degree of mutation. In general, if the degree of mutation is too low, evolutionary algorithms (EAs) may return local optimal solutions. If degree of mutation is too high, EAs focus on global search and the results may not be converged. In *EVOLT*, no degree of mutation setting requires.

Figure 8 describes how the offspring QoS parameter is mutated. One of decision variables in each gene segment is randomly selected to mutate. s denotes the number of QoS parameters at each node. The value of the mutated gene is obtained from the random number following the normal distribution $N(\eta,\sigma)$. The average(η) is the parent QoS parameter and the standard deviation (σ) (i.e., the degree of mutation) is dynamically adjusted on the run time.

procedure MUTATE(q_c)
 for $i \leftarrow 1$ **to** s
  do $\begin{cases} \textbf{if } U(0,1) \le 1/n \\ \quad \textbf{then } q_m[i] \leftarrow N(q_c[i],\sigma) \end{cases}$
 return (q_m)

Fig. 8: Adaptive Mutation

The degree of mutation (σ) is dynamically adjusted to avoid local optimal results. The σ is increased when the individuals are in the local optimal regions (i.e., the individuals cannot improve their fitness values after generations). The σ is decreased when the individuals are not in the local optimal regions in order to improve local search.

The degree of mutation at generation g (σ_g) is adjusted by considering the parents' *age values*. The the parent segment that has higher age value are copied to the offspring. When the difference of two parents *age values* is large, the degree of mutation (σ_g) is decreased in order to increase the degree of local search. When the individuals cannot improve their fitness values after generations, the parents' age values are the same. The degree of mutation (σ_g) is reset to its initial value to increase the degree of global search. The degree of mutation (λ_{g+1}) is calculated as Equation 6.

$$\sigma_{g+1} = \frac{1}{|AGE_{p_1} - AGE_{p_2}| + 1}\sigma_g \qquad (6)$$

IV. Simulation Evaluation

This section shows a set of simulation results to evaluate how *EVOLT* contributes to search for appropriate QoS parameters that satisfy QoS requirements.

A. Simulation Configurations

All simulations were carried out on the modified Java Network Simulator (JNS)[1] , a java implementation of the ns-2 simulator[2].

A simulated utility communication network consists of 34 nodes: a control center (CC), 30 substations (SS1-SS30) and 3 hydro power stations (HS1-HS3). These nodes are connected with a simple tree topology shown in Figure 1. The network bandwidth is 10 Mbps at each link. Link loss rate is 10^{-10}. The hydro power station is the special type of substation; it can generate electric power and transmits a special type of SCADA packets. Hydro power stations are randomly placed in the simulated network. The CC monitors and controls substations and hydro power stations. It periodically receives data from each SS and HS so that it can examine the SS/HS's operational status . The CC also periodically transmits control data to each SS to supervise its operation.

A SCADA application and a maintenance application are deployed on each node. Table I shows a set of packet types of the two applications. There are 16 packet types: 8 SCADA packet types (S1 to S8) and 8 maintenance packet types (M1 to M8) . All of these 16 packet types are periodically transmitted from each source node. This simulation assumes different packet types have the same data transmission frequency/interval and are transmitted at the same time. Tolerable time bound for SCADA packets (S1-S4) is 1 second; and for SCADA packets (S5-S8) is 0.25 second. There is no tolerable time bound setting for Maintenance packet.

TABLE I: Simulated Data

Data Type	Source	Destination	Data Size
S1	SS	CC	1 bytes
S2	SS	CC	6 bytes
S3	HS	CC	1 bytes
S4	HS	CC	6 bytes
S5	CC	SS	2 bytes
S6	CC	SS	6 bytes
S7	CC	HS	2 bytes
S8	CC	HS	6 bytes
M1	SS	CC	450 bytes
M2	SS	CC	3600 bytes
M3	SS	CC	200 bytes
M4	SS	CC	400 bytes
M5	SS	CC	50 bytes
M6	SS	CC	50 bytes
M7	SS	CC	200 bytes
M8	SS	CC	300 bytes

The QoS requirements of transmission latency, jitter, success rate for SCADA packets are 1, 0.3, and 0.99 respectively. The QoS requirements of transmission latency, jitter, and success rate for maintenance packets are 2, 0.7, and 0.95 respectively.

Table II shows the simulation configurations in *EVOLT* and NSGA-II. The results show the average from the 10 independent runs of each algorithm.

TABLE II: The Simulation Configurations

Configuration	NSGA-II	*EVOLT*
EWMA coefficient (Equation 4)	0.8	0.8
g_{max}	100	100
μ	100	100
mutation rate	$1/n$	$1/n$
crossover rate	0.9	x
degree of SBX crossover	15	x
degree of polynomial mutation	20	x

B. Simulation Results

The simulation results are discussed in this section.

Table III shows the comparison of the QoS measures from *EVOLT* and NSGA-II at the last generation in distribution, span, and convergence. Distribution and span equations are modified from spread metric equation in [7] in order to measured the metrics without the knowledge true optimal solutions (Pareto front). Distribution indicator is a diversity metric that measures how the individuals are uniformly distributed. Distribution is measured as the standard deviation of the Euclidean distances between an individuals to its two adjacent neighbors. It is calculated as Equation 7. d_i is the Euclidean distance between consecutive individuals on objective space. $\bar{d}$ is the mean of these distances. This metric takes a zero value for an ideal distribution. Before applying this metric, the objective function values are normalized.

$$D = \sqrt{\frac{\sum_{i=1}^{N-1}(d_i - \bar{d})^2}{N - 1}} \qquad (7)$$

Span indicates how the algorithm can find the different extreme solutions. It is calculated as the maximum Euclidean distance between two individuals (Equation 8). n denotes the number of objectives. $x_i[o]$ denotes the objective k value of individual i. The higher span, the bigger the non-dominated frontier is.

$$S = \max_{i,j \in \mu}\left(\sqrt{\sum_{k=1}^{n}(x_i[k] - x_j[k])^2}\right) \qquad (8)$$

Violation indicates how many individuals violate constraint at the last generation. The upper line shows the average and the lower line shows the standard deviation of ten independent runs. The better value is marked in bold. The results show that the *EVOLT* contributes to better span and violation than that of NSGA-II. This happens because the proposed mechanisms (i.e., adaptive-fitness crossover, gene age mechanism, and adaptive mutation) contribute to improve the individuals (sets of QoS parameters). The distributions of *EVOLT* and NSGA-II are close. This happens because the proposed mechanism, *EVOLT*, uses the same diversity mechanism as NSGA-II (i.e., crowding distance).

TABLE III: The Comparison of *EVOLT* and NSGA-II

Algorithm		Distribution	Span	Violation
EVOLT	Avg.	0.003	**0.118**	**17.12**
	SD	**0.002**	**0.070**	**35.94**
NSGA-II	Avg.	0.003	0.061	23.7
	SD	0.004	0.075	41.67

Tables IV and V show the comparison of *EVOLT* and NSGA-II at the last generation in average of QoS objectives (i.e., latency, jitter and success rate) of the SCADA application and Maintenance application respectively. The results show that the *EVOLT* contributes to better QoS measures than that of NSGA-II. The novel genetic operations in *EVOLT* contribute to find appropriate sets of QoS parameters.

TABLE IV: The Comparison of QoS results in SCADA application

Algorithm		Latency (sec)	Jitter (sec)	Success Rate (%)
EVOLT	Avg.	0.757	**0.361**	1
	SD	**0.272**	**0.158**	0
NSGA-II	Avg.	**0.749**	0.401	1
	SD	0.282	0.216	0

C-metric [8] represents how the individuals of an algorithm outperform the individuals of the other algorithm. It

TABLE V: The Comparison of QoS results in Maintenance application

Algorithm		Latency (sec)	Jitter (sec)	Success Rate (%)
EVOLT	Ave.	**1.60**	**0.604**	1
	SD	**0.602**	**0.200**	0
NSGA-II	Ave.	1.80	0.671	1
	SD	0.604	0.342	0

is calculated as Equation 9. *C*-metric of algorithm *A* to *B* is represented as $C(A, B)$. $\succ$ operator denotes constraint-dominate (i.e., $x \succ y$ represent individual x constraint-dominates individual y . $C(A, B)$ is calculated as the fraction of *B*'s individuals that at least one individual of *A* constraint-dominates. Thus, if $C(A, B) = 1$, all of *B*'s individuals are constraint-dominated by at least one of *A*'s individuals. In average of 10 independent runs, $C(EVOLT, \text{NSGA-II})$ is 0.13 and $C(\text{NSGA-II}, EVOLT)$ is 0.08. This result shows that the novel genetic operations in *EVOLT* contribute to better non-dominated frontier than that of NSGA-II.

$$C(A, B) = |\{b \in B \mid \exists a \in A : a \succ b\}|/|B| \qquad (9)$$

Generation distance(GD) is measured as in Equation 10. This generation distance metric represents how fast an algorithm can optimized the solutions. It is calculated the minimum of Euclidean distance from the non-dominated individuals to the Utopian point in objective space (U) at the end of each generation. Since the success rate is a maximized objective, it is converted to a minimized objective by using the value of ($1 - successrate$) instead. Before applying this metric, the objective function values are normalized.

$$GD = \min_{i \in \mu} \sqrt{\sum_{k=1}^{n} (x_i[k])^2} \qquad (10)$$

Figure 9 shows the generation distance at the end of each generation. The result shows that,in average of 10 independent runs, the proposed novel genetic operations in *EVOLT* contribute to reduce the generation distance faster than the NSGA-II does.

Table VI shows the number of generations of each algorithm to achieve the desired GD. This table concludes that *EVOLT* always achieve the desired generation distance before NSGA-II does. The speed up ratio is calculated as the number of required generations to meet the desired generation distance in NSGA-II over the number of required generations in *EVOLT*. The average of speed up ratio is 1.98. This result show that the proposed novel genetic operations in *EVOLT* contribute to converges approximately two times faster than NSGA-II does.
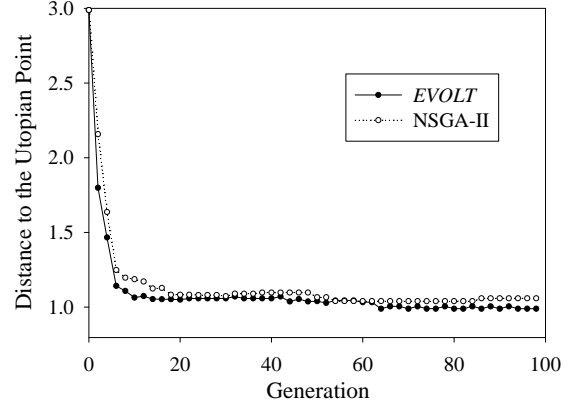
Fig. 9: The Generation Distance

TABLE VI: Generation Distance Achievement

GD		*EVOLT*	NSGA-II		Speed Up Ratio
3		0	0		0
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

The search ratio which is calculated as the number of individuals evaluated through generations over the total number of combinations of QoS parameters is $\frac{(100*100)}{(10*100*10*100*10*100*2*5*100*2*5*100*2)^{33}} = \frac{1}{6.4716*10^{34}}$. This information shows that using evolutionary algorithm can reduce the search space enormously.

V. Related Work

There are several researches apply genetic algorithms to tune parameter settings in power electric systems [9]–[11]. The results show that genetic algorithms can find suitable parameters and sufficiently close to the global optima values. However, they focus on parameter tuning of controllers in substation/control center. The proposed genetic algorithm, *EVOLT*, is newly designed and focus on the power utility communication networks.

[12] applies genetic algorithms for network topology reconfiguration. Each gene is considered as a list of reconfigurable link allocations. The objectives are transmission latency, the number of dropped packets, and the cost of reconfiguring the network. These three objective values are combined in to a single fitness function using a weighted sum. These weight values are carefully designed. The obtained results may not be in non-dominated rank. In contrast, *EVOLT* does not require any weighted sum values in objective functions. The obtained solutions are always in non-constraint dominated ranking.

[13] proposed a GA-based multi-purpose optimization

algorithm for QoS routing. There are multiple routes between source node and destination node. The proposed algorithm searches for the optimal routes that satisfy objectives (i.e., transmission latency and communication cost).There is no concept of constraints. Thus, the obtained results can be infeasible. Searching time to find optimal solutions will be very long since it alters the genes by performing genetic operations in each objective one by one. In contrast, *EVOLT* considers large number of objectives by using domination ranking. Novel genetic operations are also proposed to meet QoS requirements in a short time.

Algorithms [14], [15] are focusing on reducing the genetic operation parameters. However, there are some parameters that left to be tuned in their proposed algorithms. Parameter-Less GA [14] adjusts population size dynamically. There are some parameters (e.g.,crossover rate, mutation rate) left to be tuned. Moreover, the period to add new population need to be carefully designed. In Meta-GA [15], the upper level genetic algorithm is added to find the parameters. However, the parameters of the upper level genetic algorithm itself need to be tuned. In *EVOLT*, there is no genetic operation parameters. The population size is the desired number of the sets of QoS parameters from the administrators.

VI. Conclusion

This paper investigates the proposed evolutionary algorithm to find appropriate sets of QoS parameters to satisfy QoS requirements in power utility communication networks. Simulation results show that the proposed genetic operations in emphEVOLT work properly and is able to find appropriate sets of QoS parameters. *EVOLT* is designed to operate at the same overhead as regular evolutionary algorithm while reducing the number of nontrivial parameter setting burdens. The simulation results show that the *EVOLT* finds sets of QoS parameters that satisfy the QoS requirements faster than NSGA-II does.

References

[1] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected areas in communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[2] J. Northcote-Green and R. Wilson, *Control and Automation of Electrical Power Distribution Systems*. CRC Press, 2006.

[3] Y. W. M. Shahidehpour, *Communication and Control in Electric Power Systems: Applciations of Parallel and Distributed Processing*. Wiley-IEEE, 2003.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Prof., 1989.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Tans. on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[6] J. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[7] A. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. Durillo, and A. Beham, "AbYSS: Adapting scatter search to multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 4, pp. 439–457, 2008.

[8] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study And the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[9] J. Finch and M. Besmi, "Genetic algorithms applied to a power system stabiliser," in *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*, 1995, pp. 100–105.

[10] I. Ramírez-Rosado and J. Bernal-Agustín, "Genetic algorithms applied to the design of large power distribution systems," *IEEE Trans. on Power Systems*, vol. 13, no. 2, pp. 696–703, 1998.

[11] Y. Abdel-Magid and M. Abido, "Optimal multiobjective design of robust power system stabilizers using GA," *IEEE Trans. on Power Systems*, vol. 18, no. 3, pp. 1125–1132, 2003.

[12] D. Montana and T. Hussain, "Adaptive reconfiguration of data networks using genetic algorithms," *Int'l Con. on Genetic and Evolutionary Computation Conference*, 2002.

[13] A. Koyama, L. Beralli, K. Matsumoto, and B. O. Apduhan, "A ga-based multi-purpose optimization algorithms for qos routing," *18th Int'l Con. on Advances Information Networking and Applications*, 2004.

[14] G. Harik and F. Lobo, "A parameter-less genetic algorithm," in *Proc. of the Genetic and Evolutionary Computation Conference*, vol. 1, 1999, pp. 258–265.

[15] J. Clune, S. Goings, B. Punch, and E. Goodman, "Investigations in meta-GAs: panaceas or pipe dreams?" in *Proc. of the 2005 workshops on Genetic and evolutionary computation*. ACM, 2005, pp. 235–241.