Master's Thesis

Title

Evolutionary Network Power-Saving Method based on Pareto Optimal Control

Supervisor

Professor Masayuki Murata

Author

Yosuke Akishita

February 9th, 2018

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Evolutionary Network Power-Saving Method
based on Pareto Optimal Control

Yosuke Akishita

Abstract

The power consumption of networks has been increasing as the service over the Internet becomes popular. Many methods to reduce the power consumption have been proposed. The power consumption of networks can be saved by shutting down unnecessary network devices, following the changes in the traffic demands; when the traffic demands are small, only a small number of nodes are required to be powered on to accommodate the traffic. On the other hand, when the traffic demands become large, more nodes should be powered on to accommodate the traffic without congestion.

These methods minimize power consumption considering the number of powered-on nodes and the maximum link utilization as constraints. However, the actual network should satisfy multiple objectives. One of the important objectives is the reliability. The reliability may be degraded by powering-off links. For example, the network, some of whose nodes are shut down, can be disconnected in case of failures. In this case, the network service becomes unavailable until the connectivity is repaired by powering up links or nodes. But powering up the nodes takes a time.

Therefore, we propose a network power saving method that handles multiple complex objectives, following the environmental changes such as traffic fluctuations and failures. Our method calculates a set of Pareto optimal solutions (Pareto front) to consider all of these objectives. Then our method select a solution whose power consumption is the smallest among the solution on the Pareto front satisfying required performance and reliability.

In our method, the Pareto front should be updated so as to follow the environmental fluctuation. If traffic rate does not change significantly and no failure occurs, the Pareto front for the current time slot is close to the Pareto front of the previous time slot. However, if sudden environmental changes occur, the previous Pareto front will be far from the true Pareto front after the change.

Even in this case, an appropriate Pareto front should be obtained as soon as possible, because the suitable network configuration cannot be obtained unless an appropriate Pareto front is obtained. Therefore, we propose a method to accelerate the calculation of the Pareto front. In this method, we introduce the evolvable solutions (ES) which are the solutions with high evolvability. By calculating the Pareto front from the solutions in the previous Pareto front and ES, an appropriate Pareto front can be obtained in a small number of generations.

We evaluate our method by simulation, and demonstrate that our method reduces the power consumption without violating the constraints, following the traffic changes. In addition, we also demonstrate that our method can satisfy the constraints within about 4 generations and save power consumption within about 650 generations even in the case of the sudden traffic changes.

Keywords

Network power saving
Pareto optimization
Pareto front
Multi-objective evolutionary algorithms
Evoluvability

Contents

List of Figures

List of Tables

1 Introduction

Network traffic has been increasing as the service over the Internet such as streaming and cloud service becomes popular [1], and the power consumption of networks has also been increasing [2, 3]. The power consumption has become one of the important problems in networks.

The power consumption of networks can be saved by shutting down unnecessary network devices, following the changes in the traffic demands; when the traffic demands are small, only a small number of nodes are required to be powered on to accommodate the traffic. On the other hand, when the traffic demands become large, more nodes should be powered on to accommodate the traffic without congestion.

Many methods to reduce the power consumption have been proposed [4–7]. Wei et al. formulated the optimization problem considering SDN/IP hybrid routing mode to minimize the number of powered-on links under the constraints that the link load does not exceed the threshold of link capacity [4]. They also proposed a heuristic algorithm that finds the optimal setting of OSPF link weight and splitting ratio of SDNs to solve the problem. The method proposed by Amaldi et al. sets the OSPF link weights so that the energy consumption is minimized by solving the Mixed Integer Linear Problem (MILP) periodically [5]. Chiaraviglio et al. also formulated the MILP that minimizes the number of powered-on nodes under the constraints that the full connectivity should be kept and maximum link utilization should be less than the predefined threshold [6]. They also proposed a heuristic method to solve the problem.

These methods minimize power consumption considering the number of powered-on nodes and the maximum link utilization as constraints. However, the actual network should satisfy multiple objectives. One of the important objectives is the reliability. The reliability may be degraded by powering-off links. For example, the network, some of whose nodes are shut down, can be disconnected in case of failures. In this case, the network service becomes unavailable until the connectivity is repaired by powering up links or nodes. But powering up the nodes takes a time.

One approach to considering such multiple objectives is to solve the optimization problem including them as the constraints; a network is configured so as to minimize the power consumption under the constraints that the delay should be less than a predefined threshold and the number of distinct paths on the powered-on paths should be larger than the predefined value. However, this approach has the following problems. The first problem is calculation time; it takes a long time

to solve the optimization problem; the optimization problem includes as many binary variables as the number of links. Some of the objective functions are non-linear and complex. A large calculation time may cause the difficulty in configuring the network following the traffic changes. Thus, we need a heuristic method that can obtain the suitable solution immediately after traffic change occurs.

Another problem is caused by that this approach includes only a single objective function; a solution obtained by this approach minimizes only one objective function, but does not minimize the others. For example, let us consider the case that the network operator wants to configure the network so that the power consumption is minimized under the constraint that the required bandwidths can be provided for all node pairs. In this case, there may be multiple solutions that minimize the power consumption and satisfy the constraint. The solution that maximizes the provided bandwidth for each node pair among such multiple solutions is preferable. However, the optimization problem that minimizes the power consumption under the constraint of the provided bandwidth cannot obtain such a solution. By setting the objective functions to the weighted sum of multiple objective functions, all of the objective functions may be considered. However, this approach also cannot solve the problem; the optimization problem with the weighted sum of multiple objectives may be even unable to obtain the solution that minimizes the energy consumption, because it only minimizes the weighted sum of multiple objectives.

A suitable solution considering multiple objectives is a Pareto optimal solution. Pareto optimal solutions are the solutions that cannot be improved in any of the objectives without degrading at least one of the other objectives. There are several methods to save the power consumption by obtaining the Pareto optimum solutions. Gomes et al. proposed a method to deploy virtual machines considering the Pareto optimum solutions of bandwidth and power consumption, and demonstrated that this method improves the available bandwidth and power efficiency [8]. Zaheeruddin et al. also proposed a method to control multicast routes by obtaining the Pareto optimum solution of QoS and power efficiency in wireless networks [9]. However, these methods do not consider the environmental changes. When the traffic changes, the Pareto optimum solutions changes. Failures of nodes also cause the change of the Pareto optimum solutions. Therefore, the Pareto optimum solutions should be updated immediately and network should be reconfigured based on the updated Pareto optimum solutions, following such changes.

In this thesis, we propose a network power saving method that handles multiple complex

objectives, following the environmental changes. In our method, we store candidate network configurations and evolve them so as to follow the environmental changes in the network. Then, we select the network configuration that minimizes the energy consumption and satisfies the requirements from the candidate network configurations.

In the above steps, we evolve the candidate network configurations based on the Multi-Objective Evolutionary Algorithms (MO-EA) [10–15], which calculate the set of the Pareto optimum solutions, which is called the *Pareto front*, by evolutional algorithm. Though these papers use evolutional algorithms to obtain the Pareto front for a given case and evolve the solutions without changing the environment, we evolve the solutions from those of the previous time slot at each time slot. By doing so, we adapt the set of solutions so that they become the Pareto front for the current time slot, following the changes.

In our approach, the computational complexity and the number of the generations required to achieve suitable solutions are important; it may take a long time to achieve the suitable solutions if they are large, which causes a large time to recover the performance of the network after the traffic changes or failures occur. Therefore, we discuss methods to achieve suitable solutions immediately. In this thesis, we accelerate the evaluations of the solutions by encoding the information required by the evaluations, because the evaluation of solutions takes most of time in each generation in our method. We also shape the evolution strategy to generate the suitable solutions for the current time slot considering the objectives.

However, when sudden environmental changes occur, the number of generations required to achieve an appropriate solution increases especially in a large network. This is because the actual Pareto front after the change becomes far from the previous Pareto front. Therefore, we propose a method to achieve an appropriate Pareto front in a small number of generations even in such cases. In this method, we generate *Evolvable Solutions (ES)*, which is the solutions with high evolvavility. The solution with high evolvability accelerates the evolution process. Thus, the ES promotes the evolution and reduces the generation required to achieve an appropriate Pareto front.

Mengistu et al. proposed a method to generate the solutions with high evolvabiliy, called *Evolvability Search* [16]. Evolvability Search generates the solutions with high evolvability by the evolutional algorithm selecting solutions with higher Evolvability as fitness. In Evolvability Search, the evolvability is quantified as the number of unique individuals among offspring generated from the individual to be evaluated. Mengistu et al. applied Evolvability Search to the

robotics, and demonstrated that Evolvability Search generates solutions with higher evolvability than those generated by the conventional objective-based search algorithm. In this thesis, we introduce Evolvability Search to generate the ES, and use it as initial solutions with the previous Pareto front to achieve appropriate solutions quickly even after the sudden changes.

The rest of this thesis is organized as follows. Section 2 explains related work, and Section 3 proposes a method to save power consumption that handles the environmental changes based on MO-EA. We accelerate our method in Section 4. We evaluate our method in Section 5. Finally we conclude this thesis in Section 6.

2 Related work

2.1 Network power saving

Power consumption of networks can be saved by shutting down nodes and links. Many methods to reduce the power consumption have been proposed [4–7].

Wei et al. formulated the optimization problem considering SDN/IP hybrid routing mode to minimize the number of powered-on links under the constraints that the link load does not exceed the threshold of link capacity [4]. They proposed a heuristic algorithm that optimizes OSPF link weight of IP routers and traffic flow splitting ratio of SDN enabled switch. The algorithm enables traffic flow to be aggregated onto partial links and the underutilized links can be turned off.

Amaldi et al. proposed a method to minimize power consumption under the constraint that network congestion is avoided [5]. The method proposed by Amaldi et al. sets the OSPF link weights and shutting down unused devices so that the energy consumption is minimized by solving the Mixed Integer Linear Problem (MILP) periodically.

Chiaraviglio et al. also formulated the MILP that minimizes the number of powered-on nodes under the constraints that the full connectivity should be kept and maximum link utilization should be less than the predefined threshold [6, 7]. They also proposed a heuristic method to solve the problem. This method searches the node that can be shut down first because shutting down a node saves more energy than shutting down a link.

Giroire et al. proposed a method to reduce power consumption in a network composed of routers whose function can remove redundant traffic [17, 18]. By using the function to remove redundant traffic, the bandwidth that the network can provide can be improved. However, this function increases the power consumption of the router [17]. Therefore, they propose a method to decide which router to operate this function, considering the link that can sleep [18].

These methods minimize power consumption considering the number of powered-on nodes and the maximum link utilization as constraints. However, the actual network should satisfy multiple objectives. One of the important objectives is the reliability. The reliability may be degraded by powering-off links. For example, the network, some of whose nodes are shut down, can be disconnected in case of failures. In this case, the network service becomes unavailable until the connectivity is repaired by powering up links or nodes. But powering up the nodes takes a time. Therefore, we propose a network power saving method that handles multiple complex objectives,

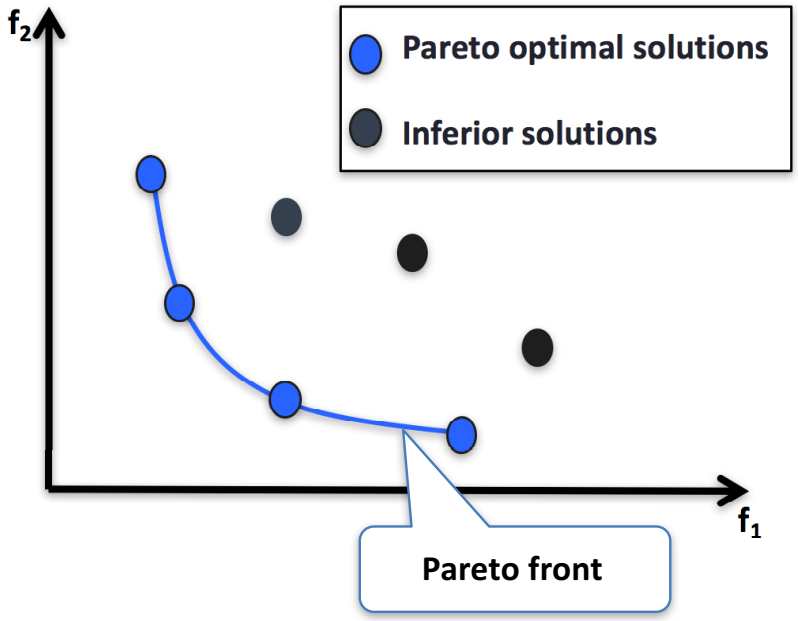


Figure 1: Pareto optimal solutions

following the environmental changes such as traffic fluctuations and failures.

2.2 Multi-objective optimization problem

2.2.1 Multi-objective evolutionary algorithm

Pareto optimal solutions and Pareto front In multi-objective optimization problems, it is impossible to obtain a complete optimal solution to all of the given objective functions, because the objective functions compete with each other. Therefore, the Pareto solutions are obtained. x^* is a Pareto optimal solution when there is no x that satisfies Eqs. (1) and (2).

$$f_i(x) \leq f_i(x^*) \; \forall i = 1, ..., p \tag{1}$$

$$f_i(x) < f_i(x^*) \; \exists i \in \{1, ..., p\} \tag{2}$$

where $f_i(x)$ is ith objective function. In general, there are a number of Pareto optimal solutions, and these solutions form surface which is called Pareto front.

Calculation of Pareto optimal solutions based on the evolutional algorithm MO-EA obtains the Pareto optimal solutions by an evolutional algorithm [19]. In MO-EA, solutions are coded as a gene. MO-EA evolves the genes by using the mutation and crossover operators so that the genes approach to the Pareto optimal solutions.

MO-EA performs the following steps.

1. Initialization : generate N individuals, and denote the set of individuals as P.
2. Evaluation : rank the individuals based on the non-dominated sort, and calculate the density in each rank.
3. Generation of offspring : generate N offspring by manipulating genes (selection, crossover, and mutation), and denote them as Q
4. Replacement of the old solutions : update P to N solutions selected from $P \cup Q$, and save the solutions in rank 1 to Pareto archive.
5. Finish of determination : End when the end conditions are satisfied. Then, the solutions in Pareto archive form the Pareto front. Go back to Step.3 otherwise.

MO-EA ranks the individuals by using non-dominated sort [19]. When a solution A is superior to another solution B in all objective functions, the solution A *dominates* the solution B. The non-dominated sort ranks the solutions based on the number of dominating solutions. The non-dominated sort procedures are summarized as follows.

1. Initialize n to 1.
2. For each solution, count the number of the solutions dominating the solution, and the number of the solutions dominated by the solution.
3. For each solution that is not included in the lists, if the number of solutions dominating the solution is 0, add it to the lists F_n.
4. For each solution, subtract 1 from the number of the solutions dominating the solution if the solution is dominated by the solutions in F_n.
5. End if the number of the solutions that are not included in any lists is 0. Otherwise, increment n and go back to Step 2.

After the above steps, the list of F_n includes the solutions of the rank n.

Priorities between the individuals with the same rank are determined by using crowding distance [19]. Crowding distance c_x is a metric indicating the density of the individuals near the individual x, and is defined by

$$c_x = \begin{cases} \infty & \text{max or min on} \\ & \text{any criteria of } x \\ \sum_m \frac{f_m(I_m^{\text{next}}(x)) - f_m(I_m^{\text{prev}}(x))}{f_m^{max} - f_m^{min}} & \text{otherwise} \end{cases} \tag{3}$$

where $I_m^{\text{prev}}(x)$ is the individual whose value of the mth objective function is the largest among the individuals whose values are smaller than the value of x, and $I_m^{\text{next}}(x)$ is the individual whose value of the mth objective function is the smallest among the individuals whose values are larger than the value of x. f_m^{max} is the maximum value of the mth objective function, and f_m^{min} is the minimum value of mth objective function. By selecting the solution with a large c_x, we select the solutions that are different from the other solutions.

2.2.2 Application of Multi-objective optimization problem

Kessaci et al. proposed a method to optimize the three objectives of scheduling High Performance Computing (HPC) application, energy consumption, CO_2, provider's profit [10]. In that method, the Pareto front, which is a set of Pareto optimal solutions, is derived by using *Multi-Objective Evolutionary Algorithms (MO-EA)* [11–13, 20]. Then, they select one solution according to a certain selection criterion from the Pareto front. They simulated their method and the result shows that their method enables scheduling to increase the profit of providers while suppressing power consumption and CO_2 emissions.

In the renewable energy system, Ahmadi et al. formulated minimization of the total cost rate of the system and maximization of the exergy efficiency of the system as a multi-objective optimization problem [14]. They showed that the Pareto front obtained by MO-EA method can determine the best design parameters for the system.

Wang et al. formulated 12 kinds of design problems in the water supply system that consider the cost and the performance [15]. Then they use five MO-EA methods to each problem and compare solutions. The result shows that Nondominated Sorting Genetic Algorithm-II (NSGA-II) [21] yielded the best among the five method of MO-EA. NSGA-II ranks the individuals in the

group and selects individuals based on elitist principles.

As described above, the methods for obtaining a Pareto front and selecting an appropriate solution among them are applied in various fields. In this thesis, we apply multi-objective optimization problem to network power saving control.

There are several methods to save the power consumption in networks by obtaining the Pareto optimum solutions. Gomes et al. proposed a method to deploy virtual machines considering the Pareto optimum solutions of bandwidth and power consumption, and demonstrated that this method improves the available bandwidth and power efficiency [8]. Zaheeruddin et al. also proposed a method to control multicast routes by obtaining the Pareto optimum solution of QoS and power efficiency in wireless networks [9].

However, these methods do not consider the environmental changes. When the traffic changes, the Pareto optimum solutions change. Failures of nodes also cause the change of the Pareto optimum solutions. Therefore, the Pareto optimum solutions should be updated immediately and network should be reconfigured based on the updated Pareto optimum solutions, following such changes. We use MO-EA in order to calculate an appropriate solution following the environmental fluctuations.

2.3 Evolvability Search

The evolvability is one of the important elements in the evolution of living things, in addition to accelerating the evolution process as a whole, The evolvability gives benefits to species by avoiding extinction or establishing a new niche [22]. In general, the evolvability is not required if the fitness function is static, and degrades the performance of the evolutionary algorithms. However, the evolvability is important in the case of dynamically changing conditions.

Mengistu et al. proposed the method called *Evolvability Search*, which creates solutions with high evolvability [16]. Evolvability Search directly selects solutions with higher evolvability by using the evolvability as the fitness in the evolutionary algorithms. Mengistu et al. approximated the evolvability by the individual's capacity to generate future phenotypic variation which is quantified by the number of unique behaviors.

In Evolvablity Search, the evolvability of the individual i is determined by the following procedure.

1. Select individuals j randomly.

2. Manipulate genes with individual i and individual j to generate individuals k and l.

3. Determine whether individuals k and l are unique individuals by the indicator $B(x)$ ($x \in \{k, l\}$). If it is a unique individual, it is added to the list U_p, and if it is not a unique individual, it is discarded.

4. If the number of descendants of generated i has reached a certain number, go to step 5, otherwise go to step 1.

5. End Evolvability of individual i as the population of U_p.

Mengistu et al. make experiments in the robotics and set $B(x)$ as the distance between them according to a domain-specific behavioral distance metric. The results show that Evolvability Search generates solutions with higher Evolvability than the conventional objective-based search algorithm. They also demonstrated that it is evolvable after transferring to the new environment from the environment where the Evolvability Search was executed. Therefore, Evolvability Search significantly increases the evolvability of the solution.

In this thesis, we apply the Evaovability Search to promote the evolution of the Pareto front. Hereafter, we call the solution obtained by Evolvability Search *Evolvable Solution (ES)*.

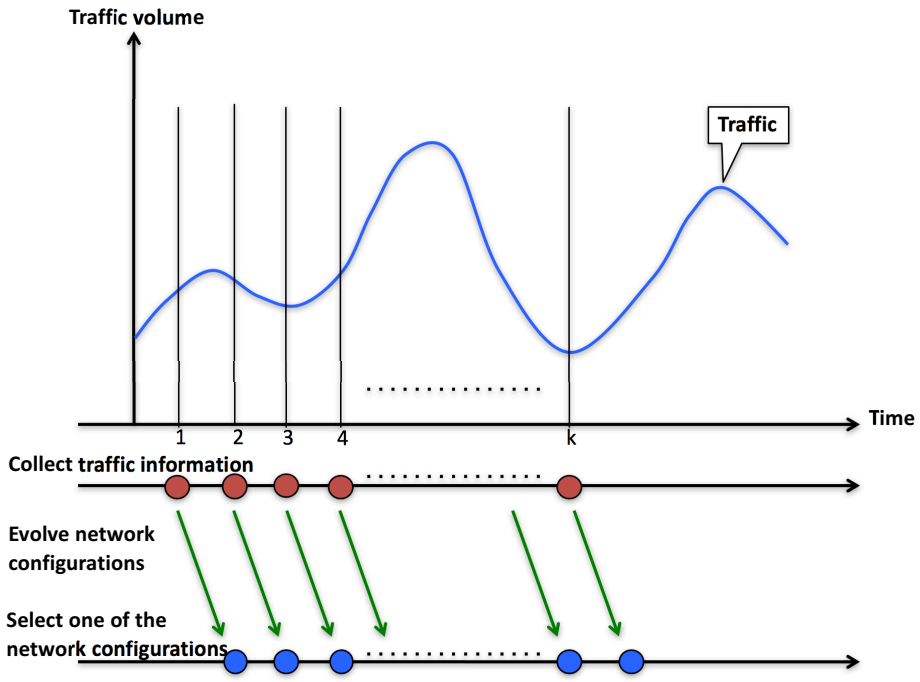


Figure 2: The overview of our method

3 Network power saving method following the environmental changes

3.1 Overview of network power saving method based on Pareto optimal control

Figure 2 shows the overview of our method. In our method, we store the candidate network configurations and evolve them following the environmental changes at each time slot. Then, we select one of the network configurations that minimizes the energy consumption and satisfies the requirements as shown in Figure 3. At each time slot, the evolution of the candidate network configurations is performed based on the MO-EA, considering the multiple objectives. By continuing these steps, our method follows the environmental changes.

In this thesis, we consider multiple objectives such as energy consumption, performance, and reliability. Therefore, the suitable network configuration is a Pareto optimal solution of such multiple objectives. However, the objective functions depend on the current situation; for example, the performance of the network depends on the traffic. That is, the Pareto optimal solutions at previous time slots may no longer be the Pareto optimal solutions at the current time slot. Therefore, the Pareto optimal solutions should be calculated for each time slot.

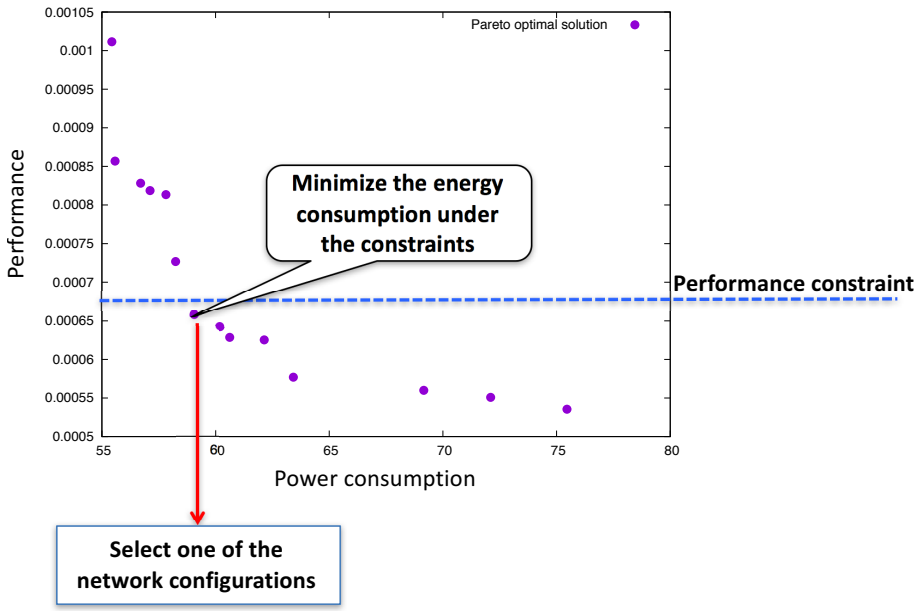


Figure 3: Selection of the network configuration

In this thesis, we obtain the Pareto optimal solutions for the current time slot by evolving the previous Pareto optimal solutions. If the environmental change is small, the current Pareto optimal solutions are obtained immediately by evolving the previous solutions. If the sudden change occurs and the current Pareto optimum solutions become far from the previous solutions, the evolved solution may be still far from the current Pareto optimum. But even in such cases, by continuing the evolution, we obtain the solutions near the current Pareto optimum.

The problem solved in this thesis is to set the powered-on links and nodes and routes between the nodes. In this thesis, we configure a route between a node pair from the candidate path calculated in advance by k-shortest path over the network topology where all nodes and links are powered on, because the paths traversing many nodes consumes much resources and should not be selected. By evolving the network configurations focusing on the k-shortest paths, long routes are avoided, and evolved network configurations become close to suitable ones fast.

At each time slot, our method evolves the candidate network configurations based on the MO-EA. That is, we perform the following step several times in each time slot.

1. Evaluation : rank the network configurations based on the non-dominated sort, and calculate the density in each rank.

2. Generation of offspring : generate offspring by manipulating network configurations

3. Replacement of the old solutions : update the set of network configuration, and save the solutions in rank 1 to Pareto archive.

Finally, we store the solutions in Pareto archive for the next time slot. The details of each operation are as follows.

3.2 Evolution of network configurations

3.2.1 Evaluation

Each candidate network configuration is evaluated by calculating the values of the objective functions. Then, we rank the candidate configurations by the non-dominated sort. The solutions of Rank 1 are the solutions which are the nearest to the Pareto front. We also calculate the Crowding distance for each solution.

Then we select one best solution, whose power consumption is the smallest among the solutions that satisfy the other constraints, from the solutions of Rank 1. Hereafter we call the selected solution *priority solution*. The priority solution is the best network configuration among the current candidate configurations.

3.2.2 Generation of offspring

In the above steps, our method generates new offspring by selection, crossover and mutation as follow.

Selection and crossover We select network configurations based on a tournament strategy; we select k network configurations randomly, and select the best individual from them. The best individual is selected as follows. If the k selected network configurations include the priority solution, the priority solution is selected, because the priority solution is the best solution among the current solutions. Otherwise, we compare the ranks of the solutions, and the select the solution with the smallest rank. If there exist multiple solutions with the smallest rank, we compare the Crowding distances and select the solution with the largest Crowding distance among the solutions with the smallest rank.

By performing the above steps twice, we select two network configurations. Then, we perform the crossover operation, which swaps randomly selected path in the selected two network configurations to generate new offspring. After swapping the selected paths, we modify the information on the powered-on links so that any link included in the selected paths are powered on.

Mutation Mutation generates candidate network configurations different from the candidates of the previous generation. In this thesis, the mutation has two roles; generating the network configurations with a small power consumption by shutting down nodes and generating the network configurations that accommodate more traffic by balancing the loads. Considering the above roles, we perform the following mutations.

- Shutting down a node: Select one powered-on node. Then, update the routes so that all paths do not include the selected node; if a path for a flow includes the selected node, a new path for the flow that includes only the powered-on nodes and links is selected randomly from the candidate paths for the flow. If no candidate paths includes only the powered-on nodes are found, discard the generated configuration.

- Balancing loads: Select one congested link which is calculated in evaluation phase. Then, paths including the target links are updated; a new route for each flow whose path required to be updated is randomly selected from the candidate paths for the flow.

3.3 Selection of network configuration

In this thesis, we aim to minimize the energy consumption under the constraint of the other objectives. Therefore, our method selects the configuration whose power consumption is the smallest among the candidates that satisfy the constraint. The selected network configuration is the solution that minimizes only the energy consumption but the solution that is nearly optimal also for the other objectives, because the candidate network configurations are updated so that the set of candidate network configurations become close to the Pareto front.

4 Acceleration of calculation considering multi-objective functions

Our method requires the evolutions of candidate network configurations at each time slot. The calculation time of the evolution is important, because it is difficult to set the length of each time slot to short value, which may cause the difficulty in following the changes, if the evolution takes a long time. Therefore, this section discuss the computational complexity of evolution.

Because the computational complexity of evolution depends on the objective functions, we first explain the objective functions assumed in this discussion. Then, we discuss the computational complexity. Finally, we propose a method to accelerate the calculation and the evolution.

4.1 Multi-objective functions

Though our method is applicable to any objective functions, the following discussions are based on the following three objective functions.

Power Consumption

In this thesis, the network power consumption is defined as the sum of the power consumption of the powered-on links and nodes, and is calculated by

$$E^{\mathrm{net}}(x) = \sum_{(i,j)\in L} E^{\mathrm{Link}} p_{i,j}(x) + \sum_{k\in V} E^{\mathrm{Node}} p_k(x) \tag{4}$$

where L is the set of links, V is the set of nodes, E^{Link} is the power consumption at each link, E^{Node} is power consumption at each node, $p_{i,j}(x)$ is a variable which is 1 when gene x uses link between i and j, 0 otherwise, and $p_k(x)$ is variable which is 1 when x uses node k, 0 otherwise.

Reliability

One approach to keeping the connectivity in case of failures is to prepare the distinct backup paths on the powered-on links. If a failure occurs, the routes of flows passing the failed link are changed to the distinct backup paths. By doing so, we can keep the connectivity immediately after the failure is detected without powering-on a new node, which takes a time.

By preparing more distinct backup paths, we can keep the connectivity in case of large failures. Thus, we set the metric of the reliability based on the number of distinct paths on the powered-on

links. We define the reliability by

$$R(x) = \frac{1}{\min_{i,j} r_{i,j}(x) + \alpha \sum_{i,j} r_{i,j}(x)} \tag{5}$$

where $r_{i,j}$ is the number of distinct paths between i and j, and α is a parameter. In our evaluation, we set α to $\max_x \sum_{i,j} r_{i,j}(x)$.

Though we select the paths used in the case without failures from k-shortest paths, we allow the distinct backup paths that are not included in the k-shortest paths. By allowing such backup paths, more nodes and links can be powered off while preparing a required number of distinct backup paths.

Performance

In this thesis, we use maximum link utilization as the metric of performance. That is, the objective function of performance $P(x)$ is defined by

$$P(x) = \max_{(i,j) \in L} \rho_{i,j}(x) \tag{6}$$

where $\rho_{i,j}(x)$ is utilization of link i–j.

4.2 Computational complexity without acceleration

4.2.1 Encoding Strategy

A network configuration includes the set of the powered-on nodes, the set of the powered-on links, and the paths between all node pairs. Therefore, we should encode the network configuration so that all of such information are included. The simple encoding strategy encodes such information as a gene. That is, each gene include the set of powered-on links and the set of the pairs of the flow and the path used by the flow.

4.2.2 Computational Complexity

We denote the number of flows by f, and the number of candidate paths. We denote the number of nodes by N and the number of links by L. We denote the number of candidates by m. We denote the number of distinct backup paths prepared for each flow by b. F is the maximum number of rank calculated by the non-dominant sort.

We discuss the computational complexity of each operation performed to evolve the candidate configurations below.

Evaluation In each generation, all candidates are required to be evaluated by calculating the objective functions and non-dominated sort.

To calculate the power consumption, we need to check whether each link is powered on or off. This process takes $O(L)$ for each candidate.

To evaluate the reliability, we need to calculate the distinct backup paths. The distinct backup paths for each flow can be calculated by using Dijkstra's algorithm. First, we remove the intermediate nodes passed by the flow from the topology. Then we calculate one backup path from the source node to the destination node by Dijkstra's algorithm. After removing the nodes passed by the calculated new backup path, another backup path is calculated by the same way. These steps are continued until no backup paths are found or a sufficient number of backup paths are found. The computational complexity of Dijkstra's algorithm is $O(L + N \log N)$ [23]. The Dijkstra's algorithm is performed b times for each flow. That is, the computational complexity to evaluate the reliability is $O(bL + bN \log N)$.

To evaluate the performance, we calculate the maximum link utilization. The link utilizations are calculated by adding the traffic rate of the flows passing the link, which takes $O(f)$. Then, we select the maximum of the calculated link utilizations, which takes $O(L)$.

The above processes to evaluate the objective functions are performed for each candidate. Among the above processes, the evaluation of the reliability takes most of calculation time. That is, the computational complexity of evaluation of the objective functions for each candidate is $O(bL + bN \log N)$.

After the evaluation of the objective functions, we perform the non-dominant sort. Non-dominant sort is performed by counting the number of solutions dominating it and the number of solutions dominated by it. This process is continued F times. That is, the computational complexity of the non-dominant sort is Fm.

To evaluate the candidates, we also use the Crowding distance. The Crowding distance is calculated by comparing the values of the objective functions of the nearest candidates, after sorting the candidates in order of the values of the objective functions. The sort takes $O(m \log m)$.

Generation of offspring Offspring is generated by the crossover and mutation.

The crossover is performed after selecting two candidates. The selection is performed by selecting the best solution among k randomly selected solutions. This process takes $O(k)$. Then, the crossover operation swaps randomly selected paths, which takes only $O(1)$. That is, the crossover operation to generate one offspring takes $O(k)$.

In this thesis, we perform two kinds of mutations; shutting down a node and balancing loads. To shutting down a node, we need to select the flows passing the selected node, which is to be shut down. Then, we select a new path that does not include the selected node for each of the selected flows. This process takes $O(f)$. To balancing the loads, we need to select the flows passing the congested links, and we select new paths randomly for each of the selected flows. This process also takes $O(f)$.

4.3 Acceleration of calculation

According to the above discussion, the evaluation of the reliability, which requires to calculate the distinct backup paths, takes most of time. Therefore, we accelerate the calculation by accelerating the evaluation of the reliability.

The distinct backup paths do not change unless the network topology of the powered-on nodes and links is changed. Therefore, we encode the distinct backup paths in genes, and calculate the paths only when the calculation is required.

Each time a new candidate is generated by crossover or mutation, we update the backup paths by the following steps. First, we check whether the network topology of the powered-on links is changed since the previous update of the backup paths. If the network topology is not changed, any backup paths do not require the updates. Otherwise, we select the flows whose backup paths are required to be updated, and update only backup paths of only the selected paths. The flows whose backup paths are required to be updated are the following flows.

- The flows which do not have a sufficient number of backup paths. (i.e., the flows whose number of backup paths is less than a threshold.)
- The flows whose backup paths pass the links that no longer exist.
- The flows whose backup paths pass the node that are passed by the newly calculated routes for the flow

Then, only the backup paths of the selected flows are updated by Dijkstra's algorithm.

The above steps to select the flows whose backup paths are required to be updated takes $O(bf)$, because selection can be done by checking all of the previous backup paths. Then, if f' flows are selected as the flows whose backup paths are required to be updated, the computational complexity of the updating the backup paths for each candidate is $O\left(f'\left(L + N\log N\right)\right)$, which is much smaller than the computational complexity of the original version if the number of selected flows are small.

Though the above discussion is based on the objective function described in Section 4.1, the same method to accelerate the calculation, which encode the information required to calculate the objective function in the gene, can be used in the case of the other objective functions that require a large calculation time.

4.4 Acceleration of evolution using evolvable solutions

4.4.1 Overview of control method using ES

In this thesis, we use the evolvable solutions to accelerate the evolution of the Pareto front. In this approach, we evolve the candidate network configurations from the candidate network configurations at the previous time slot and the evolvable solutions obtained by the Evolvability Search in each time slot.

In this approach, ES should not be removed at the first selection in each time slot to accelerate the evolution. However, Evolvability Search does not consider the objectives of our network control such as the number of distinct paths. As a result, ES may be removed at the first selection. To avoid the removal of ES at the first selection, we need the ES that is close to the Pareto front. In this thesis, we generate the ES, which is the close to the Pareto front, by evolving the solutions on the Pareto front at a time slot.

Figure 4 shows the overview of the network control using ES. Evolvability Search is a computationally time-consuming method because a sufficient number of offspring samples are required to be evaluated to estimate the evolvability of an individual. Therefore, Evolvability Search is executed in parallel to the evolution of the candidate network configuration to follow the changes. The evolution of the candidate network configuration is performed using the previous network configuration and ESs that are the results of the previously completed Evolvability Search. On

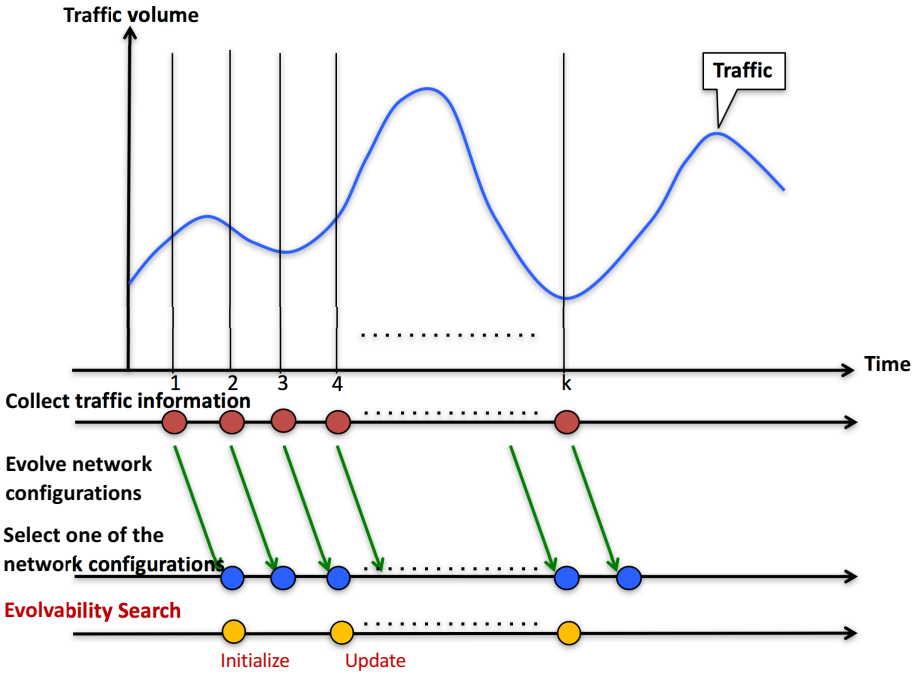


Figure 4: Control method with Evolvability Search

the other hand, Evolvability Search uses the network candidate configurations generated by the process of the evolution of the candidate network configurations at the time slot when Evolvability Search starts as the initial solutions. We limit the number of generations in Evolvability Search so as to avoid the solutions far from the Pareto front.

4.4.2 Calculation of evolvable solutions

In this thesis, the ES is obtained by Evolvability Search. To perform the Evolvability Search, the evolvability must be defined. As discussed in Section 2.3, the evolvability is quantified by the number of unique individuals born from the parent individual. That is, the evolvability is defined by defining what is unique.

In this thesis, we define the uniqueness of individuals based on the ability to handle different situations. In the network, the flows passing the bottleneck links of each solution are important features indicating the situation the solution can handle; the solution can accommodate traffic without congestion unless the flow passing the bottleneck links becomes large. The flows passing the bottleneck links also indicate the capability to save the power; if the bottleneck link is passed

by a large number of flows, we can shut down the other links.

Therefore, we define the uniqueness by the combination of the flows passing the bottleneck links. We evaluate whether the new individual x is unique based on following $B(x)$, and add individuals considered as unique to the list U_p if Bx is above a pre-specified threshold (here 1000000 or 2000000).

$$B(x) = \min_{g \in U_p} b(x, g) \tag{7}$$

where $b(x, g)$ is the difference between the behavior of the individual x and the individual g, and is defined by

$$b(x, g) = \|\boldsymbol{b_x} - \boldsymbol{b_g}\|_2 \tag{8}$$

where $\boldsymbol{b_x} = (b_x(1), b_x(2), ..., b_x(f), ..., b_x(n))$, n is the number of flows in the network, $b_x(f)$ is a value indicating the rate of congested links the flow f in x goes through. $b_x(f)$ is defined by

$$b_x(f) = \min_{k \in L_f} \frac{C(k)}{N(k)} \tag{9}$$

where L_f is the set of links that go through in flow f, $N(k)$ is the number of flows through link k, and $C(k)$ is the capacity of link k .

Table 1: Simulation Environment

Power consumption of node	0.7 [kw]
Power consumption of link	0.07 [kw]
Maximum transmission speed at each port (FatTree)	1 [Gbps]
Maximum transmission speed at each port (Internet2)	100 [Gbps]

Table 2: Parameters of our method

Parameter	Value
Number of saved configurations	50
Generation in each time slot	1
Crossover ratio	0.5
Mutation ratio	0.5

5 Evaluation

In this section, we evaluate our method. We first demonstrate that our method saves the energy consumption, following the actual daily traffic changes. Then, we demonstrate that our method configures the suitable network configuration, even when sudden traffic changes or failures occur. Finally, we discuss the scalability of our method.

In our evaluation, we set the power consumption of the links and switches based on the fact that the power consumption of nodes is 10 times as large as that of links [5]. Table 1 shows the power consumption and capacity of links and nodes used in our evaluation.

The parameters used in our evaluation are shown in Table 2. And in this thesis, we define k of k-shortest path by the hop length of the shortest path; we calculate k-shortest path while the hop of path is less than $(hop\ of\ the\ shortest\ path) * 1.5$.

5.1 The case of actual daily traffic changes

5.1.1 Evaluation environment

Network topology and traffic In this evaluation, we use the traffic trace monitored at the Internet2, and the network topology based on the Internet2 topology [24].The traffic data at the

Internet2 are collected by the Netflow protocol. The sampling rate is one out of every 100 packets, and aggregated data are exported every 5 minutes. We use the traffic data monitored from 1 November 2011 to 4 November 2011.

The network topology used in this evaluation is generated based on the PoP-level topology of the Internet2. Network topologies without redundant nodes are not suitable to our evaluation; if a network topology does not include redundant node, any node cannot be shut down under the constraint that all flows can be accommodated. Therefore, we generate a network topology so as to include redundant nodes by the following steps. We deploy one access router and three backbone routers for each PoP. Then, we connect each access router to all of the backbone routers in the same PoP. The backbone routers in the near PoPs are connected randomly based on degree; select lowest degree backbone node for each PoP. Then, we check if the generated topology has a sufficient number of distinct paths between all access routers. If a pair of access routers does not has a sufficient number of distinct paths, we connect PoPs (unconnected backbone routers) which are passed by the shortest distinct path because we demonstrate that our method saves the energy consumption under the constraint that a sufficient number of distinct paths are provided.

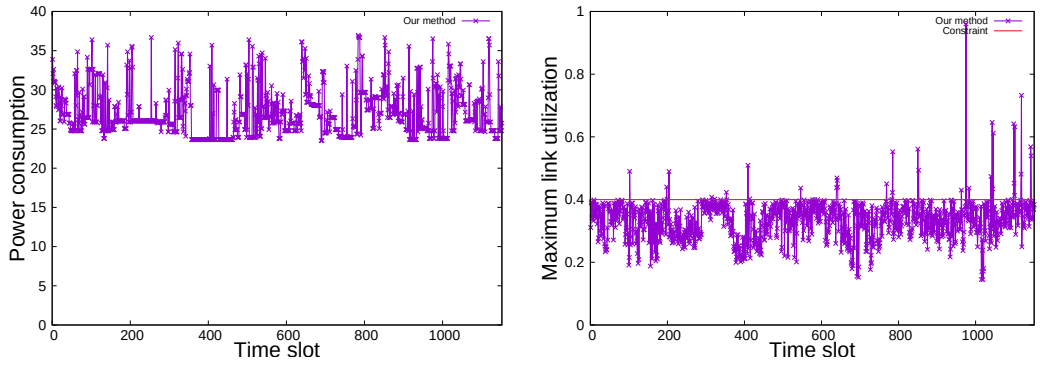
The generated topology includes the redundant nodes; all flows between access routers can be accommodated even when some backbone routers are shut down. While shutting down backbone routers saves the energy consumption, powering on more backbone routers provides more capacities to flows.

SLA requirements In this evaluation, we use the objective functions defined in Section 4.1. By using these objective functions, we set the following requirements.

- Maximum link utilization should be less than 0.4.
- At least 2 distinct paths should be provided between any access node pairs.

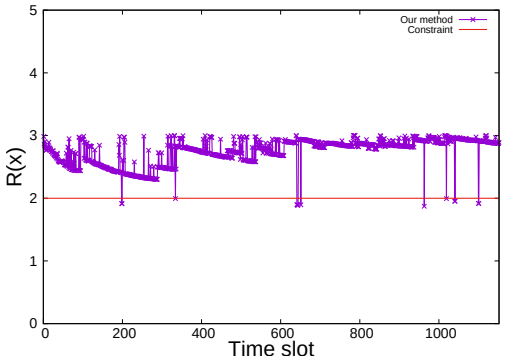
5.1.2 Result

Figure 5 shows the results for 4 days. In these figures, the horizontal axis is the time slot. The vertical axis indicates the values of the objective functions. This figure indicates that our method follows the traffic variation; our method reduces the power consumption when traffic volume is



(a) Power consumption

(b) Performance



(c) Fault tolerance

Figure 5: Result for daily traffic changes

low, and satisfies the SLA by turning up more devices even when traffic volume becomes large in most of time slots.

This figure also indicates that the SLA are violated at the several time slots. This is caused by the sudden traffic changes. In this case, all of the candidate configurations calculated by using the previously monitored traffic are far from the suitable network configuration. However, even in such cases, our method finds the network that satisfies the SLA and the SLA becomes satisfied at the next time slot. The candidate configurations found at the next time slot may not be different from the Pareto optimal solutions, which causes the increases of the power consumption. However, the candidate configurations evolve to achieve the Pareto optimal solutions. As a result, the power consumption becomes saved in several time slots. The details in the case with the sudden traffic changes are investigated in Section 5.2.

Though we set the number of generations in each time slot to 1 in this evaluation, the number

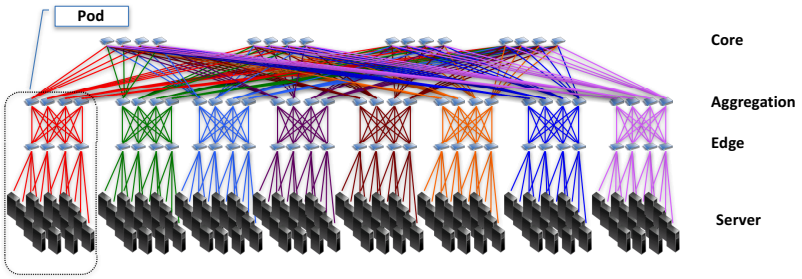


Figure 6: Network topology: FatTree

of generations in each time slot can be set to a larger value. If we set the number of generations to a large value, we can reduce the number of time slots required to achieve the Pareto optimal solutions.

5.2 The case of sudden traffic changes

In this subsection, we evaluate our method in the case of sudden traffic changes, and investigate the number of time slots to achieve the suitable network configurations.

5.2.1 Evaluation environment

In this evaluation, we use two network topologies in addition to the network topology based on the Internet2 used in Section 5.1.1. By using these network topologies, we demonstrate that our method works in any network topologies.

For each network topology, we generate the traffic to simulate the case with sudden traffic changes. The details of the traffic used in this evaluation is explained in the latter in this subsection. The other parameters including the SLA are set to the same values as Section 5.1.

Network topology

FatTree The FatTree topology is a typical network architecture in data centers. This network topology is symmetric and has a large number of distinct paths with a similar number of hops. In this evaluation, we use the FatTree topology with 8 ports for each switch shown in Figure 5.2.1. In this topology, each pod consists of 16 servers and 2 layers of 4 switches. Each edge switch

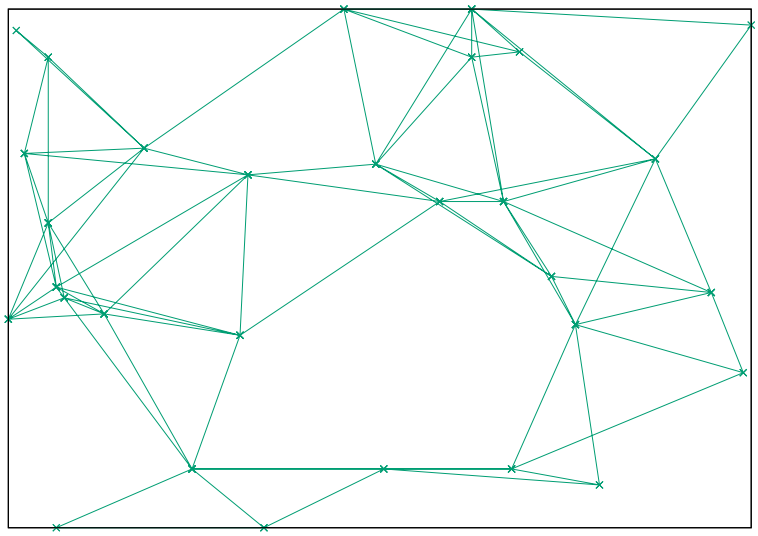


Figure 7: Topology created by Waxman model

connects to 4 servers and 4 aggregation switches. Each aggregation switch connects to 4 edge switches and 4 core switches. Each core switch connects to 8 pods.

Topology created by Waxman model Waxman model is a typical network model. By using the Waxman model, we can generate the network topology randomly. Unlike the FatTre, the network generated by the Waxman model is not symmetry. That is, the flows has the different number of distinct paths, and the number of hops of the distinct paths are different.

Figure 5.2.1 shows the network topology generated by the Waxman moel. In this evaluation, we set the average degree to 4. Similar to the case of the Internet2 topology, we use the topology generated by the Waxman model as a PoP level topology; we deploy one access router and three backbone routers, and connect them by the same way as Section 5.1.1.

The topology generated by the Waxman model is also used to discuss the scalability in Section 5.4. That is, we compare the result for the network topologies with the different sizes. The network size may have an impact on the number of flows passing a link, and the probabilities that congestion occurs may change. To avoid such impacts of the number of flows passing each link, we set the capacity of each link based on the betweenness centrality. The capacity of the link l is

set to

$$l_l = T * b(l)$$

where T is a parameter, and $b(l)$ is the betweenness centrality of the link l. In this evaluation, T is set to 100000.

Traffic In this evaluation, we set the same traffic demand for D time slots and investigate the values of the objective functions achieved at each time slot. By doing so, we investigate the number of time slots to satisfy the SLA or to achieve the suitable network configurations after the traffic change occurs. After D time slots, we generate new traffic demand, and investigate the values of the objective functions.

We have the actual traffic traces for the Internet2. Thus, we generate the traffic demand at each time slot based on the traffic traces. In the following discussion, we use traffic demands at 0:00 am on 1 November 2011, 7:55 pm on 1 November 2011 and 6:40 am on 2 November 2011. By using such traffic, we can generate the sudden traffic changes at time slot 300, 600.

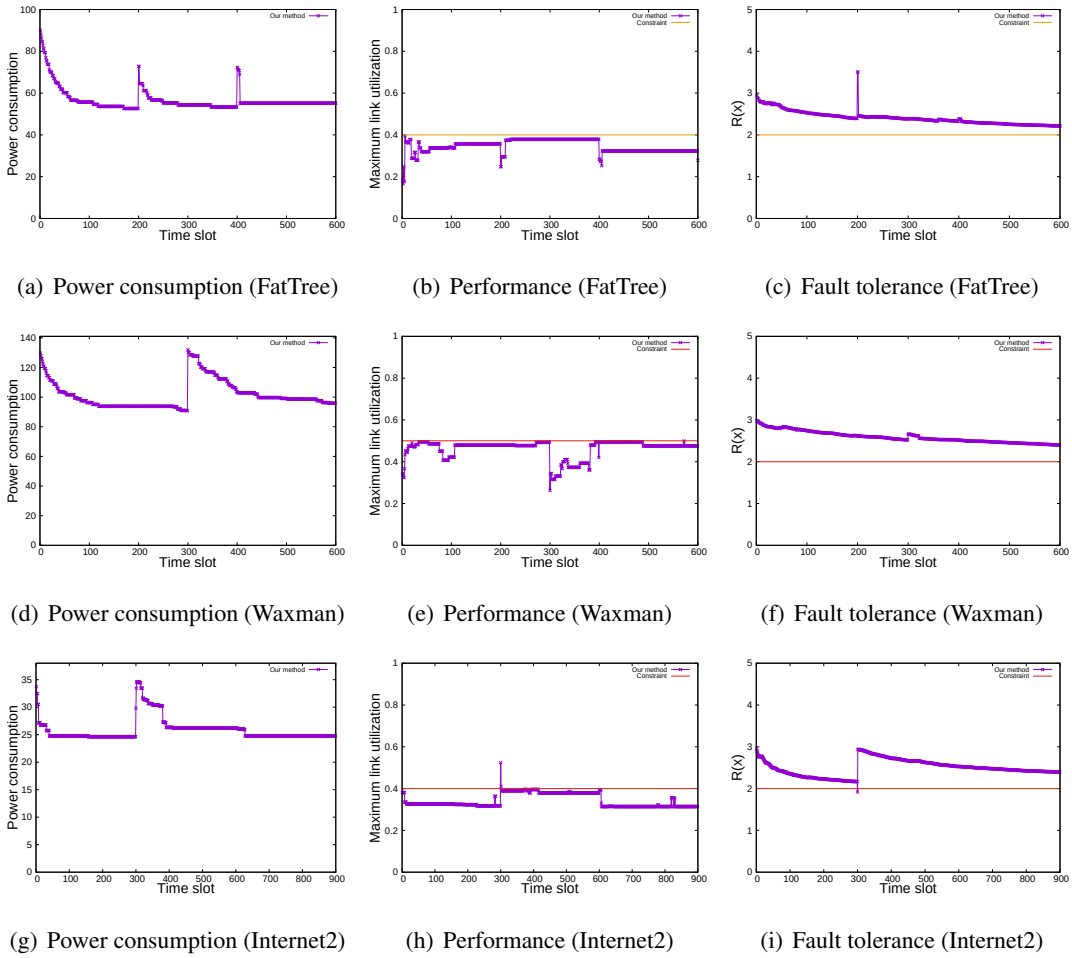
We have no actual traffic data for the other topologies. Thus, we generate traffic demand randomly by the following steps.

1. Define maximum of traffic demand at each link l: $T_l = \frac{capacity}{2(N_{PoP}-1)}$.
2. Select one link of PoP randomly.
3. Generate traffic: $T_l * (0 - -1)$. End if all PoPs are selected. Go to step2 otherwise.

5.2.2 Result

Figure 8 shows the results. The results indicate that all of the SLA constraints are satisfied in most cases even when sudden traffic changes occur. This is because our method has multiple candidate configurations, and if one of them satisfies all of the SLA, our method select it to satisfy the SLA. As a result, even if no candidates satisfying the SLA are found and the SLA is violated when the sudden traffic changes occur, the SLA constraints become satisfied in a few time slots.

On the other hand, the power consumption remains large unless the suitable network configurations that saves the power consumption without violating the SLA are found. Figure 8(a), 8(d) and 8(g) shows the power consumption becomes small as the time slot goes on. This is because the candidates becomes close to Pareto optimal as the time slot goes on.



(a) Power consumption (FatTree) (b) Performance (FatTree) (c) Fault tolerance (FatTree)

(d) Power consumption (Waxman) (e) Performance (Waxman) (f) Fault tolerance (Waxman)

(g) Power consumption (Internet2) (h) Performance (Internet2) (i) Fault tolerance (Internet2)

Figure 8: Result for sudden traffic fluctuation

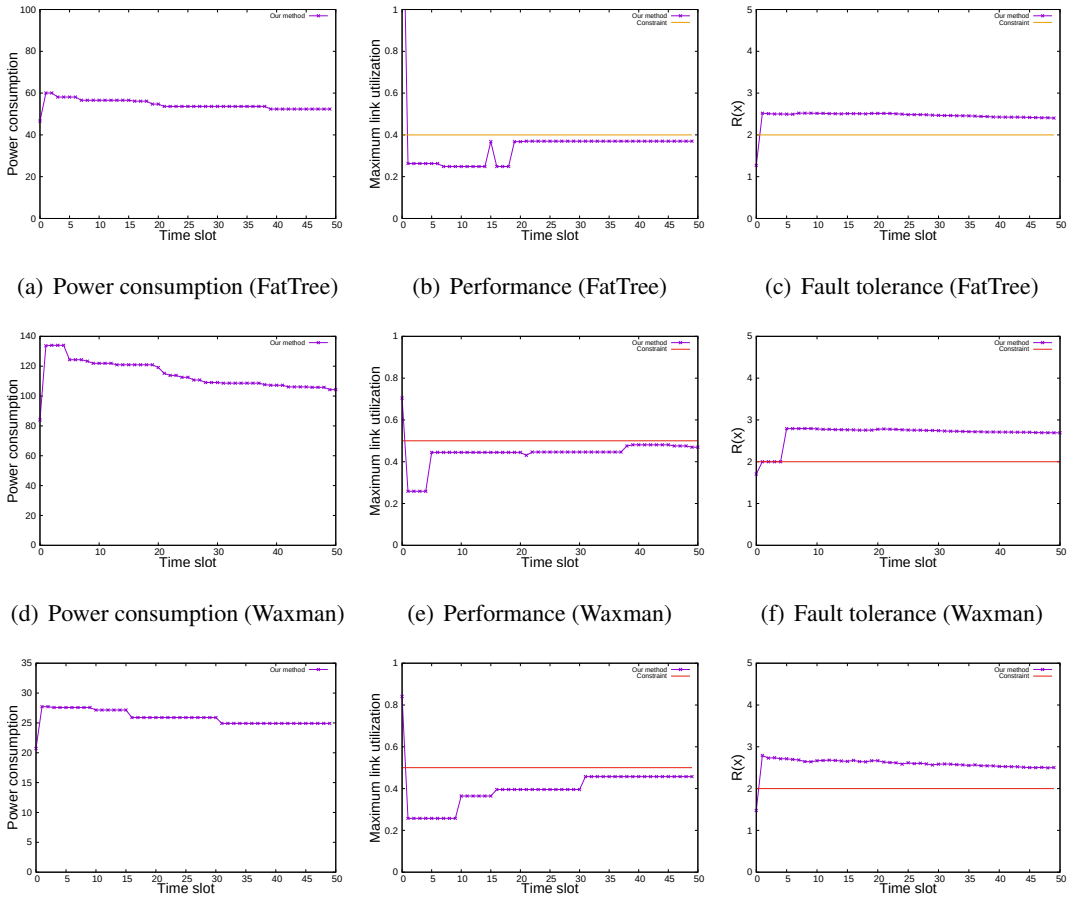
Figure 8 also shows that our method works similarly in any network topologies; the power consumption becomes large after the sudden traffic changes but decreases as the time slot goes on in any topologies. This result indicates that our method can work in any network topologies.

5.3 Case of failures

5.3.1 Evaluation environment

In this evaluation, we investigate the case with failures. We generate the failure on the powered-on node with the highest degree, which has the largest impact on the network. To focus on the impact on the failures, we do not change the traffic during the evaluation.

The other parameters are set to the same values as Section 5.1.



(a) Power consumption (FatTree) (b) Performance (FatTree) (c) Fault tolerance (FatTree)

(d) Power consumption (Waxman) (e) Performance (Waxman) (f) Fault tolerance (Waxman)

(g) Power consumption (Internet2) (h) Performance (Internet2) (i) Fault tolerance (Internet2)

Figure 9: Evaluation values with failures

5.3.2 Result

Figure 9 shows the results. In this figure, the vertical axis indicates the power consumption, maximum link utilization, and the reliability, and the horizontal axis indicates the time slots after the failure occurs.

Figure 9(c), 9(f) and 9(i) shows that the reliability constraints are not satisfied at the time slot 0 due to the failure. Even in this case, our method can keep the connectivity between all access nodes by immediately rerouting the flow passing the failed node to the backup paths. However, this reroute causes the increases of the maximum link utilization at the time slot 0 shown in Figure 9(b), 9(e) and 9(h). However, the candidate network configurations satisfying the SLA can be found at the next time slot, but the power consumption becomes large, similar to the case with the sudden

Table 3: Topology created by Waxman model

Topology ID	Number of flows
I	90
II	380
III	870
IV	1560
V	2540

traffic changes. This is because our method finds the network configuration satisfying the SLAs, but the current candidates are far from the Pareto optimal. Finally, the power consumption also becomes small as the time slot goes on. This is because the candidates are evolved to be close to the Pareto optimal. That is, even in case of failure, our method achieves suitable network configuration that saves power consumption without violating the SLAs if we have a sufficient number of nodes and links that are not broken.

5.4 Scalability

Finally, we discuss the scalability of our method. In a large network, the calculation time to evolve the candidates may become large. In addition, the number of generations to achieve a sufficient solution may also become large. Therefore, in this section, we discuss the calculation time and the number of generations to achieve a sufficient solution.

5.4.1 Evaluation environment

In this evaluation, we generate the network topology by Waxman model to generate the network topology with various sizes as shown in Table 3. The other parameters are set to the same values as Section 5.1.

5.4.2 Calculation time

We investigate the calculation time required for one generation. To investigate the calculation time, we implemented our method by C++ and run it on the computer with four Intel(R) Xeon(R) CPUs (E7- 4870) and 512 GB.

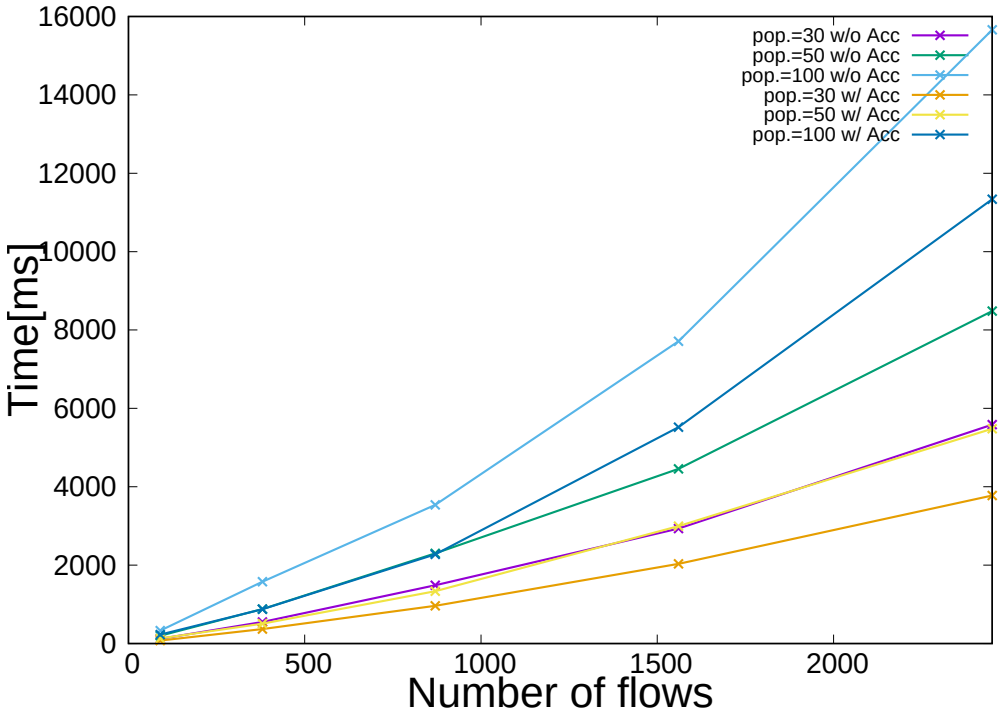


Figure 10: Calculation time

Figure 10 compares the calculation time of the method with and without the acceleration proposed in Section 4.3. According to the discussion in Section 4.2.2, the calculation time depends on the number of candidates generated in one generation and the number of flows whose routes are to be calculated. Thus, we plot the results of the various number of candidates, and the relation between the calculation time and the number of flows in a network. The vertical axis indicates the calculation time in one generation and the horizontal axis indicates the number of flows in the network. The results indicate that the calculation time become large as the number of candidates becomes large. The results also indicate that the calculation time increases as the number of flows increases.

The results also show that the acceleration reduces the calculation time. By using the acceleration, we can generate 100 new candidates of the next generation in 12 seconds even for a network including 2450 flows.

5.4.3 Convergence time

In our approach, the number of the generations required to achieve suitable solutions are important; if a large number of generations are required, it takes a long time to achieve a suitable network configuration after the sudden changes occur.

The number of generations required to achieve a suitable solution depends on the topology. Therefore, we use two topologies to evaluate the convergence time.

For each topology, we generate two kind of traffic, and the traffic change from the first traffic to the second one is generated after performing a sufficient number of evolutions for the first traffic. Then, we investigate the time required to achieve a suitable solution. In this evaluation, we generate two patterns of traffic changes.

- Case I: The traffic between the access routers are changed randomly without changing the total traffic volume of all access router pairs.
- Case Ⅱ: The traffic between the access routers are changed randomly so that the total traffic volume of all access router pairs becomes double of the total traffic volume before change

To investigate the time required to achieve a suitable solution, we define two kinds of suitable solutions.

- Solution satisfying constrains: A solution satisfying all of the constraints.
- Energy-efficient solution: A solution that satisfies all of the constraints and saves the energy consumption sufficiently.

The energy consumption that can be saved depends on the topology and traffic. Thus, we define the solution that saves the energy consumption sufficiently by using the energy consumption achieved by running our method for a sufficient number of generations. To define an energy-efficient solution, we perform our methods until the energy consumption becomes unable to be improved for 500 generations. Then, we define the finally achieved energy consumption as the lowest energy consumption ($P^{\min}$). Then, we define the energy-efficient solutions as the solutions whose saved energy consumptions are $(P^{\max} - P^{\min})(1 - \epsilon)$, where $P^{\max}$ is the energy consumption when all links are powered on, and ϵ is a parameter. That is, the energy consumption achieved by the energy-efficient solution is less than

$$P^{\text{efficient}} = P^{\max} - \big((P^{\max} - P^{\min})(1 - \epsilon)\big)$$

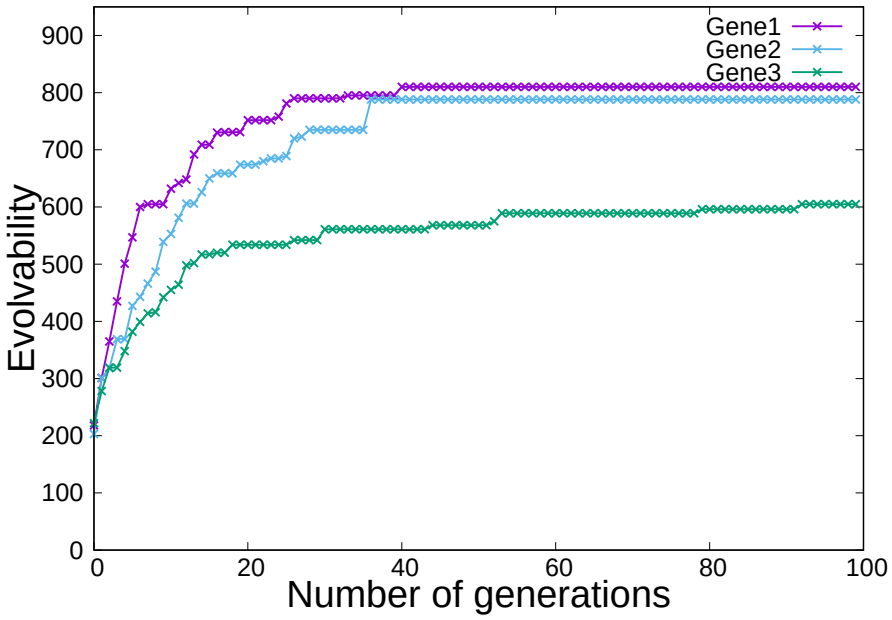


Figure 11: Transition of Evolvability

In this evaluation, we set ϵ to 0.1

In this evaluation, we compare three methods.

- Method with ES

  The method uses ES proposed in Section 4.4 in addition to the candidate network configuration in the previous time slot.

- Method with RS

  This method uses the randomly generated initial solutions in addition to the candidate network configuration in the previous time slot instead of ES. By comparing with this method, we demonstrate the impact of using ES.

- Method without ES nor RS

  This method evolves the candidate network configurations from only the candidates in the previous time slot.

To compare these methods, the Pareto front before the traffic change should be the same for all of three methods. In this evaluation, we prepare three Pareto Fronts that are sufficiently evolved in the environment before fluctuation for each topology.

Table 4: Number of generations for convergence to solution satisfying constrains

(a) Case I (Topology I)

	w/ ES	w/ RS	w/o ESRS
Avg.	1	1	1
Max	1	1	1
Min	1	1	1

(b) Case II (Topology I)

	w/ ES	w/ RS	w/o ESRS
Avg.	3.93	2.93	3.96
Max	14	8	29
Min	1	1	1

(c) Case I (Topology II)

	w/ ES	w/ RS	w/o ESRS
Avg.	1	1	1
Max	1	1	1
Min	1	1	1

Figure 11 shows the evolvability achieved by Evolvability Search from three Pareto front in Topology I. This figure indicates that the evolvability rapidly increases. That is, ES can be calculated with a small number of generations.

Figure 12 compares the number of generations required to converge to energy-efficient solution. In this figure, the horizontal axis shows the number of generations of ES in the case of method with ES, and the vertical axis shows the average number of generations required to converge to energy-efficient solution. The figure shows that ES reduces the number of generations to converge to energy-efficient solution, comparing the methods with ES, with RS and without ESRS.

We also investigate the impact of the number of generations of ES. Figure 12 shows that the increase of the number of generations of ES does not always reduce the number of generations to converge to energy-efficient solution. This is caused by that the Evolvability Search generates the solutions far from the Pareto front. Such solutions are removed at the first selection. As a result, the effectiveness of ES is degraded.

Figure 12 also shows the number of generations of ES that minimizes the number of generations to converge to energy-efficient solution depends on the case. ES of 10 generations is the most effective in Figure 12(a) while ES of 20 generations is the most effective in Figure 12(b). This is caused by the difference of how far the appropriate Pareto front after the change and the

previous Pareto front are. When the traffic fluctuation is large, the previous Pareto front is far from the appropriate Pareto front. In this case, the previous Pareto front does not work effectively on the convergence to the power saving solution, and the ES that are far from the previous Pareto front contribute the convergence to the energy-efficient solution.

The ES reduces the number of generations to converge to energy-efficient solution in the larger topology as shown in Figure 12(c). However, as the size of the topology increases, the solution space becomes larger. As a result the number of required generations becomes large.

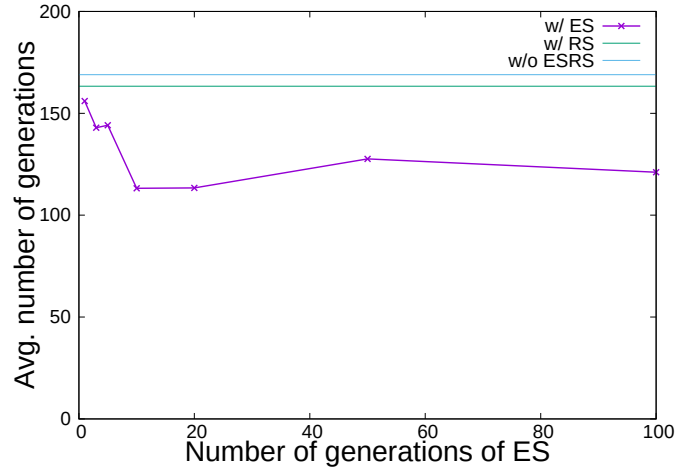
Figure 13 shows the complementary cumulative distribution function of the number of generations required for convergence to energy-efficient solution. For the method with ES, we plot the result of using ES of the generation with the highest performance in Figure 12. This figure also demonstrates that the methods with ES reduces the number of generations required for convergence, compared with the methods with RS and without ESRS. However, there are cases where a large number of generations are required even in the methods with ES.

This is caused by that the candidate network configuration at a time slot become the local optimum solutions. In this case, the solution that sufficiently saves the energy consumption can be generated only by mutations. Because the ES is generated from the candidate network configurations at a previous time slot, the ES may also be close to the local optimum solutions if the candidate network configurations used to generate ES is local optimum solutions. One approach to avoiding such a situation is to generate new initial solutions. However, the evolution from new initial solutions also requires a large number of generations. Therefore, how to handle the case with the local optimal solutions is one of our future work.

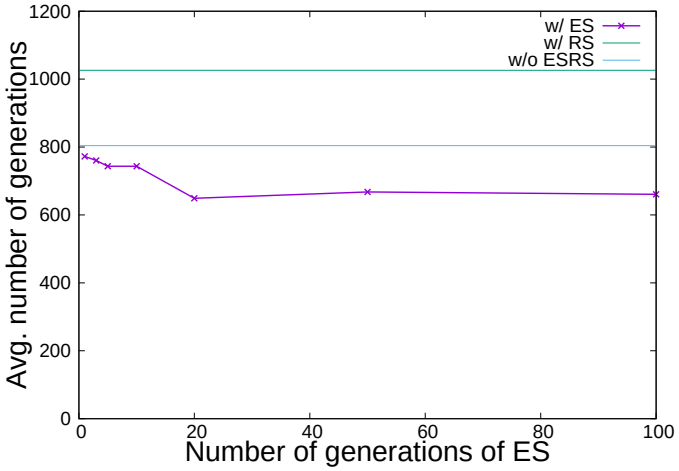
Figure 13 indicates that the method with RS requires a larger number of generations than even the method without ES nor RS. This is because a large number of solutions that cannot reduce the power consumption are generated by adding the randomly generated initial solution.

Table 4 indicates the number of generations for convergence to solution satisfying constraints. For the method with ES, we fill out the table with the result of using ES of the generation with the highest performance in Figure 12. The result shows that all methods can obtain the solution satisfying the constraints in one generation when total traffic volume does not change. On the other hand, when large fluctuation occurs, at most 14 generations are required by the method with ES. This is cause by that the suitable network reconfiguration becomes far from the previous Pareto front. However, the average number of generations to obtain solution satisfying the constraints is

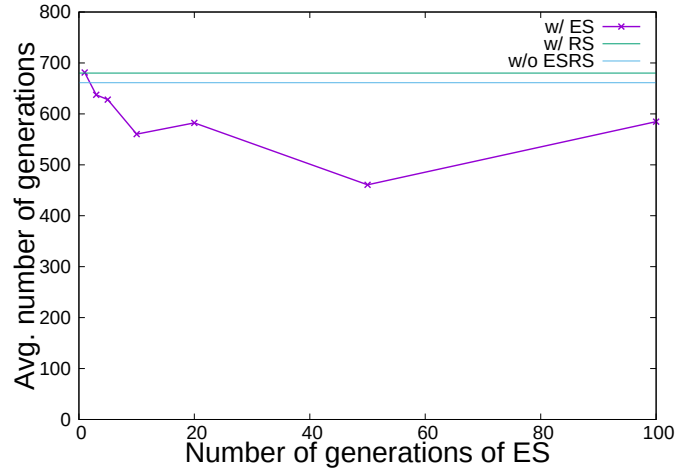
less than 4 in all methods. That is, the solution satisfying the constraints can be found immediately after the changes in most cases.



(a) Case I (Topology I)

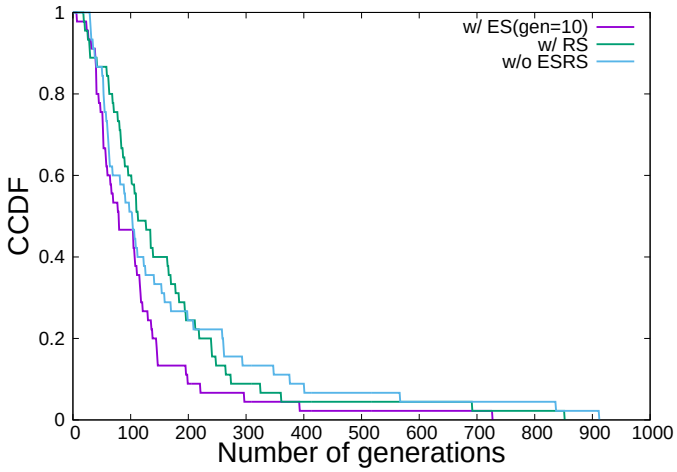


(b) Case II (Topology I)

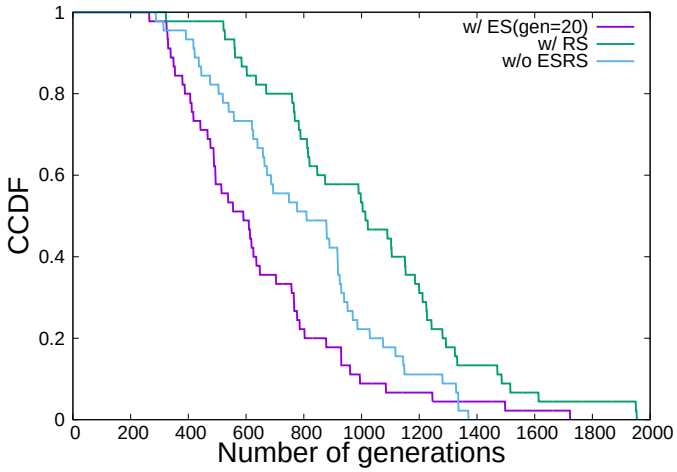


(c) Case I (Topology II)

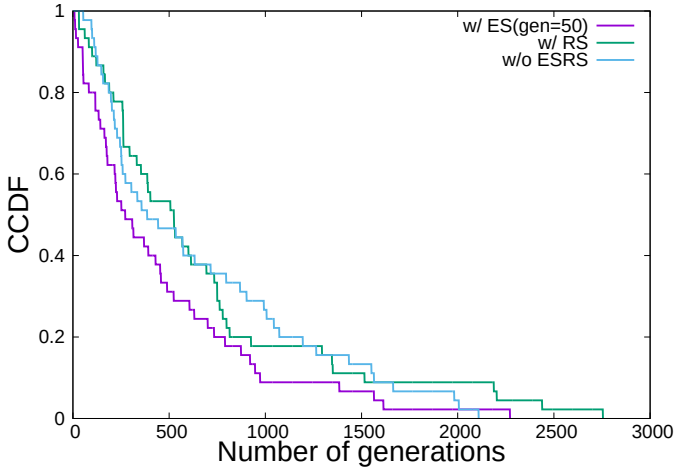
Figure 12: Average number of generations for convergence to energy-efficient solution



(a) Case I (Topology I)



(b) Case II (Topology I)



(c) Case I (Topology II)

Figure 13: Complementary cumulative distribution function of number of generations for convergence to energy-efficient solution

6 Conclusion

In this thesis, we proposed a network power saving method that handles multiple objectives, following the environmental changes. Our method calculates Pareto front to consider all of these objectives. Then our method select a solution whose power consumption is the smallest among the solution on the Pareto front satisfying required performance and reliability.

We evaluated our method by simulation. The results show that our method reduces the power consumption without violating the SLA constraints, following the actual and rapid traffic changes in general networks. In addition, even when a failure occurs, our method rebuilds paths and gradually recovers the network configurations so that the energy consumption is minimized under the SLA constraints.

We also proposed a method to accelerate the evolution of the Pareto front. In this method, we introduced the evolvable solutions (ES) which are the solutions with high evolvability. By calculating the Pareto front from the solutions in the previous Pareto front and ES, an appropriate Pareto front can be obtained in a small number of generations.

We also evaluated the method to accelerate the evolution of the Pareto front by simulation. The results show that the method with ES obtains the solution satisfying the constraints in about 4 generations, and the solutions that saves the power consumption sufficiently in about 650 generations.

Our future work includes to reduce the number of generations required to achieve the energy-efficient solution especially by handling the case that the candidate network configurations fall into local optimal. Another future research topic is to investigate the policy to set the number of generations of ES.

Acknowledgements

This thesis would not accomplish without a lot of great supports of many people. First, I would like to express my deepest gratitude to Professor Masayuki Murata of Osaka University, for his valuable comments, insights and continuous encouragement. Furthermore, I would like to show my greatest appreciation to Assistant Professor Yuichi Ohsita of Osaka University. He devoted a great deal of time for me and gave me a lot of advices about my research. Without his continuous support, my work would not be accomplished. In addition, he enthusiastically taught me the way of thinking and writing through this thesis. They must be invaluable skills in the future of my life. Moreover, I would like to show my appreciation to Associate Professor Shin'ichi Arakawa, Assistant Professor Daichi Kominami, Specially Appointed Assistant Professor Naomi Kuze and Tatsuya Otoshi of Osaka University. Finally, I would like to thank all the members of Advanced Network Architecture Research Laboratory at the Graduate School of Information Science and Technology, Osaka University, for their support and advices.

References

[1] Cisco, "Cisco global cloud index:forecast and methodology, 2014-2019," tech. rep., Cisco Systems Inc., Oct. 2015.

[2] Van Heddeghem, Ward and Lambert, Sofie and Lannoo, Bart and Colle, Didier and Pickavet, Mario and Demeester, Piet, "Trends in worldwide ICT electricity consumption from 2007 to 2012," *Computer Communications*, vol. 50, pp. 64–76, Sep. 2014.

[3] Lambert, Sofie and Van Heddeghem, Ward and Vereecken, Willem and Lannoo, Bart and Colle, Didier and Pickavet, Mario, "Worldwide electricity consumption of communication networks," *Optics express*, vol. 20, pp. B513–B524, Dec. 2012.

[4] Y. Wei, X. Zhang, L. Xie, and S. Leng, "Energy-aware traffic engineering in hybrid sdn/ip backbone networks," *Journal of Communications and Networks*, vol. 18, pp. 559–566, Sep. 2016.

[5] Amaldi, E. and Capone, A. and Gianoli, L. G., "Energy-aware IP traffic engineering with shortest path routing," *Comput. Netw.*, vol. 57, pp. 1503–1517, Apr. 2013.

[6] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP network energy cost: formulation and solutions," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, pp. 463–476, Apr. 2012.

[7] Chiaraviglio, Luca and Mellia, Marco and Neri, Fabio, "Reducing power consumption in backbone networks," in *Proceedings of Communications, 2009. ICC'09. IEEE International Conference on*, pp. 1–6, IEEE, Jun. 2009.

[8] R. L. Gomes, L. F. Bittencourt, E. R. Madeira, E. Cerqueira, and M. Gerla, "Two-criteria pareto frontier for virtual network allocation on edge-as-a-service networks," *Computer Communications*, vol. 102, pp. 58 – 66, Apr. 2017.

[9] Zaheeruddin, D. Lobiyal, and S. Prasad, "Ant based pareto optimal solution for qos aware energy efficient multicast in wireless networks," *Applied Soft Computing*, vol. 55, pp. 72 – 81, Jun. 2017.

[10] Kessaci, Yacine and Melab, Nouredine and Talbi, El-Ghazali, "A pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation," *Cluster Computing*, vol. 16, pp. 451–468, Sep. 2013.

[11] Battiti, Roberto and Passerini, Andrea, "Brain–computer evolutionary multiobjective optimization: a genetic algorithm adapting to the decision maker," *Evolutionary Computation, IEEE Transactions on*, vol. 14, pp. 671–687, Sep. 2010.

[12] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, pp. 992–1007, Sep. 2006.

[13] Fadaee, M and Radzi, MAM, "Multi-objective optimization of a stand-alone hybrid renewable energy system by using evolutionary algorithms: a review," *Renewable and Sustainable Energy Reviews*, vol. 16, pp. 3364–3369, Jun. 2012.

[14] Ahmadi, Pouria and Dincer, Ibrahim and Rosen, Marc A, "Thermoeconomic multi-objective optimization of a novel biomass-based integrated energy system," *Energy*, vol. 68, pp. 958–970, Apr. 2014.

[15] Q. Wang, M. Guidolin, D. Savic, and Z. Kapelan, "Two-objective design of benchmark problems of a water distribution system via moeas: Towards the best-known approximation of the true pareto front," *Journal of Water Resources Planning and Management*, vol. 141, p. 04014060, Jul. 2014.

[16] H. Mengistu, J. Lehman, and J. Clune, "Evolvability search: directly selecting for evolvability in order to study and produce it," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pp. 141–148, ACM, Jul. 2016.

[17] Giroire, Frédéric and Moulierac, Joanna and Phan, Truong Khoa and Roudaut, Frédéric, "Minimization of network power consumption with redundancy elimination," *Computer communications*, vol. 7298, pp. 247–258, May 2012.

[18] Giroire, Frédéric and Moulierac, Joanna and Phan, Truong Khoa and Roudaut, Frédéric, "Minimization of network power consumption with redundancy elimination," *Computer communications*, vol. 59, pp. 98–105, Mar. 2015.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, pp. 182–197, Apr. 2002.

[20] T. D. Prasad and N.-S. Park, "Multiobjective genetic algorithms for design of water distribution networks," *Journal of Water Resources Planning and Management*, vol. 130, pp. 73–82, Dec. 2004.

[21] Deb, Kalyanmoy and Agrawal, Samir and Pratap, Amrit and Meyarivan, Tanaka, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," *Lecture notes in computer science*, vol. 1917, pp. 849–858, Sep. 2000.

[22] J. Brookfield, "Evolution: the evolvability enigma," *Current Biology*, vol. 11, pp. R106–R108, Feb. 2001.

[23] M. Barbehenn, "A note on the complexity of dijkstra's algorithm for graphs with weighted vertices," *IEEE Transactions on computers*, vol. 47, p. 263, Feb. 1998.

[24] M. H. Gunes and K. Sarac, "Inferring subnets in router-level topology collection studies," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 203–208, ACM, Oct. 2007.