Exploiting Linkage Information and Problem-Specific Knowledge in Evolutionary Distribution Network Expansion Planning

Ngoc Hoang Luong   Hoang.Luong@cwi.nl
Centrum Wiskunde & Informatica (CWI), 1098 XG Amsterdam, The Netherlands

Han La Poutré   Han.La.Poutre@cwi.nl
Centrum Wiskunde & Informatica (CWI), 1098 XG Amsterdam, The Netherlands
Delft University of Technology, 2628 CD Delft, The Netherlands

Peter A. N. Bosman   Peter.Bosman@cwi.nl
Centrum Wiskunde & Informatica (CWI), 1098 XG Amsterdam, The Netherlands

Abstract

This article tackles the Distribution Network Expansion Planning (DNEP) problem that has to be solved by distribution network operators to decide which, where, and/or when enhancements to electricity networks should be introduced to satisfy the future power demands. Because of many real-world details involved, the structure of the problem is not exploited easily using mathematical programming techniques, for which reason we consider solving this problem with evolutionary algorithms (EAs). We compare three types of EAs for optimizing expansion plans: the classic genetic algorithm (GA), the estimation-of-distribution algorithm (EDA), and the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA). Not fully knowing the structure of the problem, we study the effect of linkage learning through the use of three linkage models: univariate, marginal product, and linkage tree. We furthermore experiment with the impact of incorporating different levels of problem-specific knowledge in the variation operators. Experiments show that the use of problem-specific variation operators is far more important for the classic GA to find high-quality solutions. In all EAs, the marginal product model and its linkage learning procedure have difficulty in capturing and exploiting the DNEP problem structure. GOMEA, especially when combined with the linkage tree structure, is found to have the most robust performance by far, even when an out-of-the-box variant is used that does not exploit problem-specific knowledge. Based on experiments, we suggest that when selecting optimization algorithms for power system expansion planning problems, EAs that have the ability to effectively model and efficiently exploit problem structures, such as GOMEA, should be given priority, especially in the case of black-box or grey-box optimization.

1 Introduction

A typical power system consists of power plants that generate the electricity, transmission networks that carry the high-voltage (HV) electricity from far-off generating sites to substations near residential and industrial areas, and distribution networks that feed the medium-voltage (MV) and low-voltage (LV) electricity from the substations to

homes and businesses. In this article, the focus is on distribution networks, but the methodologies can also be extended to transmission networks. In order for distribution networks to work properly, distribution network operators (DNOs) have to ensure that the capacities of network cables are sufficient to handle the magnitude of the power flows that are carried through the cables to satisfy customers' power demands. Otherwise, bottlenecks can cause overloads, which heat up the cable wires. This is detrimental to the normal operation and safety of the networks, and may cause blackouts or earlier cable replacements. Therefore, DNOs need to perform Distribution Network Expansion Planning (DNEP) to determine *what* kinds of network enhancements should be made and *where* these enhancements should be made. The dynamic DNEP formulation also involves the question *when* those reinforcement activities should be started during the planning period while in the static DNEP formulation this time-dependent decision making issue is omitted. The goal of DNEP is to find the most economical expansion plan, in terms of investment and operation costs, for which the network satisfies the power demand over the planning period.

DNEP is sometimes simplified to be scalably solved by classical mathematical programming methods, compromising the true non-linear nature of DNEP that is due to its complicated properties and constraints, and thus leading to unsatisfactory representations of the real problem (Ramirez-Rosado and Bernal-Agustin, 1998). On the other hand, evolutionary algorithms (EAs) have been widely applied and achieved practical results in DNEP with more realistic formulations, see e.g., (Borges and Martins, 2012; Diaz-Dorado et al., 2002; Ramirez-Rosado and Bernal-Agustin, 1998). This is mostly due to the straightforward implementation and broad applicability of EAs. However, most DNEP studies in literature overlook several important issues. First, experiments are usually conducted by using only one, arbitrarily chosen, EA with a customized problem-specific variation operator (VO), omitting both questions why that specific EA should be chosen over other available EAs and what the advantages that VO has compared to other alternatives. Second, the comparison of how effective various constraint-handling mechanisms help the solvers traverse the search space is often disregarded. In this paper, while aiming to solve a formulation of the DNEP problem that captures many important real-world considerations, we also address these issues. We employ 3 EA solvers: a classic genetic algorithm (GA), a estimation-of-distribution algorithm (EDA), and a Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) (Bosman and Thierens, 2013; Thierens and Bosman, 2011). The GA is arguably the most popular EA in DNEP literature, but it is rarely used out of the box in practice. Practitioners often customize VOs (i.e., crossover and mutation) with expert and problem-specific knowledge (PSK) so that important problem structures are respected during variation, e.g., cables in the same feeder group in the network should be treated together when constructing new networks. Taking the perspective of black-box optimization, where such PSK is assumed to be hardly available, linkage learning (LL) can be performed to identify, during optimization, which variables are inter-dependent and should thus be jointly considered when generating new solutions. EDAs, such as BOA (Pelikan et al., 2000) or ECGA (Harik et al., 2006), are well-known examples of EAs that build probabilistic models that exhibit a degree of variable dependency that is aligned with variable linkage to effectively generate high-quality solutions. However, large population sizes are often required so that probabilistic models can be properly constructed. Building a high-order probabilistic model also introduces significant additional computation time requirements. Being a recently-developed LL EA, GOMEA focuses specifically on linkage, without estimating associated probability distributions, allowing higher-order

models to be built much more efficiently. GOMEA also has an efficient variation operator that exploits the learned linkage model to create new solutions that are better or at least equal to existing solutions. GOMEA has been shown to have superior performance and scalability on laboratory benchmarks and recently in power system optimization as well (Grond et al., 2014b; Luong et al., 2013). Linkage learning does not exclude the possibility of combining linkage knowledge with PSK if available. In this paper, we show how to combine the strength of LL with PSK exploitation.

Population size parameter settings have big impacts on how effective and how efficient problem instances are solved. Population sizes of EAs are often chosen arbitrarily or are customized to the specific problem instance at hand in DNEP literature (Borges and Martins, 2012; Diaz-Dorado et al., 2002; Ramirez-Rosado and Bernal-Agustin, 1998). Practitioners often need to manually try different population sizes to figure out a suitable population size for each problem instance, which is both time-consuming and difficult for comparing the performance of different EAs fairly. This approach is also difficult to generalize to other applications. To get rid of this troublesome parameter, Harik and Lobo (1999) proposed the parameter-less GA with a population sizing-free scheme. Pelikan et al. (2007) then proposed a simplified implementation of the original scheme. Recently, Pereira and Lobo (2015) adopted the implementation of (Pelikan et al., 2007) to develop a framework for parameter-less EAs. The scheme, however, has been mainly applied to unconstrained problems. Here, we adapt the Harik-Lobo scheme in the context of DNEP, a highly constrained optimization problem. We then also employ the adapted population sizing-free scheme as a framework for comparing the performance of GA, EDA, and GOMEA.

This article is an extension to our previous publication at GECCO 2015 (Luong et al., 2015). Beside GA and GOMEA, the present manuscript considers the application of EDA for solving DNEP. We also present additional results to support our design choice in adapting the population sizing-free scheme. Moreover, we include a more detailed description and pseudo-code of the employed EAs and their variation operators.

The remainder of this article is organized as follows. Section 2 formulates the DNEP problem. Section 3 introduces linkage learning and three linkage models: univariate, marginal product, and linkage tree. Section 4 outlines the three optimization algorithms GA, EDA and GOMEA. In Section 5, we present the adapted Harik-Lobo population sizing-free scheme for the DNEP problem. Section 6 describes different variation operators and constraint-handling techniques for DNEP. Section 7 presents the experimental results on the performance of GA, EDA, and GOMEA when being combined with different variation operators. Section 8 concludes the article.

2 DNEP Formulation

In this article, we focus on optimizing expansion plans for medium voltage distribution (MV-D) networks. A typical MV-D network consists of cables branching out from HV/MV substations connecting MV nodes (MV/LV substations and MV customer substations) (Grond et al., 2014b). These cables form different feeder (cables) groups in ring-shaped or meshed structures. However, some cables are opened on one side, called normally open points (NOPs). Electricity cannot flow through those cables, so that every node is supplied its power demand through a single feed path. The whole network thus operates radially in normal situations. Figure 1 shows an example of a distribution network. We focus on expansion options for network cables as our main asset category. How the cables are connected and what types of cables are used, determine the capacity of the network. Increasing power demands in the future can create

bottlenecks in some parts of the network, where the power flows have magnitudes that are greater than the nominal capacities of the cables. The goal of capacity planning is to find the best expansion options to solve these bottlenecks. Possible expansion options are: replacing existing cables with new cables of higher capacities or adding new cable connections and thus creating new feed paths to the network (Grond et al., 2014b). Adding new cables connections requires placements of new NOPs to satisfy the radiality constraint (Grond et al., 2014b). In the following, we formulate the problem in more detail.
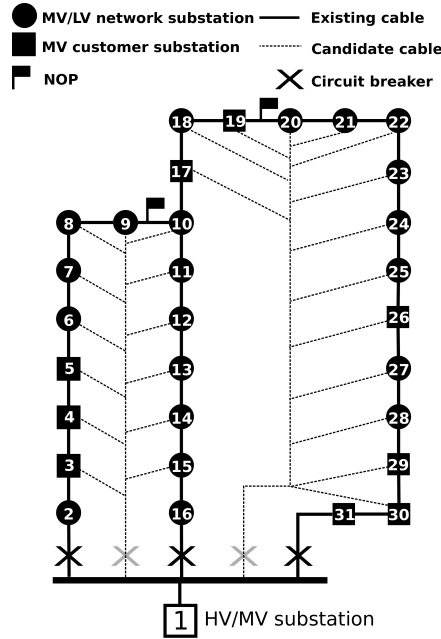
Figure 1: A distribution network example (Grond et al., 2014b)

2.1 Decision Variables

We specify all the existing and potential cable connections (branches) that we want to consider for capacity planning. Let l denote the total number of branches. A distribution network can be represented as a vector of l integer elements:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_l), \quad |x_k| \in \Omega(k) \cup \{0\}, \quad k \in \{1, 2, \ldots, l\} \tag{1}$$

where $\Omega(k)$ is the set of cable types that can be installed at the kth branch ($\Omega(k) \subseteq \mathbb{N}$). The value of x_k indicates the status and the type of cable installed:

- $x_k = ID > 0$: A cable of type $ID \in \Omega(k)$ is installed.
- $x_k = 0$: No cable is installed at the kth branch.
- $x_k = -ID < 0$: A cable of type $ID \in \Omega(k)$ is installed but is out of normal operation. This is an NOP.

2.2 Constraints

Let n denote the total number of nodes (substations) in an MV network. The following constraints must be satisfied for any candidate network to be considered feasible:

1. **Connectivity**: For each consuming node (i.e., an MV/LV network substation or MV customer substation), there exists a path of concatenating cables connecting that consuming node to an HV/MV supplying substation.

2. **Power flow constraint**: In normal operation, the voltage at each node stays within allowable limits (i.e., $0.9 \cdot V_i^{nom} \leq V_i \leq 1.1 \cdot V_i^{nom}$, $i \in \{1, 2, \ldots, n\}$) and the magnitude of the power flow through each cable stays within the nominal capacity of that cable (i.e., $|S_i| \leq S_i^{nom}$, $i \in \{1, 2, \ldots, l\}$).

3. **Radiality constraint**: In normal operation, the power demand of each node is supplied by a single feed path.

4. **Reconfigurability constraint**: When an active cable ($x_k > 0$) fails, the part of the feeder group (from an HV/MV substation with circuit breaker to an NOP) containing the failed cable is disconnected from the network. All customers connected to that feeder group are then out of service. The DNO has to bring the network back to operation by closing NOPs to temporarily re-route the power flow through other paths while the failed cable is being repaired. During this emergency situation, the radial operation constraint can be compromised and the network is allowed to endure a mild overload of 130% nominal capacity (as in (Grond et al., 2014b)).

5. **Substation capacity constraint**: Each HV/MV substation has limited physical space to install new outgoing cables. We assume that at most 3 new outgoing cables are allowed for each HV/MV substation (as in (Grond et al., 2014b)).

Alternating-current (AC) power flow calculations (PFCs) (Grainger and Stevenson, 2003) are required to check constraints 2 and 4 for each solution plan. PFCs, which involve solving AC power flow models, are computationally expensive and dominate the computing time of the optimization process. To compute a full reconfigurability constraint evaluation, all cables in a network solution need to be checked for failures, which requires many time-consuming PFCs. There exists a reasonably accurate, but much more efficient, method for reconfigurability check, named Line Outage Distribution Factor AC (LODF-AC) (Grond et al., 2014a), which needs to perform only one true AC-PFC for the base case while each single cable check can be approximately computed. We employ the true AC-PFC to evaluate the power flow constraint (i.e., constraint 2) and the LODF-AC method to verify the reconfigurability constraint (i.e., constraint 4).

PFCs can only be performed for connected networks. The connectivity constraint is thus a crucial constraint that needs to be separately handled so that constraints 2 and 4 can be properly evaluated (see also Section 6). If the network encoded in a solution plan is unconnected, we do not evaluate other constraints but we quantify its disconnectivity by comparing it with the topology of the existing network x^{now}. Specifically, we loop through all the decision variables and count the number of positions where the existing network has a positive value (i.e., an active branch) but the solution has a negative value (i.e., an NOP) or where the existing network has a non-positive value (i.e., no cable connection or an NOP) and the solution has a positive value. This number is considered to be the disconnectivity value. Note that the case when a position in the existing network has a positive value and that position in the candidate solution has value 0 does not exist because such a case implies that an existing cable connection would be removed, which is generally undesirable according to network operators (see more details in Section 6.1). Connected networks do not need this disconnectivity quantification and are assigned the disconnectivity value 0. Figure 2 shows the pseudo-code for the disconnectivity quantification procedure.

```
DisconnectivityQuantification(x)
1  if ConnectivityCheck(x) then
2     Return( 0 )
3  count ← 0
4  for k ∈ {1, 2, . . . , l} do
5     if [x_k^now > 0 and x_k < 0] or [x_k^now ≤ 0 and x_k > 0] then
6        count ← count + 1
7  Return(count)
```

Figure 2: Disconnectivity Quantification

To deal with solutions that violate constraints, we will, as a basis, consider the use of constraint domination (Deb, 2000). Because we have a cascade of importance in the constraints, we modify constraint domination slightly to work as follows. When comparing two candidate networks, if both networks are unconnected, the one with a smaller disconnectivity value is the better one. If only one network is unconnected, then the connected network is the better solution. If both networks are connected, they can then be compared by using the other evaluated constraint values and their objective values. The network with smaller total constraint violation is the better one. If both candidate networks are feasible (i.e., no constraint violation), the one having smaller cost is preferred. Figure 3 shows the pseudo-code for our implementation of constraint domination in DNEP. The total constraint violation of a candidate network is taken as the aggregate of the amounts of violation of constraints 2, 3, 4, and 5. The radiality constraint and the reconfigurability constraint are evaluated as Boolean values, i.e., 1-value indicates the constraint is satisfied while 0-value indicates the constraint is violated. More refined methods to quantify DNEP constraint violations are worth further investigation but are outside the scope of this work.

2.3 Objective Function

Solving DNEP involves finding the feasible expansion plan that minimizes the total cost, which is typically comprised of the capital expenditure $CAPEX$ and the operational expenditure $OPEX$. If $x^0 = (x_1^0, x_2^0, x_3^0, \ldots, x_l^0)$ is the configuration of the currently existing network, then each x_k^0 for which k corresponds to a potential cable connection is assigned the value 0. For the static DNEP problem, in which the time of investment is disregarded, any $x \neq x^0$ can be seen as a candidate expansion plan. The element-wise differences between x^0 and x indicate which reinforcement activities need to be carried out to transform the current network x^0 into the network x. $CAPEX$ is then the investment cost that a DNO needs to spend to acquire and construct new assets. Note that the cost of closing an NOP (i.e., $x_k^0 = -ID < 0$ and $x_k = ID > 0$) or placing an NOP (i.e., $x_k^0 = ID > 0$ and $x_k = -ID < 0$) is negligible and can be disregarded. $OPEX$ is the operational cost of the new network configuration x.

We employ the annuity method (Verzijlbergh et al., 2012) to calculate the capital expenditures $CAPEX$ for new assets. The investment cost on a new asset is converted into a series of uniform annual payments, called annuities. We assume the length of this series to be equal to the (uniform) economic lifetime of the new asset t_{life}. The annuity AN_a of the asset a with a discount rate $i = 4.5\%$ (as in (Grond et al., 2014b)) can be computed as:

$$AN_a = Price_a \cdot \frac{i}{1 - (1 + i)^{-t_{life}}} \tag{2}$$

```
ISBETTER(x, x')
 1  dq ← DISCONNECTIVITYQUANTIFICATION(x)
 2  dq' ← DISCONNECTIVITYQUANTIFICATION(x')
 3  if dq < dq' then
 4     RETURN (TRUE)
 5  else if (dq > dq' ≥ 0) or (dq = dq' > 0) then
 6     RETURN (FALSE)
 7  else
 8     con ← CONSTRAINTVIOLATIONCALCULATION(x)
 9     con' ← CONSTRAINTVIOLATIONCALCULATION(x')
10     if con < con' then
11        RETURN (TRUE)
12     else if (con > con' ≥ 0) or (con = con' > 0) then
13        RETURN (FALSE)
14     else
15        cost ← TOTALCOSTCALCULATION(x)
16        cost' ← TOTALCOSTCALCULATION(x')
17        if cost < cost' then
18           RETURN (TRUE)
19        else
20           RETURN (FALSE)
```

```
CONSTRAINTVIOLATIONCALCULATION(x)
// n: number of nodes; l: number of branches
// V = (V₁, V₂, ..., Vₙ): vector of voltage at each node
// S = (S₁, S₂, ..., Sₗ): vector of power flow through each branch
 1  con ← 0
 2  V, S ← POWERFLOWCALCULATION(x)
 3  for i ∈ {1, 2, ..., n} do
 4     con ← con + max(Vᵢ^min − Vᵢ, Vᵢ − Vᵢ^max, 0)
 5  for i ∈ {1, 2, ..., l} do
 6     con ← con + max(Sᵢ − Sᵢ^nom, 0)
 7  if con > 0 then
 8     con ← con + 1
 9  else
10     con ← BOOLEANTOINT(¬RECONFIGURABILITYCHECK(x))
11     con ← con + BOOLEANTOINT(¬RADIALITYCHECK(x))
12     for i ∈ {1, 2, ..., n} do
13        con ← con + max(NUMBEROFOUTGOINGCABLES(i) − 3, 0)
14  RETURN(con)
```

Figure 3: Constraint Domination for DNEP

$CAPEX$ for the asset a in a year t can be calculated as:

$$CAPEX_a(t) = \begin{cases} AN_a & \text{if } t_{inst_a} \le t < t_{inst_a} + t_{life} \\ 0 & \text{else} \end{cases} \qquad (3)$$

with t_{inst_a} is the time of installing the asset a. Let A denote the set of all new assets a's that will be installed in the planning period $[t_0, t_{horizon}]$. The total $CAPEX$ on the whole network in a year t is defined as:

$$CAPEX(t) = \sum_{a \in A} CAPEX_a(t) \qquad (4)$$

The operational expenditure $OPEX$ can be calculated by capitalizing the energy loss. The energy loss of the network in year t can be taken as in (Grond et al., 2014b;

Verzijlbergh et al., 2012):

$$E_{loss}(t) = P_{peak\,loss}(t) \cdot T_{loss}(t) \qquad (5)$$

where $P_{peak\,loss}(t)$ is the peak loss which can be obtained from the PFC regarding the peak loads in year t. $T_{loss}(t)$ is the service time of peak loss for year t, defined by the area of the yearly loss profile (Grond et al., 2014b; Verzijlbergh et al., 2012). In this article, we assume that $T_{loss}(t) = 2000$ hours, which is a typically reasonable value for MV distribution networks in the Netherlands (Grond, 2011). Given the forecast electricity price in year t, we can capitalize the energy loss and regard it as the $OPEX$ in year t.

$$OPEX(t) = E_{loss}(t) * Price_{electricity}(t) \qquad (6)$$

In this study, we take the price of electricity for energy loss capitalization as 0.068 EUR/kWh during the planning period.

2.3.1 Total Cost Formulation

We want to minimize the net present value (NPV) (i.e., at time t_0) of the total cost of both investment cost $CAPEX$ and operation cost $OPEX$ over a planning period of $t_{horizon}$ years with a discount rate i.

$$COST_{NPV} = \sum_{t=t_0}^{t_{horizon}} \frac{CAPEX(t) + OPEX(t)}{(1+i)^{t-t_0}} \qquad (7)$$

2.3.2 Static Planning

The goal in static DNEP is to find the most economical solution plan that satisfies the peak load profile at the final year of the planning period. Static DNEP gives DNOs a general picture about what kinds of network reinforcements can be expected. However, the actual total cost objective function includes the important operation cost $OPEX$, which is the capitalized energy loss, that depends on the load profile and the specific network configuration at each year. We employ the following method, which has been proposed in (Grond et al., 2014b), to estimate the $OPEX$ for a solution plan. Based on the (predicted) annual peak load growth rate R, we compute the peak load profile for each year from the beginning year t_0 until the final year $t_{horizon}$. We determine the year $t_{overload}$ when the first bottleneck happens in the network. We then assume that all expansion options in the solution plan are installed at the same time in year $t_{overload}$, i.e., $t_{inst} = t_{overload}$ for all new assets. Therefore, to evaluate the total cost, from t_0 until $t_{overload}$, we use the current network topology, and from $t_{overload}$ until $t_{horizon}$ we switch to the new network topology.

2.3.3 Dynamic Planning

The question *when* each expansion option should be carried out is also an important issue that DNOs have to address. Dynamic DNEP formulations often contain time-dependent decision variables and complicated problem models that are much more difficult to solve than those of static DNEP. A general dynamic formulation can, in principle, encode any expansion plans, even the solutions that are considered impractical by DNOs such as installing a thin cable first and then upgrading with a different cable of higher capacity, or removing a previously-upgraded cable during the same planning period. Network cables (i.e., the main asset category in DNEP) are generally expensive and have very long lifetimes (e.g., normally 30 years) while planning periods of more than 30 years are regarded as impractical because it is difficult to properly

predict power demand scenarios for a very distant future. Therefore, during a practical planning period, a network branch might require reinforcement only once (e.g., cable replacement or installation of a new cable connection). These facts motivate an approach that precludes the aforementioned impractical expansion options while at the same time can be solved more efficiently than the general dynamic formulation. Here, we propose a method, called the *decomposition heuristic*, to find a suitable schedule for network reinforcement activities in a feasible expansion plan as follows. Similarly to the static planning, we determine the year $t_{overload}$ when the first bottleneck occurs and assume that the whole solution plan is carried out in that year. Such simultaneous reinforcements might include some asset installations which are required at some later time t but are not necessary at the year $t_{overload}$ (i.e., $t > t_{overload}$). If these early investments can be postponed as further as possible, the net present value (NPV) for the total cost of the expansion plan might be reduced because investments made at a later time are discounted more (see Equation 7). To this end, we loop through the list of all new assets in the solution plan in a random order and check if we can delay each expansion option by one year. If the postponement of an asset investment reduces the NPV of the total cost and all constraints are still satisfied, then we accept that postponement. Otherwise, that expansion option cannot be postponed further. We continue this postponement checking until no expansion option can be postponed any more. Each postponement checking requires calculations of the changes in $CAPEX$ and $OPEX$ (Equations 4 and 6) and evaluations of network constraints in the related years. Finally, we obtain a solution plan with an investment time for each expansion option and the NPV of the total cost (i.e., the objective value) is evaluated accordingly. By employing this *decomposition heuristic*, we can bring the decision making about investment time into the static DNEP framework while still satisfying the requirements of DNOs in real-world practice.

3   Problem Structures and Linkage Models

A key part that characterizes an evolutionary algorithm (EA) is its variation operators (VOs), i.e., how new solutions are generated. VOs can be model-based, meaning that the way variation is performed is governed by a (learnable) model. Models of particular interest are linkage models, which are used to match the dependency structure of the optimization problem instance at hand. A linkage model often contains information about groups of inter-dependent decision variables, also known as linkage sets. Variables in the same linkage set are dependent on each other in the sense that when being considered together, they have a significant contribution to the quality of a solution (Thierens and Bosman, 2011). These variables should thus be jointly considered when performing variation. Variation operators can, for instance, make use of the information in linkage models to juxtapose partial solutions from existing solutions to generate new solutions. A linkage model that matches correctly with the problem dependency structure is crucial for variation operators to efficiently mix and preserve good partial solutions (i.e., building blocks) in the population in order to efficiently create high-quality solutions. Problem-specific knowledge (PSK), if available, can be used to directly construct the linkage model of the problem. If such valuable PSK is not available, or difficult to transcribe into linkage information, linkage information can be inferred from the working population of EAs by linkage learning (LL) procedures (Thierens and Bosman, 2011), in which problem variables having some degree of dependency are identified. In the following we give a general definition of a linkage model, called the Family of Subset (FOS) model, and subsequently describe three variants that can be used in practice.

3.1 Family of Subset Linkage Model

Let $L = \{1, 2, \ldots, l\}$ be the set of all l decision variables of the problem instance at hand. We use the Family of Subsets (FOS) concept proposed by (Thierens and Bosman, 2011) to encode different linkage models. A FOS, denoted $\mathcal{F}$, consists of subsets of the set L, i.e., $\mathcal{F} = \{\boldsymbol{F}^1, \boldsymbol{F}^2, \ldots, \boldsymbol{F}^{|\mathcal{F}|}\}$ where $\boldsymbol{F}^i \subseteq L$, $i \in \{1, 2, \ldots |\mathcal{F}|\}$. Thus, $\mathcal{F} \subseteq \mathcal{P}(L)$, i.e., $\mathcal{F}$ is a subset of the power set of L. Each subset $\boldsymbol{F}^i$ is a linkage set containing decision variables that exhibit some degree of dependency and should thus be jointly treated when performing variation. A FOS $\mathcal{F}$ is said to be complete if every problem variable is contained in at least one linkage set, i.e., $\forall i \in L$, $\exists j \in \{1, 2, \ldots, |\mathcal{F}|\} : i \in \boldsymbol{F}^j$. The completeness property ensures that all problem variables are considered when performing variation following linkage sets in $\mathcal{F}$. We here consider 3 complete FOS models: univariate, marginal product, and linkage tree.

3.2 Univariate Model

The univariate factorization (UF) model contains only singleton sets. It therefore expresses the assumption that all problem variables are independent from each other, i.e., $\boldsymbol{F}^i = \{i\}$, $i \in L = \{1, 2, \ldots, l\}$. Because there is only one possible configuration, the univariate model does not require any linkage learning.

3.3 Marginal Product Model

The marginal product (MP) model is a partitioning of the set L of all problem variables, i.e., $\bigcup_{\boldsymbol{F}^i \in \mathcal{F}} \boldsymbol{F}^i = L$ and $\forall \boldsymbol{F}^i, \boldsymbol{F}^j \in \mathcal{F}, \boldsymbol{F}^i \neq \boldsymbol{F}^j : \boldsymbol{F}^i \cap \boldsymbol{F}^j = \emptyset$. Variables in the same linkage set have some degree of dependency while variables in different linkage sets are considered to be independent (Thierens and Bosman, 2011). The well-known Extended Compact Genetic Algorithm (ECGA) (Harik et al., 2006) employs the MP as its linkage model.

The MP model is often learned from a population $\mathcal{P}$ of n individuals using a greedy algorithm that optimizes a model scoring metric as follows. First, the FOS $\mathcal{F}$ assumes the univariate structure and is then scored by the chosen metric. Next, all possible merges of two linkage sets $\boldsymbol{F}^i$ and $\boldsymbol{F}^j$ are tried out and the merge that improves the scoring metric the most is chosen. The merged linkage set is added into $\mathcal{F}$, replacing the two constituent linkage sets, i.e., $\mathcal{F} \leftarrow (\mathcal{F} \setminus \{\boldsymbol{F}^i, \boldsymbol{F}^j\}) \cup \{\boldsymbol{F}^i \cup \boldsymbol{F}^j\}$. This procedure continues until no merging event can improve the scoring metric anymore. The scoring metric employed by ECGA is named the Combined Complexity Criterion (CCC), which is the sum of the Compressed Population Complexity and the Model Complexity (Harik et al., 2006). The Compressed Population Complexity (CPC) is the cost of representing the whole population of n individuals with FOS model $\mathcal{F}$ and is calculated as

$$CPC = n \sum_{i=1}^{|\mathcal{F}|} H(X_{\boldsymbol{F}^i}) \tag{8}$$

where $X_{\boldsymbol{F}^i}$ is a set of random variables, in which each random variable X_k corresponds with a problem variable x_k in the linkage set $\boldsymbol{F}^i$. $H(X_{\boldsymbol{F}^i})$ is the entropy of the marginal distribution of $X_{\boldsymbol{F}^i}$. We have

$$H(X_{\boldsymbol{F}^i}) = - \sum_{\boldsymbol{x} \in \Omega(X_{\boldsymbol{F}^i})} P(X_{\boldsymbol{F}^i} = \boldsymbol{x}) \log_2 P(X_{\boldsymbol{F}^i} = \boldsymbol{x}) \tag{9}$$

where $\Omega(X_{\boldsymbol{F}^i})$ is the the sample space for random variables $X_{\boldsymbol{F}^i}$, i.e., all possible string values of the problem variables in $\boldsymbol{F}^i$ when being jointly considered. The probability

$P(X_{\boldsymbol{F}^i} = \boldsymbol{x})$ can be computed by counting the frequency of $\boldsymbol{x}$ in the population $\mathcal{P}$. The Model Complexity (MC) is the cost of representing or storing FOS model $\mathcal{F}$ and is calculated as

$$MC = \log_2(n+1) \sum_{i=1}^{|\mathcal{F}|} (|\Omega(X_{\boldsymbol{F}^i})| - 1) \tag{10}$$

The greedy model building algorithm, which has a worst-case runtime $\mathcal{O}(nl^3)$, needs to minimize the CCC metric. Instead of computing this scoring metric for the whole FOS $\mathcal{F}$ at every merging trial, the CCC metric improvement (i.e., CCC metric decrease) can be computed more efficiently considering only problem variables in the two involved linkage sets $\boldsymbol{F}^i$ and $\boldsymbol{F}^j$ (Thierens and Bosman, 2011) as

$$CCC(\boldsymbol{F}^i, \boldsymbol{F}^j) = n[H(X_{\boldsymbol{F}^i}) + H(X_{\boldsymbol{F}^j}) - H(X_{\boldsymbol{F}^i \cup \boldsymbol{F}^j})] + $$
$$\log_2(n+1)[(|\Omega(X_{\boldsymbol{F}^i})| - 1) + (|\Omega(X_{\boldsymbol{F}^j})| - 1) - (|\Omega(X_{\boldsymbol{F}^i \cup \boldsymbol{F}^j})| - 1)] \tag{11}$$

3.4 Linkage Tree Model

Because each configuration of the MP model is a partitioning of the set L, every problem variable can exist in only one linkage set. Therefore, any two variables are either dependent (if they are in the same linkage set) or independent (if they are in different linkage sets). In the linkage tree (LT) model, a problem variable can exist in multiple linkage sets. The LT model is thus more expressive than the MP model in the sense that any two variables can be both dependent and independent. While the MP model has a flat structure, the LT model arranges its linkage sets in a hierarchical tree structure. The lowest level (i.e., leaf nodes) contains univariate linkage sets of each variable separately, i.e., $\boldsymbol{F}^i = \{i\}, i \in L = \{1, 2, \ldots, l\}$. Higher levels contain multivariate linkage sets, in which each linkage set $\boldsymbol{F}^i$ is formed by merging two mutually exclusive linkage sets $\boldsymbol{F}^j$ and $\boldsymbol{F}^k$ at lower levels, i.e., $\boldsymbol{F}^j \cap \boldsymbol{F}^k = \emptyset, |\boldsymbol{F}^j| < |\boldsymbol{F}^i|, |\boldsymbol{F}^k| < |\boldsymbol{F}^i|$ and $\boldsymbol{F}^j \cup \boldsymbol{F}^k = \boldsymbol{F}^i$. The highest level (i.e., the root node) contains the set L itself. Thus, an LT can encode different levels of dependency, from the totally independent state in the leaf nodes to the all-dependent state in the root node (Bosman and Thierens, 2013). The root node can be removed from the LT because it assumes that variables should be jointly copied when performing variation, which means no new solution is created. An LT over a set of problem variables $\{1, 2, 3, 4, 5\}$ can be, for example, $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1, 3\}, \{2, 5\}, \{1, 3, 4\}\}$. An LT model configuration for a set L of l problem variables is thus a FOS $\mathcal{F}$ of exactly $2l - 2$ linkage sets.

The LT model can be learned from the population of n individuals by a hierarchical clustering procedure named the Unweighted Pair Group Method with Arithmetic Mean (UPGMA). The FOS $\mathcal{F}$ is built in a bottom-up manner. First, $\mathcal{F}$ is initialized with l univariate linkage sets $\boldsymbol{F}^i = \{i\}, i \in L = \{1, 2, \ldots, l\}$. Then, UPGMA iteratively merges the two linkage sets $\boldsymbol{F}^i$ and $\boldsymbol{F}^j$ that are the most similar. The newly created linkage set $\boldsymbol{F}^i \cup \boldsymbol{F}^j$ is added to the LT $\mathcal{F}$. The two constituents linkage sets $\boldsymbol{F}^i$ and $\boldsymbol{F}^j$ are still kept in the LT $\mathcal{F}$ but they are not considered for merging anymore. The merging operations continue until the root node (i.e., the set L itself) is created. To calculate the similarity between two linkage sets $\boldsymbol{F}^i$ and $\boldsymbol{F}^j$, we take the average of the mutual information (MI) over all pairs of problem variables (X, Y) where $X \in \boldsymbol{F}^i$ and $Y \in \boldsymbol{F}^j$ (Bosman and Thierens, 2013) as follows

$$MI^{UPGMA}(\boldsymbol{F}^i, \boldsymbol{F}^j) = \frac{1}{|\boldsymbol{F}^i||\boldsymbol{F}^j|} \sum_{X \in \boldsymbol{F}^i} \sum_{Y \in \boldsymbol{F}^j} MI(X, Y) \tag{12}$$

where $MI(X, Y) = H(X) + H(Y) - H(X, Y)$. UPGMA can be optimally implemented by using the reciprocal nearest-neighbor chain technique such that an LT can be built in $\mathcal{O}(nl^2)$ time (Gronau and Moran, 2007).

4 Evolutionary Algorithms

DNEP is difficult for simple k-opt local search (LS) with a small neighborhood k to solve. For example, adding a new cable connection requires the addition of a new NOP to make the network radial, which in turn requires relocations of other existing NOPs to obtain the topology with lowest energy loss. Such multivariate decision making cannot be found by a typical LS like 1/2/3-opt hillclimbing while higher-order LS would require a drastic increase in computational effort. Global search meta-heuristics, like EAs, are therefore potentially more suitable for DNEP.

4.1 Genetic Algorithm (GA)

```
GA  //population size n
1  for i ∈ {1, 2, . . . , n} do
2      P_i ← CREATERANDOMSOLUTION()
3      EVALUATEFITNESS(P_i)
4  while ¬TERMINATIONCRITERIASATISFIED() do
5      F ← LEARNMODELFROMPOPULATION(P)
6      π ← RANDOMPERMUTATION({1, 2, . . . , n})
7      k ← 1
8      for i ∈ {1, 2, . . . , n} do
9          O_i ← RECOMBINE(P_{π_k}, P_{π_{k+1}})
10         EVALUATEFITNESS(O_i)
11         k ← k + 2
12         if k ≥ n then
13             k ← 1
14             π ← RANDOMPERMUTATION({1, 2, . . . , n})
15     P ← TOURNAMENTSELECTION(P + O, n, 4)
```

```
RECOMBINE(p^0, p^1)
1  o ← p^0
2  for i ∈ {1, 2, . . . , |F|} do
3      if RANDOM01() < 0.5 then
4          o ← COPYVALUES(o, p^1, F^i)
5  RETURN(o)
```

```
COPYVALUES(x, d, F^i)
1  o ← x
2  for k ∈ F^i do
3      o_k ← d_k
4  RETURN(o)
```

Figure 4: Genetic Algorithm

The GA is started with a population $\mathcal{P}$ of n randomly generated candidate solutions. Next, for every generation, a FOS $\mathcal{F}$ is learned from the current population $\mathcal{P}$. An offspring population $\mathcal{O}$ of n new solutions is created from $\mathcal{P}$ by performing recombination (i.e., crossover), guided by the linkage information in $\mathcal{F}$, n times on 2 parent

solutions that are randomly picked each time, giving 1 offspring solution. Then, both the parent population $\mathcal{P}$ and the offspring population $\mathcal{O}$ are combined into a selection pool $\mathcal{P} + \mathcal{O}$ of $2n$ solutions in total. Tournament selection with tournament size 4 is performed on this pool to select n survivor solutions, which form the new parent population $\mathcal{P}$ for the next generation, ensuring convergence by logistic growth of the best solution (Thierens and Bosman, 2011). If the univariate factorization (UF) model is chosen for linkage learning, we automatically have the simple GA with uniform crossover, in which all dependencies among problem variables are disregarded. If the MP model is employed, we have a GA variant with a recombination operator that can exchange blocks of values at the positions indicated by the linkage sets in $\mathcal{F}$. We do not combine the LT model with GA because the simple crossover operator is not compatible with the hierarchical structure of an LT FOS. Figure 4 shows pseudo-code for our GA implementation.

Note that, in our previous work (Luong et al., 2015), we employed a slightly different version of GA, in which two offspring solutions are created from two parent solutions in every recombination event. In this article, recombining two parent solutions results in one offspring. We use this implementation so that GA can be presented in sync with both EDA and GOMEA in the sense that one offspring solution is constructed each time. There is no significant difference in performance if two offspring were to be generated in recombination.

4.2 Estimation-of-Distribution Algorithm (EDA)

The EDA starts with a population $\mathcal{P}$ of n randomly generated candidate solutions. Every generation, a FOS $\mathcal{F}$ is learned from the current population $\mathcal{P}$. The EDA also derives a probability distribution from the population $\mathcal{P}$ following the obtained structure $\mathcal{F}$. In the case of the univariate model or the MP model, all linkage sets of $\mathcal{F}$ are mutually exclusive, so the probability distribution can be formulated as $P_{\mathcal{F}}(\boldsymbol{X}) = \prod_{i=1}^{|\mathcal{F}|} P(X_{\boldsymbol{F}^i})$ where a random variable X_i corresponds with each problem variable x_i. Each linkage set $\boldsymbol{F}^i$ corresponds to a marginal $X_{\boldsymbol{F}^i}$ whose distribution $P(X_{\boldsymbol{F}^i})$ can be estimated by counting frequencies of all possible value strings of the variables in $\boldsymbol{F}^i$ in the population $\mathcal{P}$. Note that for the LT model such a joint distribution $P_{\mathcal{F}}(\boldsymbol{X})$ cannot be defined directly in terms of $P(X_{\boldsymbol{F}^i})$. The EDA does not use the recombination operator (i.e., crossover) like the GA to create offspring from existing solutions. Instead, the offspring population $\mathcal{O}$ of n new solutions is generated by sampling the estimated probability distribution. The selection pool $\mathcal{P} + \mathcal{O}$ of $2n$ solutions is again created by combining $\mathcal{P}$ and $\mathcal{O}$. A tournament selection with tournament size 4 is performed on $\mathcal{P} + \mathcal{O}$ to select n survivors, forming the new population $\mathcal{P}$ for the next generation. If the univariate model is employed, we have an EDA that corresponds with the UMDA (Mühlenbein, 1997) while an EDA with the MP model can be considered to be similar to the ECGA (Harik et al., 2006). Figure 5 shows pseudo-code for the EDA.

4.3 Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA)

The GOMEA starts with a population $\mathcal{P}$ of n randomly generated candidate solutions. Every generation, the linkage model building procedure is performed on $\mathcal{P}$ to construct a FOS $\mathcal{F}$. Using the obtained FOS $\mathcal{F}$, GOMEA transforms each existing parent solution $\boldsymbol{p} \in \mathcal{P}$ into a new offspring solution $\boldsymbol{o} \in \mathcal{O}$ whose fitness value is equal to or better than the fitness value of $\boldsymbol{p}$. The offspring population $\mathcal{O}$ completely replaces $\mathcal{P}$ and becomes the new parent population $\mathcal{P}$ for the next generation. Figure 6 shows pseudo-code for GOMEA.

```
EDA   //population size n
1  for i ∈ {1, 2, . . . , n} do
2      Pi ← CREATERANDOMSOLUTION()
3      EVALUATEFITNESS(Pi)
4  while ¬TERMINATIONCRITERIASATISFIED() do
5      F, PF(X) ← LEARNDISTRIBUTIONFROMPOPULATION(P)
6      for i ∈ {1, 2, . . . , n} do
7          Oi ← SAMPLEDISTRIBUTION()
8          EVALUATEFITNESS(Oi)
9      P ← TOURNAMENTSELECTION(P + O, n, 4)
```

```
SAMPLEDISTRIBUTION()
1  for i ∈ {1, 2, . . . , |F|} do
2      oFⁱ ← SAMPLESUBSETDISTRIBUTION(Fⁱ, P(X_{Fⁱ}))
3  RETURN(o)
```

Figure 5: Estimation-of-Distribution Algorithm

Instead of fully creating new solutions and then evaluating them like in GA, the variation operator of GOMEA, called Gene-pool Optimal Mixing (GOM) (Thierens and Bosman, 2011), uses the learned FOS to evolve each existing parent p into a new off-spring o in an iterative manner. First, o and a backup b are cloned directly from p. Then, each linkage set in the FOS $\mathcal{F}$ is traversed iteratively in a random order. For each linkage set, a donor d is randomly selected from the current population $\mathcal{P}$. If the values of the donor for the variables indicated by the linkage set differ from those in o in at least one position, these values are copied from d into o. This partially-altered solution o is evaluated and compared against its backup b. If o is equally good or better than b (i.e., $fitness[o] \geq fitness[b]$), the changes are accepted (i.e., the values are copied from d) and updated into b as well. Otherwise, the changes are undone and o reverts to its backup state b. Note that the acceptance of solutions having equal fitness can be beneficial to move across a fitness plateau (Bosman and Thierens, 2013). It can be seen that each linkage set corresponds with a mixing event, in which the current solution is recombined with a random donor solution and the variables in the same linkage set are treated together, preserving the building-block structure (insofar such building blocks exist for the problem at hand and are correctly represented by the FOS). After traversing the whole FOS, an offspring o is then fully constructed, replacing the original parent p in the next generation. Note that GOMEA does not need to perform selection over the combined pool $\mathcal{P} + \mathcal{O}$ like GA or EDA because the GOM operator ensures that the quality of the offspring solution is better or at least equal to the parent solution.

It can happen that GOM cannot improve the current parent solution p or that, because of a significant plateau, GOM keeps transforming back and forth solutions of different genotypes but with the same fitness value. To overcome this, if GOM cannot yield a new offspring or when the number of subsequent generations that the best solution at the end of generation t $x^{best}(t)$ does not change, i.e., the no-improvement stretch (NIS), exceeds a certain threshold, we invoke the Forced Improvement (FI) procedure (Bosman and Thierens, 2013). In essence, FI is similar to GOM but we always use $x^{elitist}$ as the only donor solution. $x^{elitist}$ is the best-found-so-far solution, which is constantly checked for possible updates every time a fitness evaluation is performed. FI only accepts the mixing event that results in a strict improvement (i.e., $fitness[o] > fitness[b]$) and FI stops as soon as such mixing event occurs. Previous research on GOMEA sug-

```
GOMEA  //population size n
1  for i ∈ {1, 2, ..., n} do
2      Pᵢ ← CREATERANDOMSOLUTION()
3      EVALUATEFITNESS(Pᵢ)
4  xᵇᵉˢᵗ(0) ← arg min_{x∈P}{fitness[x]}; t ← 0; tᴺᴵˢ ← 0
5  while ¬TERMINATIONCRITERIASATISFIED() do
6      F ← LEARNMODELFROMPOPULATION(P)
7      for i ∈ {1, 2, ..., n} do
8          Oᵢ ← GENEPOOLOPTIMALMIXING(Pᵢ)
9      P ← O
10     t ← t + 1
11     xᵇᵉˢᵗ(t) ← arg min_{x∈P}{fitness[x]}
12     if fitness[xᵇᵉˢᵗ(t)] > fitness[xᵇᵉˢᵗ(t − 1)] then
13         tᴺᴵˢ ← 0
14     else
15         tᴺᴵˢ ← tᴺᴵˢ + 1
```

```
GENEPOOLOPTIMALMIXING(p)
1  b ← o ← p; fitness[b] ← fitness[o] ← fitness[p];
2  changed ← false
3  for i ∈ {1, 2, ..., |F|} in a random order do
4      d ← RANDOM({P₁, P₂, ..., Pₙ})
5      o ← COPYVALUES(o, d, Fⁱ)
6      if o ≠ b then
7          EVALUATEFITNESS(o)
8          if fitness[o] ≥ fitness[b] then
9              b ← o; fitness[b] ← fitness[o]; changed ← true
10         else
11             o ← b; fitness[o] ← fitness[b]
12 if ¬changed or tᴺᴵˢ > 1 + ⌊log₁₀(n)⌋ then
13     changed ← false
14     for i ∈ {1, 2, ..., |F|} in a random order do
15         o ← COPYVALUES(o, xᵉˡⁱᵗⁱˢᵗ, Fⁱ)
16         if o ≠ b then
17             EVALUATEFITNESS(o)
18             if fitness[o] > fitness[b] then
19                 b ← o; fitness[b] ← fitness[o]; changed ← true
20             else
21                 o ← b; fitness[o] ← fitness[b]
22             if changed then breakfor
23 if ¬changed then
24     o ← xᵉˡⁱᵗⁱˢᵗ; fitness[o] ← fitness[xᵉˡⁱᵗⁱˢᵗ]
25 RETURN(o)
```

```
COPYVALUES(x, d, Fⁱ)
1  o ← x
2  for k ∈ Fⁱ do
3      oₖ ← dₖ
4  RETURN(o)
```

Figure 6: Gene-pool Optimal Mixing Evolutionary Algorithm

gests a threshold for NIS of $1 + \lfloor \log_{10}(n) \rfloor$ (Bosman and Thierens, 2013). If FI does not succeed in improving p, $x^{elitist}$ is returned as the new offspring.

The GOMEA with the linkage tree model is the most popular variant of GOMEA, and is also known as the Linkage Tree Genetic Algorithm (LTGA) (Thierens and Bosman, 2011). The concept of constructing an offspring in a step-wise manner by iteratively improving a parent solution is compatible with the hierarchical structure

of linkage trees. GOMEA can also be straightforwardly combined with the univariate or marginal product model. Note that the local search-like mechanism of the GOM operator causes GOMEA to use more fitness evaluations per generation compared to the GA or the EDA. However, for problems that are efficiently solvable with correctly detected linkage, it has been shown that GOMEA has population sizing requirements that are much smaller than those of GA and EDA. GOMEA moreover also requires far fewer generations, resulting in more efficient overall performance (Thierens and Bosman, 2011).

5 Population sizing-free scheme

5.1 The Harik-Lobo scheme

Setting the population size parameter in real-world applications of EAs is hard because a suitable population size setting depends on the structure of the problem instance at hand and also on the specific EA being used. Practitioners often need to experiment with different population sizes in an *ad hoc* manner to find a setting that works well, given the amount of time available. Here, we adapt a population sizing-free scheme that was previously proposed and tested on academic benchmarks (Harik and Lobo, 1999). In essence, we run multiple instances of the EA in parallel. Each instance has a different population size but larger populations have a slower generational cycle. We start with the first population P_0 of some small size n_0. Then, by doubling the population size, the next population P_i is twice as large as the previous one, i.e., $n_i = 2n_{i-1}$ for $i > 0$. All the populations are scheduled with the principle that for every b generations of population P_i, 1 generation of population P_{i+1} is run. If P_{i+1} does not exist yet, it will be initialized before running its first iteration. Having no maximum population size, the EA runs and grows its populations until the computing time budget is used up. Recently, this population sizing-free scheme has been revised with a slightly simpler implementation (Pelikan et al., 2007; Pereira and Lobo, 2015). We refer to this population-sizing free mechanism as the Harik-Lobo scheme in reference to the original authors that first proposed the idea. The pseudo-code for the Harik-Lobo scheme is given in Figure 7.

POPULATION SIZING FREE FRAMEWORK
1   $\mathbf{P}_0 \leftarrow$ INITIALIZENEWPOPULATION(n_0)
2   $generation[0] \leftarrow 0$
3   $max\_population\_index \leftarrow 0$
4   $i \leftarrow 0$
5   **while** ¬TERMINATIONCRITERIASATISFIED() **do**
6   EXECUTEONEGENERATION($\mathbf{P}_i$)
7     $generation[i] \leftarrow generation[i] + 1$
8   **if** $generation[i]$ **mod** $b = 0$ **then**
9       $i \leftarrow i + 1$
10   **if** $i > max\_population\_index$ **then**
11        $\mathbf{P}_i \leftarrow$ INITIALIZENEWPOPULATION($2^i n_0$)
12        $generation[i] \leftarrow 0$
13        $max\_population\_index \leftarrow i$
14   **else**
15        $i \leftarrow 0$

Figure 7: Harik-Lobo Population Sizing-Free Framework (Pereira and Lobo, 2015)

5.2 Adaptations of the Harik-Lobo Scheme and Linkage Model Selection for EAs solving DNEP

5.2.1 Adaptations of the Harik-Lobo Scheme

For reasons of efficiency, the Harik-Lobo scheme terminates old populations of smaller sizes when they converge or when they are shown to be inefficient in solving the problem instance at hand. If mutation is not employed, a converged population, in which all individuals become identical, should be terminated because new offspring cannot be generated any more. A smaller population P_i always spends the same number of fitness evaluations (in case $b = 2$) or more (in case $b > 2$) than a larger population P_{i+1}. Therefore, a smaller population is regarded as inefficient if its average fitness is worse than the average fitness value of any larger population. Such smaller and inefficient populations should be terminated as well. However, the Harik-Lobo scheme has only been tested on unconstrained problems previously. For constrained optimization problems like DNEP, it is uninformative to compare average fitness values of different populations when a population contains both feasible and infeasible solutions. Therefore, we will benchmark two variants of the Harik-Lobo scheme: the original scheme with the termination of converged *or* inefficient (smaller) populations and the adapted scheme with the termination of only converged populations.

We employ Network 3 (see Section 7.1 and the Appendix for more details) as the benchmark network. We combine different linkage models (see Section 3) with different optimization algorithms (see Section 4) to create 7 EA variants: GA-UF, GA-MP, EDA-UF, EDA-MP, GOMEA-UF, GOMEA-MP, and GOMEA-LT. Each EA variant is run 30 times separately with a computing budget of 1,000,000 evaluations for solving the static DNEP problem. Each EA variant is put into the two variants of the Harik-Lobo scheme (both use the generation base $b = 4$ as in the original implementation of the Harik-Lobo scheme (Harik and Lobo, 1999)) so that setting the population size parameter is not required. The average results of the elitist solutions over 30 runs are shown in Figure 8. To support our conclusions from the experimental results, we perform statistical hypothesis testing. In particular, we perform the Mann-Whitney-Wilcoxon statistical hypothesis test for equality of medians with $p < \alpha = 0.05$ to see whether the final result obtained by one EA is statistically different from that of another EA.

For all GOMEA variants, there is no statistically significant difference in performance of the original Harik-Lobo scheme and the adapted scheme (p-values are .665, .947, and .935 for GOMEA-UF, GOMEA-MP, and GOMEA-LT, respectively). On the other hand, for GAs and EDAs, regardless of the employed FOS models, the variants with the adapted scheme always obtain better solutions within the same computing budget. These differences are found to be statistically significant ($p < .001$). Because DNEP is a constrained optimization problem, comparing average fitness values of populations containing both feasible and infeasible solutions might not be an effective method to determine whether a population is inefficient in solving the problem instance at hand. Besides, Figure 8 also indicates that, compared to GA and EDA, GOMEA is the most consistent algorithm, obtaining the same results, regardless of which population sizing-free scheme is chosen. The experiment suggests that the adapted Harik-Lobo scheme is the better population sizing-free framework for EAs solving our DNEP problem. We will use the adapted Harik-Lobo scheme in all following experiments.

Harik and Lobo (1999) employed the generation base $b = 4$ for their parameterless GA. Pelikan et al. (2007) used the generation base $b = 2$ for their implementation of the population sizing-free framework. The generation base $b = 2$ is also used in our
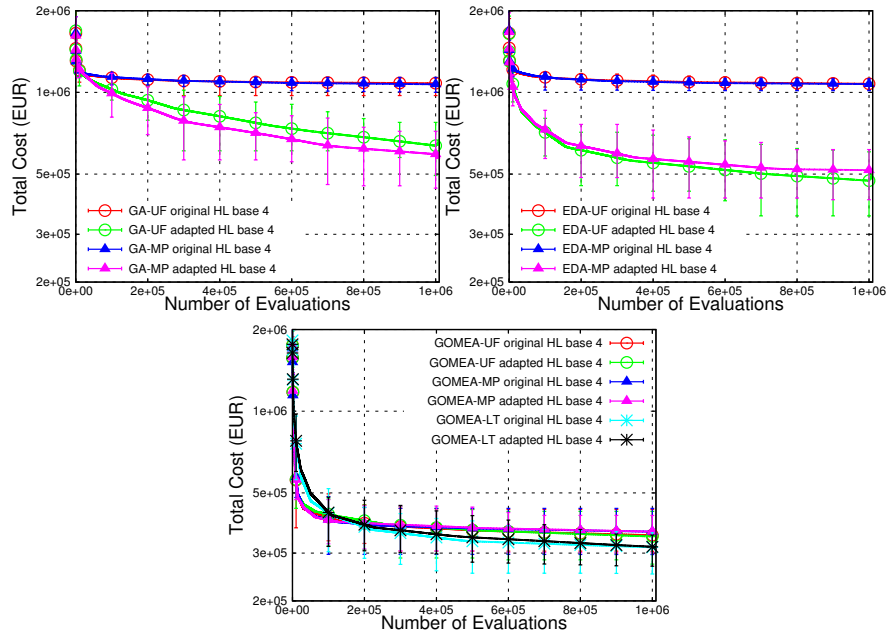
Figure 8: Benchmarking the original Harik-Lobo population sizing-free scheme and the adapted scheme on solving DNEP by GA, EDA, and GOMEA. Error bars show the maximum and minimum values of the Net Present Value of total cost.

previous work (Luong et al., 2015). In this article, we benchmark the adapted Harik-Lobo scheme with both generation bases 2 and 4 to understand which is the better setting for DNEP. We also conduct experiments with 7 EA variants on Network 3 as outlined above. Figure 9 shows the results. The difference between $b = 2$ and $b = 4$ is negligible when GOMEA is the employed optimizer together with the UF model ($p = .053 > \alpha$) or the MP model ($p = .633 > \alpha$). For GOMEA-LT, the result obtained when $b = 4$ is a little better than the result obtained when $b = 2$, and this difference is found to be statistically significant ($p = .003 < \alpha$). For GAs and EDAs, the Harik-Lobo scheme always has better performance when $b = 4$ than when $b = 2$ regardless of the employed FOS model, and the differences are statistically significant ($p < .001$ for the GA cases, $p = .009 < \alpha$ for the EDA-UF case, and $p < .001$ for the EDA-MP case). Therefore, for all following experiments, we employ the adapted Harik-Lobo scheme with the generation base $b = 4$ as the population sizing-free framework for EAs solving our DNEP problem.

5.2.2 Linkage Model Selection

Figure 10 shows that all GOMEA variants, regardless of the chosen linkage models, always obtain solutions of (statistically significantly) better quality than those found by all GA and EDA variants within the allowed computing budget ($p < .001$). This confirms that the laboratory-benchmarked superior performance of the Gene-pool Optimal Mixing operator is retained when solving the real-world optimization DNEP. GOMEA-UF and GOMEA-MP show no difference in their convergence behavior ($p = .09 > \alpha$). The big cardinality of each variable (i.e., much more than 2) causes the complexity term in Equation 10 to be large and thus prohibits the merging of small linkage sets, espe-
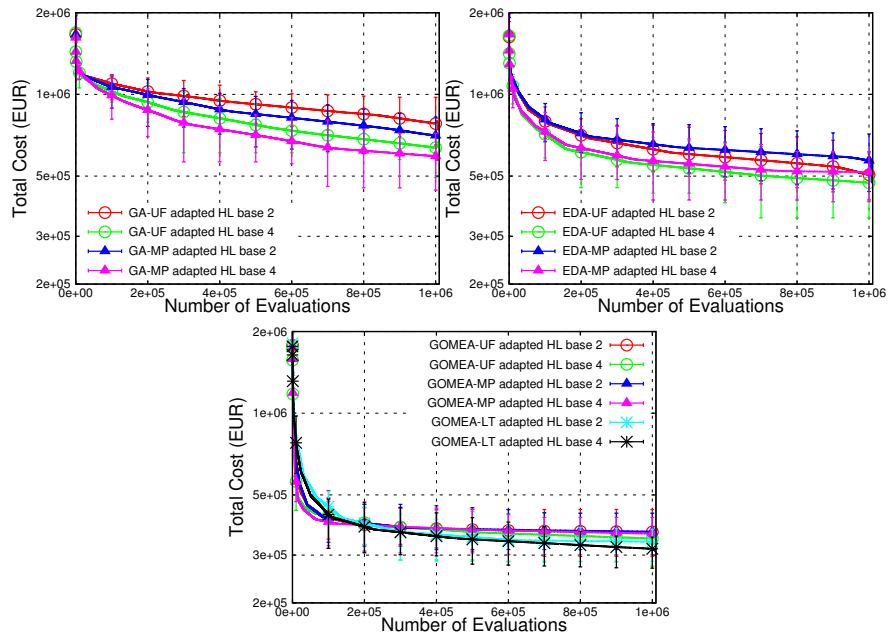
Figure 9: Benchmarking the adapted population sizing-free scheme with two generation bases $b = 2$ and 4 on solving DNEP by GA, EDA, and GOMEA. Error bars show the maximum and minimum values of the Net Present Value of total cost.
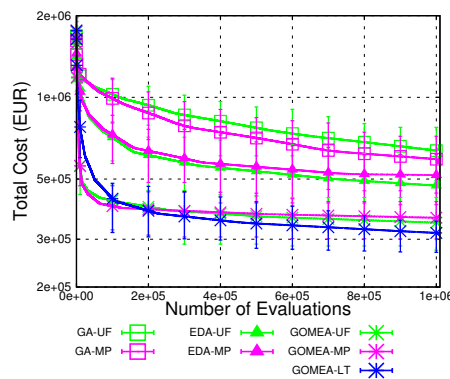

Figure 10: Performance of GA, EDA, and GOMEA when combined with the UF, MP, and LT models on solving DNEP. Error bars show the maximum and minimum values of the Net Present Value of total cost. The adapted Harik-Lobo population sizing-free scheme is employed with the generation base $b = 4$ for all EA variants.

cially when population sizes are small. The learned MP FOS therefore contains mostly univariate linkage sets, which are similar to the UF model. Figure 10 shows that GA-MP performs slightly better than GA-UF while EDA-UF performs slightly better than EDA-MP, and these differences are found to be statistically significant ($p = .019 < \alpha$ for the GA case and $p < .001$ for the EDA case). The fact that EDA-UF outperforms EDA-MP, GA-UF, and GA-MP suggests that the objective function of our DNEP for-

mulation resembles the function OneMax to some degree, i.e., $CAPEX$ is calculated as the sum of the investment cost of each new asset and $OPEX$ is computed as the sum of the capitalized energy loss of every asset. However, the DNEP constraints certainly cause dependencies to exist among problem variables, e.g., the network cables must form a connected graph such that all demand nodes can be served (the connectivity constraint), or there must exist enough redundancies in the network so that it can be reconfigured when some failure occurs (the reconfigurability constraint). These constraints are difficult and numerous. Thus, it stands to reason that the DNEP constraints have a substantial effect on the dependency structures of the feasible space. However, as said, the marginal product model can only capture dependencies to a very limited degree, and therefore its added value, if any, compared to the univariate model, is also limited. The interactions between problem constraints, objective function values, and the model building procedure are worth further research but are outside the scope of this study.

Figure 10 shows that GOMEA-LT outperforms all other EA variants, and the results are found to be statistically significant ($p = .001 < \alpha$ for the comparison between GOMEA-LT and GOMEA-UF and $p < .001$ in all the other cases). We argue that the capability of the LT to model different dependency levels at once, makes the LT much better suited for the DNEP structure. For example, upgrading a cable in one region does not affect the power flows through other cables in a different section of the distribution network (i.e., independence in terms of power flows in normal situation), but those separate cables can be reconfigured to be connected when a network failure happens (i.e., inter-dependence in terms of reconfigurability).

Selecting the best FOS structure for each EA, in all following experiments, we employ the MP model for GA, the UF model for EDA, and the LT model for GOMEA.

6 Exploiting Problem-Specific Knowledge

6.1 Population Initialization

Normally, EAs can start with randomly initialized populations. However, because DNEP is a highly-constrained engineering problem, randomly generated solutions are typically infeasible and violate many constraints. Therefore, we use a repair procedure that partially repairs infeasible solutions by comparing them with the current, i.e., starting, network situation. First, each decision variable x_k (i.e., a network branch) can only receive a random non-negative value (i.e., we do not place any NOPs yet) as long as it does not downgrade the currently existing cable. This also means that currently existing cables can only be left intact or be replaced with cables of higher capacities. Existing connections are rarely removed because cable connection removals decrease network capacity. Solutions that are generated in this way satisfy the connectivity constraint because the current network is connected. Second, we go through HV/MV substations and check the number of cables branching out from each substation. If the number of outgoing cables is more than the allowable capacity of the substation (i.e., violating constraint 5), we randomly delete outgoing cables until constraint 5 is satisfied. Third, we go through all variables that have positive values (i.e., active cables) in a random order. For each positive-value decision variable, we try to place an NOP on that cable by negating its value. If the network is still connected, then the NOP can be placed; otherwise, we undo the operation. This procedure returns a network of radial topology with random placements of NOPs (i.e., constraint 3 is satisfied). We do not repair the power flow and reconfigurability constraint (i.e., constraint 2 and 4) because they involve PFCs, which are computationally expensive. Figure 11 shows the pseudo-code

for randomly generating a network solution, which can be used in the population initialization phase.

```
DNEP::CREATERANDOMSOLUTION()
// 𝕋 = set of all cable types (including 0).
// x = (x₁, x₂, ..., x_l), |x_k| ∈ Ω(k), k ∈ {1, 2, ..., l}
1  x^now ← GETCURRENTNETWORKCONFIGURATION()
2  for k ∈ {1, 2, ..., l} do
3     Ω(k) ← {ID ∈ 𝕋 | capacity[ID] ≥ capacity[||x_k^now||]}
4     x_k ← RANDOM(Ω(k))
5  x ← REMOVEREDUNDANTCONNECTIONS(x, x^now)
6  x ← RADIALIZENETWORK(x)
7  RETURN(x)
```

```
REMOVEREDUNDANTCONNECTIONS(x, x^now)
// 𝕊 = set of all HV/MV substations.
1   o ← x
2   for s ∈ 𝕊 do
3      C_s = {}
4   for k ∈ {1, 2, ..., l} do
5      if ISOUTGOINGCABLE(k) and x_k ≠ 0 and x_k^now = 0 then
6         s ← GETSUBSTATIONINDEXOFCABLE(k)
7         C_s ← C_s ∪ {k}
8   for s ∈ 𝕊 do
9      if |C_s| > max_s then
10        π ← RANDOMPERMUTATION({1, 2, ..., |C_s|})
11        for i ∈ {1, 2, ..., |C_s| − max_s} do
12           o_{π_i} ← 0
13  RETURN(o)
```

```
RADIALIZENETWORK(x)
1   o ← x
2   π ← RANDOMPERMUTATION({1, 2, ..., l})
3   for i ∈ {1, 2, ..., l} do
4      if o_{π_i} > 0 then
5         o_{π_i} ← −o_{π_i}
6         if ¬CHECKCONNECTIVITY(o) then
7            o_{π_i} ← x_{π_i}
8   RETURN(o)
```

Figure 11: Generating a distribution network configuration.

6.2 Variation Operators

Being popularly applied in black-box optimization, EAs require little problem-specific knowledge (PSK) and their variation operators (VOs) (i.e., procedures to generate new offspring solutions) can operate on a wide range of problems. However, in real-world applications like DNEP, the problems are often highly constrained such that it is difficult for general-purpose VOs to traverse the search space of feasible solutions efficiently. Efficiently traversing the space of feasible solutions is of high importance because the evaluations of candidate solutions are typically computationally expensive. Full constraint evaluations of solution plans for DNEP involve solving PFCs (Grainger and Stevenson, 2003), which dominates the computing time of the optimization process. Thus, it is beneficial to incorporate PSK into VOs of EAs as efficiency enhancement methods. Local search heuristics can normally be used for efficiency enhancement, but a typical operator like hillclimbing with some small neighborhood (e.g., 1/2/3-opt

local search) was not found to be helpful in improving the efficiency of EAs solving our DNEP model because such local search often fails to reach expansion options that include inter-dependent activities (e.g., adding new cables and NOPs and relocating existing NOPs).

Among the constraints of DNEP, the connectivity constraint needs to be handled separately because PFCs cannot be performed on unconnected networks and the power flow constraint and reconfigurability constraint cannot be checked without PFCs. Thus, in order to compare solution plans (e.g., when performing tournament selection), we need to quantify their disconnectivity and use that as the comparison criterion or we have to repair the connectivity so that other constraints can be evaluated. Here, we introduce different VOs for GA, EDA, and GOMEA along with their corresponding connectivity constraint-handling techniques.

6.2.1 Disconnectivity Quantification

We name the out-of-the-box VO of EAs when solving our DNEP problem DQ1 because it employs the disconnectivity quantification procedure, which is part of the constraint evaluation itself. Apart from that, no other connectivity knowledge is assumed when generating offspring solutions. At every recombination, model sampling, or mixing event, a new offspring solution (or partially-new solution in case of GOMEA) is created and is evaluated for its fitness value. The connectivity constraint is checked first, and only a connected network solution is then evaluated for other constraints and objective value. Candidate solutions can be compared by the connectivity-constraint domination mechanism as presented in Section 2.2.

However, even if we recombine two connected parent networks, it is still difficult for the crossover operator of GA to generate connected offspring networks, especially for big networks with many cable connections. Thus, we also propose a different VO, in which we allow each recombination of two parent networks to retry crossover to generate connected offspring networks. After 100 times, if the offspring networks are still unconnected, they will be evaluated for the disconnectivity value as above. Similarly, for EDA, we allow maximum 100 times of model sampling to generate a connected networks before performing disconnectivity quantification. For GOMEA, during the process of constructing an offspring, for each mixing event, if the partially-altered solution is an unconnected network, we allow it to randomly select a different donor for a maximum of 100 times. We call this VO DQ100. Figure 12 shows the pseudo-code for DQ100 with its corresponding realization in GA, EDA and GOMEA.

6.2.2 Connectivity Repair

Inspired by the Forced Improvement (FI) operator of GOMEA, in which solutions that cannot be altered by GOM will be mixed with the elitist solution $x^{elitist}$, we propose a repair procedure to fix any unconnected network x by matching it with the (overall) best-found-so-far candidate network $x^{best} = x^{elitist}$. For each decision variable k, if $x_k^{best} > 0$ and $x_k < 0$, then $x_k \leftarrow -x_k$; if $x_k^{best} > 0$ and $x_k = 0$, then $x_k \leftarrow x_k^{best}$. On the other hand, if $x_k^{best} < 0$ and $x_k > 0$, then $x_k \leftarrow -x_k$; if $x_k^{best} = 0$ and $x_k > 0$, then $x_k \leftarrow 0$. In other words, we try to transform the topology of the unconnected network to the best solution's topology. We call the VO that uses this connectivity repair procedure CRB (i.e., Connectivity Repair by the Best solution). Figure 13 shows the pseudo-code for the CRB scheme. CRB can be straightforwardly employed by any EA.

We propose a second connectivity repair mechanism for unconnected offspring network by using the parent solution networks. For GA, this VO is much like DQ100,

```
GA::RECOMBINE::DQ100(p^0, p^1)
1  o ← RECOMBINE(p^0, p^1); count ← 1
2  while ¬CHECKCONNECTIVITY(o) and count < 100 do
3    o ← RECOMBINE(p^0, p^1)
4    count ← count + 1
5  RETURN(o)
```

```
EDA::SAMPLEDISTRIBUTION::DQ100()
1  o ← SAMPLEDISTRIBUTION(); count ← 1
2  while ¬CHECKCONNECTIVITY(o) and count < 100 do
3    o ← SAMPLEDISTRIBUTION()
4    count ← count + 1
5  RETURN(o)
```

```
GOMEA::COPYVALUES::DQ100(x, d, F^i)
1  o ← COPYVALUES(x, d, F^i); count ← 1
2  while ¬CHECKCONNECTIVITY(o) and count < 100 do
3    d' ← RANDOM({P_1, P_2, ..., P_n})
4    o ← COPYVALUES(x, d', F^i)
5    count ← count + 1
6  RETURN(o)
```

Figure 12: Disconnnectivity Quantification DQ100

```
CONNECTIVITYREPAIRBYBESTSOLUTION(x)
 1  x^best ← GETCURRENTBESTNETWORKCONFIGURATION()
 2  o ← x
 3  for k ∈ {1, 2, ..., l} do
 4    if x_k^best > 0 and x_k < 0 then
 5      o_k ← -x_k
 6    else if x_k^best > 0 and x_k = 0 then
 7      o_k ← x_k^best
 8    else if x_k^best < 0 and x_k > 0 then
 9      o_k ← -x_k
10    else if x_k^best = 0 and x_k > 0 then
11      o_k ← 0
12  RETURN(o)
```

Figure 13: Connectivity Repair by the Best Solution.

but after 100 trials, if the networks are still unconnected they will be reverted back to the parent solutions. Because all the candidate solutions in the initial population are connected networks (due to our solution network generator as presented in Section 6.1), the use of this VO implies that only connected offspring are allowed to be evaluated and enter tournament selection. For GOMEA, in each mixing event, if the partially-altered solution becomes unconnected, this is because there exist some variables whose positive values are replaced by some non-positive values from the donor. We can simply revert these decision variables to their backup values. We call the VO that uses this connectivity repair scheme CRP (i.e., Connectivity Repair by Parent). Note that we do not combine EDA with CRP because offspring solutions in EDA do not have direct parent solutions like GA or GOMEA, but are generated by sampling the learned probability distribution. Figure 14 shows the pseudo-code for the realization of the CRP scheme in GA and GOMEA.

```
GA::CONNECTIVITYREPAIRBYPARENT(p⁰, p¹)
1  o ← GA::RECOMBINE:DQ100(p⁰, p¹)
2  if ¬CHECKCONNECTIVITY(o) then
3      o ← p⁰
4  RETURN(o)
```

```
GOMEA::COPYVALUES::CONNECTIVITYREPAIRBYPARENT(x, d, Fⁱ)
1  o ← GOMEA::COPYVALUES(x, d, Fⁱ)
2  if ¬CHECKCONNECTIVITY(o) then
3      for k ∈ Fⁱ do
4          if xₖ > 0 and oₖ ≤ 0 then
5              oₖ ← xₖ
6  RETURN(o)
```

Figure 14: Connectivity Repair by the Best Solution.

6.2.3 Branch Exchanging

This VO aims to directly generate connected offspring networks by following the principle that during the recombination of two connected networks p^0 and p^1, if we bring a cable connection from p^1 to p^0 (i.e., $p_k^0 \leftarrow p_k^1, p_k^0 \leq 0, p_k^1 > 0$), we need to bring the corresponding NOP (or a no-connection branch) from p^1 to p^0 as well (i.e., $p_j^0 \leftarrow p_j^1, p_j^0 > 0, p_j^1 \leq 0$) and vice versa.

For GA, during the recombination of two parents p^0 and p^1, when we copy values from p^1 according to a linkage set F^i in the FOS $\mathcal{F}$ (see Figure 4), for variables whose values are both positive (i.e., both are active cables) or both non-positive (i.e., no-connection branches or NOPs), we can copy as normal since these positions have the same structures in both networks. For each variable index k where $p_k^0 \leq 0$ and $p_k^1 > 0$ (or $p_k^0 > 0$ and $p_k^1 \leq 0$), we need to find a different variable j where $p_j^0 > 0$ and $p_j^1 \leq 0$ (or $p_j^0 \leq 0$ and $p_j^1 > 0$) such that copying values at variables k and j from the network p^1 still maintains the connectivity of the network p^0. If we cannot find such a variable j then we do not perform crossover at the variable k. Note that while variable k belongs to the current linkage set F^i, variable j is searched for over the whole solution. Because the complicated connectivity structure might not be entirely captured by linkage learning, an active cable and its corresponding NOP might not always reside in the same linkage set F^i. In (Luong et al., 2015), we employed a slightly different version of branch exchange for GA, which maintained the connectivity of both networks p^0 and p^1. Our previous implementation of GA created two offspring from two parents for each recombination while in this article, GA creates one offspring from two parents.

For GOMEA, in each mixing event, this procedure works similarly for the current solution o and a donor d and we need to search for the variable j when $o_k \leq 0$ and $d_k > 0$ (or $o_k > 0$ and $d_k \leq 0$). Also, we only need to maintain the connectivity of the current solution. Similar to the CRP procedure, we do not employ branch exchange for EDA because EDA does not create offspring solutions by directly exchanging values between parent solutions but by sampling the probability distribution instead. This VO is problem-specific because it employs connectivity knowledge of DNEP when generating new offspring. We call this VO BX (i.e., branch exchanging). Figure 15 shows the pseudo-code for the realization of BX in GA and GOMEA.

Furthermore, we here experiment the combination of BX with a DNEP-specific mutation. After every mixing event with a linkage set, but before the evaluation of the partially-altered solution, we go through every variable in the linkage set and perform

```
COPYVALUES::BRANCHEXCHANGE(x, d, F^i)
1   X ← {k ∈ {1, 2, …, l} | (x_k > 0 and d_k > 0) or (x_k ≤ 0 and d_k ≤ 0)}
2   X' ← {1, 2, …, l} \ X
3   Y ← {k ∈ F^i | (x_k > 0 and d_k > 0) or (x_k ≤ 0 and d_k ≤ 0)}
4   Y' ← F^i \ Y
5   o ← x
6   for k ∈ Y do
7       o_k ← d_k
8   for k ∈ Y' do
9       for j ∈ X' \ {k} in a random order do
10          if (o_k ≤ 0 and d_j ≤ 0) or (o_k > 0 and d_j > 0) then
11              o_k ← d_k; o_j ← d_j
12              if CHECKCONNECTIVITY(o) then
13                  X' ← X' \ {k, j}; Y' ← Y' \ {k, j}
14                  break for
15              else
16                  o_k ← x_k; o_j ← x_j
17  RETURN(o)
```

Figure 15: Branch Exchange.

a mutation with probability $1/l$ (l is the length of the solution). The mutated values must have the same sign as the original values (i.e., positive or non-positive) so that the connectivity and radiality of the network are still maintained. We call the branch exchanging VO with this mutation operator BX-M.

7 Experiments

7.1 Benchmark Problems

We perform experiments on 3 benchmark networks of different sizes that represent real distribution networks of a DNO in The Netherlands. Figure 16 shows the current topologies of the 3 networks. More details about current power consumption, potential locations for installing new cables, and characteristics of different cables types can be found in the Appendix. The overall sizes of these 3 benchmarks are summarized in Table 1. The maximum number of evaluations is 50,000 for Network 1, 100,000 for Network 2, and 1,000,000 for Network 3. The planning period is 30 years, and for each problem instance and each variation operator, each EA (i.e., GA, EDA or GOMEA) is run 30 times. For computational reasons, for dynamic DNEP of network 3, the planning period is 10 years. Figures 17 and 18 show the average convergence graphs of the elitist solutions obtained by GA, EDA, and GOMEA integrated with different VOs. We perform the Mann-Whitney-Wilcoxon statistical hypothesis test for equality of medians with $p < \alpha = 0.05$ to verify if there exists a statistically significant difference between the final result achieved by one EA and that of another EA.

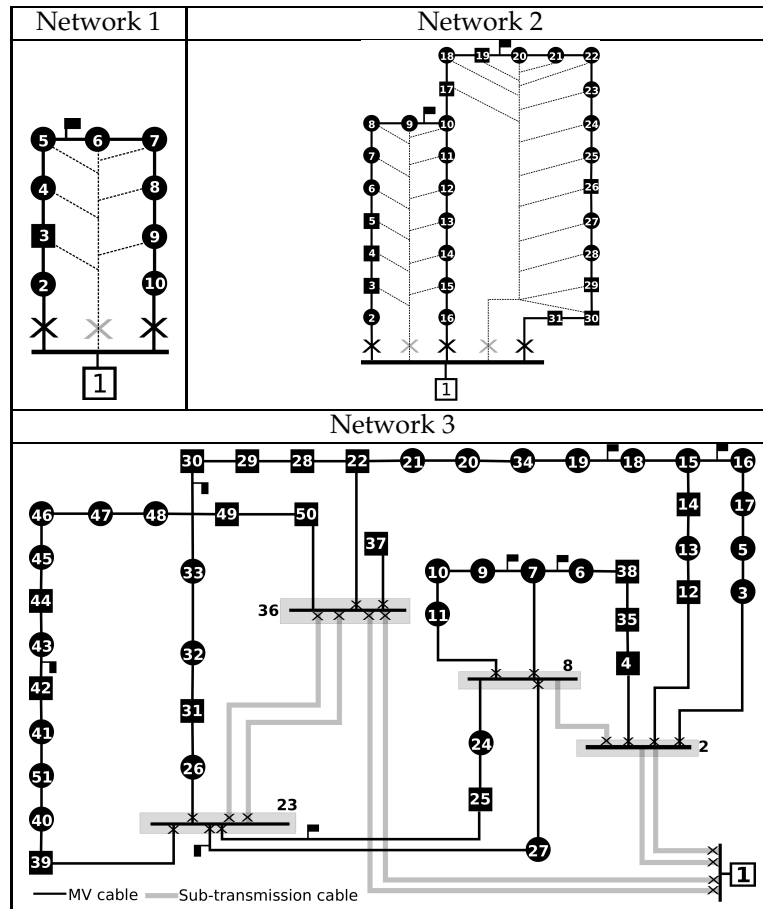| ID	# Branches (Variables)	# Nodes	# HV/MV Substations	# Cable types
1	17	10	1	3
2	59	31	1	3
3	190	51	4	12

Table 1: Benchmark Network Size

Figure 16: 3 benchmark MV distribution networks with existing assets. Potential cable connections can be found in the Appendix. Legends are explained in Figure 1.

7.2 Results

7.2.1 Static DNEP

Figure 17 shows the performance of GA, EDA and GOMEA integrated with different VOs solving the static DNEP. For Network 1 (see Figure 17a), almost all EA variants (except EDA-CRB) exhibit the same effectiveness in obtaining (near-)optimal solutions. After spending a certain number of evaluations, most solvers have similar convergence until termination. For Network 2 (see Figure 17b), GOMEA-BX(-M) performs slightly better than other EA variants, and the difference is found to be statistically significant ($p < .001$). EDA-CRB is the worst optimizer when solving Networks 1 and 2. However, on small networks, these differences between EA variants and between different VOs are practically negligible (e.g., differences of about 1,000-3,000 EUR for a planning period of 30 years).
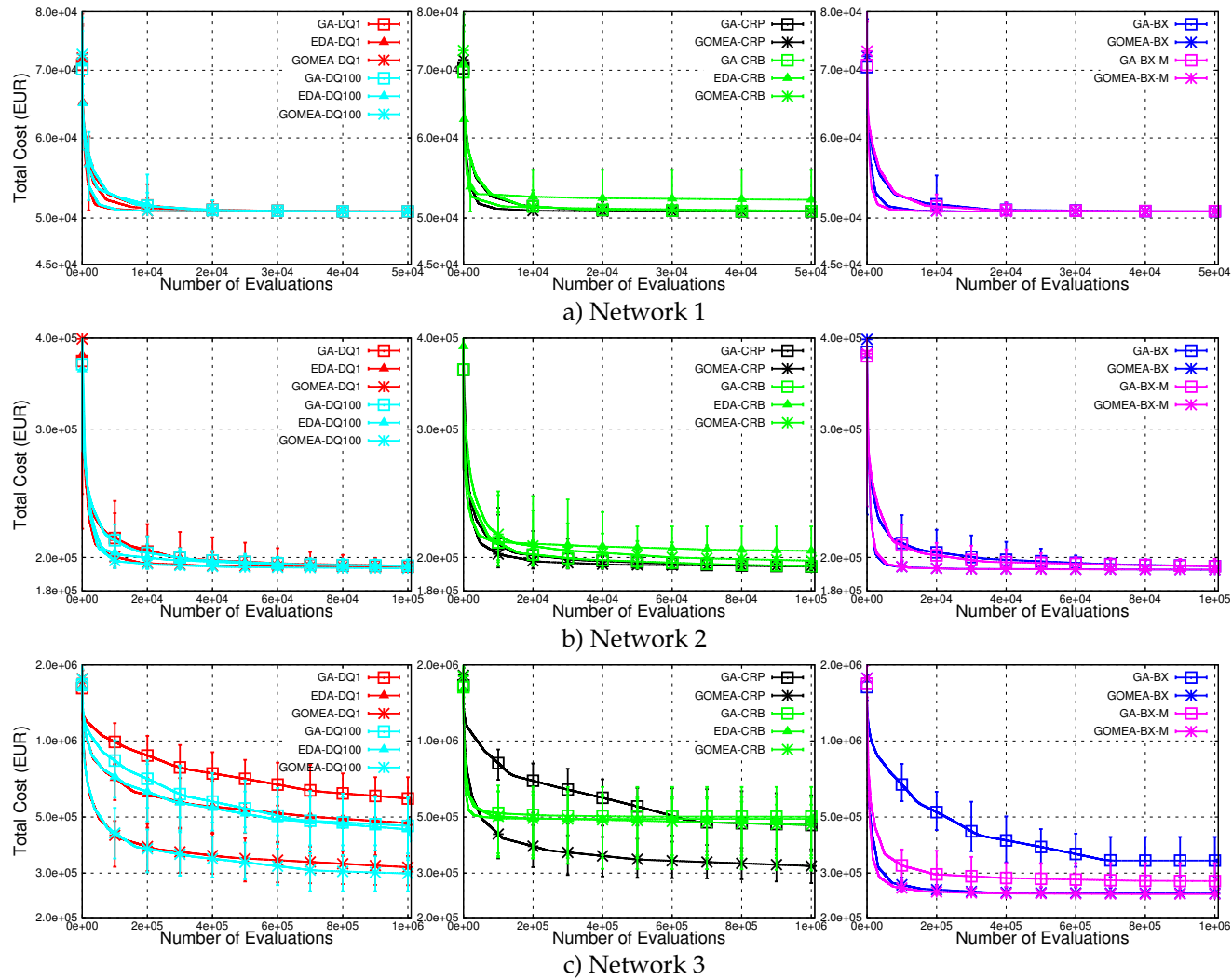
Figure 17: Static DNEP. Horizontal axis: number of evaluations. Vertical axis: Net Present Value (NPV) of total cost (EUR). Error bars show the maximum and minimum values of the NPV of total cost.

In contrast, Figure 17c shows that, when solving DNEP for a larger network, the performance gaps between different EA variants are considerable. For Network 3, GA-DQ1 is outperformed by all other EA variants ($p < .001$). Even though our EDA implementation employs the UF model, EDA-DQ1 has significantly better performance than GA-DQ1 ($p < .001$), which suggests that the UF model sampling might be more effective than the MP model-based crossover operator (see Section 5.2.2). While also assuming no connectivity knowledge, within the same number of evaluations, GOMEA-DQ1 outperforms both GA-DQ1 and EDA-DQ1, and obtains solution plans of much better quality. On average, the expansion plans obtained by GOMEA-DQ1 costs about 150,000-280,000 EUR less than those obtained by GA-DQ1 and EDA-DQ1. These differences are found to be statistically significant ($p < .001$). DQ100 gives GA multiple trials of recombination to generate new connected networks, and that indeed helps GA improve its performance significantly ($p < .001$, but still worse than GOMEA-DQ1 in terms of the quality of obtained solutions at termination). DQ100 induces some small improvements for EDA-DQ1, but the differences are not significant ($p = .119 > \alpha$).

While the connectivity repair VOs CRP and CRB significantly enhance the effectiveness and efficiency of GA ($p < .001$), they do not show any improvement over GOMEA-DQ1. For GOMEA, repairing by parent solutions (CRP) are (statistically significantly) better than repairing by the best solutions (CRB) ($p < .001$). GOMEA-CRB actually has the worst convergence behavior among all GOMEA variants, and even GOMEA-DQ1 can significantly outperform GOMEA-CRB ($p < .001$). This can be because it is difficult for general-purpose VOs to generate connected networks and keeping matching unconnected offspring networks with the slowly-changing best topology reduces the beneficial diversity in the population, making the algorithm prone to premature convergence.

Solving the issue of maintaining connectivity and radiality, the VO BX brings out significant improvements in efficiency for GA and GOMEA when solving DNEP ($p < .001$). GA-BX has excellent performance, obtaining significantly better solutions than GA-DQ1 ($p < .001$). BX also has positive impacts on GOMEA but the size of the effect is much less substantial than on GA. The results here suggest 2 conclusions. First, DNEP problem-specific VOs are crucial for GAs, to efficiently solve (larger) real-world networks. Second, GOMEA is the more robust solver, which can be used out-of-the-box and still obtain solutions of good quality close to those found by EAs customized with problem-dedicated VOs (i.e., comparing GOMEA-DQ1 in leftmost graphs with GA-BX and GOMEA-BX in rightmost graphs in Figure 17c). GOMEA-DQ1 is even found to obtain results that are significantly better than those found by GA-BX ($p = .004 < \alpha$).

The DNEP mutation operator has no significant influence over GOMEA-BX ($p = .052 > \alpha$), but results in a statistically significant improvement for GA-BX ($p < .001$). GA-BX-M, however, is still outperformed by GOMEA-BX(-M) ($p < .001$). On average, the expansion plans obtained by GOMEA-BX(-M) cost about 30,000 EUR less than the plans obtained by GA-BX-M at termination. Statistical tests support that the quality of expansion plans for Network 3 obtained by GOMEA-BX(-M) are significantly better than those obtained by all other EA variants ($p < .001$). Thus, GOMEA-BX(-M) is the best solver in this test case (i.e., the fastest solver given the budget of evaluations used in our experiments).
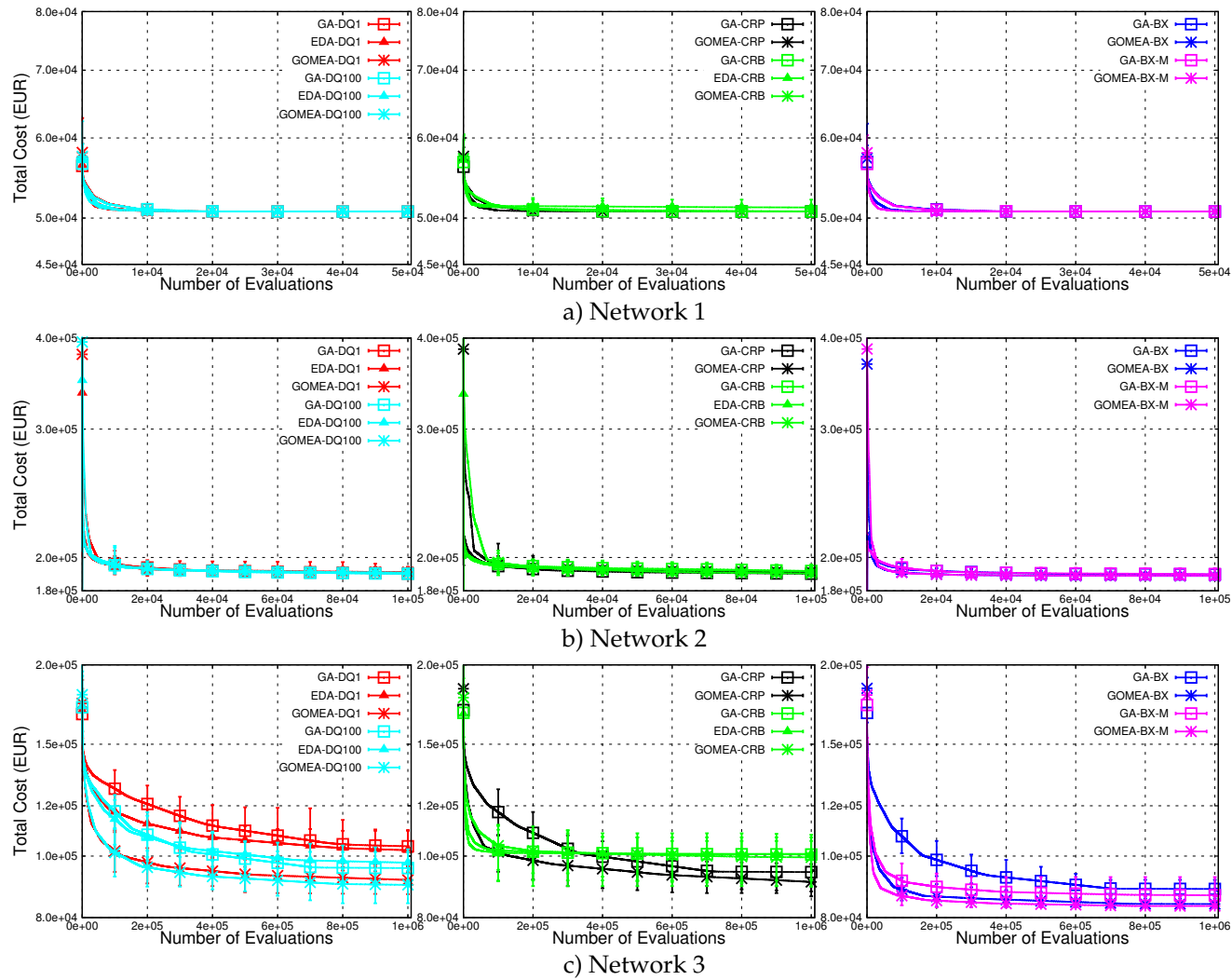
Figure 18: Dynamic DNEP. Horizontal axis: number of evaluations. Vertical axis: Net Present Value (NPV) of total cost (EUR). Error bars show the maximum and minimum values of the NPV of total cost.

7.2.2 Dynamic DNEP

Figure 18 shows the convergence performance of GAs, EDAs and GOMEAs solving dynamic DNEP. The general performances of all EA variants are similar to those observed in the case of static DNEP, suggesting that the use of the decomposition heuristic does not introduce additional complexity to the problem. For Networks 1 and 2 (see Figure 18a and 18b), the differences in the quality of solutions obtained by different EA variants are either statistically insignificant or practically negligible.

For Network 3, Figure 18c shows that GOMEA-DQ1 significantly outperforms both EDA-DQ1 and GA-DQ1 ($p < .001$). Even when only the simplest VO DQ1 is used, GOMEA can still obtain solutions of high quality, which are practically close to the ones found by the DNEP problem-specific EAs, i.e., GA-BX(-M) and GOMEA-BX(-M). The VO BX, which maintains the connectivity and radiality of offspring networks during solution recombination, is again shown to be the best VO, resulting in (statistically significantly) better performance when combined with both GA and GOMEA ($p < .001$). In general, the more domain knowledge is well-incorporated into variation operators, the better the performance of EAs. However, the improvement gaps that this problem-specific VO brings about for GOMEA-DQ1 (on average about 7,000 EUR) are not as substantial as for GA-DQ1 (on average about 14,000 EUR). For dynamic DNEP, BX-M (branch exchange with mutation) induces some small improvements for both GA-BX and GOMEA-BX, in which the differences are found to be statistically significant ($p < .001$ for the GA case, and $p = .012 < \alpha$ for the GOMEA case). GOMEA-BX-M is the overall best solver in this experiment, which (statistically significantly) outperforms other EA variants ($p = .012 < \alpha$ in the case of GOMEA-BX-M versus GOMEA-BX, and $p < .001$ in all the other cases). While it might not be necessary for GOMEA solving many laboratory benchmarks, mutation can still be beneficial when tackling real-world problems because the linkage structure is not always the only or the most important structure to be exploited.

8 Conclusions

This article contributes guidelines and methodologies for the application of EAs in tackling the real-world optimization DNEP. First, we proposed a decomposition heuristic that allows available static planning solvers to solve the dynamic DNEP in a practical manner. Second, we suggested practitioners to employ population sizing-free schemes to get rid of the notoriously-difficult-to-set population size parameter. Third, we introduced multiple variation operators that can be employed by EAs solving DNEP and showed their impact on the performance of 3 typical EAs: the classic GA, an EDA, and the GOMEA, which is capable of learning and exploiting hierarchical linkage relations. GOMEA is shown to be a far more robust solver for solving DNEP on our benchmark networks. Using the same number of solution evaluations as GA and EDA, GOMEA obtains better results, even when assuming a minimal amount of problem-specific knowledge (PSK). Adding PSK to GOMEA improves performance but the improvement gap, for a fixed budget of evaluations, is much less substantial than that for classic GA, which again confirms the usefulness of GOMEA and linkage learning (LL) to detect problem structure. As the problem size increases, LL is of great importance to ascertain efficient scalability of EAs. Lastly, based on our results, we suggest that LL EAs, like GOMEA, should be given priority when selecting EAs for solving DNEP. While this article focuses on expansion planning for medium voltage distribution networks, the methodologies presented here can be straightforwardly modified to apply to both low voltage distribution networks and high voltage transmission net-

works. Besides traditional asset installations, future works might also include smart grid technologies (e.g., battery energy storage system and demand side management) as alternative network reinforcement options, which induce more complex problem structures, potentially requiring linkage information and problem-specific knowledge to be effectively exploited to solve the problem in an efficient manner.

Acknowledgments

References

Borges, C. L. T. and Martins, V. F. (2012). Multistage expansion planning for active distribution networks under demand and distributed generation uncertainties. *International Journal of Electrical Power & Energy Systems*, 36(1):107 – 116.

Bosman, P. A. N. and Thierens, D. (2013). More concise and robust linkage learning by filtering and combining linkage hierarchies. In *Proc. of the Genetic and Evolutionary Comp. Conf. - GECCO-2013*, pages 359–366. ACM.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(24):311 – 338.

Diaz-Dorado, E., Cidras, J., and Miguez, E. (2002). Application of evolutionary algorithms for the planning of urban distribution networks of medium voltage. *Power Systems, IEEE Transactions on*, 17(3):879–884.

Grainger, J. J. and Stevenson, W. D. (2003). *Power System Analysis*. McGraw-Hill Education.

Gronau, I. and Moran, S. (2007). Optimal implementations of UPGMA and other common clustering algorithms. *Information Processing Letters*, 104(6):205 – 210.

Grond, M., Pouw, J., Morren, J., and Slootweg, H. (2014a). Applicability of line outage distribution factors to evaluate distribution network expansion options. In *Power Engineering Conference (UPEC), 2014 49th International Universities*, pages 1–6.

Grond, M. O. W. (2011). *Impact of Future Residential Loads on Medium Voltage Networks*. Master thesis, Delft University of Technology.

Grond, M. O. W., Luong, N. H., Morren, J., Bosman, P. A. N., Slootweg, J. G., and La Poutré, H. (2014b). Practice-oriented optimization of distribution network planning using metaheuristic algorithms. In *18th Power Systems Computation Conference*, pages 1–8.

Harik, G. R. and Lobo, F. G. (1999). A parameter-less genetic algorithm. In *Proc. of the Genetic and Evolutionary Comp. Conf. (GECCO 1999)*, pages 258–265.

Harik, G. R., Lobo, F. G., and Sastry, K. (2006). Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA). In Pelikan, M., Sastry, K., and Cantú-Paz, E., editors, *Scalable Optimization via Probabilistic Modeling*, volume 33 of *Studies in Computational Intelligence*, pages 39–61. Springer.

Luong, N. H., Grond, M. O. W., Bosman, P. A. N., and La Poutré, H. (2013). Medium-voltage distribution network expansion planning with gene-pool optimal mixing evolutionary algorithms. In *Artificial Evolution - 11th International Conference, EA 2013*, pages 93–105.

Luong, N. H., La Poutré, H., and Bosman, P. A. N. (2015). Exploiting linkage information and problem-specific knowledge in evolutionary distribution network expansion planning. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*, pages 1231–1238.

Mühlenbein, H. (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (2000). Linkage problem, distribution estimation, and bayesian networks. *Evolutionary Computation*, 8(3):311–340.

Pelikan, M., Hartmann, A. K., and Lin, T. (2007). Parameter-less hierarchical bayesian optimization algorithm. In Lobo, F. G., Lima, C. F., and Michalewicz, Z., editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 225–239. Springer.

Pereira, J. C. and Lobo, F. G. (2015). A java implementation of parameter-less evolutionary algorithms. *CoRR*, abs/1506.08694.

Ramirez-Rosado, I. and Bernal-Agustin, J. (1998). Genetic algorithms applied to the design of large power distribution systems. *Power Systems, IEEE Transactions on*, 13(2):696–703.

Thierens, D. and Bosman, P. A. N. (2011). Optimal mixing evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO-2011*, pages 617–624. ACM.

Verzijlbergh, R. A., Grond, M. O. W., Lukszo, Z., Slootweg, J. G., and Ilic, M. D. (2012). Network impacts and cost savings of controlled EV charging. *Smart Grid, IEEE Transactions on*, 3(3):1203–1212.

A   Extra material

ID	Type	I_{nom}	R (Ω/km)	X (Ω/km)	C (μF/km)	Cost (€/km)
1	120 mm²	215 A	0.257	0.085	0.38	50 000
2	150 mm²	295 A	0.20858	0.09592	0.3833	59 000
3	240 mm²	370 A	0.13517	0.10823	0.43553	62 000
4	400 mm²	475 A	0.08077	0.09972	0.5344	66 000
5	630 mm²	605 A	0.0511	0.09272	0.64103	73 000
6	N/A	135 A	0.53253	0.09777	0.27072	N/A
7	N/A	160 A	0.3737	0.09367	0.30671	N/A
8	N/A	195 A	0.26756	0.08995	0.34505	N/A
9	N/A	225 A	0.32829	0.10134	0.32667	N/A
10	N/A	320 A	0.13079	0.07757	0.53109	N/A
11	N/A	350 A	0.10193	0.08004	0.48485	N/A
12*	N/A	N/A	N/A	N/A	N/A	N/A

Table A.1: MV cable types. Type 12 (sub-transmission cable) and types 6-11 (legacy cables) are not used for new cable installations in MV DNEP.

ID	Load P [kW]	Load Q [kVAR]	Customers [#]	Existing Branch	Existing Length [m]	Existing Type	Potential Branch	Potential Length [m]
1	0	0	0	1 - 2	654	1	1 - 3	1235
2	271	168	131	1 - 10	710	1	1 - 4	1259
3	924	573	1	2 - 3	610	1	1 - 5	1323
4	394	244	190	3 - 4	163	1	1 - 6	1711
5	409	253	197	4 - 5	511	1	1 - 7	1904
6	394	244	190	5 - 6	496	1	1 - 8	1976
7	370	229	179	6 - 7	420	1	1 - 9	1781
8	117	72	57	7 - 8	297	1		
9	259	160	125	8 - 9	336	1		
10	431	267	208	9 - 10	690	1		

Table A.2: Network 1 Data

ID	Load P [kW]	Load Q [kVAR]	Customers [#]	Existing Branch	Existing Length [m]	Existing Type	Potential Branch	Potential Length [m]
1	0	0	0	1 - 2	481	3	1 - 3	526
2	35	17	25	1 - 16	246	3	1 - 4	469
3	1113	539	1	1 - 31	761	3	1 - 5	989
4	348	216	1	2 - 3	96	2	1 - 6	2062
5	871	286	1	3 - 4	48	2	1 - 7	738
6	332	109	232	4 - 5	498	2	1 - 8	2227
7	132	82	92	5 - 6	86	2	1 - 9	2307
8	170	82	119	6 - 7	288	2	1 - 10	2633
9	22	14	15	7 - 8	935	2	1 - 11	3041
10	202	98	141	8 - 9	200	2	1 - 12	3041
11	120	0	84	9 - 10	470	2	1 - 13	1395
12	88	55	62	10 - 11	851	3	1 - 14	2194
13	284	137	199	10 - 17	736	2	1 - 15	923
14	219	136	153	11 - 12	220	3	1 - 17	2808
15	314	152	220	12 - 13	300	3	1 - 18	2760
16	185	90	130	13 - 14	284	3	1 - 19	2653
17	127	79	1	14 - 15	479	3	1 - 20	1275
18	17	8	12	15 - 16	846	3	1 - 21	1205
19	896	434	1	17 - 18	101	2	1 - 22	1136
20	314	152	220	18 - 19	154	2	1 - 23	1131
21	125	77	88	19 - 20	283	2	1 - 24	1041
22	248	120	174	20 - 21	308	2	1 - 25	950
23	85	41	60	21 - 22	133	2	1 - 26	966
24	123	76	86	22 - 23	132	2	1 - 27	900
25	209	130	146	23 - 24	138	2	1 - 28	804
26	566	274	1	24 - 25	140	2	1 - 29	677
27	266	129	186	25 - 26	103	2	1 - 30	801
28	126	61	88	26 - 27	215	2		
29	360	174	1	27 - 28	139	2		
30	273	169	191	28 - 29	218	2		
31	263	163	1	29 - 30	136	2		
				30 - 31	160	3		

Table A.3: Network 2 Data

ID	Load P [kW]	Load Q [kVAR]	Cust-omers [#]	Existing Branch	Existing Length [m]	Existing Type	Potential Branch	Potential Length [m]	Potential Branch	Potential Length [m]
1	0	0	0	1 - 2	1	12	2 - 5	1335	8 - 38	1900
2	67	32	22	1 - 36	1	12	2 - 6	1357	8 - 40	2754
3	185	90	108	2 - 3	670	6	2 - 9	1256	8 - 41	3542
4	112	54	1	2 - 4	280	7	2 - 10	1040	8 - 42	3252
5	194	94	92	2 - 8	1	12	2 - 13	1840	8 - 43	2847
6	61	30	14	2 - 12	1820	7	2 - 14	1987	8 - 44	2696
7	152	73	216	3 - 5	570	6	2 - 15	2123	8 - 45	2510
8	158	76	143	4 - 35	380	9	2 - 16	1948	8 - 46	2148
9	282	136	317	5 - 17	570	6	2 - 17	1541	8 - 47	1900
10	193	94	153	6 - 7	1300	6	2 - 18	2507	8 - 48	1781
11	165	54	39	6 - 38	759	9	2 - 19	2437	8 - 49	1725
12	148	72	1	7 - 8	1421	6	2 - 20	3102	8 - 51	3405
13	91	44	47	7 - 9	610	7	2 - 21	2885	9 - 23	1840
14	119	57	1	8 - 11	360	7	2 - 28	3605	9 - 23	1693
15	311	150	186	8 - 24	570	8	2 - 29	3160	10 - 23	1981
16	314	152	295	8 - 27	570	8	2 - 30	3950	10 - 36	1890
17	333	161	245	9 - 10	250	7	2 - 31	3560	13 - 23	1496
18	351	170	397	10 - 11	340	7	2 - 32	3855	13 - 36	1189
19	236	114	167	12 - 13	320	7	2 - 33	3769	14 - 23	1822
20	297	144	351	13 - 14	380	7	2 - 34	2437	14 - 36	1388
21	253	122	264	14 - 15	150	7	2 - 35	372	15 - 36	1978
22	355	172	9	15 - 16	800	7	2 - 38	372	15 - 36	1496
23	492	238	208	15 - 18	510	7	2 - 40	4483	16 - 23	2726
24	152	74	34	16 - 17	570	7	2 - 41	5283	16 - 36	2277
25	156	75	1	18 - 19	280	7	2 - 42	4969	17 - 23	3063
26	186	90	41	19 - 34	510	7	2 - 43	4632	17 - 36	2704
27	310	150	211	20 - 21	300	7	2 - 44	4487	18 - 36	1887
28	292	141	4	20 - 34	510	7	2 - 45	4291	18 - 36	1321
29	11	6	1	21 - 22	530	7	2 - 46	3972	19 - 23	1524
30	230	111	8	22 - 28	955	8	2 - 47	3726	19 - 36	983
31	136	66	1	22 - 36	313	9	2 - 48	3593	20 - 36	1563
32	287	139	232	23 - 25	400	7	2 - 49	3546	20 - 36	920
33	298	144	186	23 - 26	350	7	2 - 51	5159	21 - 23	1219
34	174	84	167	23 - 27	350	7	5 - 8	1919	21 - 36	583
35	180	87	1	23 - 36	1	12	5 - 23	3038	23 - 28	1951
36	0	0	0	23 - 39	590	5	5 - 36	2718	23 - 29	2217
37	806	500	1	24 - 25	365	7	6 - 8	3086	23 - 31	2029
38	156	75	1	26 - 31	785	7	6 - 23	4250	23 - 32	1194
39	259	126	134	28 - 29	465	8	6 - 36	4112	23 - 33	1515
40	281	136	76	29 - 30	740	8	8 - 9	671	23 - 35	3061
41	310	150	69	30 - 33	685	8	8 - 10	819	23 - 34	1524
42	217	105	2	31 - 32	272	7	8 - 13	530	23 - 35	3061
43	153	74	34	32 - 33	671	7	8 - 14	974	23 - 38	3061
44	137	66	1	35 - 38	426	9	8 - 15	1190	23 - 41	2374
45	259	126	62	36 - 37	94	9	8 - 16	1772	23 - 42	2092
46	261	126	121	36 - 50	450	7	8 - 17	1977	23 - 43	1678
47	226	110	50	39 - 40	640	7	8 - 18	1342	23 - 44	1528
48	269	130	139	40 - 51	1251	8	8 - 19	1036	23 - 45	1340
49	218	106	1	41 - 42	430	7	8 - 20	1557	23 - 46	1017
50	136	66	5	41 - 51	229	9	8 - 21	1231	23 - 47	778
51	240	116	53	42 - 43	387	7	8 - 28	2088	23 - 48	629
				43 - 44	130	7	8 - 29	1947	23 - 49	592
				44 - 45	520	6	8 - 30	2354	28 - 36	1328
				45 - 46	350	6	8 - 31	1731	29 - 36	1581
				46 - 47	233	6	8 - 32	2033	30 - 36	1456
				47 - 48	180	6	8 - 33	2039	31 - 36	632
				48 - 49	150	7	8 - 34	1036	32 - 36	928
				49 - 50	435	7	8 - 35	1900	33 - 36	1009
									34 - 36	983
									35 - 36	2945
									36 - 38	2945
									36 - 40	2006
									36 - 41	2721
									36 - 42	2489
									36 - 43	1973
									36 - 44	1818
									36 - 45	1672
									36 - 46	1204
									36 - 47	971
									36 - 48	936
									36 - 49	850
									36 - 51	2567

Table A.4: Network 3 Data