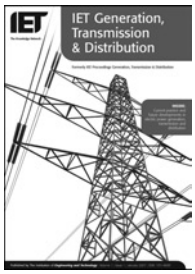
Published in IET Generation, Transmission & Distribution
Received on 23rd April 2007
Revised on 8th October 2007
doi: 10.1049/iet-gtd:20070174



ISSN 1751-8687

Power distribution network expansion scheduling using dynamic programming genetic algorithm

E.G. Carrano[5] *R.T.N. Cardoso*[1] *R.H.C. Takahashi*[2] *C.M. Fonseca*[3,4] *O.M. Neto*[1]

[1]*Department of Electrical Engineering, Universidade Federal de Minas Gerais, 31270-901 Belo Horizonte, MG, Brazil*
[2]*Department of Mathematics, Universidade Federal de Minas Gerais, 31270-901 Belo Horizonte, MG, Brazil*
[3]*Faculty of Science and Engineering, Universidade do Algarve, 8005-139 Faro, Portugal*
[4]*CEG-IST-Centre for Management Studies, Instituto Superior Técnico, Universidade Técnica de Lisboa, 2780-990 Porto Salvo, Portugal*
[5]*Centro Federal de Educação Tecnológica de Minas Gerais, 30480-000 Belo Horizonte, MG, Brazil*
E-mail: taka@mat.ufmg.br

Abstract: A genetic algorithm that is dedicated to the expansion planning of electric distribution systems is presented, with incremental expansion scheduling along a time horizon of several years and treated as a dynamic programming problem. Such a genetic algorithm (called dynamic programming genetic algorithm) is endowed with problem-specific crossover and mutation operators, dealing with the problem through a heuristic search in the space of dynamic programming variables. Numerical tests have shown that the proposed algorithm has found good solutions that considerably enhance the solutions found by non-dynamic programming methods. The algorithm has also shown to work for problem sizes that would be computationally infeasible for exact dynamic programming techniques.

1 Introduction

The electric distribution system is the most extensive part of the electrical system, and consequently, it is the mainly responsible for energy losses [1]. Therefore the use of optimisation techniques in the design of this subsystem can lead to significant economic gains, obtaining networks, which minimise the immediate costs (those related to installation and reconductoring) and further costs (costs related to energy losses and system maintenance) [2–4].

The fundamental working mechanism of the distribution system is the power flow that is established by the interaction between the power sources, the loads and the branch impedances, for the specific network topology, where the admissible topologies for distribution systems are the planar tree graphs. This means that its design becomes formulated as combinatorial optimisation problems involving non–linear constraints and non-linear objective functions. These characteristics of the problem preclude the employment of most of the traditional (deterministic) optimisation methods, and motivate the employment of the recent evolutionary computation techniques, that are able to deal with problems with such features. Most of the recent works dealing with the problem of power distribution network expansion employ algorithms belonging to this class. See for instance [3–6] and the references therein.

It is well known that distribution systems are in constant evolution, subject to load increasing in different places at different times, which leads to the

need of successive system expansions [5, 7, 8]. Rigorously speaking, this feature puts the network expansion design problem within the class of dynamic programming (DP) problems [9, 10]. This means that there is a sequence of design actions (the expansion in each step) that are interdependent (the design of the expansion in a stage affects the design problem statement for the subsequent stages). This class of problems is usually associated with a large computational complexity, since the variable space is composed of the number of decision variables in each stage multiplied by the number of stages (This is sometimes called the curse of dimensionality.). Such difficulties possibly explain why there are few works dealing with electric distribution expansion planning from the viewpoint of DP. For related problems that are treated via DP, see [7, 11, 12] for instance.

In this paper, the electric distribution expansion problem is modelled using the DP technique, with the system being expanded by incremental steps in time. Conceptually, this approach allows the optimal system expansion policy to be found, in the sense that the DP variable space includes such an optimum. A genetic algorithm (GA) with specialised mutation and crossover operators is proposed here, as a metaheuristic for dealing with such a high-complexity problem. The GA is built on the basis of the system representation that has been presented in [4, 6].

This paper is structured as follows: Section 2 discusses the formulation of expansion scheduling of power distribution systems under a DP framework, Section 3 shows the proposed approach, including the problem-specific GA that has been developed for dealing with network expansion dynamic programming problems (dynamic programming genetic algorithm (DP-GA), Section 4 presents two non-dynamic methods that could be employed in a multistage power distribution design, and that are used here for comparison, Section 5 presents the results achieved on a 100-bus distribution system. This section is subdivided into two parts: the first part presents a comparison of the results achieved by the DP-GA and the two non-dynamic approaches; the second one discusses the effect of time discretisation on final solution quality.

2 Expansion scheduling of power distribution systems

An electric distribution network can be represented as a dynamic system, in which the system variables are its branches. The system should be described as a linear discrete lumped time-invariant dynamic system, as shown in (1)

$$x[k] = x[k-1] + u[k] \quad \forall \quad k = 1, \ldots, N \tag{1}$$

in which: k is the index of the stage (discrete time), N is the total number of stages in which the time has been discretised, $x[k]$ is the network at stage k, and $u[k]$ is the increment to the network at stage k(New branch installation and reconductoring.).

The distribution network expansion scheduling is interpreted, in this case, as the system evolution in a finite time horizon. The DP problem is determined by defining:

a. the initial electric distribution system;

b. the design time horizon and the number of stages to be considered;

c. the load forecast for each stage within the design time horizon.

The proposed methodology starts by finding a target network: a network that is optimal for the load profile that is expected to exist at the end of the design time horizon. To find such a network, see for instance [4, 6]. The proposed DP-GA then searches for the optimal expansion scheduling, leading from the initial network to the final one.

Let $g_k(x[k], u[k])$ be the cost of the network at stage k (including the cost of the network increment $u[k]$). The total cost of the dynamic system evolution, in a general formulation, is

$$J = \sum_{k=1}^{N} g_k(x[k], u[k]) \tag{2}$$

In the specific case of this work, the objective function is modelled as follows

$$J_{\text{PDES}} = \sum_{k=1}^{N} \left[\text{inst cost}(u[k]) + \text{oper cost}(x[k])\right] \cdot (1 + \text{ir})^{-(k-1)} \tag{3}$$

in which J_{PDES} is the present cost of the whole system, inst cost($u[k]$) are the costs related to installations and reconductoring at stage k, oper cost($x[k]$) are the costs related to maintenance and losses at stage k and ir is the interest rate.

doi: 10.1049/iet-gtd:20070174

The optimal network expansion scheduling problem can be formulated as follows

$$\min_{u[1],\ldots,u[N]} J = \sum_{k=1}^{N} g_k(x[k], u[k]) \tag{4}$$

$$\text{subject to: } \begin{cases} x[k] = x[k-1] + u[k] \quad \forall \quad k = 1, \ldots, N \\ x[0] + \sum_{k=1}^{N} u[k] = x^* \end{cases} \tag{5}$$

in which: $x[0]$ is the initial network; x^* is the target network.

Finally, in addition to constraint (5), which is called C_1, five other constraints must be considered:

C_2: the network $x[k]$ must feed all nodes with load in all stages;

C_3: the network $x[k]$ must be radial (a tree structure) in all stages;

C_4: the immediate cost of stage k cannot be greater than the budget available for that stage

$$\text{inst cost}(u[k]) + \text{oper cost}(x[k]) \leq \text{budget}_k \quad \forall k = 1, \ldots, N$$

C_5: the voltage at load buses must comply with legal standards

$$V_{\min} \leq V_{\text{nd}k}^{\,j} \leq V_{\max} \quad \forall j = 1, 2, \ldots, n_{\text{nd}} \quad \text{and} \quad k = 1, \ldots, N$$

C_6: the current in each branch must not exceed the branch capacity

$$I_{\text{br}k}^{\,j} \leq I_{\max}{}^{j} \quad \forall j = 1, 2, \ldots, n_{\text{c}} \quad \text{and} \quad k = 1, \ldots, N$$

This procedure is illustrated in Fig. 1. Figs. 1*a* and 1*b* show the initial system and the target system, respectively. In this example, the number of stages has been set to four. This means that the target system should be reached not later than the 4th stage. Figs. 1*c*–1*f* show a possible solution for this case.

It should also be noted that the loads at the system nodes are, themselves, a dynamic system

$$P_{\text{L}}[k] = P_{\text{L}}[k-1] \cdot \Phi[k] + N_{\text{L}}[k] \tag{6}$$

where: $P_{\text{L}}[k]$ are the loads at stage k, $N_{\text{L}}[k]$ are the new loads connected to the system at stage k and $\Phi[k]$ is the expected increasing rate of the existing loads at stage k.

3 Dynamic programming genetic algorithm

Exact DP approaches are known to present high computational complexity, which renders infeasible their application to large problems. An alternative approach to DP problems that is receiving attention in recent years is the use of approximations that recursively approach the DP solution [10, 13, 14].

The use of metaheuristics, such as hybrid evolutionary / DP algorithm, has been considered recently [15–18]. However, the direct application of these algorithms to network expansion scheduling is not necessarily efficient, since they have serious problems when dealing with network structures [4, 19]. The main problem is caused by the inability of these algorithms to maintain the feasibility of their candidate solutions (the population, in the case of genetic algorithms); the use of the traditional evolutionary algorithm operators often results in infeasible networks.

In the remainder of this section, the DP-GA is presented; it is a GA that has been developed for dealing with network expansion scheduling problems. The proposed algorithm employs problem-specific crossover, mutation and fix operators, which guarantee the feasibility of the population.

Since DP-GA is a heuristic search method, there is no guarantee that a global optimal solution will be achieved. However, it is expected that at least nearly-optimal solutions will be achieved in reasonable computational time, even for problems which are too large to be solved by exact methods.

3.1 Problem specific GA

3.1.1 DP-GA solution encoding: In the proposed algorithm, each candidate solution (individual) is represented by a matrix $\boldsymbol{U}$, with dimension $N \times n_{\text{c}}$ ($n_{\text{c}} = |C_{\text{t}} \bigcup C_{\text{i}}|$ is the number of connections, N is the number of stages, C_{t} is the set of connections of the target system and C_{i} is the set of connections of the initial system).

Each cell $\boldsymbol{U}[i, j]$ represents the actions (new installations, reconductorings and branch removals) that must be performed in connection j at stage i. Therefore $\boldsymbol{U}[5, 4] = 2$ means that the connection 4 should receive an expansion of size 2 at stage 5. This expansion can be either the installation of a branch of index 2, if there is no previous connection, or the reconductoring of a previously existing branch, increasing its capacity index by 2 units. It is assumed that the branch types are ordered in increasing order of capacity, such as, for k branch types, the branch

doi: 10.1049/iet-gtd:20070174

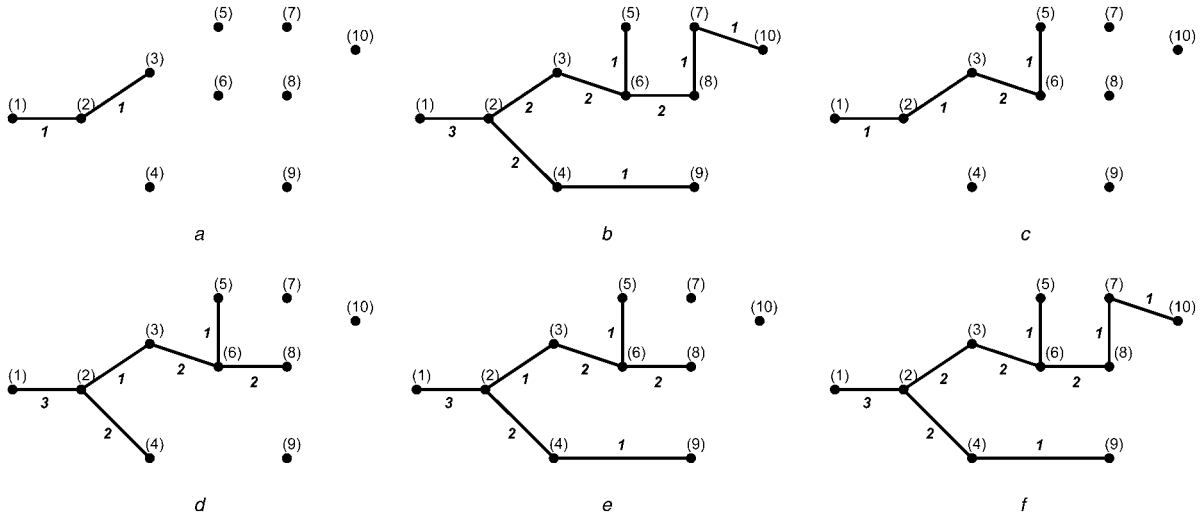


Figure 1 *Test system*

a Initial system
b Target system
c Stage 1
d Stage 2
e Stage 3
f Stage 4

with minimal capacity receives index 1, and the branch with maximal capacity receives index k.

For instance, the encoding for the test system of Fig. 1 is shown in Fig. 2.

3.1.2 DP-GA crossover operators: Two crossover operators are employed in the proposed algorithm (Fig. 3). Let P_1 and P_2 be the parent solutions and S_1 and S_2 be the offspring solutions in both cases.

Crossover operator 1: This operator chooses M lines of the two parents and switch them to obtain the offspring solutions. It can be implemented as follows:

1. $S_1 = P_1$ and $S_2 = P_2$;
2. Choose a set containing M stages ($\mathcal{S}$) randomly;
3. $S_1(\mathcal{S}, :) = P_2(\mathcal{S}, :)$ and $S_2(\mathcal{S}, :) = P_1(\mathcal{S}, :)$.

Crossover operator 2: This operator chooses L columns of the two parents and switches them to obtain the offspring solutions. It can be implemented as follows:

1. $S_1 = P_1$ and $S_2 = P_2$;
2. Choose a set containing L connections ($\mathcal{C}$) randomly;
3. $S_1(:, \mathcal{C}) = P_2(:, \mathcal{C})$ and $S_2(:, \mathcal{C}) = P_1(:, \mathcal{C})$.

3.1.3 DP-GA mutation operators: Four mutation operators are employed in the proposed algorithm. Let P be the parent solution and S be the offspring solution, in all cases.

Mutation operator 1: This operator changes the stage in which an action is intended to occur. It can be implemented as follows:

1. $S = P$;
2. Choose a connection c randomly;
3. Choose a stage k_1, among the stages in which $P(k_1, c) \neq 0$;
4. Choose another stage k_2 randomly;

$$
\begin{array}{c}
\begin{array}{cccccccccc} & [& U[i,1] & U[i,2] & U[i,3] & U[i,4] & U[i,5] & U[i,6] & U[i,7] & U[i,8] & U[i,9] &] \end{array} \\
\begin{array}{c}\text{from node} \\ \text{to node}\end{array} \begin{array}{cccccccccc} & & 1 & 2 & 2 & 3 & 4 & 5 & 6 & 7 & 7 & \\ & & 2 & 3 & 4 & 6 & 9 & 6 & 8 & 8 & 10 & \end{array} \\
\begin{bmatrix} U[1] \\ U[2] \\ U[3] \\ U[4] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}
\end{array}
$$

Figure 2 *Encoding for the example system of Fig. 1*

doi: 10.1049/iet-gtd:20070174

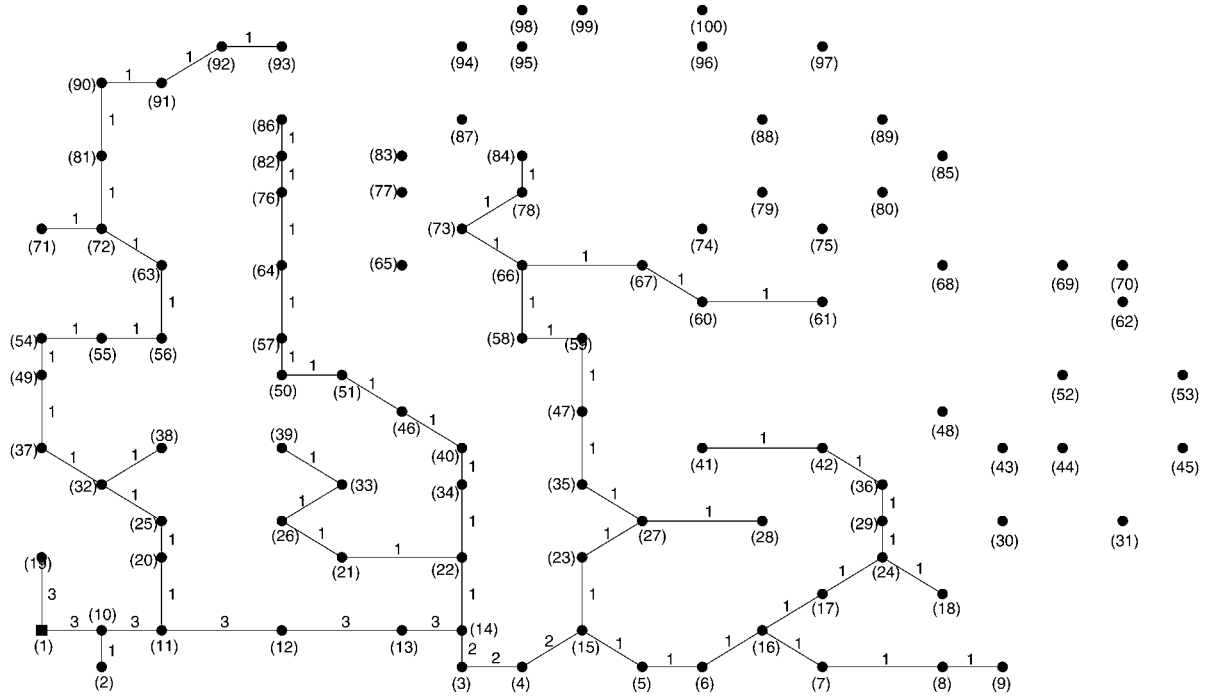


Figure 3 *Initial system (x[0])*

5. $S(k_1, c) = S(k_1, c) + P(k_2, c)$ and $S(k_2, c) = 0$.

Mutation operator 2: This operator switches two lines in a solution (at least one of those lines should be non-null). It can be implemented as follows:

1. $S = P$;

2. Choose a stage k_1, among the stages in which $P(k_1, c) \neq 0$ for at least one connection $c = 1, 2, \ldots, n_c$;

3. Choose an other stage k_2 randomly;

4. $S(k_1, j)_{j=1,2,\ldots,n_c} = P(k_2, j)_{j=1,2,\ldots,n_c}$ and $S(k_2, j)_{j=1,2,\ldots,n_c} = P(k_1, j)_{j=1,2,\ldots,n_c}$.

Mutation operator 3: This operator divides a big expansion into two expansions of smaller size. It can be implemented as follows:

1. $S = P$;

2. Choose a connection c and a stage k_1, among the stages in which $P(k_1, c) \geq 2$ randomly;

3. $S(k_1, c) = 0$;

4. Choose a stage k_2 randomly;

5. $S(k_2, c) = S(k_2, c) + a$ with $a \leq \left(\sum_{i=1}^{k} P(i, c) - \sum_{i=1}^{k} S(i, c)\right)$;

6. If $\sum_{i=1}^{k} S(i, c) < \sum_{i=1}^{k} P(i, c)$, then go to step 4.

Mutation operator 4: This operator joins two small expansions in an expansion of bigger size. It can be implemented as follows:

1. $S = P$;

2. Choose a connection c, among the connections in which $\sum_{i=1}^{k} P(c, i) \geq 2$ and $|\{P(c, 1), P(c, 2), \ldots, P(c, n_c)\}| \geq 2$, randomly;

3. Choose a stage k_1, among the stages in which $P(k_1, c) \neq 0$, randomly;

4. $S(c, k_1) = 0$;

5. Choose another stage k_2, among the stages in which $P(k_2, c) \neq 0$, randomly;

6. $S(c, k_2) = 0$;

7. Choose a stage k_3 randomly;

8. $S(c, k_3) = S(c, k_3) + P(c, k_1) + P(c, k_2)$.

3.1.4 DP-GA fix operators: Three fix operators are employed in the DP-GA, in order to guarantee the

doi: 10.1049/iet-gtd:20070174

feasibility of the new individuals that are generated in the initial population or after the application of some genetic operators of crossover and mutation.

Fix operator 1: After the employment of crossover operator 1, it is possible to obtain solutions, which do not comply with the dynamic constraint (C_1), i.e. which do not reach the target network until the last stage. This operator performs the following tasks for each offspring of crossover 1: (i) each column of the individual is checked; (ii) if the sum of actions leads to a result that is smaller than the target, an action of the exact size for correcting the difference is added to a random position of the column; (iii) if the sum of actions leads to a result that is greater than the target, the difference is removed from existing actions, chosen randomly in that column.

Fix operator 2: If a previously unloaded node *a* receives a load at stage *k* and it is not connected to the system at this point, then the action (or actions) which connects *a* to the system are moved to stage *k*. It forces the solution to comply with C_2. On the other hand, if a node *a* is connected to the system as a leaf (i.e. as a terminal connection) before receiving any load, then the action (or actions) which connects *a* to the system are moved to the stage in which it receives the load.

Fix operator 3: In problems where branch removals should be performed, it is possible to obtain intermediate

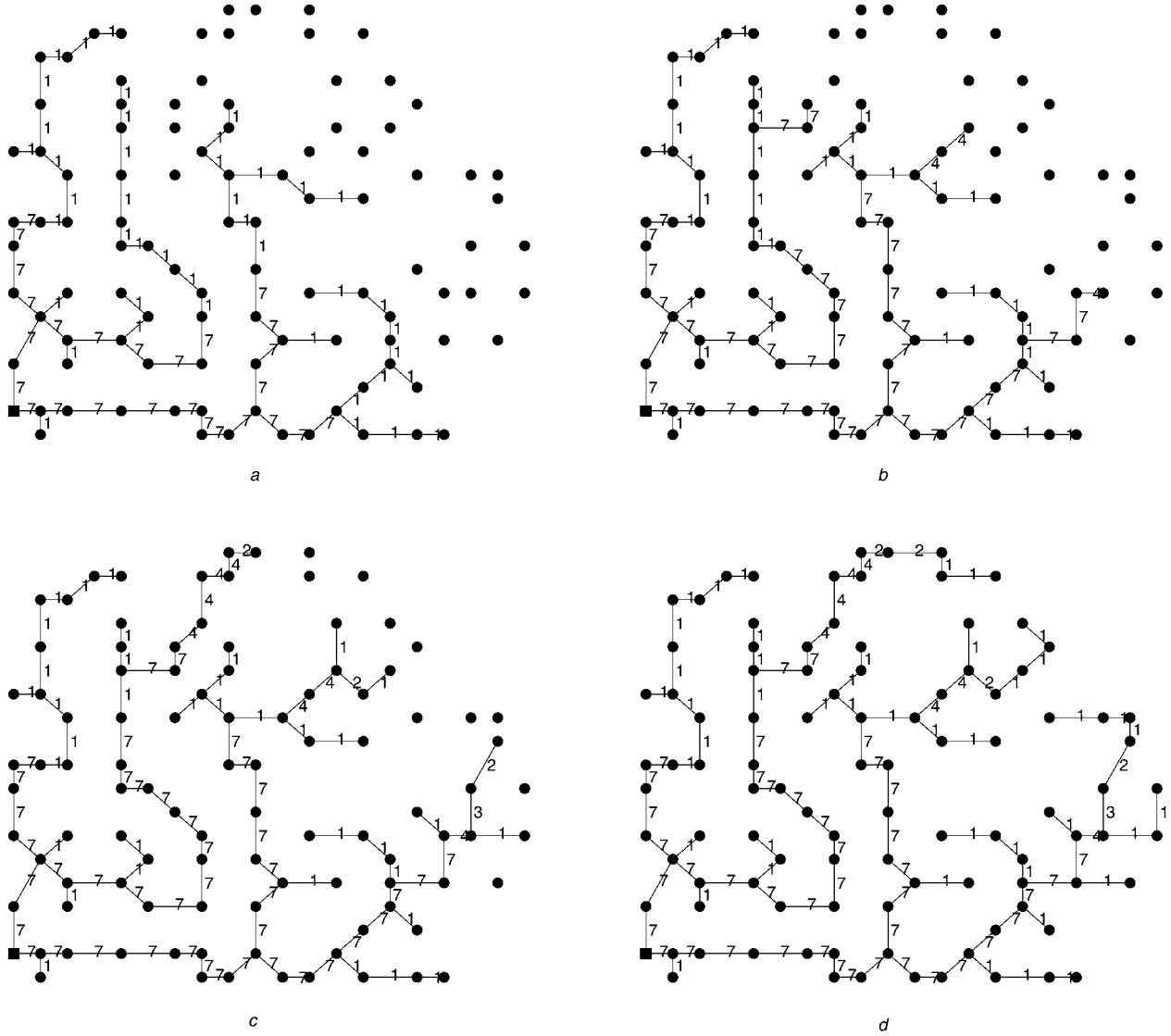


Figure 4 *DP-GA solution*

Accumulated cost of expansion, up to each stage, is presented in parenthesis
a Stage 01 ($1 098 198.70)
b Stage 03 ($2 167 864.20)
c Stage 07 ($3 627 300.00)
d Stage 10 ($4 604 416.15)

doi: 10.1049/iet-gtd:20070174

structures which either present loops or are disconnected (some nodes with load do not have a path to the root). It is possible to avoid these situations by moving the removal actions to stages in which they do not cause loops or disconnected networks. This fix operator ensures that the solutions comply with C_3.

3.1.5 DP-GA constraint handling: The constraints C_1, C_2 and C_3 are handled by fix, crossover and mutation operators. The constraints C_4, C_5 and C_6 are handled by a penalty method, as follows

$$J_i' = J_i + f_i \cdot J_i\left(1 + \frac{N_{\text{inf}}}{N}\right) \tag{7}$$

where: J_i' is the adjusted objective function value of solution i, J_i is the objective function value of solution i, f_i is 0 if the solution is feasible in all stages or 1 elsewhere, and N_{inf} is the number of stages in which the solution is infeasible.

3.1.6 DP-GA pseudo-code: The pseudo-code of DP-GA is presented in Fig. 5. The following notation is employed in that figure:

- M is the number of individuals in the population;
- N is the number of stages;
- network specific GA($P_L[N]$) represents a non-dynamic network optimisation algorithm (such as the ones presented in [3, 4, 6]) that generates the optimal target network for the load configuration $P_L[N]$ of stage N;
- random scheduling(target) returns a random feasible scheduling that leads to the target network, representing a potential solution for the DP problem;
- evaluate(P_i) returns the objective function value of solution (P_i);
- fitness(f) finds the fitness value of solutions;
- SUS(P, fit) makes the selection of solutions using stochastic universal sampling [20];
- uniform random returns a random value, following a uniform distribution probability.

DP-GA

1: $target \leftarrow$ network specific GA($P_L[N]$);
2: **for** $i \leftarrow 1$ **until** M **do**
3: $P_i \leftarrow$ random scheduling($target$); % generate individuals of initial population
4: $f_i \leftarrow$ evaluate(P_i);
5: **end for**
6: **while** $stop$ = false **do**
7: $fit \leftarrow$ fitness(f);
8: $P^{sel} \leftarrow$ SUS(P, fit); % selection
9: $P^{sel} \leftarrow$ shuffle(P^{sel}); % sorts the population randomly
10: **for** $i \leftarrow 1$ **until** $\frac{M}{2}$ **do**
11: $r_c \leftarrow$ uniform random; % decides if individuals i and $\frac{M}{2}+i$ will suffer crossover
12: **if** $r_c \leq p_{cros}$ **then**
13: $r_o \leftarrow$ uniform random; % chooses the crossover operator to be applied
14: **if** $r_o \leq 0.5$ **then**
15: $\left(P_i^{sel}, P_{\frac{M}{2}+i}^{sel}\right) \leftarrow$ crossover operator 1 $\left(P_i^{sel}, P_{\frac{M}{2}+i}^{sel}\right)$;
16: $P_i^{sel} \leftarrow$ fix operator 1 $(P_i^{sel}, target)$;
17: $P_{\frac{M}{2}+i}^{sel} \leftarrow$ fix operator 1 $\left(P_{\frac{M}{2}+i}^{sel}, target\right)$;
18: **else**
19: $\left(P_i^{sel}, P_{\frac{M}{2}+i}^{sel}\right) \leftarrow$ crossover operator 2 $\left(P_i^{sel}, P_{\frac{M}{2}+i}^{sel}\right)$;
20: **end if**
21: $P_i^{sel} \leftarrow$ fix operator 2 (P_i^{sel});
22: $P_i^{sel} \leftarrow$ fix operator 3 (P_i^{sel});
23: $P_{\frac{M}{2}+i}^{sel} \leftarrow$ fix operator 2 $\left(P_{\frac{M}{2}+i}^{sel}\right)$;
24: $P_{\frac{M}{2}+i}^{sel} \leftarrow$ fix operator 3 $\left(P_{\frac{M}{2}+i}^{sel}\right)$;
25: **end if**
26: **end for**
27: **for** $i \leftarrow 1$ **until** M **do**
28: $r_m \leftarrow$ uniform random; % decides if individual i will suffer mutation
29: **if** $r_m \leq p_{mut}$ **then**
30: $r_o \leftarrow$ uniform random; % chooses the mutation operator to be applied
31: **if** $r_o \leq 0.25$ **then**
32: $P_i^{sel} \leftarrow$ mutation operator 1 (P_i^{sel});
33: **else if** $r_o \leq 0.50$ **then**
34: $P_i^{sel} \leftarrow$ mutation operator 2 (P_i^{sel});
35: **else if** $r_o \leq 0.75$ **then**
36: $P_i^{sel} \leftarrow$ mutation operator 3 (P_i^{sel});
37: **else**
38: $P_i^{sel} \leftarrow$ mutation operator 4 (P_i^{sel});
39: **end if**
40: $P_i^{sel} \leftarrow$ fix operator 2 (P_i^{sel});
41: $P_i^{sel} \leftarrow$ fix operator 3 (P_i^{sel});
42: **end if**
43: **end for**
44: **for** $i \leftarrow 1$ **until** M **do**
45: $P_i \leftarrow P_i^{sel}$;
46: $f_i \leftarrow$ evaluate(P_i);
47: **end for**
48: **if** stop condition **then**
49: $stop \leftarrow$ true; % stop condition: may depend on f and on a maximum number of generations
50: **end if**
51: **end while**

Figure 5 *Dynamic programming genetic algorithm*

4 Non-dynamic methods

It is possible to develop non-dynamic methods which are able to deal with multistage power distribution planning. Two of those approaches are described here.

Incremental approach (INC)

1. Discretise the analysis time in N stages;
2. For each stage k (from 1 to N) do:

a. Update the loads using (6);

doi: 10.1049/iet-gtd:20070174

b. Obtain $x[k]$ connecting the new nodes (if required) and reconductoring the branches that cannot comply with the new power demand.

Non-dynamic genetic algorithm approach (NDGA)

1. Discretise the analysis time in N stages;

2. For each stage k (from 1 to N) do:

a. Update the loads using (6);

b. Obtain $x[k]$ running a GA specific for network optimisation for the new load conditions, taking the immediate costs as an optimisation criterion.

The non-dynamic genetic algorithm (NDGA) employed in this work is based on a mono-objective version of the GA described in [4], called here GANet, for performing the network static optimisation in each stage.

Both approaches described above can plan the multistage expansion of distribution systems. However, the design process is performed for each stage condition separately, regardless of the full

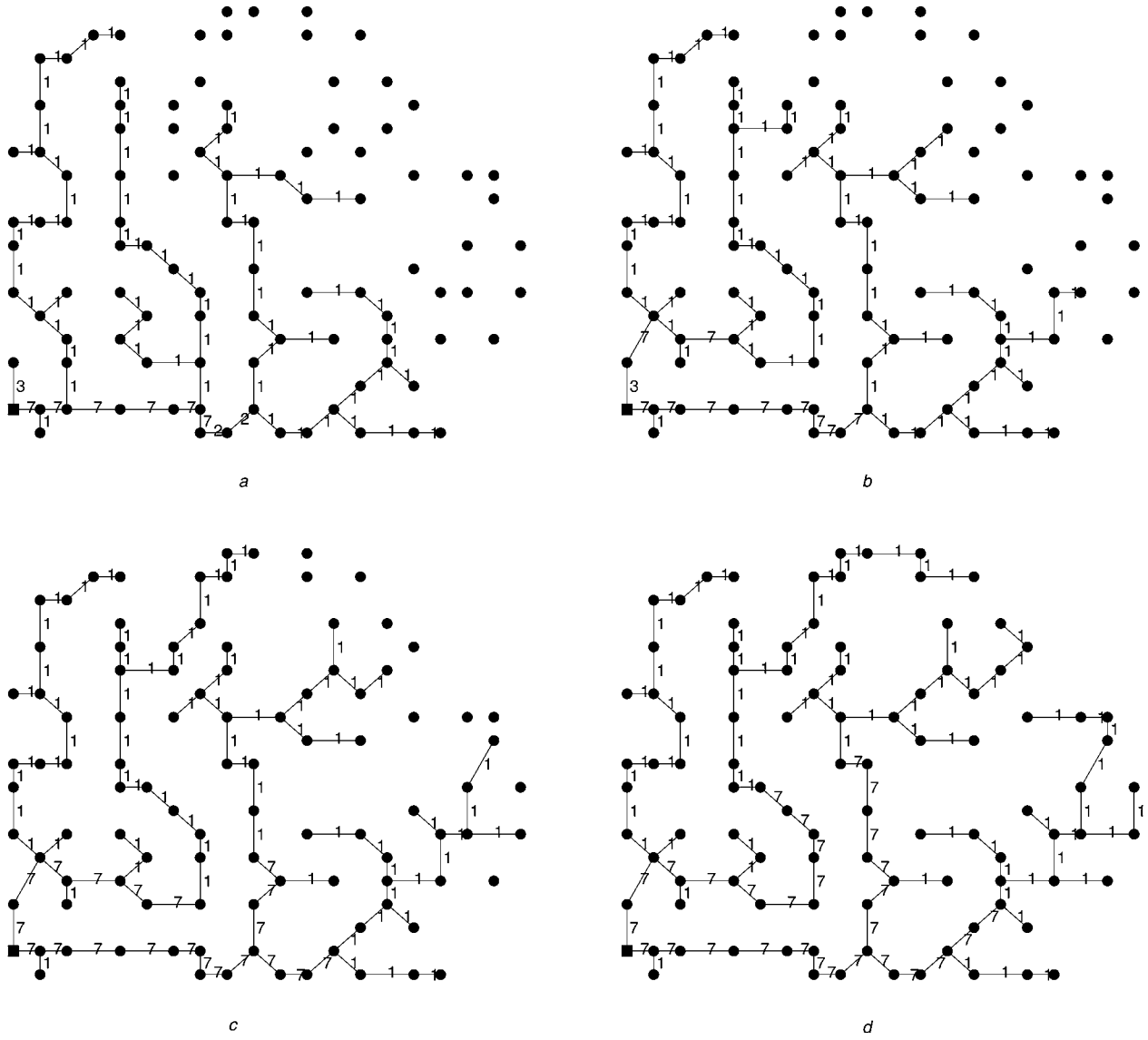


Figure 6 *INC solution*

Accumulated cost of expansion, up to each stage, is presented in parenthesis
a Stage 01 ($620 849.50)
b Stage 03 ($2 168 845.93)
c Stage 07 ($4 954 503.68)
d Stage 10 ($6 838 402.23)

doi: 10.1049/iet-gtd:20070174

context of the problem. It should be noticed that the methods can lead to different final systems, corresponding to different system costs.

5 Results

5.1 DP-GA against non-dynamic approaches

The three methods described in the previous sections have been employed here for the expansion of a 100-bus distribution system. The following design parameters have been set:

Design time: 10 years;

Pre-existing nodes: 70 nodes;

New nodes entering in the system: every year;

Expected annual load increasing of existing nodes: 5.00%;

Annual interest rate: 10.00%.

The system has some pre-existing connections as shown in Fig. 7. The target system for the DP-GA approach has been achieved using GANet (a mono-objective version of

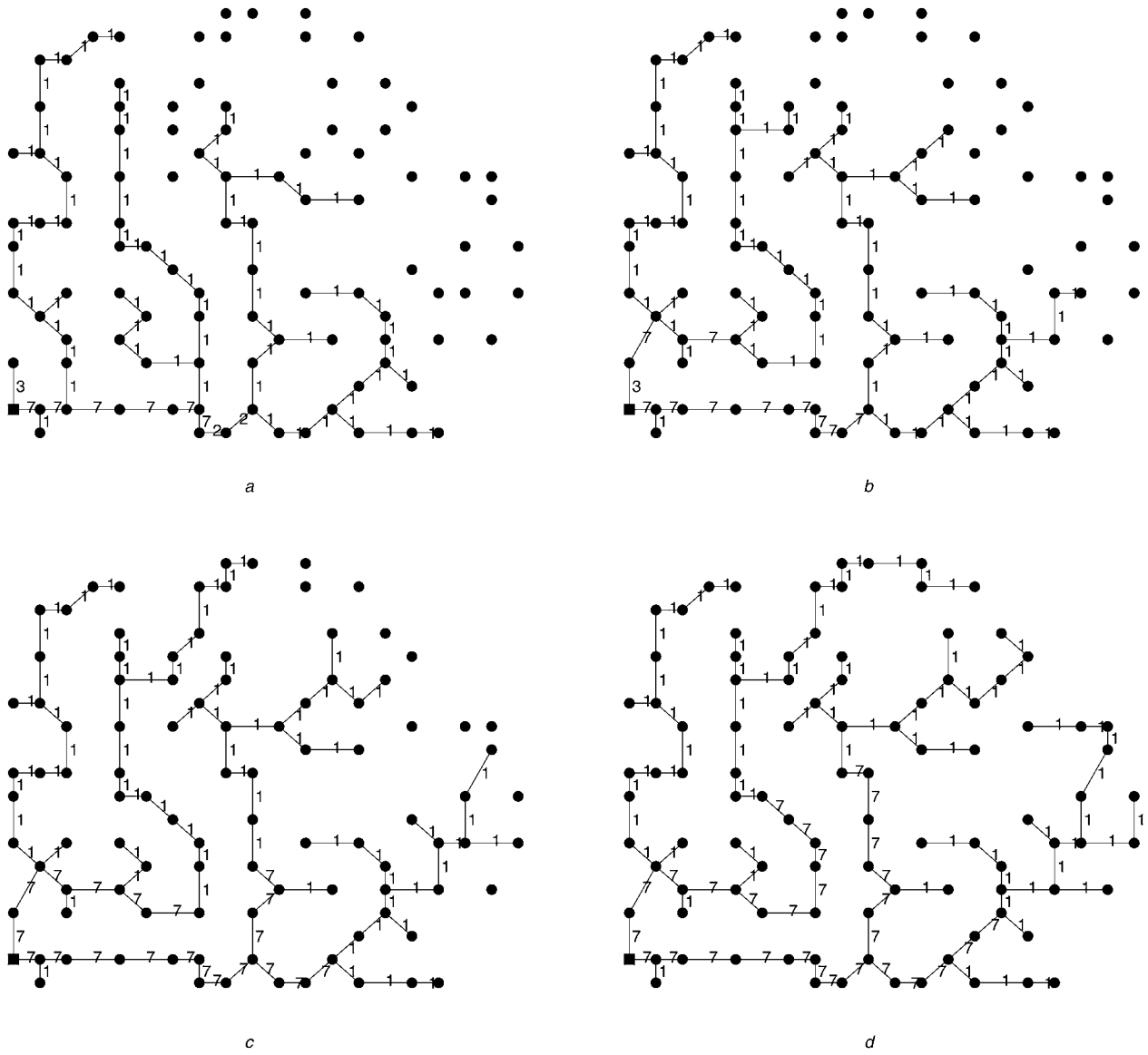


Figure 7 *NDGA solution*

Accumulated cost of expansion, up to each stage, is presented in parenthesis
a Stage 01 ($574 913.34)
b Stage 03 ($1 906 837.02)
c Stage 07 ($4 011 121.50)
d Stage 10 ($5 337 449.18)

doi: 10.1049/iet-gtd:20070174

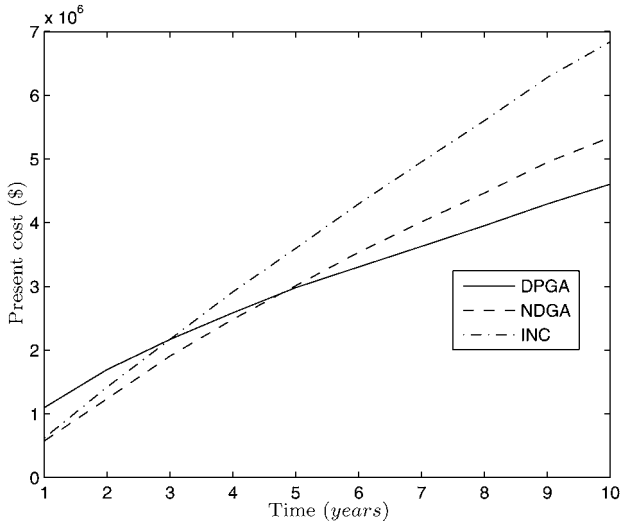
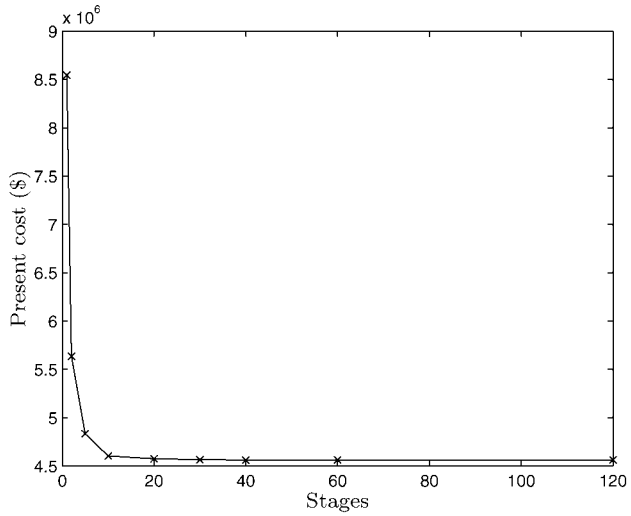


Figure 8 *Accumulated costs for the different network expansion scheduling approaches*

the GA proposed in [4]) – the same algorithm that is employed in each stage of the NDGA approach. The graphical representation of the solutions found by INC, NDGA and DP-GA approaches can be found in Figures 6, 7 and 4 respectively.

The accumulated present cost of the three approaches for each year in design time is shown in Fig. 8. It is noticeable that the DP-GA approach has the highest initial cost, which was expected, since the other approaches plan the system taking into account just the immediate horizon. After some stages, the accumulated costs of the other systems become higher than the DP-GA solution, that has considered the global conditions of the problem during the whole optimisation process. From Figs. 4, 6 and 7, one should notice that, in this specific example, the intermediate and final networks achieved by the three approaches have similar topologies but present different branch types. Notice that DP-GA tends to employ higher capacity conductors, to reduce the impact of losses and to avoid the reconductoring of new branches. The results concerning the accumulated costs support the recommendation for using DP-GA, since it has achieved a system which is about 15% cheaper than the NDGA approach and 35% cheaper than the INC approach, in the full time horizon under consideration.

The INC approach is the fastest algorithm among the three methods, since it has no optimisation process involved. On the other hand, the NDGA approach is considerably slower than the DP-GA approach. This can be explained by the fact that solving the whole network expansion scheduling problem via NDGA requires N algorithm runs of the GANet (the network specific GA must be executed once in each stage), whereas a complete design of the network using the DP approach, in turn, requires firstly one run of GANet to find the target and then one execution of DP-GA itself, to make the expansion planning. For the 100-node example problem with ten stages, both GANet and DP-GA have presented similar computational costs: the



Stages:	01	02	05	10	20
Cost ($):	8 545 547.18	5 637 242.35	4 834 213.87	4 604 416.15	4 575 654.04
Reduction:	-	34.0%	14.3%	04.8%	00.6%

Stages:	30	40	60	120
Cost ($):	4 566 171.55	4 561 447.70	4 561 447.70	4 561 447.70
Reduction:	00.2%	00.1%	00.0%	00.0%

Figure 9 *Time discretisation comparison*

doi: 10.1049/iet-gtd:20070174

GANet has spent 124 min, in average, and the DP-GA has spent 139 min, in average, on an Athlon64 with 2 GB RAM using MatLab 7 environment(Athlon64 and MatLab are registered trademarks of AMD and MathWorks, respectively.). The DP design cycle has required, in this example, about 247 min of computation time, whereas the non-dynamic GA design cycle has required about 1216 min.

5.2 Effects of time discretisation in solution quality

To evaluate the impact of the number of stages (or time discretisation) being considered, from the initial system to the final system, the DP-GA has been run for nine different numbers of stages: 1 (this corresponds to the steady-state approach), 2, 5, 10, 20, 30, 40, 60 and 120 stages.

The Fig. 9 shows the results achieved for these nine cases with different numbers of stages. As expected, the present cost becomes almost stable after ten stages. Also, the fast initial cost decay, when passing from a non-dynamic approach to the dynamic approach with increasing numbers of stages, strongly suggests that a DP approach should be performed always. Even if a very large system instance were to be designed, a DP design with some stages should be run, since the financial gain would probably be very significant.

The consistency of the results that have been obtained, leading to similar costs for rather different number of stages, seems to support the hypothesis that the proposed algorithm has reached, at least, a good sub-optimal solution of the problem. It should be noticed that an exact DP algorithm would not be able to solve this example problem within reasonable computational time.

6 Conclusion

This paper has presented a new approach for the design of electric distribution networks, using an approximate DP method solved via an evolutionary algorithm (a heuristic algorithm). A GA called DP-GA, using problem-specific fix, mutation and crossover operators has been presented for dealing with this problem.

The results obtained by this approach suggest that the algorithm can find good sub-optimal solutions for large problem instances, within reasonable computational times. The solutions that have been found, when compared with other solutions found via non-dynamic methods, represent significantly smaller costs. The consistency of the solutions obtained, for different numbers of stages, suggests that finding such good solutions is a systematic feature of the algorithm, rather than a feature reached 'by chance'.

The proposed methodology can be valuable for supporting the decisions on investment policy of energy companies. It permits a better allocation of limited financial resources, achieving solutions with smaller total costs than the ones gained with static design approaches.

7 Acknowledgments

The authors would like to thank the Brazilian agencies FAPEMIG, CAPES and CNPQ for the financial support.

8 References

[1] ĆURČIĆ S., STRBAC G., ZHANG X.-P.: 'Effect of losses in design of distribution circuits', *IEE Proc., Gener. Transm. Distrib.*, 2001, **148**, pp. 343–349

[2] RAMIREZ-ROSADO I., BERNAL-AGUSTÍN J.: 'Genetic algorithms applied to the design of large power distribution systems', *IEEE Trans. Power Syst.*, 1998, **13**, pp. 696–702

[3] CARRANO E.G., TAKAHASHI R.H.C., CARDOSO E.P., SALDANHA R.R., NETO O.M.: 'Optimal substation location and energy distribution network design using a hybrid GA-BFGS algorithm', *IEE Proc., Gener. Transm. Distrib.*, 2005, **152**, pp. 919–926

[4] CARRANO E.G., SOARES L.A.E., TAKAHASHI R.H.C., SALDANHA R.R., NETO O.M.: 'Electric distribution multiobjective network design usign a problem-specific genetic algorithm', *IEEE Trans. Power Deliv.*, 2006, **21**, pp. 995–1005

[5] CARVALHO P.M.S., FERREIRA L.A.F.M., LOBO F.G., BARRUNCHO L.M.F.: 'Optimal distribution network expansion planning under uncertainty by evolutionary decision convergence', *Electr. Power Energy Syst.*, 1998, **20**, pp. 125–129

[6] CARRANO E.G., GUIMARAES F.G., TAKAHASHI R.H.C., NETO O.M., CAMPELO F.: 'Electric distribution network expansion under load-evolution uncertainty using an immune system inspired algorithm', *IEEE Trans. Power Syst.*, 2007, **22**, pp. 851–861

[7] VAZIRI M., TOMSOVIC K., BOSE A.: 'A directed graph formulation for the multistage distribution expansion problem', *IEEE Trans. Power Deliv.*, 2004, **19**, pp. 1335–1341

[8] LEVITIN G., LISNIANSKI A.: 'Optimal multistage modernization of power system subject to reliability and capacity requirements', *Electr. Power Syst. Res.*, 1999, **50**, pp. 183–190

[9] BELLMAN R.E.: 'Dynamic programming' (Princeton University Press, 1957)

doi: 10.1049/iet-gtd:20070174

[10] BERTSEKAS D.P.: 'Dynamic programming – deterministic and stochastic models' (Prentice-Hall, 1985)

[11] YEHIA M.A., MATAR E.E., HOBEILA N.Y., AVEDIKIAN A.D.: 'A heuristic algorithm for electric distribution networks optimal feeder configuration using geographic information system', *IEEE Trans. Power Syst.*, 2002, **17**, pp. 1232–1237

[12] MONTEIRO C., RAMIREZ-ROSADO I.J., MIRANDA V., ZORZANO-SANTAMARIA P.J., GARCIA-GARRIDO E., FERNANDEZ-JIMENEZ L.A.: 'Gis spatial analysis applied to electric line routing optimization', *IEEE Trans. Power Deliv.*, 2005, **20**, pp. 934–942

[13] LINCOLN B., RANTZER A.: 'Relaxing dynamic programming', *IEEE Trans. Autom. Control*, 2006, **51**, pp. 1249–1260

[14] DE FARIAS D.P., VAN ROY B.: 'The linear programming approach to approximate dynamic programming', *Oper. Res.*, 2003, **51**, pp. 850–865

[15] MAK T.S.T., LAM K.P.: 'Equivalence-set genes partitioning using an evolucionary-DP approach', *IEEE Trans. Nano Biosciense*, 2005, **4**, pp. 295–300

[16] LAKSHMINARASIMMAN L., SUBRAMANIAN S.: 'Short-term scheduling of hydrothermal power system with cascaded reservoirs by using modified differential evolution', *IEE Proc., Gener. Transm. Distrib.*, 2006, **153**, pp. 693–700

[17] TSE S., LIANG Y., LEUNG K., LEE K., MOK T.S.: 'A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization', *IEEE Trans. Syst. Man Cybern. B*, 2007, **37**, pp. 84–91

[18] LIAO G.: 'Short-term thermal generation scheduling using improved immune algorithm', *Electr. Power Syst. Res.*, 2006, **76**, pp. 360–373

[19] SMITH D.K., WALTERS G.A.: 'An evolutionary approach for finding optimal trees in undirected networks', *Eur. J. Oper. Res.*, 2000, **120**, pp. 593–602

[20] BAKER J.E.: 'Reducing bias and inefficiency in the selection algorithm'. Proc. Int. Conf. Genetic Algorithms, Massachusetts, US, 1987, pp. 14–21

doi: 10.1049/iet-gtd:20070174