Hindawi
Journal of Engineering
Volume 2018, Article ID 1601239, 8 pages
https://doi.org/10.1155/2018/1601239

Research Article

Improved Genetic Algorithm for Distribution System Performance Analysis by Taking Advantage of Essential Spanning Trees

Yanzhu Ji,[1] Zhuoqun Shi,[2] and Robert M. O'Connell[1]

[1] *Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, MO, USA*
[2] *Black & Veatch, Kansas City, MO, USA*

Correspondence should be addressed to Robert M. O'Connell; oconnellr@missouri.edu

Received 26 January 2018; Accepted 30 April 2018; Published 3 June 2018

Academic Editor: Roberto Brighenti

Growing interest in the smart grid, increasing use of distributed generation, and classical distribution system reconfiguration (DSR) and restoration problems have led to the search for efficient distribution automation tools. One such tool, the improved Fast Nondominated Sorting Genetic Algorithm (FNSGA), not only is effective in finding system configurations that are optimal with respect to voltages, currents, and losses, but also considered parametric study to determine minimum values of N and Gen. In this paper, the essential spanning tree concept is expanded to improve the computational efficiency of the algorithm. Results of the study show that for relatively small test systems, optimum system configurations are obtained using values of N and Gen that require very small CPU times. In larger systems, optimum values of N and Gen requiring reasonable CPU times can also be found, provided that certain carefully chosen branches are removed from the pool of possibilities when producing the initial population in the algorithm. By using essential trees, the efficiency of the calculation is improved.

1. Introduction

Growing interest in the smart grid [1] and increasing use of distributed generation, along with classical problems of DSR, have led to the need for efficient and reliable power distribution system simulation tools. For system performance optimization with respect to multiple objectives, such as voltage profiles, system load balancing index, and power losses, various formulations of genetic algorithms (GA) [2–7] have been shown to be very effective.

One recently developed GA, the Fast Nondominated Sorting Genetic Algorithm (FNSGA) [8], converges relatively efficiently to reliable Pareto-optimal solution sets in the multiobjective DSR problem when applied to relatively small test systems. When applied to larger test systems, however, the original FNSGA is restricted by the use of the Newton-Raphson numerical load flow solution method [9] and the need for large initial population sizes (N) and numbers of iterations (Gen), which leads to burdensome memory requirements and long computation time. Computational efficiency is increasingly important in large (real) systems [2, 3, 10]. In the improved FNSGA discussed here, the Newton-Raphson load flow program is replaced with a revised version of the direct load flow method [11]. Also, a parametric study was conducted to determine minimum values of N and Gen that lead to reasonably repeatable configurations of a distribution system that are optimized for the multiple objectives of voltages, currents, and power losses [12]. In that study it was determined that computational efficiency could be improved by judiciously removing certain branches from the pool of possibilities when forming the initial population for the algorithm.

A method for further improving algorithm computational efficiency, the principal subject of this paper, involves the use of essential spanning trees. The method involves replacing the so-called M matrix [8] and its associated process in the algorithm with the random selection of b essential branches, where b is the number of fundamental loops in the system. Essential branches are those between essential nodes, where essential nodes are buses that connect at least

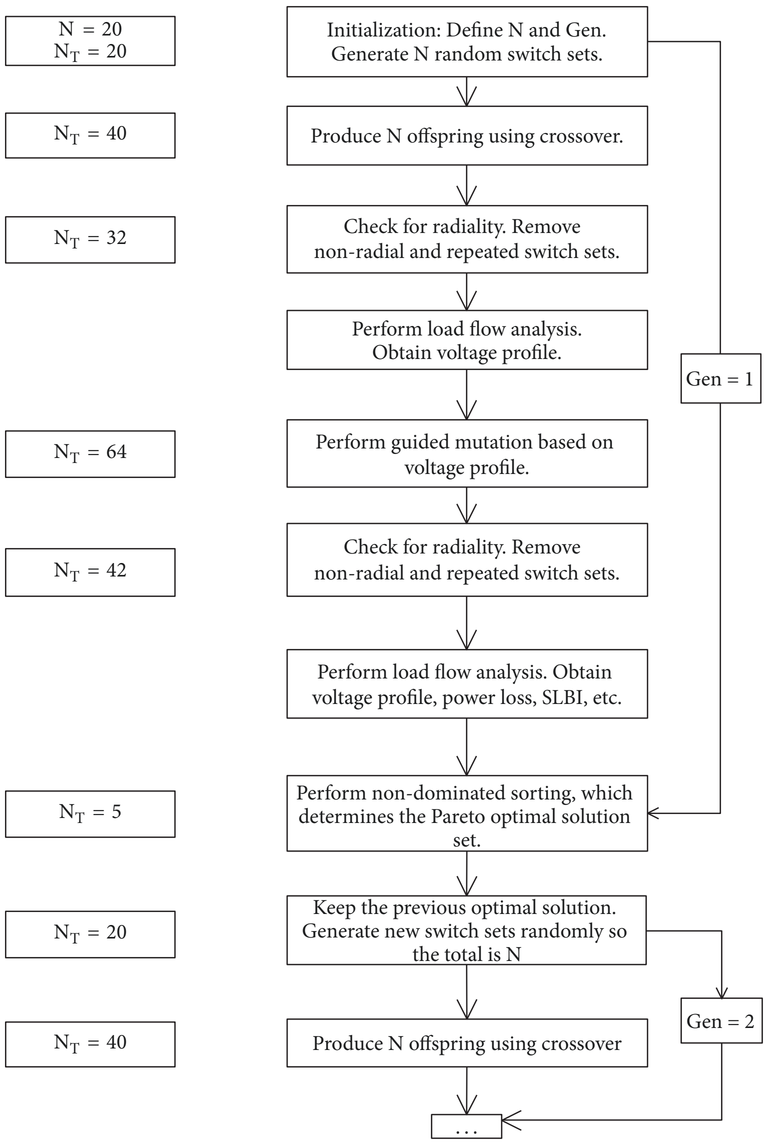
N = 20
N_T = 20
Initialization: Define N and Gen. Generate N random switch sets.
N_T = 40
Produce N offspring using crossover.
N_T = 32
Check for radiality. Remove non-radial and repeated switch sets.
Perform load flow analysis. Obtain voltage profile.
Gen = 1
N_T = 64
Perform guided mutation based on voltage profile.
N_T = 42
Check for radiality. Remove non-radial and repeated switch sets.
Perform load flow analysis. Obtain voltage profile, power loss, SLBI, etc.
N_T = 5
Perform non-dominated sorting, which determines the Pareto optimal solution set.
N_T = 20
Keep the previous optimal solution. Generate new switch sets randomly so the total is N
Gen = 2
N_T = 40
Produce N offspring using crossover
...

Figure 1: Illustration of the first two iterations (generations) of the genetic algorithm as applied to the 16-bus system shown in Figure 2.

three regular branches. The usual check for system radiality is then made based on the b chosen essential branches. If the system is radial, one regular branch within each essential branch is then chosen randomly to form a possible solution switch set. This switch set is then checked for repetition. Finally, this process is repeated N times to determine the initial population for application of the genetic algorithm.

In Section 2, the FNSGA is briefly described, with emphasis on the roles of the parameters N and Gen in the algorithm. In Section 3, the use of essential systems to further reduce CPU time is described. In Section 4, the results of parametric studies to optimize the values of N and Gen via full switch pool and essential tree approaches are presented. It is shown that the essential tree approach can result in significant computation time reductions as compared to the full switch pool approach. Section 5 concludes the paper with a summary.

2. Brief Description of the FNSGA

The FNSGA is described in detail in [8, 13]. Here, a brief description, illustrating its application to the above-mentioned 16-bus test system and emphasizing the roles of the two important parameters N and Gen, is provided instead. The FNSGA is illustrated in Figure 1 for application to the 16-bus test system shown in Figure 2. The example illustrates the first two generations, i.e., iterations of the main body program, with an initial population of 20 randomly generated sets of tie or open switches in the initially meshed test system. One such possible switch set consists of branches 7, 9, and 16, as in Figure 2.

The left column shows the current total population N_T of open switch sets under consideration during the process. During the first iteration, the initial population, which is equal to the size predefined by the user (20 in the example),

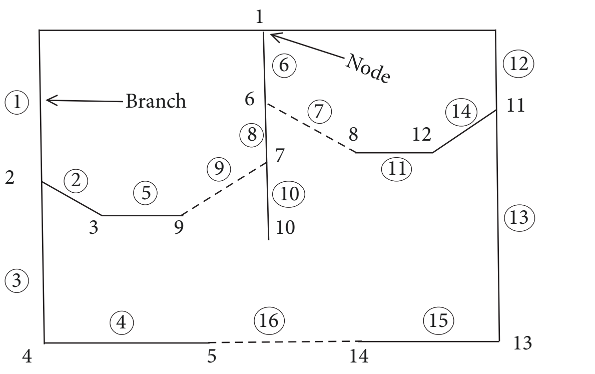
FIGURE 2: 16-bus test system consisting of three fundamental loops. Open (tie) switches in branches 7, 9, and 16 determine one possible radial system.

is generated randomly. In the next step, an equal population of offspring is generated from the initial population using the genetic crossover operator. The total population in the example now consists of 40 individuals. Due to the randomness involved in the first two steps, there might be repeated individuals among the 40, and some of them may not actually create a radial topology. Thus, the next step (the third box in the algorithm) is to detect and remove both the repeated individuals and those that do not produce radial topologies. At this point the population will consist of fewer than twice the original number, e.g., 32, as shown in Figure 1.

Next, load flow analysis is conducted on all 32 radial systems defined by the 32 valid switch sets, to generate voltage profiles in preparation for the next step, which is guided mutation. During guided mutation [8, 13], the branch that is connected to the lowest voltage bus is replaced randomly with one that is also connected to the lowest voltage bus, but not present in the current switch set combination. If there is not such an alternate, mutation for this bus is skipped. This mutation process is conducted on the voltage profiles of all 32 radial systems in question. Thus, the total population is now doubled, to 64 in the example. At this point, there might be repeated individuals among the 64, and some of them may not actually create a radial topology. Thus, it is again necessary to remove both the repeated individuals and those that do not produce radial topologies. For the sake of the example, it is assumed that 42 valid individuals remain, as shown in Figure 1.

After the guided mutation step, load flow analysis is conducted again, this time to determine all the performance objectives defined by the user. For this study, the defined multiple objectives were voltage profiles, total system power losses, and the system load balancing index (SLBI) [5]. These are the data needed for the nondominated sorting process [13], which produces the Pareto-optimal (nondominated) solution set. For the example, the population of the Pareto-optimal set is assumed to be 5, as shown in Figure 1.

At this point, one genetic generation is complete, i.e., one iteration of the user-specified number of generations Gen. The entire process is repeated for each remaining generation, with the only change being that the Pareto-optimal population set from each generation is kept and

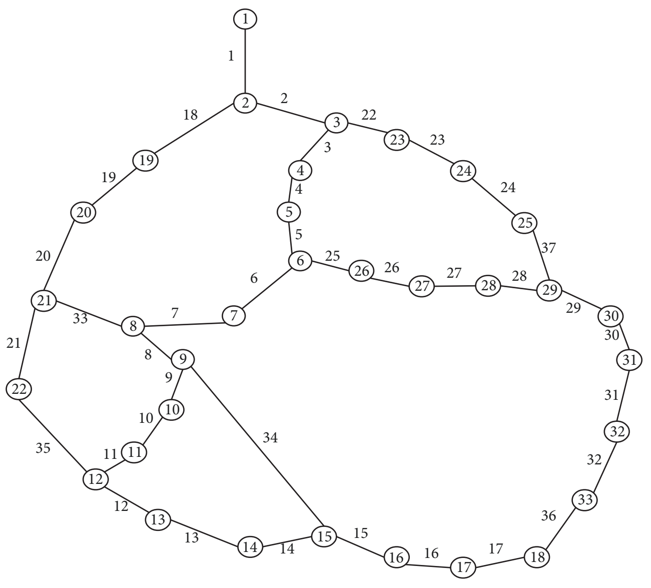
FIGURE 3: Meshed 32-bus test system consisting of five fundamental loops.

included as part of the population N at the start of the succeeding generation. Thus, the solution "evolves" to better and better results with each generation, and, thus, larger values of Gen lead to better results. Similarly, larger values of N allow the algorithm to consider more potential optimal solutions, which also leads to better results. Of course, larger values of N and Gen result in increased CPU times.

3. Applying Essential Spanning Tree to Reduce Switch Pool Size

As described previously, the FNSGA randomly selects one branch from each fundamental loop in the system to be made radial; it checks to see whether the selected set of open branches (a switch set) defines a radial system, and if so, whether it has been selected already; it then repeats the process N times. The N sets of open branches then evolve to better solutions as the algorithm proceeds. If N is very small in comparison with the number of possibilities, a very large number of generations will be required in order for the algorithm to consider a sufficient number of switch sets to produce repeatable results. This will result in the consumption of excessive CPU time. Thus, it should be possible to decrease CPU time by reducing the number of switches to be considered in forming the initial populations, i.e., the size of the eligible switch pool.

Here, we use the essential spanning tree concept [14] to simplify the original graph describing the system in question to an essential graph. For example, the complete 32-bus test system and its corresponding essential graph or mesh are shown in Figures 3 and 4, respectively. To make use of the essential mesh, the random switch sets are created using it, rather than the complete system. The FNSGA shown in Figure 1 is then accordingly modified, as shown in Figure 5. Thus, following initialization, radiality of the essential tree in

Table 1: Results summary from 25 runs of the FNSGA on the 16-bus test system with various values for N with Gen = 5.

Gen	N	Average CPU time (seconds)	Total no. of Solutions	Number of unique solutions	Maximum occurrence rate (percent)
5	2	0.048	49	17	24
5	4	0.073	66	18	48
5	8	0.146	97	16	44
5	12	0.203	110	12	80
5	16	0.265	118	11	92
5	20	0.330	118	9	96
5	24	0.369	118	10	96
5	28	0.436	118	7	100
5	30	0.448	122	8	100

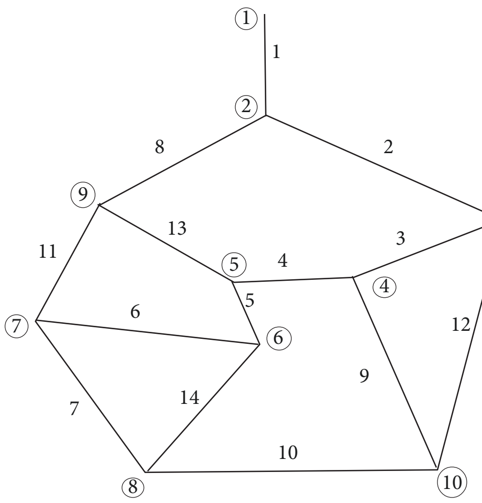

Figure 4: Essential graph of the 32-bus test system shown in Figure 3.

question is determined, followed by random selection of one switch from each branch in the radial essential tree. Load flow analysis is then performed on the associated system to calculate voltage profile, power loss, and SLBI information. Next, guided mutation based on the voltage profile is performed and repeated branches are deleted. Because mutation may cause the radiality situation to change, new essential branch sets are found, and the above steps are repeated. At this point nondominated sorting to obtain the Pareto-optimal solution sets is performed, and one iteration of the entire algorithm is complete.

After the algorithm has been repeated the specified number of times (Gen), a reduced switch pool will have been produced, which consists of only the branches in selected essential tree branches. The FNSGA algorithm can then be applied to this reduced size switch pool, rather than either the full 37 branches in the system or the 22 branches described in [12].

4. Parametric Study to Reduce CPU Time

In order to optimize N and Gen by finding minimum values of the two (for CPU time purposes) that lead to highly repeatable Pareto-optimal configurations, the above-described FNSGA was used to determine Pareto-optimal radial configurations of the initially meshed (no initially open switches) 16- and 32-bus test systems. N and Gen were defined as variable parameters. The algorithm was programmed in MATLAB and implemented on a 2.7 GHz, 8 GB RAM laptop personal computer.

4.1. 16-Bus Test System

4.1. 16-Bus Test System. To optimize the initial population size N, Table 1 shows a summary of the results of applying the FNSGA to the 16-bus system shown in Figure 2. All switches, including numbers 7, 9, and 16, were initially closed, so as not to bias any one possible solution. Bus and branch input data describing the system were taken from [15]. The number of generations was fixed at 5 and the initial population size was varied from 2 to 30. For each (Gen, N) pair the algorithm was run 25 times. Data in the table include, for each (Gen, N) pair, the average CPU time per run, the total number of Pareto-optimal solutions in the 25 runs, the number of unique solutions, and the occurrence rate of the most frequently occurring solution in the 25 runs (the maximum occurrence rate). Thus, for example, in the case of N = 16, the average CPU time was 0.265 seconds, eleven unique three-open-switch Pareto-optimal solutions appeared 118 times, and at least one such solution appeared 92 percent of the time or 23 times in the 25 runs of the algorithm.

The data in Table 1 show that CPU time is acceptably small, and it increases linearly with N, at least over the range of N considered in the study. Also, the maximum occurrence rate saturates at well over 90 percent as N increases beyond approximately 20, shown by the plotted data in Figure 6. Using the criterion that N is sufficiently large when at least one solution appears at least 90 percent of the time, the fourth order curve fitted to the data in Figure 6 shows that an initial population size of approximately 20 is sufficient to produce repeatable solution sets (with Gen = 5) in the 16-bus test system.

To optimize the number of generations Gen, Table 2 shows results (from 25 runs of the algorithm) of varying Gen with N fixed at 20, the approximately optimum value found via Figure 6. The data show that CPU time increases linearly with Gen, at least over the range of Gen considered in the study. It also shows that the maximum occurrence rate

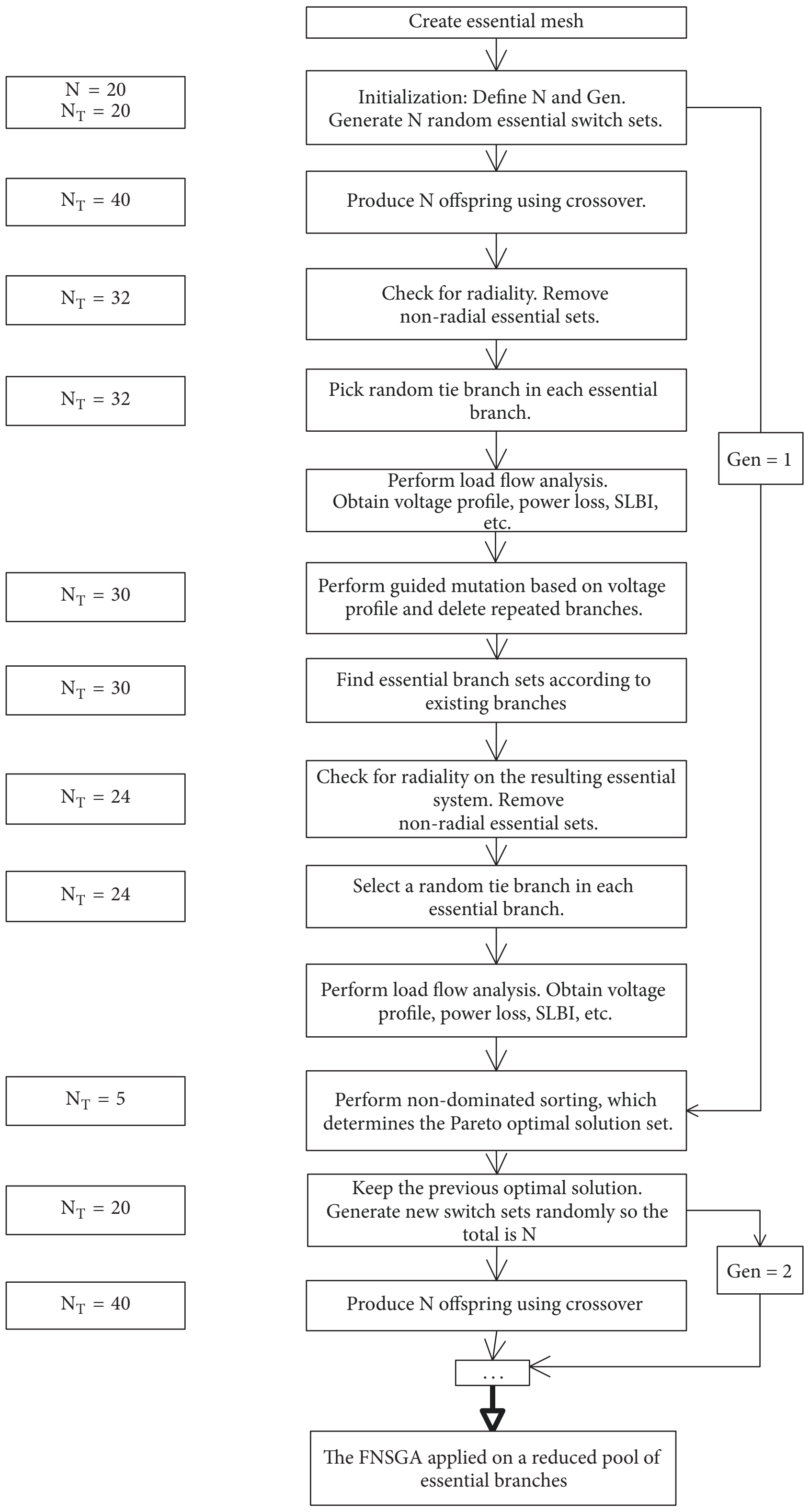
Create essential mesh
N = 20
N_T = 20
Initialization: Define N and Gen. Generate N random essential switch sets.
N_T = 40
Produce N offspring using crossover.
N_T = 32
Check for radiality. Remove non-radial essential sets.
N_T = 32
Pick random tie branch in each essential branch.
Gen = 1
Perform load flow analysis. Obtain voltage profile, power loss, SLBI, etc.
N_T = 30
Perform guided mutation based on voltage profile and delete repeated branches.
N_T = 30
Find essential branch sets according to existing branches
N_T = 24
Check for radiality on the resulting essential system. Remove non-radial essential sets.
N_T = 24
Select a random tie branch in each essential branch.
Perform load flow analysis. Obtain voltage profile, power loss, SLBI, etc.
N_T = 5
Perform non-dominated sorting, which determines the Pareto optimal solution set.
N_T = 20
Keep the previous optimal solution. Generate new switch sets randomly so the total is N
Gen = 2
N_T = 40
Produce N offspring using crossover
...
The FNSGA applied on a reduced pool of essential branches

Figure 5: Illustration of the first two iterations (generations) of the application of the FNSGA using the essential spanning tree concept.

Table 2: Results summary from 25 runs of the FNSGA on the 16-bus test system with various values for Gen with N = 20.

N	Gen	Average CPU time (seconds)	Total no. of Solutions	Number of unique solutions	Maximum occurrence rate (percent)
20	1	0.079	81	22	36
20	2	0.146	96	14	68
20	3	0.203	106	18	76
20	4	0.263	111	9	80
20	5	0.318	118	9	92
20	6	0.359	119	9	96
20	7	0.400	118	11	92
20	8	0.435	122	8	100
20	9	0.442	121	9	100
20	10	0.500	122	5	100

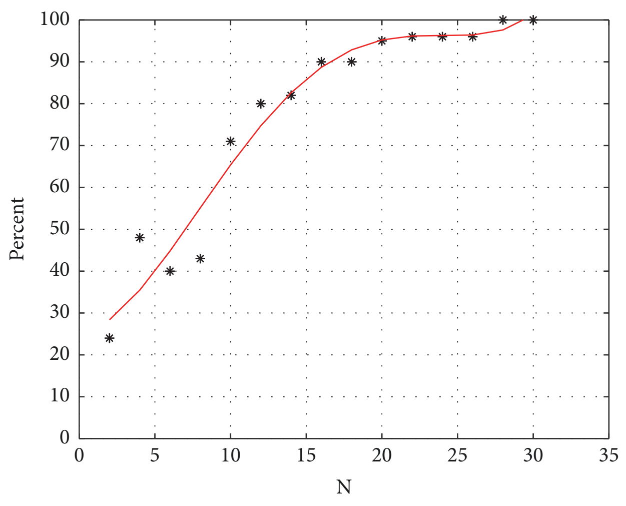

Figure 6: Maximum occurrence rate versus N, with Gen = 5, from Table 1.

saturates at well over 90 percent as Gen increases beyond approximately five; i.e., five generations are sufficient to produce repeatable solution sets (with N = 20) in the 16-bus test system.

4.2. 32-Bus Test System. The 16-bus parametric study showed that the maximum occurrence rate, which indicates the achievement of repeatable Pareto-optimal solutions, saturates above the 90 percent level for reasonable values of N and Gen, approximately 20 and 5, respectively. Also, the average CPU time for these two parameters is only approximately 0.32 seconds, as shown in Tables 1 and 2. Thus, there is no considerable computational burden with the 16-bus test system.

A similar parametric study to that done with the 16-bus test system was conducted on the 32-bus system shown in Figure 3. Bus and branch input data describing this system were taken from [16]. Note that since the system consists of five fundamental loops, Pareto-optimal solutions consist of five open switches, rather than three, as in the 16-bus case.

Tables 3 and 4 summarize the results of 25 runs of the FNSGA on the 32-bus system with N varied and Gen fixed at 16, and with Gen varied and N fixed at 160, respectively. The data in the tables show that CPU times are relatively long and increase approximately linearly over the range of values covered by N and Gen and that the maximum occurrence rates are no higher than approximately 60 percent in any case considered. Thus, the values of N and Gen required for saturation of the maximum occurrence rate (and thus repeatable Pareto-optimal solution sets) are much larger than any of the values considered, and the corresponding CPU time will be impractically large. A way of dealing with this is discussed in the next section.

4.3. Use of Essential Spanning Trees. To examine the computation efficiency when using the spanning tree approach described previously, CPU times were compared for application of the algorithm to the 32-bus test system using both the full switch pool and that determined with the spanning tree approach. As shown in Table 5, essential branch set (4, 6, 7, 9, and 10) clearly has the maximum occurrence rate, so this set was used in the comparison study. (If multiple essential branch sets have similar occurrence rates, a combination of them can also be used.)

Having determined a reduced switch pool via the above-identified essential branch set, the FNSGA was simulated 25 times using a Mac Pro with a dual-core intel Core i5 2.3 GHz CPU. The results are shown in Table 6 and compared with corresponding results using the full 37-switch pool. The data show that, as the (N, Gen) values increase from (30, 5) to (40, 16) to (80, 16), the maximum occurrence rates improve from 12 percent to 64 percent, from 36 percent to 96 percent, and from 52 percent to 100 percent, respectively.

In the full 37-switch pool case, a maximum occurrence rate of only 52 percent is obtained with 1.96 seconds of CPU time, but in the essential branch case, the desired 100 percent maximum occurrence rate is achieved with a very reasonable CPU time of 1.266 sec.

5. Conclusion

The paper shows that parametric studies of the initial population size and number of generations in the FNSGA can be used to find minimum values of the two that lead to highly repeatable Pareto-optimal configurations with reasonable

Table 3: Results summary from 25 runs of the FNSGA on the 32-bus test system with various values for N with Gen = 16.

Gen	N	Average CPU time (seconds)	Total no. of Solutions	Number of unique solutions	Maximum occurrence rate (percent)
16	30	2.75	113	58	24
16	50	3.94	117	52	32
16	70	6.07	123	51	36
16	90	8.33	133	49	40
16	110	10.6	147	44	40
16	120	12.8	171	60	44
16	140	13.3	155	46	48
16	160	14.4	162	48	44
16	170	16.4	154	36	52
16	180	15.1	164	38	56
16	190	18.3	161	33	60

Table 4: Results summary from 25 runs of the FNSGA on the 32-bus test system with various values for Gen with N = 160.

N	Gen	Average CPU time (seconds)	Total no. of Solutions	Number of unique solutions	Maximum occurrence rate (percent)
160	5	6.64	137	78	28
160	10	12.6	165	54	40
160	15	14.9	152	43	44
160	20	15.6	145	34	52
160	25	18.3	170	43	52
160	30	19.1	172	38	56

Table 5: Occurrence rate of the essential switch sets.

Essential switch sets					Occurrence in 15 runs
4	5	9	7	10	2
4	5	9	14	10	9
4	6	9	7	10	26
4	6	9	14	10	7
4	6	12	7	10	3
4	6	12	14	10	7
4	11	9	7	10	2
13	5	9	14	10	5
13	6	9	7	10	3
13	6	9	14	10	6

Table 6: Comparing average CPU time and maximum occurrence rates in 25 runs obtained with the full 37-switch pool and the essential switches for the 32-bus test system.

Gen	N	Full (37) switch pool		Reduced switch pool from selected essential switches	
		Average CPU time (seconds)	Maximum Occurrence rate (percent)	Average CPU time (seconds)	Max Occurrence rate (percent)
5	30	0.314	12	0.192	64
16	40	0.958	36	0.652	96
16	80	1.96	52	1.27	100

CPU times. As the size of the system increases, the number of eligible switches to be used in the algorithm may have to be limited. This can be done with some preliminary simulations to determine and remove from the eligible-switch pool those switches that never appear in preliminary simulations. The idea could be extended to larger systems by reducing the size of the eligible switch pool even further by also removing those switches that appear rarely, e.g., only one to four percent of the time in the preliminary simulations.

A more radical way to improve the computational efficiency of the algorithm is to take advantage of essential spanning trees [16]. This is accomplished by replacing the so-called M matrix and its associated process in the FNSGA [8] with the random selection of b essential branches, where b is the number of fundamental loops in the system. The usual check for radiality is then made based on the b chosen essential branches and the process continued as usual. The results of parametric studies show the improvement in computational efficiency due to the use of the spanning tree approach. Continuation of this work should focus on ways to improve the algorithm and further decrease CPU times in larger, i.e., realistic, test systems.

Data Availability

Underlying data is contained in published articles, which are cited in the manuscript.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] R. Bayindir, I. Colak, G. Fulli, and K. Demirtas, "Smart grid technologies and applications," *Renewable & Sustainable Energy Reviews*, vol. 66, pp. 499–516, 2016.

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

[3] Y. Kumar, B. Das, and J. Sharma, "Service restoration in distribution system using non-dominated sorting genetic algorithm," *Electric Power Systems Research*, vol. 76, no. 9-10, pp. 768–777, 2006.

[4] Y. Hong and S. Ho, "Determination of Network Configuration Considering Multiobjective in Distribution Systems Using Genetic Algorithms," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1062–1069, 2005.

[5] E. G. Carrano, L. A. E. Soares, R. H. C. Takahashi, R. R. Saldanha, and O. M. Neto, "Electric distribution network multiobjective design using a problem-specific genetic algorithm," *IEEE Transactions on Power Delivery*, vol. 21, no. 2, pp. 995–1005, 2006.

[6] D. Singh, D. Singh, and K. S. Verma, "GA based energy loss minimization approach for optimal sizing &amp; placement of distributed generation," *International Journal of Knowledge –based and Intelligent Engineering Systems*, vol. 112, no. 2, pp. 147–156, 2008.

[7] J. Torres-Jiménez, J. L. Guardado, F. Rivas, S. Maximov, and E. Melgoza, "Reconfiguration of power distribution systems using genetic algorithms and spanning trees," in *Proceedings of the 2010 7th IEEE Electronics, Robotics and Automotive Mechanics Conference, CERMA 2010*, pp. 779–784, mex, October 2010.

[8] A. M. Eldurssi and R. M. O'Connell, "A fast nondominated sorting guided genetic algorithm for multi-objective power distribution system reconfiguration problem," *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 593–601, 2015.

[9] W. Lin and J. Teng, "Phase-decoupled load flow method for radial and weakly-meshed distribution networks," *IEE Proceedings—Generation, Transmission & Distribution*, vol. 143, no. 1, p. 39, 1996.

[10] J. Zhang, X. Yuan, and Y. Yuan, "A novel genetic algorithm based on all spanning trees of undirected graph for distribution network reconfiguration," *Journal of Modern Power Systems and Clean Energy*, vol. 2, no. 2, pp. 143–149, 2014.

[11] J.-H. Teng, "A direct approach for distribution system load flow solutions," *IEEE Transactions on Power Delivery*, vol. 18, no. 3, pp. 882–887, 2003.

[12] Z. Shi, *Improved genetic algorithm for distribution system performance analysis , Masters Thesis [Master, thesis]*, University of Missouri, 2017.

[13] A. M. Eldurssi, *A Fast Non-dominated Sorting Guided Genetic Algorithm for Multi-Objective Power Distribution System Reconfiguration Problem [Ph.D. dissertation]*, Department of Electrical Engineering, University of Missouri, Columbia, MO, USA, 2015.

[14] K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms*, New York, NY, USA, Wiley, 2001.

[15] S. Civanlar, J. J. Grainger, H. Yin, and S. S. H. Lee, "Distribution feeder reconfiguration for loss reduction," *IEEE Transactions on Power Delivery*, vol. 3, no. 3, pp. 1217–1223, 1988.

[16] M. E. Baran and F. F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1401–1407, 1989.