4.3 Parameter Values

The algorithm developed here has a number of parameters that can be adjusted to improve the performance of the system. This section details the tests that were run to determine good values for the parameters.

The first important consideration is the component ranges that were used. Smaller component ranges will lead to faster convergence because a smaller range of values has to be searched. The component ranges were previously given in Table 3.3 on page 87.

Unless otherwise stated, each test was run 100 times to obtain statistics about the effect of each parameter. Individuals with one to six elements were considered. Each test was run for 40 generations and had a population size of 30000 for the pure genetic algorithm case and 3000 for the hybrid genetic algorithm case. The default genetic algorithm parameter values given in Table 3.1 on page 82, the default local optimiser values given in Table 3.5 on page 103, and the default component ranges given in Table 3.3 on page 87, were used.

The six individual length results are combined by normalising each length's parameters to its maximum value and adding the results. This result is then also normalised to its maximum value. This procedure is followed to ensure that large values, like those obtained with shorter individuals, do not dominate the results.

The minimum, median, maximum, mean, and standard deviation were determined for each test. The median is favoured over the mean as many of the results are heavily skewed towards good results because the algorithm usually obtains good results, and outliers can have a disproportionate effect on the mean. In most cases the difference between the mean and median is negligible, but there are some cases where the difference is significant. For example, running the hybrid algorithm on Problem 5 produced the cumulative distribution shown in Figure 4.4 for a two-element network. The median of the distribution is 0.194, but the mean is 0.237 despite the fact that 86% of the results are better than this value.

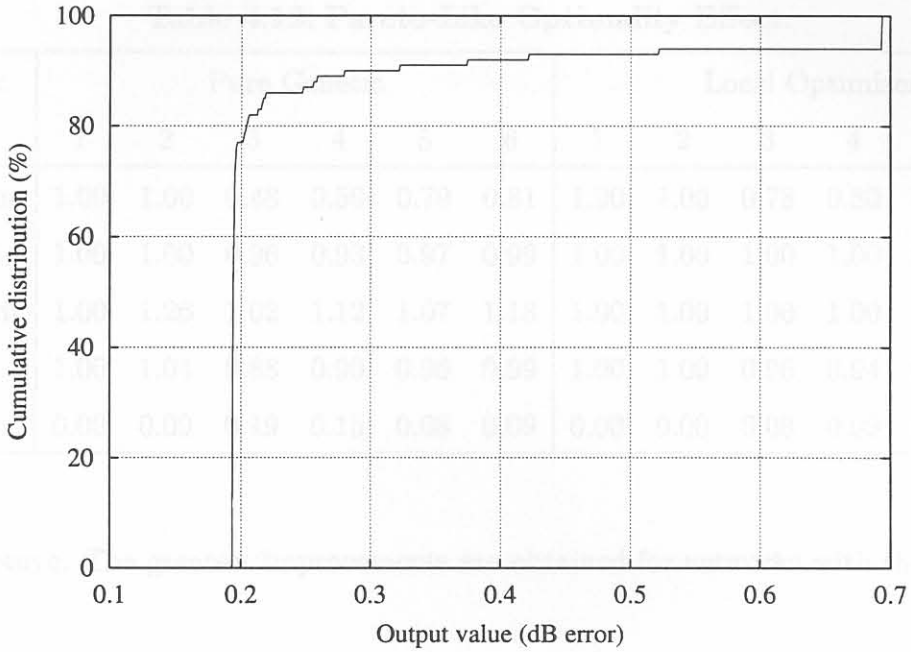


Figure 4.4: Example of a highly skewed result.

4.3.1 Pareto Effect

The effect of the Pareto-like optimality criterion discussed in Section 3.3.3 is considered here. The results with and without the Pareto-like optimality are compared to show how this approach affects the results.

The algorithm was firstly tested with the Pareto-like optimality enabled, and then with the Pareto-like optimality disabled. The algorithm had to be run once for each of the six lengths when Pareto-like optimality was disabled because each run only considers one length. Obviously this is not an entirely fair comparison because the algorithm is run six times when Pareto-like optimality is disabled, but only once when Pareto-like optimality is enabled. The results in Table 4.12 are determined from the ratio of the values obtained using Pareto-like optimality to the values that do not use Pareto-like optimality. The results are remarkable with the Pareto-like optimality actually improving the results in most cases despite requiring significantly fewer calculations. The addition of a local optimiser to the genetic algorithm decreases the effect of the Pareto-like optimality, but the results are

Table 4.12: Pareto-Like Optimality Effect.

Statistic	Pure Genetic						Local Optimiser					
	1	2	3	4	5	6	1	2	3	4	5	6
Minimum	1.00	1.00	0.48	0.59	0.79	0.81	1.00	1.00	0.78	0.80	0.89	0.72
Median	1.00	1.00	0.96	0.93	0.97	0.99	1.00	1.00	1.00	1.00	0.98	1.00
Maximum	1.00	1.26	1.02	1.12	1.07	1.18	1.00	1.00	1.00	1.00	1.02	1.07
Mean	1.00	1.04	0.88	0.90	0.96	0.99	1.00	1.00	0.96	0.94	0.97	0.97
Std Dev.	0.00	0.09	0.19	0.15	0.08	0.09	0.00	0.00	0.08	0.09	0.04	0.10

still impressive. The greatest improvements are obtained for networks with three and four elements.

4.3.2 Local Optimiser Effect

The effect of the local optimiser discussed in Section 3.4 is considered here. The results with and without the local optimiser are compared to show how it affects the results.

The algorithm was run with the default parameters in both cases. The only exception to this is that the pure genetic algorithm used a population of 50,000 and was run for 60 generations to ensure that similar numbers of fitness evaluations were used in both cases (approximately 2.6 million in the local optimiser case and approximately 2.3 million in the pure genetic algorithm case). The results in Table 4.13 are obtained from the ratio of the values using the local optimiser to the values that only use the genetic algorithm. The results obtained are better when the local optimiser is used and this in agreement with the results obtained by Renders and Flasse [34]. The effect of the local optimiser increases as the individuals become longer with significant differences being obtained for individuals with lengths of five and six elements.

Table 4.13: Local Optimiser Effect.

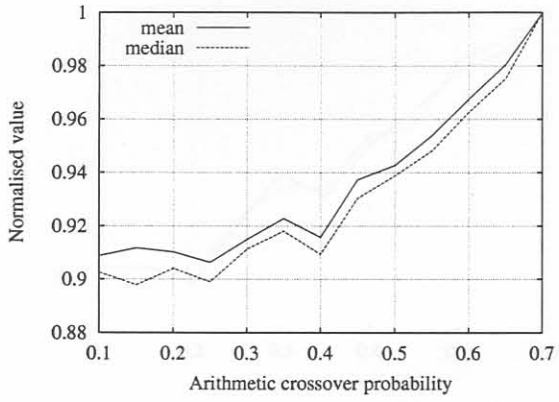
Statistic	Individual Length					
	1	2	3	4	5	6
Minimum	0.99	0.86	0.83	0.60	0.50	0.29
Median	1.00	0.99	0.99	0.96	0.88	0.80
Maximum	1.00	0.99	0.99	0.99	0.98	0.98
Mean	1.00	0.98	0.96	0.91	0.85	0.74
Std Dev.	0.00	0.04	0.05	0.12	0.14	0.23

4.3.3 Genetic Algorithm Operators

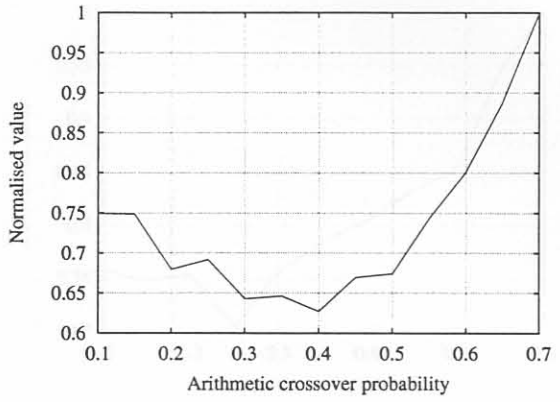
This section will consider the effect of changing the various genetic algorithm parameters. Only lumped elements were used for this round of testing because the results will be similar in the distributed and mixed cases, and lumped element tests are much faster. The genetic operators considered are arithmetic crossover, binary crossover, boundary mutation, uniform mutation, non-uniform mutation, and binary mutation.

The default values for the genetic operator probabilities were determined during the development of the algorithm and are given in Table 3.1. The probabilities were then varied round their default values to determine the effect of such variations. Both a pure genetic algorithm and a hybrid genetic algorithm with an integrated local optimiser were used for these tests, allowing the effect of the local optimiser to be observed.

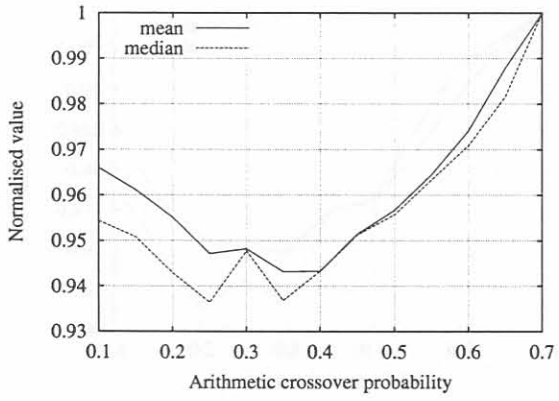
The arithmetic and binary crossover results are given in Figures 4.5 and 4.6 respectively. The arithmetic crossover probability is varied to maximum value of 0.7 because the probability of crossover (arithmetic and binary) is one at this value. The maximum probability used for binary crossover is 0.6 for the same reason. The average results show that the error initially decreases to a minimum and then increases to a maximum as the crossover probability increases. The standard deviation results show a similar trend. The default binary and arithmetic crossover values of 0.3 and 0.4 are seen to fall inside the minima of



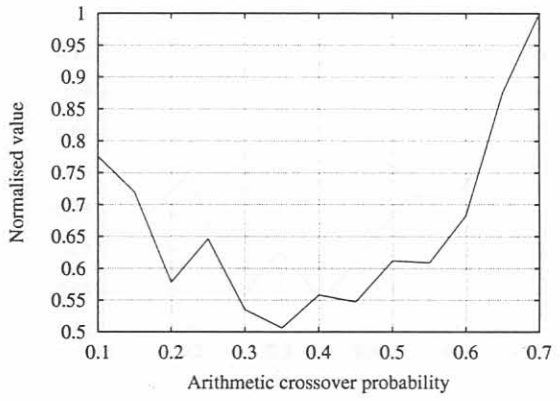
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.

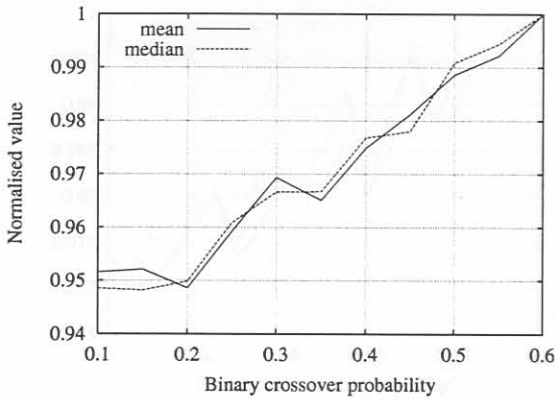


(d) Hybrid algorithm standard deviation.

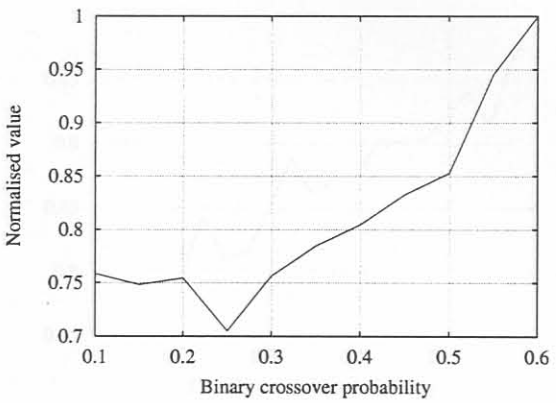
Figure 4.5: Arithmetic crossover results.

the average and standard deviation plots.

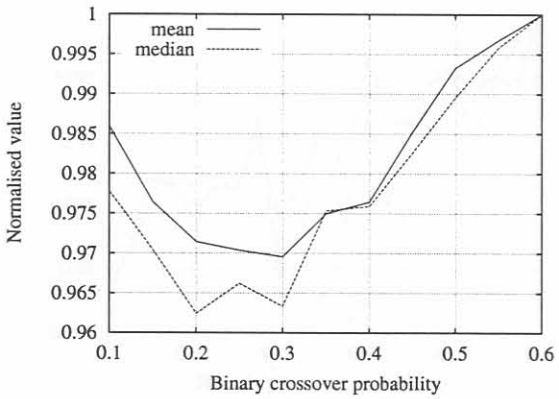
The results for boundary, binary, uniform, and non-uniform mutation are given in Figures 4.7 to 4.10. The average results show little or no trend as the mutation probability increases and the standard deviation results also show only a small trend. Where there is a trend, the error is seen to increase with mutation probability. The only exception to this is obtained with uniform mutation where the error decreases as the probability of uniform mutation increases. The default values for each type of mutation (given in Table 3.1) are



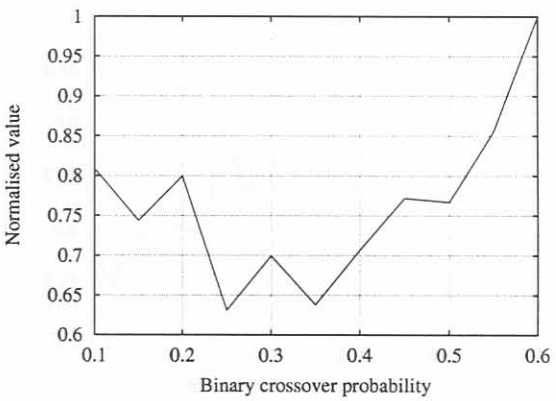
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.

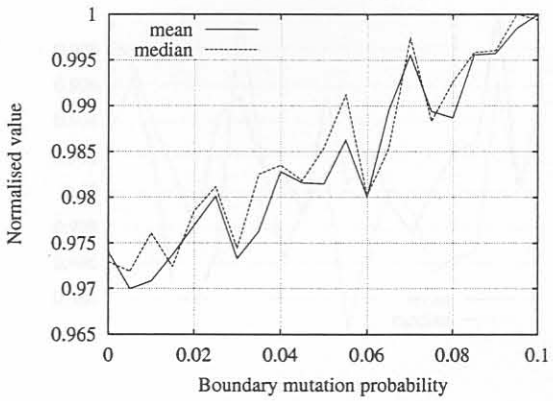


(d) Hybrid algorithm standard deviation.

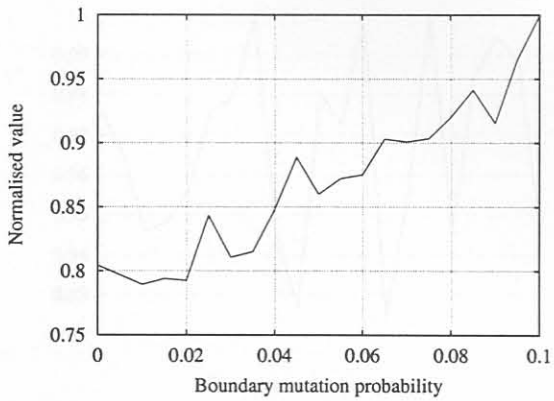
Figure 4.6: Binary crossover results.

thus shown to have only a very small effect on the performance of the algorithm.

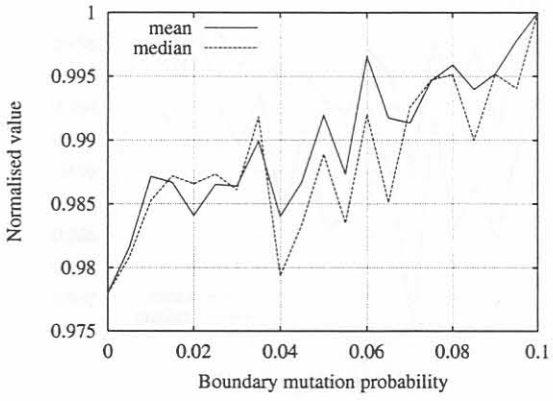
This does not mean that mutation should be ignored as shown in Table 4.14. The values in Table 4.14 are the error obtained without mutation divided by the error with mutation, averaged over all possible lengths. The standard deviation for networks with only one element is zero, so the error cannot be calculated in that case. It is clear from these results that, while mutation has only a small effect on the algorithm, it is still necessary with the mean, median, and standard deviation being higher without mutation in almost all cases.



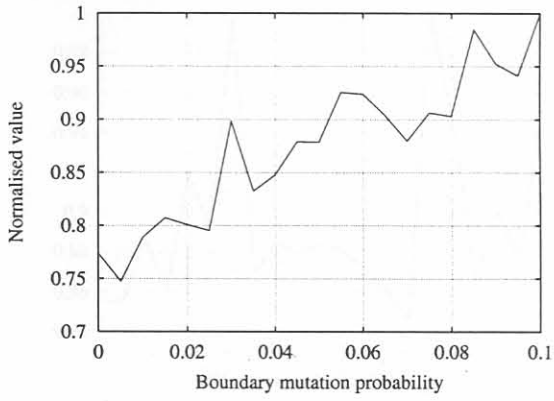
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



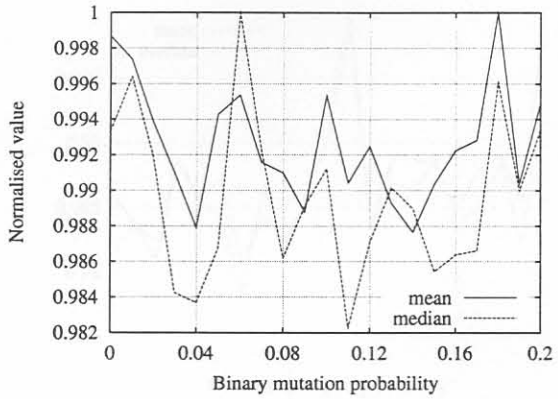
(d) Hybrid algorithm standard deviation.

Figure 4.7: Boundary mutation results.

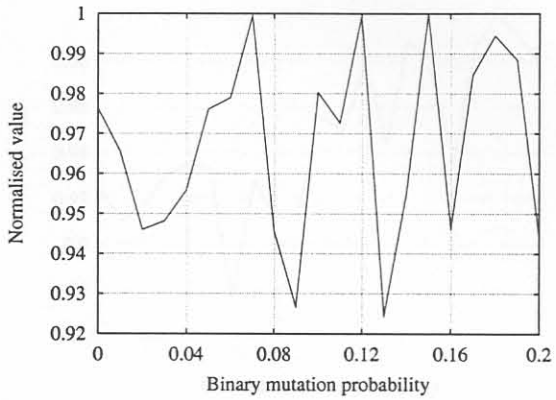
4.3.4 Tournament Size

The tournament size was varied around its default value of three to determine the effect of the bias towards good individuals. Both a pure genetic algorithm and a hybrid genetic algorithm with an integrated local optimiser were considered.

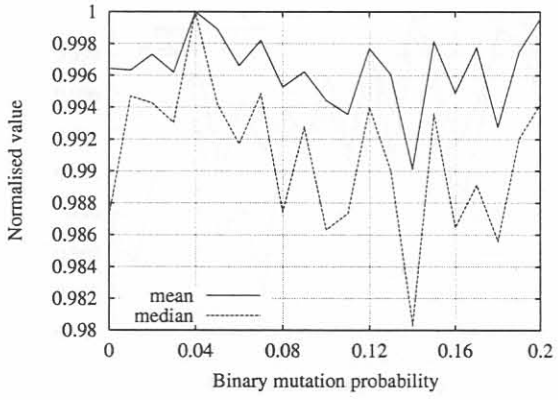
The combined results as the tournament size varies are given in Figure 4.11. The average results show that the error decreases rapidly and then slowly increases as the tournament



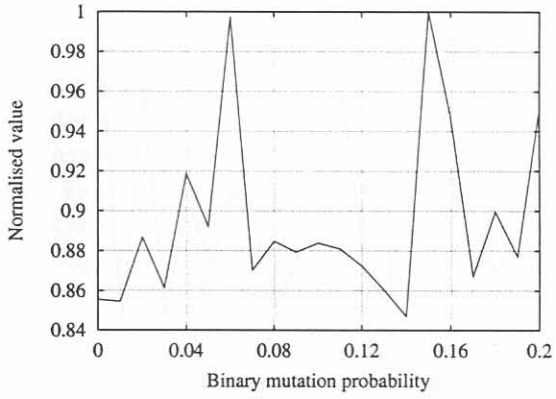
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



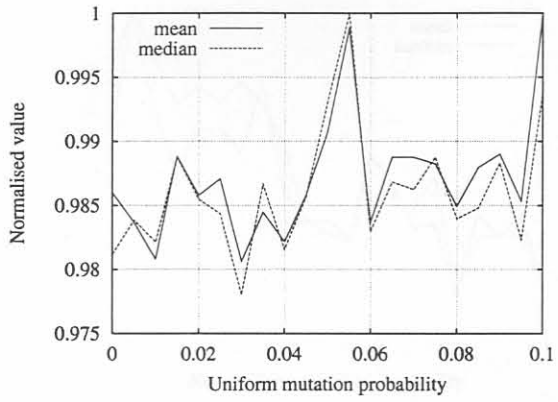
(d) Hybrid algorithm standard deviation.

Figure 4.8: Binary mutation results.

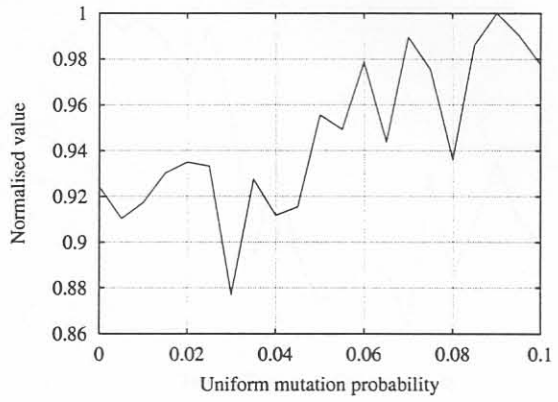
size increases. The standard deviation results show a similar trend. The default value of three for the tournament size is inside the minima of all of these results.

4.3.5 Number of Optimisation Steps

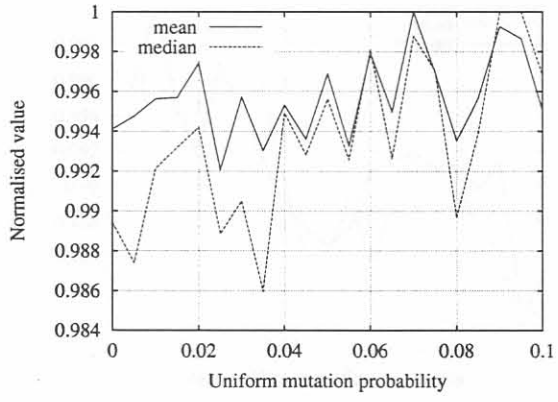
The number of local optimisation steps per iteration of a hybrid genetic algorithm is considered here. This parameter is very important because it has a major influence on both



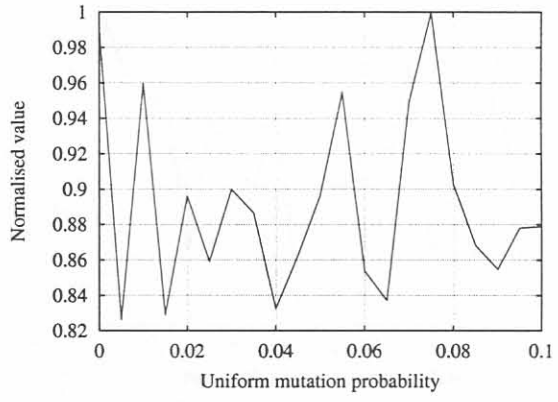
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



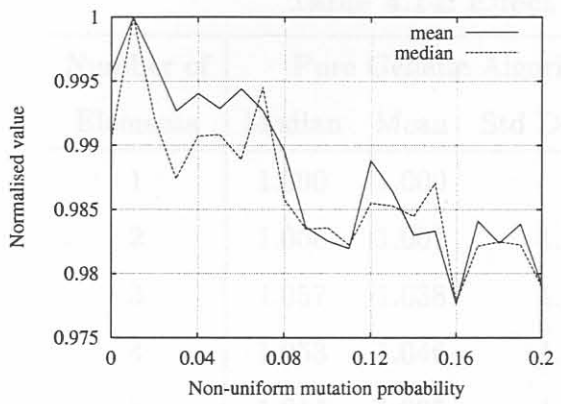
(d) Hybrid algorithm standard deviation.

Figure 4.9: Uniform mutation results.

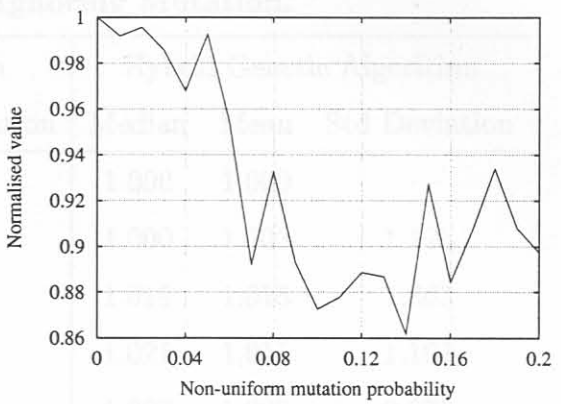
the performance of the algorithm and the amount of time required to run the algorithm.

The default values for the local optimiser parameters were determined during the development of the algorithm and were previously given in Table 3.5 on page 103. These parameters were then fixed and the number of local optimiser steps was varied around its default value of two. Obviously only the hybrid genetic algorithm was considered because the pure genetic algorithm does not have a local optimiser.

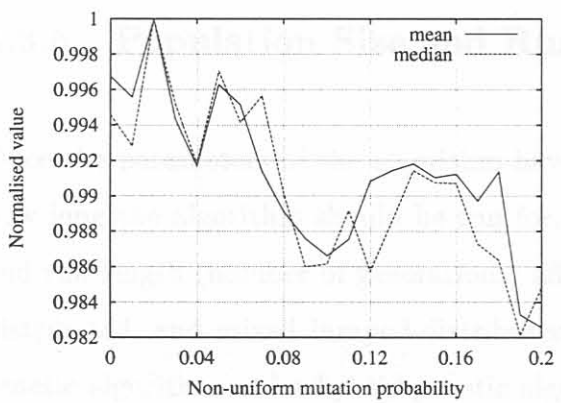
The results as the number of local optimisation steps are varied are given in Figure 4.12.



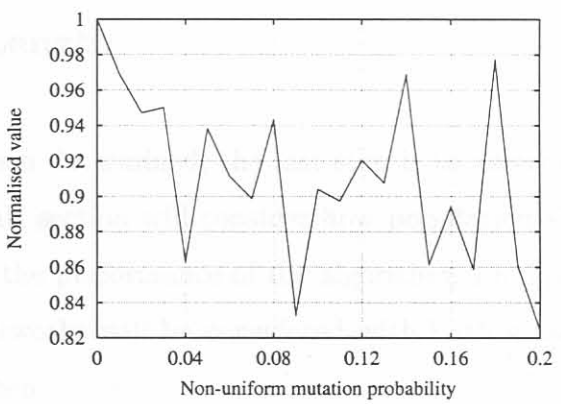
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



(d) Hybrid algorithm standard deviation.

Figure 4.10: Non-uniform mutation results.

The average results show that the error decreases as the number of local optimisation steps increases and the standard deviation results show a similar trend. The default value of two is seen to be a little low, but it must be borne in mind that the run time increases linearly with the number of local optimisation steps.

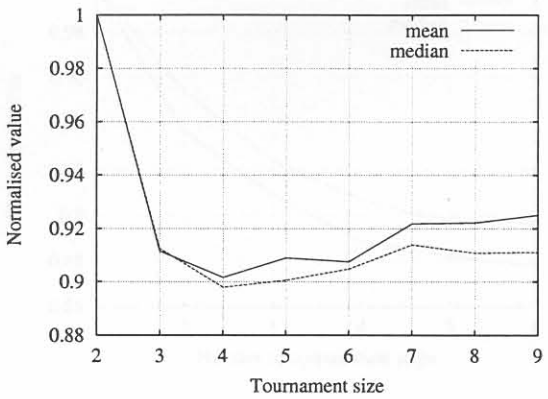
Table 4.14: Effect of Ignoring Mutation.

Number of	Pure Genetic Algorithm			Hybrid Genetic Algorithm		
Elements	Median	Mean	Std Deviation	Median	Mean	Std Deviation
1	1.000	1.000	–	1.000	1.000	–
2	1.008	1.007	1.110	1.000	1.003	1.175
3	1.057	1.038	1.188	1.019	1.015	1.303
4	1.053	1.046	1.274	1.021	1.011	1.107
5	1.014	1.025	1.075	1.007	1.008	0.973
6	1.007	1.012	1.084	0.983	1.005	0.981

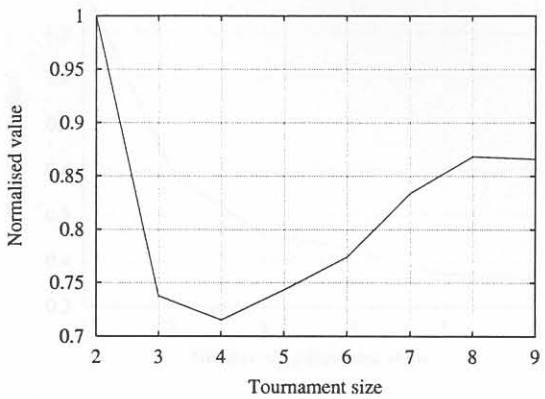
4.3.6 Population Size and Run Length

Once the parameters of the algorithm have been determined, the last step is to ascertain how long the algorithm should be run for. This section will consider how population size and run length (number of generations) affect the performance of the algorithm. Lumped, distributed, and mixed lumped-distributed networks will be considered with both a pure genetic algorithm and a hybrid genetic algorithm.

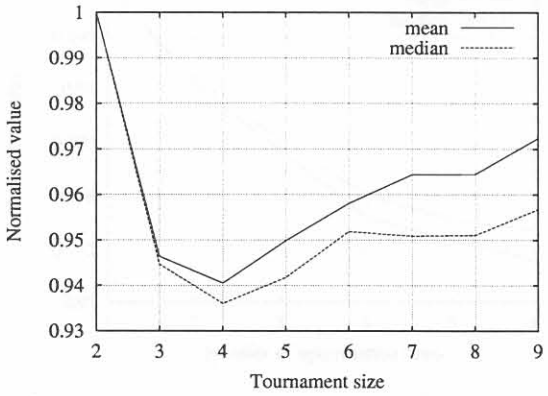
The combined results as the number of local optimisation steps are varied are given in Figures 4.13 and 4.14 on pages 133 and 134. The results show that the error decreases as the population size and the number of generations increase. The change in error slows down dramatically once a certain threshold has been reached, so increasing the population size and number of generations only leads to a small improvement once good results are obtained.



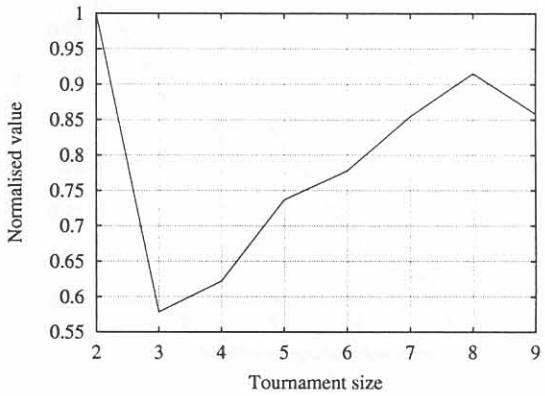
(a) Pure genetic algorithm mean and median.



(b) Pure genetic algorithm standard deviation.



(c) Hybrid algorithm mean and median.



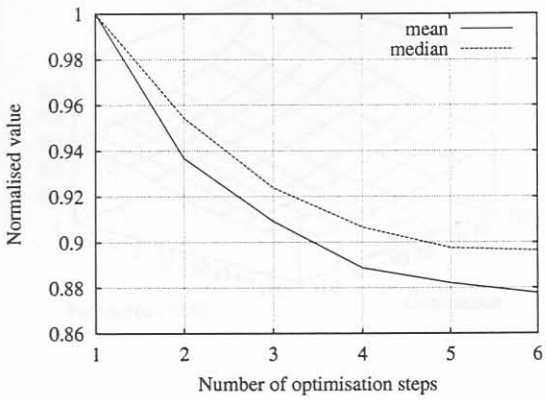
(d) Hybrid algorithm standard deviation.

Figure 4.11: Tournament size results.

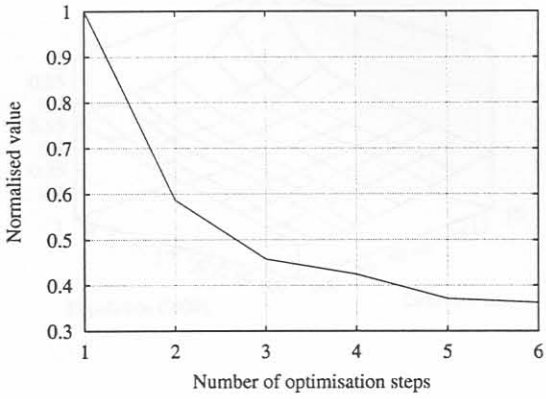
4.4 Run Times

The time required to run the algorithm is considered in this section. The lumped, mixed, and distributed cases are considered, with the distributed case being considered for both perfect transmission lines and microstrip lines.

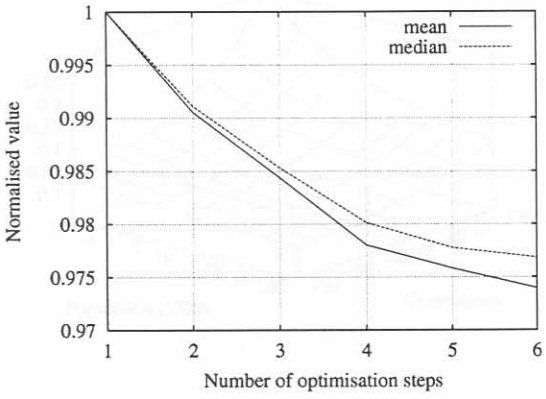
The time was determined using a 500 MHz Personal Computer (PC) with 192 MB of memory running Linux with kernel version 2.4.8–26mdk and version 2.96 of the GNU C++



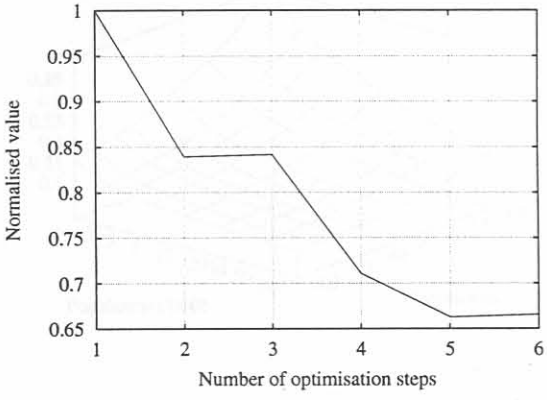
(a) Lumped elements mean and median.



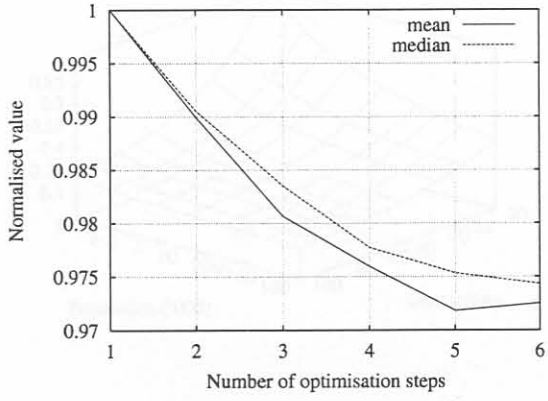
(b) Lumped elements standard deviation.



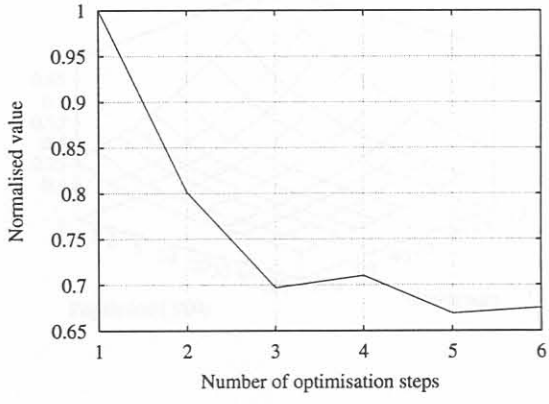
(c) Distributed elements mean and median.



(d) Distributed elements standard deviation.

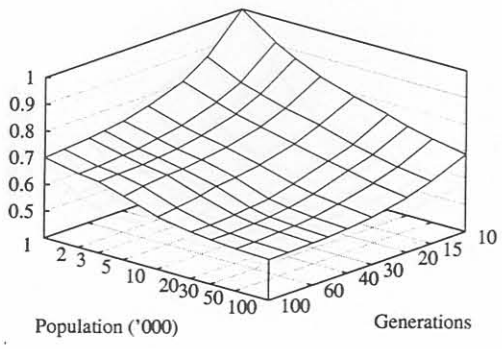


(e) Mixed elements mean and median.

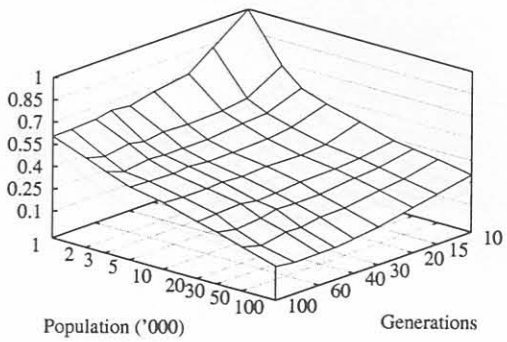


(f) Mixed elements standard deviation.

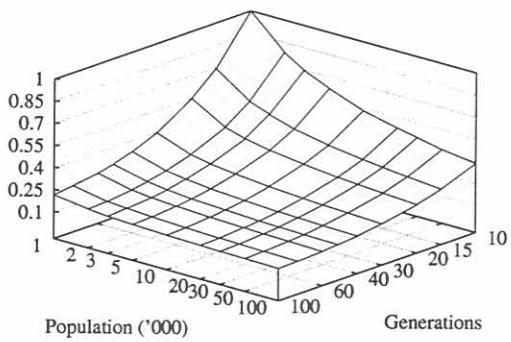
Figure 4.12: Number of local optimisation steps results.



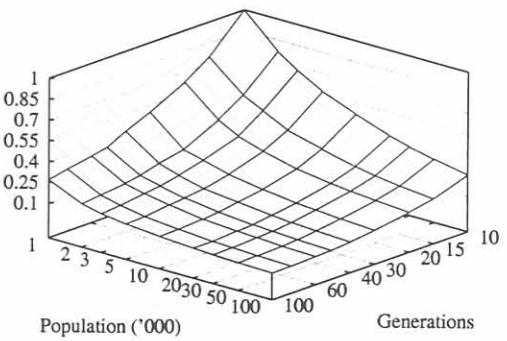
(a) Lumped elements median.



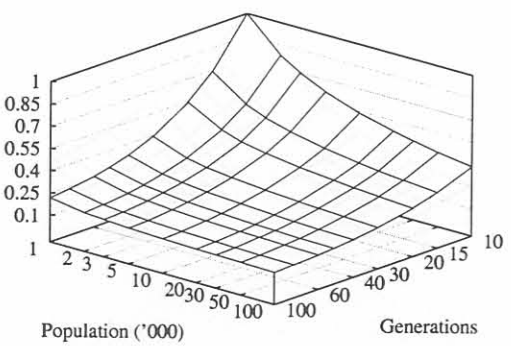
(b) Lumped elements standard deviation.



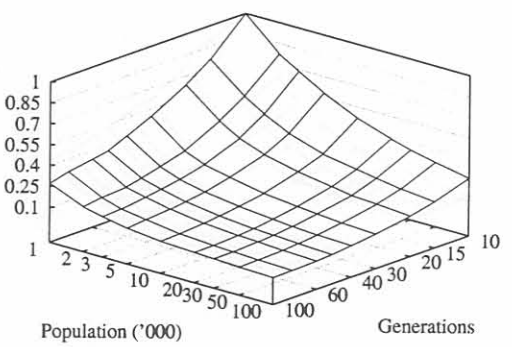
(c) Distributed elements median.



(d) Distributed elements standard deviation.

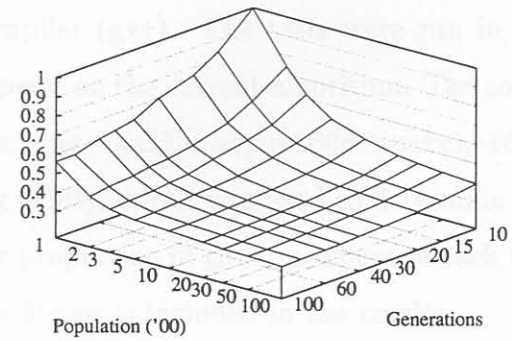


(e) Mixed elements median.

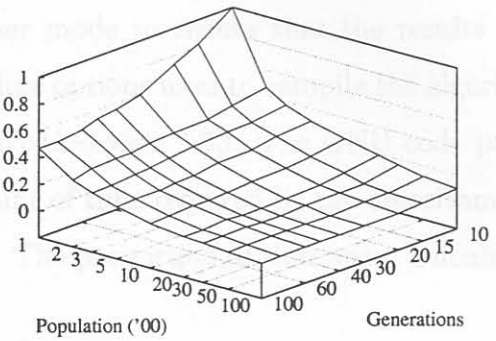


(f) Mixed elements standard deviation.

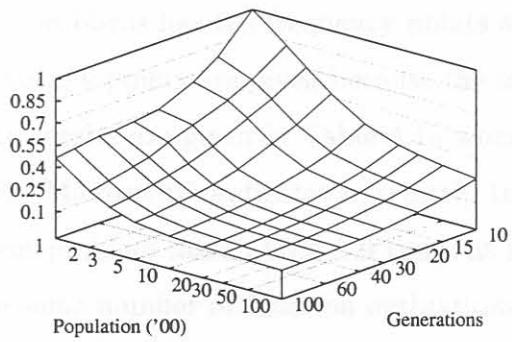
Figure 4.13: Results as population size and number of generations are varied for a pure genetic algorithm.



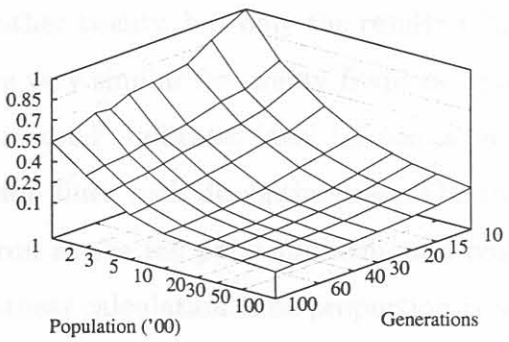
(a) Lumped elements median.



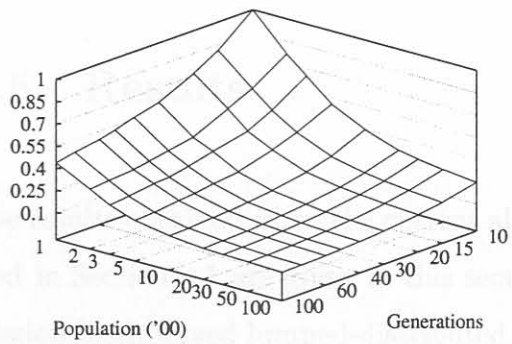
(b) Lumped elements standard deviation.



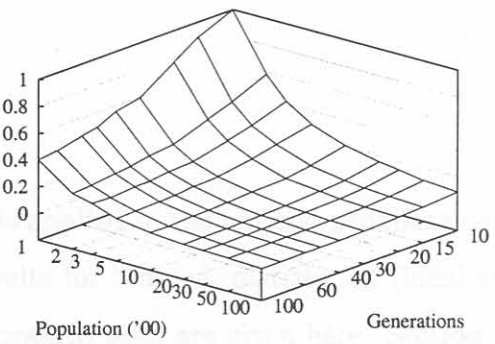
(c) Distributed elements median.



(d) Distributed elements standard deviation.



(e) Mixed elements median.



(f) Mixed elements standard deviation.

Figure 4.14: Results as population size and number of generations are varied for a hybrid genetic algorithm.

compiler (g++). The tests were run in single user mode to ensure that the results only depend on the current algorithm. The command line options used to compile the algorithm are: `-pg -Wall -mcpu=i686 -march=i686 -funroll-loops -O3`. The GNU code profiling utility, `gprof` was used to determine the amount of time required by the algorithm and the proportion of the time spent in each function. The proportion of time spent calculating the fitness is included in the results.

The results were averaged over ten runs of the algorithm in each case to characterise variations that can occur. The test problems have complex impedances at both the input and output with the impedances being constant over the whole frequency range. One of the test problems has ten frequency points and the other twenty, but only the results with ten frequency points are given because the trends are very similar for twenty frequency points. The results are given in Table 4.15 where "Distributed" indicates ideal transmission lines and "Microstrip" indicates microstrip transmission lines with discontinuities. The twenty point problem takes about 1.9 times as long to run as the ten point problem, and requires the same number of function evaluations. The fitness calculation time proportion is about 2 percentage points higher in the microstrip case than in the ideal transmission line case. Significantly, discontinuity calculations (microstrip impedance, width step, and T-junction effects) take almost 80% of the total time in the microstrip case.

4.5 Results

The results obtained when the current algorithm is applied to the ten test problems considered in Section 4.1 are given in this section. Results for lumped, distributed (ideal transmission line), mixed lumped-distributed, and microstrip tests are given here. Section 4.5.1 compares the results obtained by the current algorithm with published results.

The default genetic algorithm parameter values given in Table 3.1 on page 82, the default local optimiser values given in Table 3.5 on page 103, and the default component ranges

Table 4.15: Run Time Results.

Parameter	Lumped	Mixed	Distributed	Microstrip
Minimum time	98.6 s	98.8 s	97.9 s	1700.7 s
Median time	99.7 s	99.7 s	99.7 s	1706.2 s
Maximum time	101.7 s	100.5 s	102.1 s	1717.7 s
Minimum fitness %	97.0%	96.9%	96.9%	99.5%
Median fitness %	97.2%	97.1%	97.2%	99.5%
Maximum fitness %	97.4%	97.2%	97.4%	99.6%
Minimum evaluations	3,408,870	3,042,606	3,406,437	3,362,304
Median evaluations	3,415,986	3,415,792	3,418,184	3,368,012
Maximum evaluations	3,435,256	3,429,292	3,430,717	3,376,443

given in Table 3.3 on page 87 were used for these tests. The microstrip substrate parameters used are given in Table 3.4 on page 87. The transmission line elements in the distributed results given below do not use the default parameters given in Table 3.3, but rather have the same parameters as the microstrip lines to allow comparisons between the ideal case and the case where dispersion and discontinuities are accounted for. The microstrip parameters were converted to characteristic impedances and line lengths using the low-frequency equations given in Sections 2.4.1 and 2.4.2 on pages 66 and 68. The ratios of the widths of microstrip lines at a discontinuity was limited to a maximum value of 5, and crosses were not allowed to ensure that the discontinuity calculations are accurate.

The mixed results do not consider discontinuities and dispersion for the transmission lines, and solder pads for the lumped components. These effects could have been included, but the inclusion of these effects would have dramatically increased the time to run the algorithm to closer to the time required for microstrip tests (see Section 4.4) without greatly adding to the value of the results.

The microstrip networks can start with a parallel element because the algorithm assumes that all microstrip networks are bounded by 50 Ω transmission lines. This is necessary to

Table 4.16: Results for Problem 1 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	6	1.180	1.184	1.257
Distributed	3	1.242	1.273	1.324
Mixed	3	1.245	1.285	1.347
Microstrip	3	1.245 / 1.268	1.277 / 1.281	1.339 / 1.263

Table 4.17: Results for Problem 2 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	3	2.222	2.222	2.222
Distributed	3	2.197	2.340	2.433
Mixed	3	2.189	2.222	2.587
Microstrip	6	1.433 / 6.202	1.654 / 13.81	1.934 / 374.7

complete the discontinuities (width steps or T-junctions) at the extremities of the networks.

Note that two results are given in the microstrip cases. The first set of results are those obtained using the present algorithm while the second set are those obtained using EEsof. A comparison between these two sets of results has already been given in Section 4.2.2.

The results for Problems 1, 2, 3, 4, 9, and 10 are listed as VSWR values, and the results for Problems 5, 6, 7, and 8 are given as decibel errors. This was done because the first group of problems require the match to be as good as possible (matching network gain must be as high as possible), and the second group of problems requires a specific gain to be obtained at each frequency.

The results for some specific cases are given in Tables 4.16 to 4.25. The best networks obtained in these cases are given in Figures B.1 to B.10 on pages 175 to 184. The number of elements used in each case is the same as published results where these are available, and produces a VSWR of less than 1.5 or a gain error of less than 0.1 dB where published

Table 4.18: Results for Problem 3 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	3	1.676	1.676	2.024
Distributed	3	1.779	1.848	2.020
Mixed	5	1.586	1.680	1.795
Microstrip	2	1.308 / 2.514	1.643 / 9.728	2.293 / 33.74

Table 4.19: Results for Problem 4 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	5	2.929	2.940	2.988
Distributed	6	1.478	2.387	3.892
Mixed	6	2.455	2.980	3.430
Microstrip	6	1.269 / 49.83	1.708 / 251.9	2.324 / 952.7

Table 4.20: Results for Problem 5 (dB error).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	4	4.538×10^{-2}	0.1302	0.1589
Distributed	3	8.192×10^{-2}	0.1046	0.1353
Mixed	3	8.796×10^{-2}	0.1021	0.1336
Microstrip	3	0.0846 / 0.1017	0.0978 / 0.1150	0.1278 / 0.1336

results are not available. Six element solutions are given where a VSWR of less than 1.5 or a gain error of less than 0.1 dB is not possible. The number of elements in the microstrip case is the same as in the distributed case to allow comparisons to be made. The only exceptions are the solutions to Problems 2 and 3, and the lumped and mixed solutions to Problem 7, where adding extra elements has a small ($< 5\%$) effect.

The complete lumped, distributed, mixed, and microstrip results are given in Tables A.1

Table 4.21: Results for Problem 6 (dB error).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	5	7.960×10^{-2}	0.1750	0.2046
Distributed	4	6.513×10^{-2}	0.2289	0.3273
Mixed	3	9.332×10^{-2}	0.2335	0.4957
Microstrip	4	0.2302 / 0.2855	0.3165 / 0.3578	0.4621 / 0.4773

Table 4.22: Results for Problem 7 (dB error).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	5	0.3166	0.4611	0.7700
Distributed	6	0.2214	0.2625	0.2891
Mixed	5	0.2442	0.2861	0.3390
Microstrip	6	0.2041 / 0.2277	0.2558 / 0.2835	0.2778 / 0.3398

Table 4.23: Results for Problem 8 (dB error).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	6	0.2449	0.3005	0.4640
Distributed	6	6.850×10^{-2}	0.1841	0.3166
Mixed	6	9.805×10^{-2}	0.2451	0.4516
Microstrip	6	0.0967 / 0.2389	0.2304 / 0.3059	0.4620 / 0.5000

to A.40 on pages 159 to 172.

A number of trends can be observed in the results. The worst results are typically very poor, but the good median results show that the solutions are heavily skewed towards good results. Solutions with more elements typically have a higher standard deviation than solutions with fewer elements. The distributed results are usually worse than the lumped results because the range of parameter values is much more limited to ensure that the

Table 4.24: Results for Problem 9 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	4	1.214	1.268	1.333
Distributed	4	1.377	1.619	1.845
Mixed	3	1.259	1.484	1.867
Microstrip	6	1.576 / 3.050	2.662 / 18.83	3.498 / 460.2

Table 4.25: Results for Problem 10 (VSWR).

Present algorithm	Elements	Minimum	Median	Maximum
Lumped	3	1.467	1.557	2.063
Distributed	3	1.427	1.551	1.711
Mixed	3	1.416	1.592	1.816
Microstrip	3	1.604 / 1.680	1.807 / 1.832	1.964 / 1.987

distributed elements are realisable. The mixed results are typically at least as good as the better of the lumped and distributed results, but this is not always the case. Some of the solutions to Problems 2, 3, 4, and 7 have a number of series transmission lines that all have the same impedance in the ideal transmission line case, and the same line widths in the microstrip case.

The maximum line width ratio at discontinuities in the microstrip case is violated by a small amount in some of the solutions as shown in Table 4.26. The best solutions of every length for each run of the algorithm are considered in Table 4.26. Of a total of 19650 width steps, only 216 (1.1%) violate the maximum width ratio, with 203 (94%) of these violations being accounted for by problem 8. There are 3675 T-junctions in the best solutions with only 6 of these (0.2%) having width ratio violations.

The microstrip results are comparable to the distributed results in most cases, but there are some cases where the microstrip results are worse than the distributed results. The best

Table 4.26: Maximum Line Width Violations.

Problem	Width Step			T-Junction		
Number	Max. Ratio	Total	Violations	Max. Ratio	Total	Violations
1	4.89	2664	0 (0.0%)	3.03	18	0 (0.0%)
2	5.02	1450	1 (0.1%)	5.59	625	1 (0.2%)
3	4.94	1654	0 (0.0%)	5.02	523	1 (0.2%)
4	3.38	1450	0 (0.0%)	5.43	625	1 (0.2%)
5	5.48	2498	8 (0.3%)	3.55	101	0 (0.0%)
6	5.01	1458	1 (0.1%)	5.10	621	1 (0.2%)
7	5.00	2534	0 (0.0%)	4.23	83	0 (0.0%)
8	5.10	2578	203 (7.9%)	4.88	61	0 (0.0%)
9	5.01	2308	3 (0.1%)	5.01	196	1 (0.5%)
10	4.90	1056	0 (0.0%)	5.01	822	1 (0.1%)
total	5.48	19650	216 (1.1%)	5.59	3675	6 (0.2%)

solutions for Problems 1, 2, 5, 7, and 8 are examples of solutions that are very similar in the distributed and microstrip cases, and the best solutions for Problems 3, 5, 8, 9, and 10 are examples of solutions where the microstrip case gives worse results than the distributed case. The best solutions to Problems 2, 4, and 6 show that the microstrip solutions can be better than the distributed solutions.

4.5.1 Comparison to Published Results

This section will compare the results obtained using this algorithm with those that have been published in the literature. Obviously only those test problems that were obtained from the literature can be considered here. The main difficulty with these comparisons is that the vast majority of published results only consider lumped network synthesis, but the current algorithm can consider other cases.

Most of the published results use the transducer gain (G_T) or VSWR to determine how good

Table 4.27: Results for Problem 1 (VSWR).

Network	Result	Elements	Transformers
Abrie (lumped)	1.19	6	0
Abrie (commensurate)	1.06	4	0
Abrie (non-commensurate)	1.25	6	0
Abrie (microstrip)	1.41/1.40	4	0
Present algorithm	Minimum	Median	Maximum
Lumped (6 elements)	1.180	1.184	1.257
Distributed (4 elements)	1.089	1.156	1.258
Distributed (6 elements)	1.048	1.090	1.146
Mixed (4 elements)	1.089	1.179	1.295
Mixed (6 elements)	1.048	1.098	1.184
Microstrip (4 elements)	1.25/1.26	1.28/1.30	1.32/1.35
Microstrip (6 elements)	1.06/1.11	1.11/1.08	1.23/1.16

a match is. Note that, unlike the other error functions considered here, a higher transducer gain indicates a better match, and a perfect match corresponds to a transducer gain of one. The results for the current algorithm given below were derived from the decibel errors obtained during the tests. This was done by using equations (3.25) and (3.28) to convert to transducer gain, and (3.20) and (3.29) to calculate the VSWR. The mean, median, and maximum results were then calculated.

Note that non-ideal transformers are considered as a single element in the determination of the number of elements in the network. Many of the references do not consider an ideal transformer to be an element and this convention will be followed here, but the number of ideal transformers required will be given. The present algorithm will use the same number of elements as the published results without any ideal transformers.

The algorithm parameters used for these tests are the same as the parameters used in Section 4.5.

Table 4.28: Results for Problem 2 (G_T).

Author	Result	Elements	Transformers
Fano [57, 58]	0.816	4	0
Carlin [4]	0.837	3	0
Chen [5]	0.831	4	0
Carlin and Amstutz [5]	0.849	3	0
Dedieu *et al.* [82]	0.855	3	0
Present algorithm	Minimum	Median	Maximum
Lumped (3 elements)	0.8562	0.8562	0.8562

Problem 1 is taken from Abrie [2]. This problem in unique because lumped and distributed results are available. Unfortunately, the distributed results were obtained by fixing either the line length (commensurate case) or the line impedances (non-commensurate case), so the results are not completely comparable. The results published by Abrie and the results obtained using the present algorithm are given in Table 4.27 with the errors given as VSWR values. The best solutions obtained are shown in Figure B.1 on page 176. The lumped solution is similar to the solution obtained by Abrie [2]. The microstrip line results were obtained using one of the examples that is supplied with the demonstration version of MultiMatch Mosaic which was obtained from Ampsa's web page (www.ampsa.com). The only differences in the MultiMatch example are that two extra transmission lines are included at the load and source, and the line impedances are constrained to be between 25 and 85 Ω. These changes were also made to the current algorithm in the microstrip case to allow the results to be compared directly. The second value given in each of the microstrip cases is the value obtained using EEsof. The performance of the current algorithm on this problem is seen to be comparable to the published results, although the worst results are poor. The microstrip results are particularly impressive because they include discontinuity effects.

Problem 2 was originally proposed by Fano [57, 58], but has also been used by Carlin [4],

Table 4.29: Results for Problem 3 (G_T).

Author	Result	Elements	Transformers
Carlin and Yarman [8] (analytic)	0.847	3	1
Carlin and Yarman [8] (real-frequency)	0.932	3	1
Yarman and Fettweis [9]	0.927	3	1
Yarman and Aksen [87]	≈0.94	3	0
Dedieu *et al.* [82]	0.938	3	1
Present algorithm	Minimum	Median	Maximum
Lumped (3 elements)	0.8853	0.9362	0.9362
Mixed (3 elements)	0.9105	0.9362	0.9362

Carlin and Amstutz [5], and Dedieu *et al.* [82]. Carlin and Amstutz [5] have also applied the formulae developed by Chen [63] to this problem, and these analytic results are also given. The results are shown in Table 4.28 with the errors given as transducer gains. The best solutions are shown in Figure B.2 on page 177 where the component values have been scaled back to the original frequency band. The results are similar to the solutions obtained by Carlin [4], Carlin and Amstutz [5], and Dedieu *et al.* [82]. The performance of the algorithm developed here is seen to at least as good as the published results.

Problem 3 is used by Carlin and Yarman [8], Yarman and Aksen [87], Yarman and Fettweis [9], and Dedieu *et al.* [82]. Yarman and Carlin [8] give results for both the analytic and real-frequency algorithms developed in that work. The results by Yarman and Aksen [87] are for a mixed lumped-distributed network where a lumped prototype is transformed to a mixed network with three elements, two of which are series transmission lines, and then optimised. Yarman and Aksen [87] do not give a value for the final network obtained, so the value given in Table 4.29 was read off the frequency response of the final network. The results are shown in Table 4.29 in the form of transducer gains, and the current algorithm is again seen to achieve results that are at least as good as the published results in the vast majority of cases. The best solutions are shown in Figure B.3 on page 178 where the

Table 4.30: Results for Problem 4 (G_T).

Author	Result	Elements	Transformers
Carlin and Yarman [8]	0.742	5	1
Yarman and Fettweis [9]	0.722	5	1
Dedieu *et al.* [82]	0.750	5	1
Present algorithm	Minimum	Median	Maximum
Lumped (5 elements)	0.7515	0.7576	0.7590

Table 4.31: Results for Problem 5 (decibel error).

Network	Result	Elements	Transformers
Abrie (lumped)	0.05 dB	4	0
Present algorithm	Minimum	Median	Maximum
Lumped (4 elements)	0.04538 dB	0.1302 dB	0.1481 dB

component values have been scaled back to the original frequency band. The results are similar to the solutions obtained by Carlin and Yarman [8], Yarman and Fettweis [9], and Dedieu *et al.* [82].

Problem 4 appears in the papers by Carlin and Yarman [8], Yarman and Fettweis [9], and Dedieu *et al.* [82]. The results are shown in Table 4.30 as transducer gains, and the current algorithm is seen to obtain better results than the published algorithms. The best solutions are shown in Figure B.4 on page 179 where the component values have been scaled back to the original frequency band. The results are similar to the solutions obtained by Carlin and Yarman [8], but have a different structure to the solutions obtained by Yarman and Fettweis [9], and Dedieu *et al.* [82].

Problem 5 is a double matching problem taken from Abrie [2]. The results are given in Table 4.31, where the results are given as maximum gain errors in dB. The best solutions are shown in Figure B.5 on page 180. The lumped solution is seen to be similar to one

of the solutions obtained by Abrie [2]. The current algorithm has tremendous difficulty obtaining a good result on this problem, despite the fact that the best result is comparable to the best published result.

Problem 6 is also taken from the literature, but unfortunately, Abrie [2] only gives results for the amplifier this network is part of, not the network itself. The best solutions are shown in Figure B.6 on page 181. The lumped solution has a different structure to the solution obtained by Abrie [2].

4.6 Summary

A large number of tests were conducted to quantify the performance of the algorithm developed here, with the results being summarised in this chapter.

Ten test problems from a variety of sources were used to test the algorithm as thoroughly as possible. The problems include many different cases, and are a mix of established and new problems.

The accuracy of the fitness calculations was verified by comparing results to those obtained by EEsof. The models used for ideal components were found to be exact, and the models used for microstrip dispersion and discontinuities were found to be in very good agreement with the EEsof models. The only notable exception is the cross model which is poor. The results obtained using complete circuits are also excellent except where the substrate is thick and has a low dielectric constant.

The algorithm options and parameters were then modified to determine their effect on the performance of the algorithm. The Pareto-like optimality and local optimiser were found to improve the results obtained, and the default parameters were found to be very close to the optimal values.

The results obtained are impressive with the best results being at least as good as the best published results. The main drawback is that the results are not consistent, and the worst results are usually poor. Results for longer individuals, and mixed and distributed networks display more variance than the results for shorter individuals in the lumped case.

The implications of these results are discussed in Chapter 5.