

CHAPTER SIX

6.1 Results

Several runs were made. The following results presented were obtained by running the program for 325 generations.

The coefficients used in fitness function were;

Parameter	Coefficient	Declaration
Load	0.4	giving the idea of the amount Over loaded
NoLines	0.35	giving the idea of number of Non – over loading lines
TC	0.3	Total cost
L_F	0.3	Line fitness
n	0.15	Number of under loaded lines

The average program run time for 325 generations have been around 9 minutes.

These are the results obtained for 10 runs of program. Summary has been tabulated;

No	Tcost	Nlines	TLineLength	Nover LoadLines	Avg OverLoad %	Nunder LoadLines	Avg UnderLoad%
1	3065	9	510	4	170.51	2	14
2	3175	8	520	3	144.17	2	11.48
3	3175	8	520	3	144.18	2	15.27
4	3730	10	620	2	133	3	11.86
5	5445	13	890	2	135.65	6	15.62
6	5540	13	890	0	0	5	9.89
7	5720	14	980	2	115.43	6	13.15
8	6250	15	1090	0	0	6	13.74
9	6330	15	1090	1	111.8	7	14.16
10	6475	15	1090	1	119.37	5	11.29

Using this type of result table designer can select suitable solution according to his requirement.

Detailed No 1 result;

From To Line Type

1	2	Lynx
1	5	Lynx
2	3	350 Oil Filled
2	4	300 XLPE
2	5	Lynx
2	6	1xZebra
3	5	Lynx
3	6	Bear
4	6	300 XLPE


Total cost (in millions) = 3065

Total number of lines = 9

Total Line length (in km) = 510

System Summary

How many?	How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0 -900.0 to +900.0
Generators	3	On-line Capacity	1110.0 -900.0 to +900.0
Committed Gens	3	Generation (current)	770.0 118.5
Loads	5	Load	770.0 0.0
Branches	9	Losses ($I^2 * Z$)	0.00 118.53
Transformers	0	Branch Charging (inj)	- 0.0
Areas	1	Shunt (inj)	0.0 0.0
Inter-ties	0	Total Inter-tie Flow	0.0 0.0


 University of Moratuwa, Sri Lanka
 Electronic Theses & Dissertations
 Minimum Maximum
art.ac.lk

	Minimum	Maximum
Voltage Magnitude	0.986 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	-1.77 deg @ bus 5	14.85 deg @ bus 6
P Losses ($I^2 * R$)	-	0.00 MW @ line 1-2
Q Losses ($I^2 * X$)	-	33.74 MVAR @ line 2-6

Bus Data

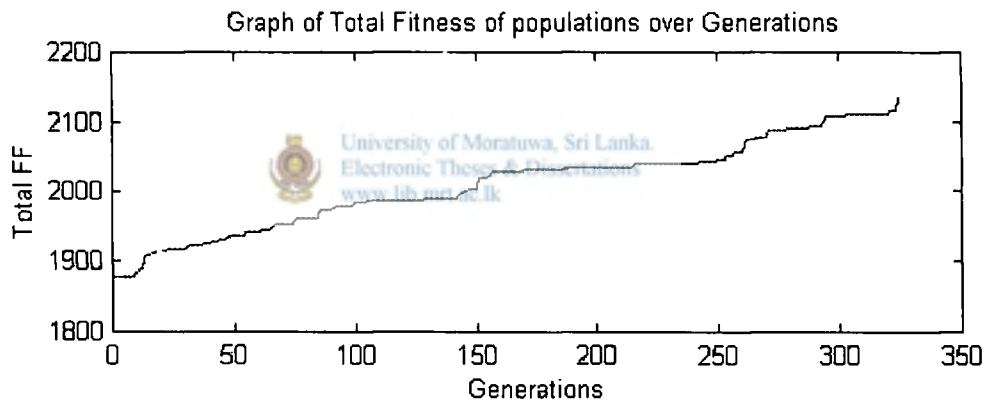
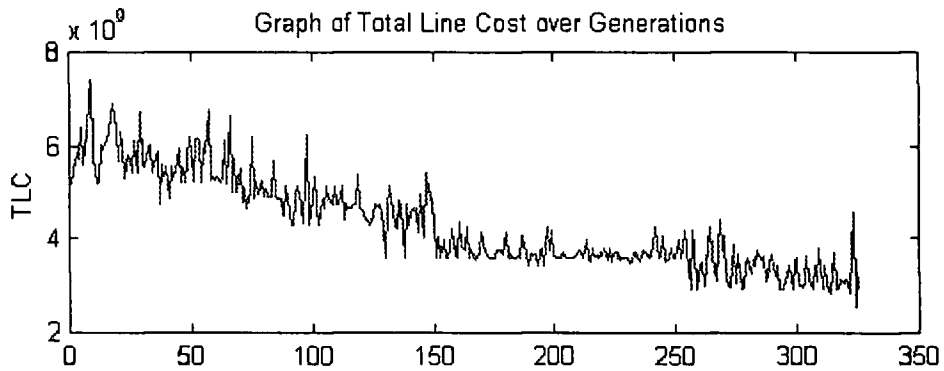
Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	13.48	90.00	0.00
2	0.993	3.715	-	-	240.00	0.00
3	1.000	6.992	300.00	50.62	40.00	0.00
4	0.986	3.409	-	-	160.00	0.00
5	0.992	-1.770	-	-	240.00	0.00
6	1.000	14.851	400.00	54.43	-	-
Total:			770.00	118.53	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 * Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	-46.65	6.31	46.65	-3.25	0.000	3.06

1	5	26.65	7.17	-26.65	-6.29	0.000	0.88
2	3	-199.23	-17.39	199.23	28.94	0.000	11.55
2	4	3.50	4.80	-3.50	-4.75	0.000	0.05
2	5	81.93	4.92	-81.93	2.93	0.000	7.85
2	6	-172.84	10.92	172.84	22.81	0.000	33.74
3	5	131.43	16.83	-131.43	3.36	0.000	20.19
3	6	-70.66	4.85	70.66	4.85	0.000	9.71
4	6	-156.50	4.75	156.50	26.77	0.000	31.51

Total: 0.000 118.53



Detailed No 2 result;

From	To	Line Type
1	2	300 XLPE
2	3	Lynx
2	4	300 XLPE
2	5	Lynx
3	4	1xGoat
3	5	Bear
4	6	1xZebra
5	6	Lynx

Total cost (in millions) = 3175
 Total number of lines = 8
 Total Line length (in km) = 520

System Summary

How many?	How much?	P (MW)	Q (MVAR)	
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	176.2
Loads	5	Load	770.0	0.0
Branches	8	Losses ($I^2 * Z$)	0.0	176.23
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

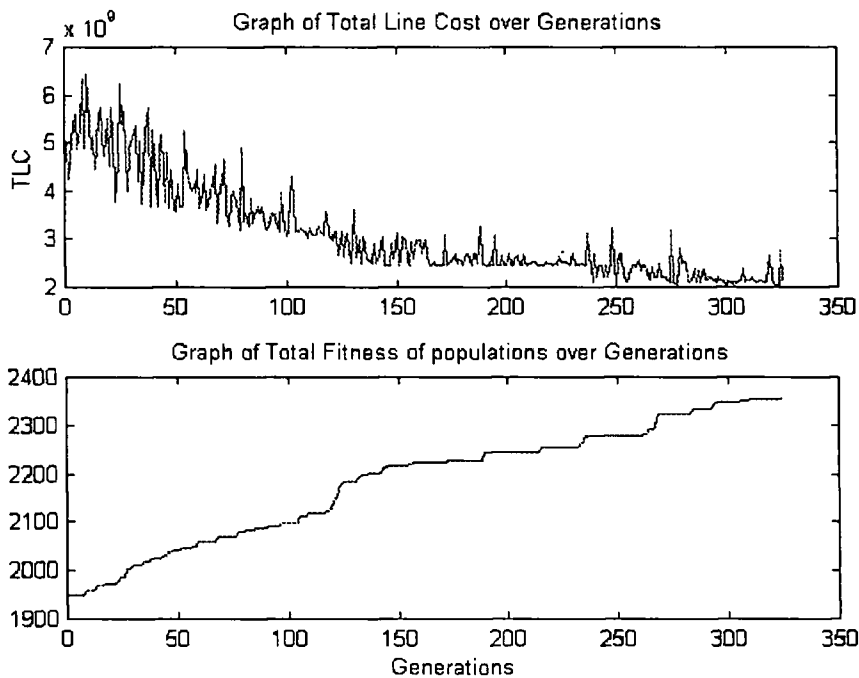
	Minimum	Maximum
Voltage Magnitude	0.974 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	0.00 deg @ bus 1	24.87 deg @ bus 6
P Losses ($I^2 * R$)	-	0.00 MW @ line 1-2
Q Losses ($I^2 * X$)	-	63.99 MVAR @ line 4-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	12.60	90.00	0.00
2	0.982	1.751	-	-	240.00	0.00
3	1.000	11.136	300.00	66.67	40.00	0.00
4	0.974	9.421	-	-	160.00	0.00
5	0.976	4.212	-	-	240.00	0.00
6	1.000	24.866	400.00	96.96	-	-
Total:			770.00	176.23	770.00	0.00

Branch Data

From Bus	To Bus	From Bus P (MW)	Injection Q (MVAR)	To Bus P (MW)	Injection Q (MVAR)	Loss ($I^2 * Z$)	
						P (MW)	Q (MVAR)
1	2	-20.00	12.60	20.00	-11.77	0.000	0.84
2	3	-139.17	-4.32	139.17	27.46	0.000	23.14
2	4	-85.06	10.63	85.06	0.81	0.000	11.44
2	5	-35.77	5.46	35.77	-3.90	0.000	1.56
3	4	11.37	10.31	-11.37	-9.70	0.000	0.60
3	5	109.45	28.89	-109.45	-15.12	0.000	13.78
4	6	-233.69	8.89	233.69	55.09	0.000	63.99
5	6	-166.31	19.01	166.31	41.87	0.000	60.89
Total:						0.000	176.23



Detailed No 3 result;

From	To	Line Type
1	2	300 XLPE
2	3	Lynx
2	4	300 XLPE
2	5	Lynx
3	4	1xGoat
3	5	Bear
4	6	1xZebra
5	6	Lynx



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Total cost (in millions) = 3175
Total number of lines = 8
Total Line length (in km) = 520

System Summary

How many?	How much?	P (MW)	Q (MVAR)	
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	176.2
Loads	5	Load	770.0	0.0
Branches	8	Losses ($I^2 * Z$)	0.00	176.23
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

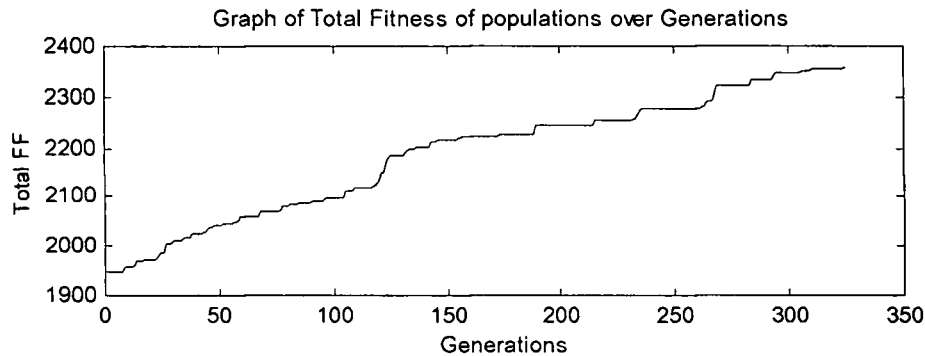
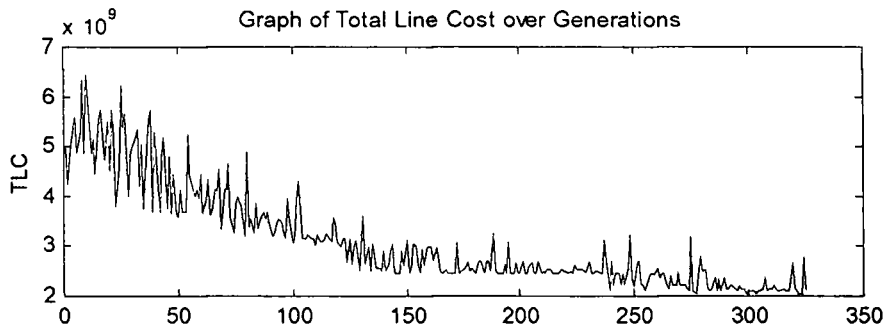
	Minimum	Maximum
Voltage Magnitude	0.974 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	0.00 deg @ bus 1	24.87 deg @ bus 6
P Losses ($I^2 \cdot R$)	-	0.00 MW @ line 1-2
Q Losses ($I^2 \cdot X$)	-	63.99 MVAR @ line 4-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	12.60	90.00	0.00
2	0.982	1.751	-	-	240.00	0.00
3	1.000	11.136	300.00	66.67	40.00	0.00
4	0.974	9.421	-	-	160.00	0.00
5	0.976	4.212	-	-	240.00	0.00
6	1.000	24.866	400.00	96.96	-	-
Total:			770.00	176.23	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 \cdot Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	-20.00	12.60	20.00	-11.77	0.000	0.84
2	3	-139.17	-4.32	139.17	27.46	0.000	23.14
2	4	-85.06	10.63	85.06	0.81	0.000	11.44
2	5	-35.77	5.46	35.77	-3.90	0.000	1.56
3	4	11.37	10.31	-11.37	-9.70	0.000	0.60
3	5	109.45	28.89	-109.45	-15.12	0.000	13.78
4	6	-233.69	8.89	233.69	55.09	0.000	63.99
5	6	-166.31	19.01	166.31	41.87	0.000	60.89
Total:						0.000	176.23



Detailed No 4 result;

From	To	Line Type
1	2	Lynx
1	5	350 Oil Filled
2	3	300 XLPE
2	4	350 Oil Filled
2	5	300 XLPE
2	6	Lynx
3	4	300 XLPE
3	5	300 XLPE
4	6	300 XLPE
5	6	Lynx



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Total cost (in millions) = 3730
Total number of lines = 10
Total Line length (in km) = 620

System Summary

How many?		How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	112.9
Loads	5	Load	770.0	0.0
Branches	10	Losses ($I^2 * Z$)	0.00	112.90
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

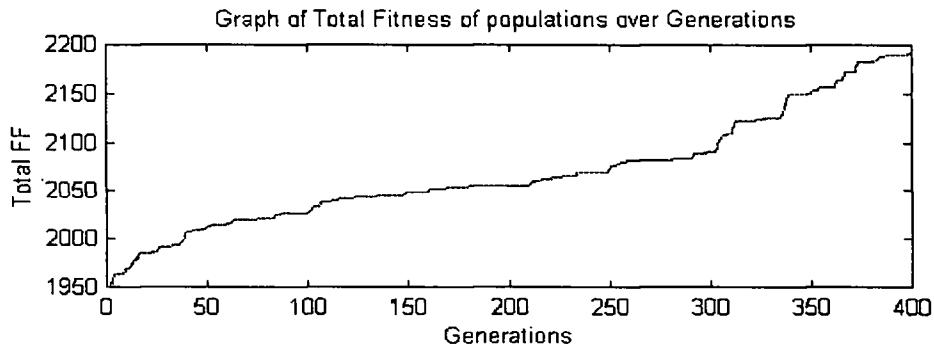
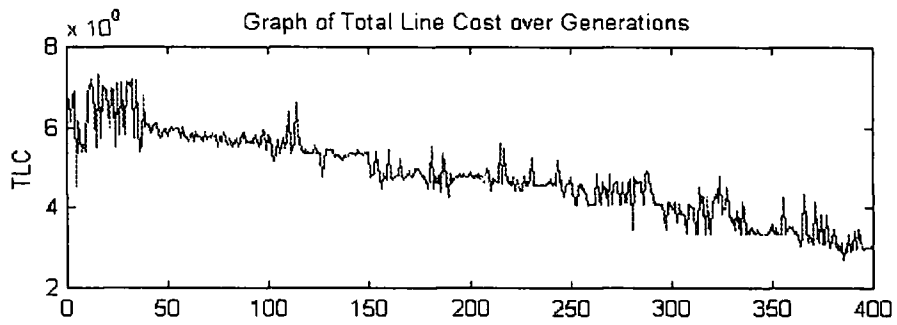
	Minimum	Maximum
Voltage Magnitude	0.990 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	0.00 deg @ bus 1	12.14 deg @ bus 6
P Losses ($I^2 \cdot R$)	-	0.00 MW @ line 1-2
Q Losses ($I^2 \cdot X$)	-	29.65 MVAR @ line 2-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	23.99	90.00	0.00
2	0.991	1.512	-	-	240.00	0.00
3	1.000	8.651	300.00	32.50	40.00	0.00
4	0.990	1.953	-	-	160.00	0.00
5	0.995	0.017	-	-	240.00	0.00
6	1.000	12.142	400.00	56.41	-	-
Total:			770.00	112.90	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 \cdot Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	-18.95	6.82	18.95	-6.26	0.000	0.56
1	5	-1.05	17.17	1.05	-17.09	0.000	0.08
2	3	-98.52	-1.04	98.52	13.40	0.000	12.36
2	4	-22.06	3.37	22.06	-3.20	0.000	0.17
2	5	20.58	-3.04	-20.58	3.59	0.000	0.55
2	6	-158.95	6.97	158.95	22.67	0.000	29.65
3	4	41.98	6.17	-41.98	-1.21	0.000	4.95
3	5	119.50	12.94	-119.50	5.12	0.000	18.06
4	6	-140.08	4.41	140.08	20.65	0.000	25.06
5	6	-100.97	8.37	100.97	13.09	0.000	21.46
Total:						0.000	112.90



Detailed No 5 result;

From To Line Type

1	2	300 XLPE
1	3	350 Oil Filled
1	5	Lynx
2	3	Lynx
2	4	Lynx
2	5	300 XLPE
2	6	300 XLPE
3	4	300 XLPE
3	5	Lynx
3	6	300 XLPE
4	5	300 XLPE
4	6	1xZebra
5	6	300 XLPE



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Total cost (in millions) = 5445
Total number of lines = 13
Total Line length (in km) = 890

System Summary

How many?	How much?	P (MW)	Q (MVAR)	
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	93.9
Loads	5	Load	770.0	0.0
Branches	13	Losses (I ² * Z)	0.00	93.92
Transformers	0	Branch Charging (inj)	-	0.0



Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

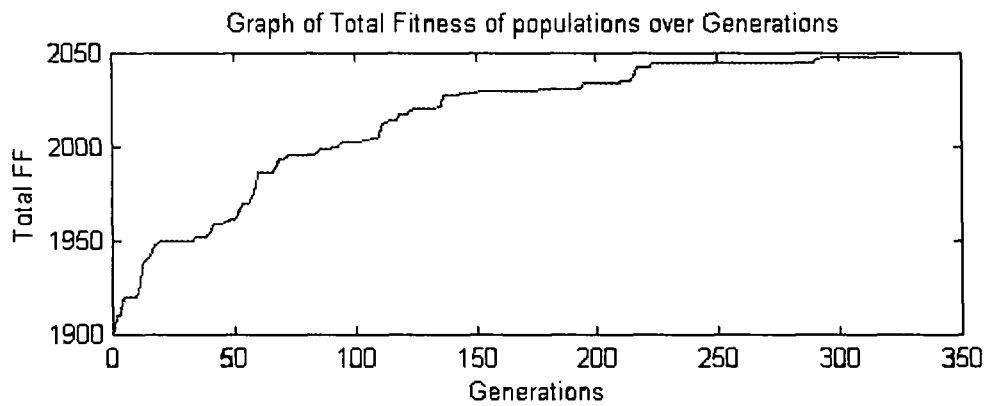
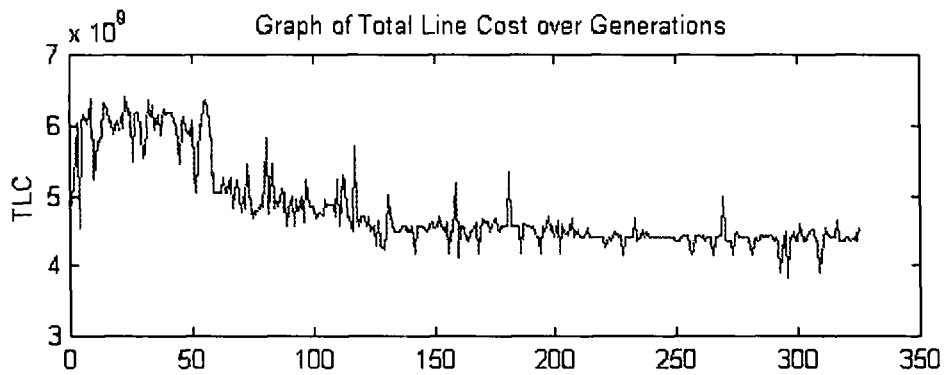
	Minimum	Maximum
Voltage Magnitude	0.991 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	-3.33 deg @ bus 5	7.32 deg @ bus 6
P Losses ($I^2 \cdot R$)	-	0.00 MW @ line 1-2
Q Losses ($I^2 \cdot X$)	-	23.97 MVAR @ line 4-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	15.29	90.00	0.00
2	0.993	-2.573	-	-	240.00	0.00
3	1.000	2.609	300.00	28.77	40.00	0.00
4	0.991	-2.067	-	-	160.00	0.00
5	0.993	-3.327	-	-	240.00	0.00
6	1.000	7.318	400.00	49.86	-	-
Total:			770.00	93.92	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 \cdot Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	29.71	5.55	-29.71	-4.18	0.000	1.37
1	3	-99.83	2.27	99.83	2.27	0.000	4.55
1	5	50.12	7.47	-50.12	-4.51	0.000	2.95
2	3	-77.97	-2.79	77.97	9.89	0.000	7.10
2	4	-6.30	1.23	6.30	-1.17	0.000	0.06
2	5	10.38	-0.25	-10.38	0.39	0.000	0.14
2	6	-136.41	5.99	136.41	17.66	0.000	23.65
3	4	29.38	4.47	-29.38	-2.04	0.000	2.43
3	5	89.31	10.64	-89.31	-1.34	0.000	9.30
3	6	-36.48	1.50	36.48	1.50	0.000	3.00
4	5	8.66	-0.73	-8.66	0.92	0.000	0.19
4	6	-145.58	3.93	145.58	20.04	0.000	23.97
5	6	-81.53	4.54	81.53	10.67	0.000	15.21
Total:						0.000	93.92



Detailed No 6 result;



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

From To Line Type

1	2	1xZebra
1	3	300 XLPE
1	5	Lynx
2	3	300 XLPE
2	4	Lynx
2	5	350 Oil Filled
2	6	2xZebra
3	4	300 XLPE
3	5	300 XLPE
3	6	300 XLPE
4	5	Lynx
4	6	1xGoat
5	6	300 XLPE

Total cost (in millions) = 5540
Total number of lines = 13
Total Line length (in km) = 890

System Summary

How many?		How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0

Committed Gens	3	Generation (current)	770.0	94.8
Loads	5	Load	770.0	0.0
Branches	13	Losses (I ² * Z)	0.00	94.84
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

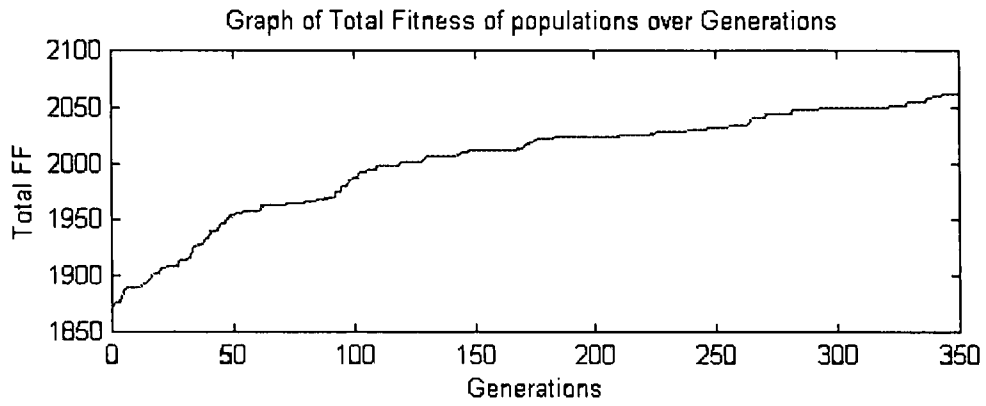
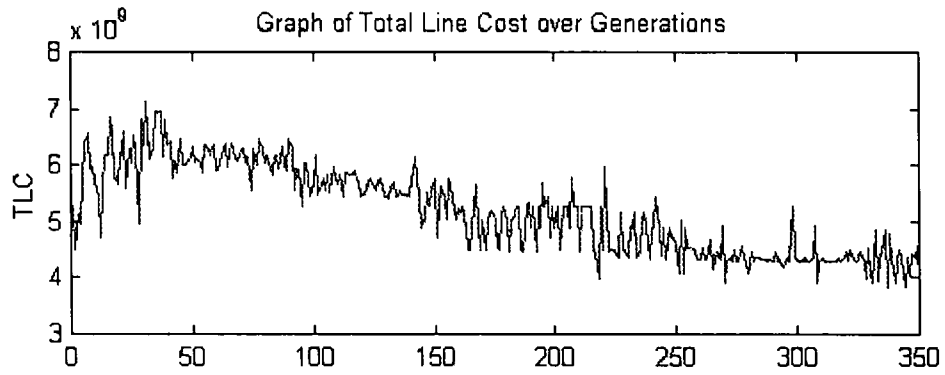
	Minimum	Maximum
Voltage Magnitude	0.991 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	-1.38 deg @ bus 5	8.28 deg @ bus 6
P Losses (I ² *R)	-	0.00 MW @ line 1-2
Q Losses (I ² *X)	-	27.91 MVAR @ line 2-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	14.40	90.00	0.00
2	0.993	-0.734	-	-	240.00	0.00
3	1.000	5.787	300.00	31.05	40.00	0.00
4	0.991	-0.505	-	-	160.00	0.00
5	0.993	-1.385	-	-	240.00	0.00
6	1.000	8.277	400.00	49.38	-	-
Total:			770.00	94.84	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection	To Bus Injection	Loss (I ² * Z)
		P (MW)	Q (MVAR)	P (MW) Q (MVAR)
1	2	9.55	5.52	-9.55 -5.36 0.000 0.16
1	3	-50.42	2.55	50.42 2.55 0.000 5.10
1	5	20.87	6.33	-20.87 -5.79 0.000 0.55
2	3	-90.20	-0.63	90.20 10.95 0.000 10.32
2	4	-2.85	1.00	2.85 -0.99 0.000 0.01
2	5	39.30	-0.74	-39.30 1.18 0.000 0.45
2	6	-176.70	5.72	176.70 22.18 0.000 27.91
3	4	39.51	5.32	-39.51 -0.95 0.000 4.37
3	5	99.18	11.81	-99.18 0.66 0.000 12.47
3	6	-19.31	0.42	19.31 0.42 0.000 0.84
4	5	6.57	-0.67	-6.57 0.77 0.000 0.10
4	6	-129.92	2.61	129.92 17.41 0.000 20.02
5	6	-74.07	3.17	74.07 9.37 0.000 12.54
Total:				0.000 94.84



Detailed No 7 result;

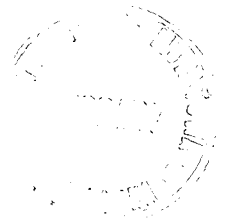


University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

From To Line Type

1	2	300 XLPE
1	3	Lynx
1	4	300 XLPE
1	5	1xZebra
1	6	Lynx
2	3	1xZebra
2	4	Lynx
2	5	350 Oil Filled
2	6	Lynx
3	5	300 XLPE
3	6	Lynx
4	5	300 XLPE
4	6	300 XLPE
5	6	300 XLPE

Total cost (in millions) = 5720
Total number of lines = 14
Total Line length (in km) = 980



System Summary

How many?		How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	99.7
Loads	5	Load	770.0	0.0
Branches	14	Losses (I ² * Z)	0.00	99.69
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

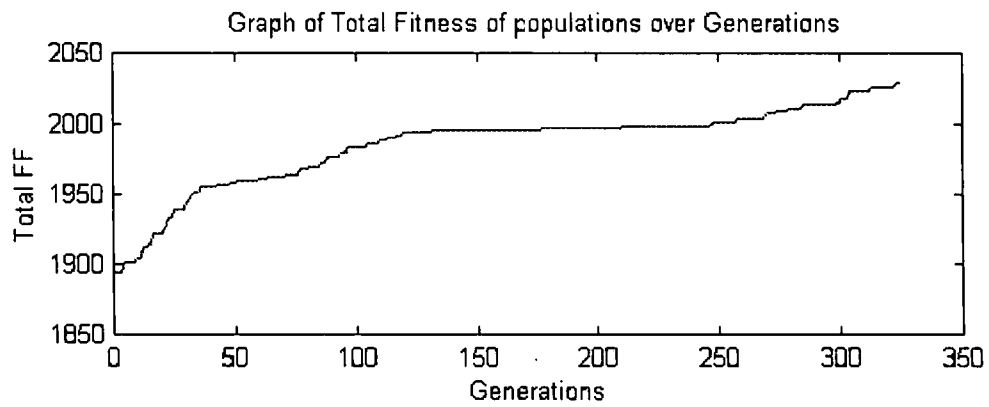
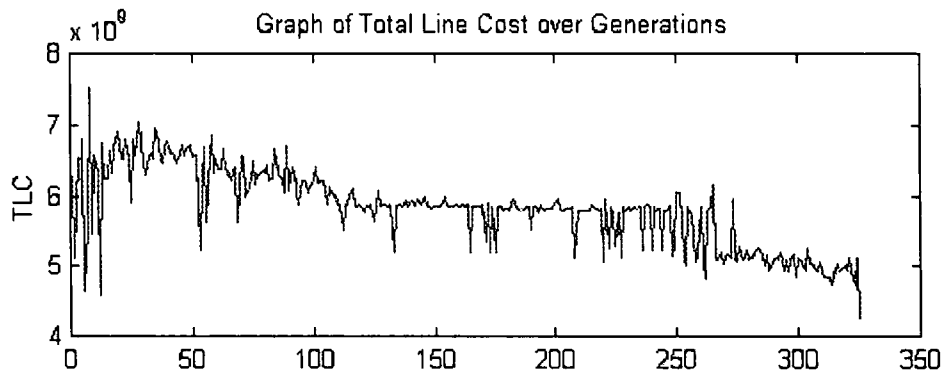
	Minimum	Maximum
Voltage Magnitude	0.992 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	-2.38 deg @ bus 4	7.20 deg @ bus 6
P Losses (I ² *R)	-	0.00 MW @ line 1-2
Q Losses (I ² *X)	-	22.18 MVAR @ line 4-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	22.50	90.00	0.00
2	0.992	-1.856	-	-	240.00	0.00
3	1.000	5.506	300.00	29.97	40.00	0.00
4	0.992	-2.384	-	-	160.00	0.00
5	0.993	-2.273	-	-	240.00	0.00
6	1.000	7.195	400.00	47.23	-	-
Total:			770.00	99.69	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection		To Bus Injection		Loss (I ² * Z)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	21.43	5.36	-21.43	-4.62	0.000	0.73
1	3	-52.15	2.51	52.15	2.51	0.000	5.02
1	4	20.63	4.55	-20.63	-3.65	0.000	0.89
1	5	35.48	7.23	-35.48	-5.78	0.000	1.46
1	6	-45.38	2.85	45.38	2.85	0.000	5.71
2	3	-114.58	0.65	114.58	14.14	0.000	14.79
2	4	6.58	0.55	-6.58	-0.49	0.000	0.06
2	5	25.19	-0.84	-25.19	1.02	0.000	0.18
2	6	-135.77	4.26	135.77	17.28	0.000	21.54
3	5	107.51	13.11	-107.51	1.55	0.000	14.66
3	6	-14.24	0.21	14.24	0.21	0.000	0.42
4	5	-0.76	-0.39	0.76	0.39	0.000	0.00
4	6	-132.03	4.53	132.03	17.65	0.000	22.18
5	6	-72.58	2.81	72.58	9.23	0.000	12.05
Total:						0.000	99.69



Detailed No 8 result;



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

From To Line Type

1	2	500 XLPE
1	3	300 XLPE
1	4	300 XLPE
1	5	300 XLPE
1	6	Lynx
2	3	Lynx
2	4	300 XLPE
2	5	300 XLPE
2	6	300 XLPE
3	4	Lynx
3	5	1xZebra
3	6	Lynx
4	5	Lynx
4	6	300 XLPE
5	6	Lynx

Total cost (in millions) = 6250

Total number of lines = 15

Total Line length (in km) = 1090

System Summary

How many?		How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	93.4
Loads	5	Load	770.0	0.0
Branches	15	Losses ($I^2 * Z$)	0.00	93.37
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

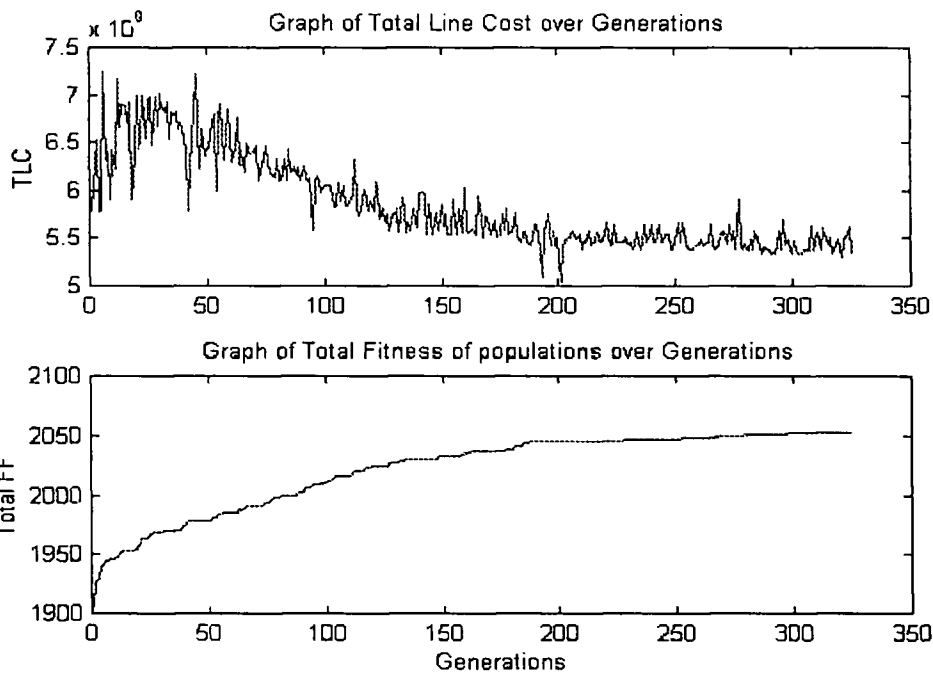
	Minimum	Maximum
Voltage Magnitude	0.994 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	-2.09 deg @ bus 5	7.75 deg @ bus 6
P Losses ($I^2 * R$)	-	0.00 MW @ line 1-2
Q Losses ($I^2 * X$)	-	19.45 MVAR @ line 2-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	22.52	90.00	0.00
2	0.995	-1.213	-	-	240.00	0.00
3	1.000	4.896	300.00	28.68	40.00	0.00
4	0.994	-1.014	-	-	160.00	0.00
5	0.994	-2.089	-	-	240.00	0.00
6	1.000	7.749	400.00	42.17	-	-
Total:			770.00	93.37	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 * Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	33.75	8.61	-33.75	-7.85	0.000	0.76
1	3	-42.67	1.82	42.67	1.82	0.000	3.65
1	4	8.79	3.25	-8.79	-3.07	0.000	0.18
1	5	28.98	5.53	-28.98	-4.44	0.000	1.09
1	6	-48.85	3.31	48.85	3.31	0.000	6.62
2	3	-92.06	0.46	92.06	9.39	0.000	9.85
2	4	-2.29	0.80	2.29	-0.79	0.000	0.01
2	5	12.09	0.97	-12.09	-0.78	0.000	0.19
2	6	-123.98	5.62	123.98	13.83	0.000	19.45
3	4	40.44	4.60	-40.44	-0.40	0.000	4.19
3	5	108.87	12.27	-108.87	1.05	0.000	13.32
3	6	-24.04	0.60	24.04	0.60	0.000	1.20
4	5	8.05	0.03	-8.05	0.12	0.000	0.15
4	6	-121.10	4.23	121.10	14.36	0.000	18.59
5	6	-82.02	4.06	82.02	10.08	0.000	14.14
Total:						0.000	93.37



Detailed No 9 result;

From To Line Type

1	2	350 Oil Filled
1	3	300 XLPE
1	4	300 XLPE
1	5	Lynx
1	6	Lynx
2	3	500 XLPE
2	4	300 XLPE
2	5	300 XLPE
2	6	500 XLPE
3	4	Lynx
3	5	Lynx
3	6	300 XLPE
4	5	Lynx
4	6	1xZebra
5	6	Lynx



Total cost (in millions) = 6330
 Total number of lines = 15
 Total Line length (in km) = 1090

System Summary

How many?	How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0 -900.0 to +900.0
Generators	3	On-line Capacity	1110.0 -900.0 to +900.0
Committed Gens	3	Generation (current)	770.0 68.3
Loads	5	Load	770.0 0.0
Branches	15	Losses ($I^2 * Z$)	0.00 68.28



Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

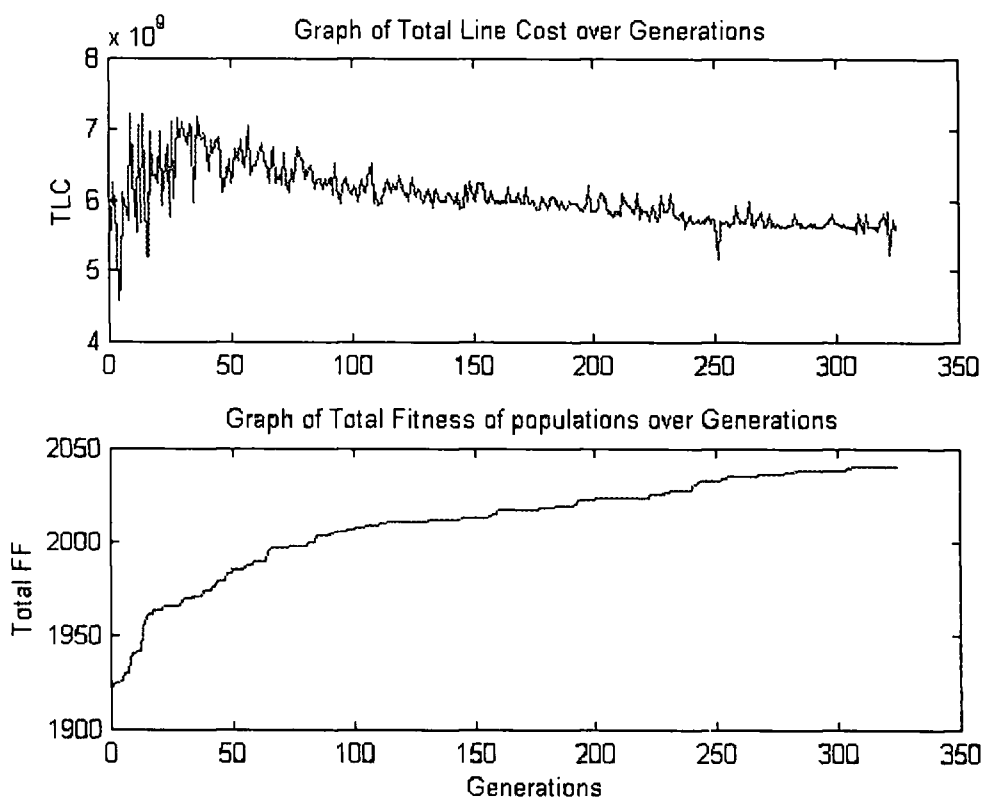
	Minimum	Maximum
Voltage Magnitude	0.996 p.u. @ bus 5	1.000 p.u. @ bus 1
Voltage Angle	-2.50 deg @ bus 5	5.43 deg @ bus 6
P Losses ($I^2 \cdot R$)	-	0.00 MW @ line 1-2
Q Losses ($I^2 \cdot X$)	-	16.74 MVAR @ line 2-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	16.09	90.00	0.00
2	0.998	0.078	-	-	240.00	0.00
3	1.000	3.585	300.00	20.92	40.00	0.00
4	0.996	-1.361	-	-	160.00	0.00
5	0.996	-2.496	-	-	240.00	0.00
6	1.000	5.430	400.00	31.27	-	-
Total:			770.00	68.28	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 \cdot Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	-3.98	6.79	3.98	-6.77	0.000	0.02
1	3	-31.27	0.98	31.27	0.98	0.000	1.96
1	4	11.83	2.15	-11.83	-1.86	0.000	0.29
1	5	37.71	4.55	-37.71	-2.89	0.000	1.66
1	6	-34.29	1.63	34.29	1.63	0.000	3.25
2	3	-117.37	-0.86	117.37	8.06	0.000	7.20
2	4	16.54	1.34	-16.64	-0.92	0.000	0.42
2	5	35.69	2.37	-35.69	-0.77	0.000	1.61
2	6	-178.96	3.91	178.96	12.83	0.000	16.74
3	4	33.95	3.05	-33.95	-0.12	0.000	2.94
3	5	91.73	8.60	-91.73	1.16	0.000	9.76
3	6	-14.31	0.23	14.31	0.23	0.000	0.46
4	5	8.54	0.20	-8.54	-0.03	0.000	0.17
4	6	-106.11	2.69	106.11	9.92	0.000	12.61
5	6	-66.33	2.53	66.33	6.67	0.000	9.20
Total:						0.000	68.28



Detailed No 10 result;



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

From	To	Line Type
1	2	Lynx
1	3	500 Oil Filled
1	4	300 XLPE
1	5	Lynx
1	6	Lynx
2	3	Lynx
2	4	300 XLPE
2	5	300 XLPE
2	6	300 XLPE
3	4	300 XLPE
3	5	1xGoat
3	6	Lynx
4	5	300 XLPE
4	6	Lynx
5	6	1xZebra

Total cost (in millions) = 6475
 Total number of lines = 15
 Total Line length (in km) = 1090

System Summary

|

How many?		How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	86.7
Loads	5	Load	770.0	0.0
Branches	15	Losses ($I^2 * Z$)	0.00	86.69
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

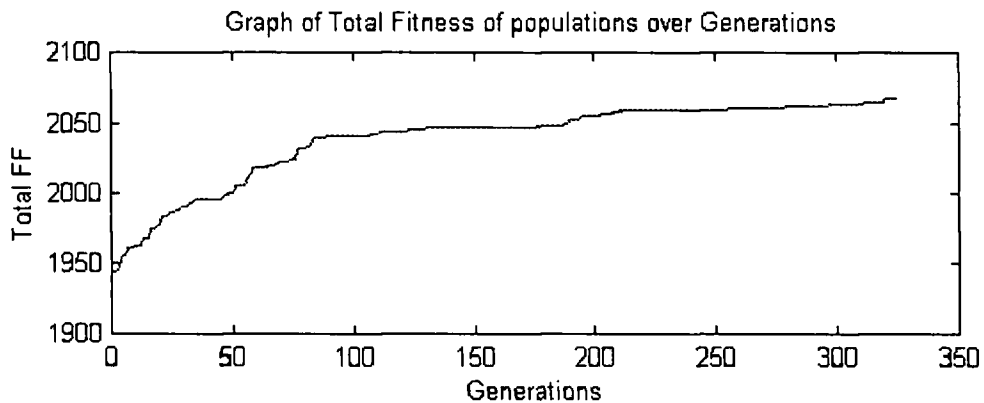
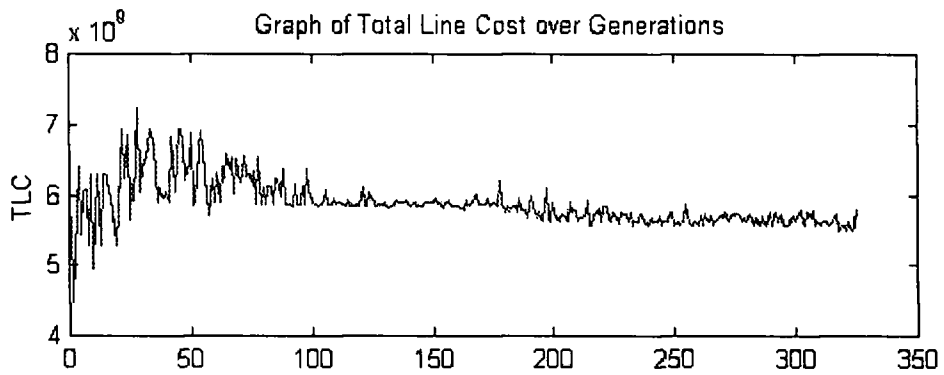
	Minimum	Maximum
Voltage Magnitude	0.994 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	-3.55 deg @ bus 5	5.99 deg @ bus 6
P Losses ($I^2 * R$)	-	0.00 MW @ line 1-2
Q Losses ($I^2 * X$)	-	19.41 MVAR @ line 2-6

Bus Data

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	19.68	90.00	0.00
2	0.994	-2.963	-	-	240.00	0.00
3	1.000	2.352	300.00	25.06	40.00	0.00
4	0.994	-2.349	-	-	160.00	0.00
5	0.994	-3.548	-	-	240.00	0.00
6	1.000	5.991	400.00	41.95	-	-
Total:			770.00	86.69	770.00	0.00

Branch Data

From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 * Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	37.23	5.37	-37.23	-3.42	0.000	1.95
1	3	-93.27	1.91	93.27	1.91	0.000	3.83
1	4	20.37	3.53	-20.37	-2.68	0.000	0.85
1	5	53.50	6.89	-53.50	-3.54	0.000	3.35
1	6	-37.81	1.98	37.81	1.98	0.000	3.96
2	3	-80.06	-1.54	80.06	9.01	0.000	7.46
2	4	-7.05	0.13	7.05	-0.05	0.000	0.08
2	5	8.08	-0.01	-8.08	0.10	0.000	0.08
2	6	-123.75	4.85	123.75	14.56	0.000	19.41
3	4	29.62	3.48	-29.62	-1.03	0.000	2.45
3	5	87.71	9.68	-87.71	-0.61	0.000	9.07
3	6	-30.66	0.97	30.66	0.97	0.000	1.95
4	5	8.27	0.00	-8.27	0.17	0.000	0.17
4	6	-125.34	3.76	125.34	14.55	0.000	18.31
5	6	-82.44	3.89	82.44	9.89	0.000	13.78
Total:						0.000	86.69



When this program run for different population sizes and different number of generations the result given by the program has some problems like increasing the number of over loaded line, under loaded lines and decrease the line fitness of the final results. By analyzing the following results that can be verified.

Detailed No 11 result;
Population size=40 and Number of generations=325

From	To	Line Type
1	2	300 XLPE
1	3	300 XLPE
1	5	1xZebra
2	3	Lynx
2	4	Lynx
2	5	1xGoat
3	5	Lynx
3	6	300 XLPE
4	5	300 XLPE
4	6	Bear

Total cost (in millions) = 3970
Total number of lines = 10
Total Line length (in km) = 640

Newton's method power flow converged in 4 iterations.

| System Summary |

Converged in 0.01 seconds

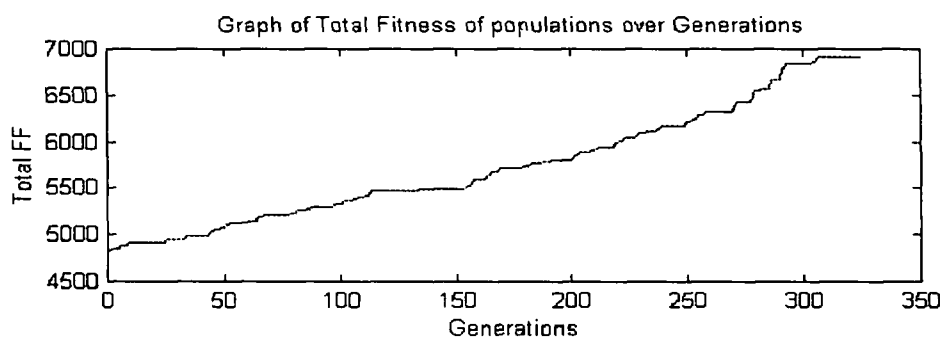
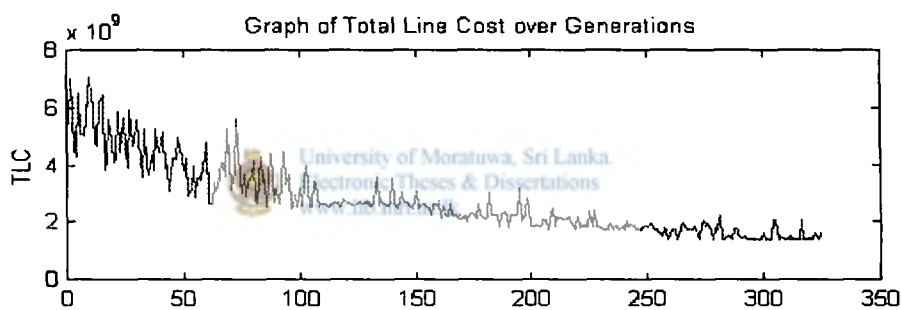
How many?		How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	207.2
Loads	5	Load	770.0	0.0
Branches	10	Losses (I ² * Z)	0.00	207.21
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0

	Minimum	Maximum
Voltage Magnitude	0.963 p.u. @ bus 4	1.000 p.u. @ bus 1
Voltage Angle	-2.12 deg @ bus 5	23.38 deg @ bus 6
P Losses (I ² *R)	0.00 MW	@ line 1-2
Q Losses (I ² *X)	93.98 MVAR	@ line 4-6

| Bus Data |

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	33.21	90.00	0.00
2	0.981	-1.702	-	-	240.00	0.00
3	1.000	8.310	300.00	78.04	40.00	0.00
4	0.963	4.867	-	-	160.00	0.00
5	0.984	-2.123	-	-	240.00	0.00
6	1.000	23.379	400.00	95.96	-	-
Total:			770.00	207.21	770.00	0.00

Branch Data							
From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 * Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	2	19.43	12.83	-19.43	-12.02	0.000	0.81
1	3	-72.27	5.25	72.27	5.25	0.000	10.50
1	5	32.84	15.12	-32.84	-13.67	0.000	1.45
2	3	-148.34	-3.06	148.34	29.36	0.000	26.30
2	4	-78.33	17.34	78.33	-8.12	0.000	9.23
2	5	6.09	-2.26	-6.09	2.31	0.000	0.05
3	5	154.94	28.15	-154.94	0.36	0.000	28.52
3	6	-115.54	15.28	115.54	15.28	0.000	30.56
4	5	46.13	-5.19	-46.13	11.00	0.000	5.81
4	6	-284.46	13.31	284.46	80.67	0.000	93.98
Total:						0.000	207.21



Detailed No 12 result;
 Population size=20 and Number of generations=1000

From	To	Line Type
1	5	Lynx
2	3	Lynx
3	5	300 XLPE
4	6	300 XLPE

5 6 Lynx

Total cost (in millions) = 1600
 Total number of lines = 5
 Total Line length (in km) = 290

Newton's method power flow converged in 4 iterations.

| System Summary |

Converged in 0.00 seconds

How many?		How much?	P (MW)	Q (MVAR)
Buses	6	Total Gen Capacity	1110.0	-900.0 to +900.0
Generators	3	On-line Capacity	1110.0	-900.0 to +900.0
Committed Gens	3	Generation (current)	770.0	241.4
Loads	5	Load	770.0	0.0
Branches	5	Losses ($I^2 * Z$)	0.00	241.42
Transformers	0	Branch Charging (inj)	-	0.0
Areas	1	Shunt (inj)	0.0	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0



University of Moratuwa, Sri Lanka.
 Electronic Theses & Dissertations
 www.theses.lanka.ac.lk

Minimum

Maximum

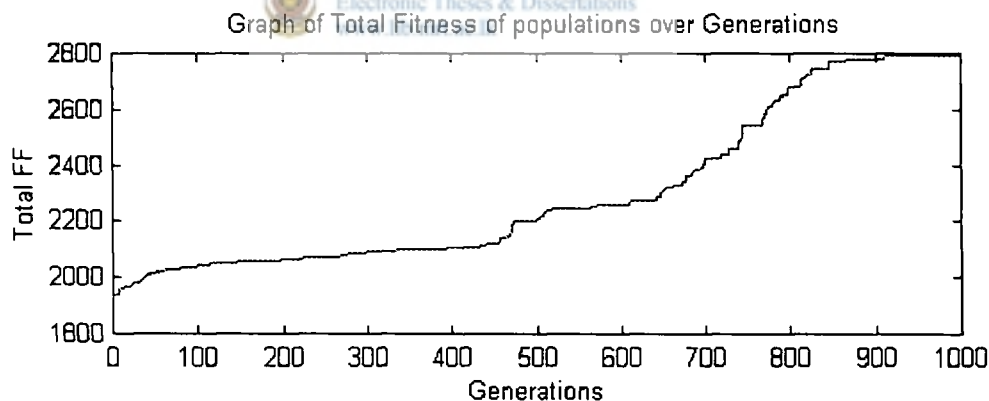
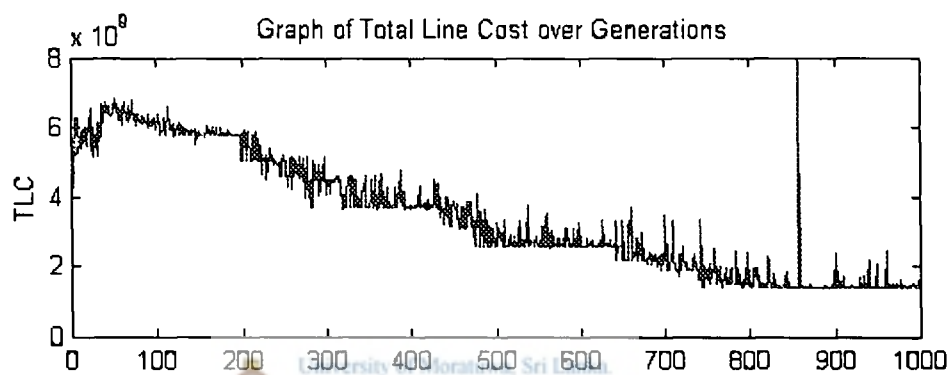
Voltage Magnitude	0.958 p.u. @ bus 2	1.000 p.u. @ bus 1
Voltage Angle	-13.91 deg @ bus 2	32.24 deg @ bus 6
P Losses ($I^2 * R$)	-	0.00 MW @ line 1-5
Q Losses ($I^2 * X$)	-	133.07 MVAR @ line 5-6

| Bus Data |

Bus #	Voltage		Generation		Load	
	Mag(pu)	Ang(deg)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	1.000	0.000	70.00	28.11	90.00	0.00
2	0.958	-13.911	-	-	240.00	0.00
3	1.000	2.842	300.00	98.14	40.00	0.00
4	0.979	20.453	-	-	160.00	0.00
5	0.968	1.362	-	-	240.00	0.00
6	1.000	32.242	400.00	115.17	-	-
Total:			770.00	241.42	770.00	0.00



Branch Data							
From Bus	To Bus	From Bus Injection		To Bus Injection		Loss ($I^2 * Z$)	
		P (MW)	Q (MVAR)	P (MW)	Q (MVAR)	P (MW)	Q (MVAR)
1	5	-20.00	28.11	20.00	-26.74	0.000	1.37
2	3	-240.00	0.00	240.00	72.24	0.000	72.24
3	5	20.00	25.90	-20.00	-24.56	0.000	1.34
4	6	-160.00	0.00	160.00	33.39	0.000	33.39
5	6	-240.00	51.30	240.00	81.77	0.000	133.07
Total:						0.000	241.42



6.2 Conclusion

The research reported in this thesis clearly demonstrates that a GA approach to a Transmission Network Planning problem is both feasible and advantageous. It provides to optimize several parameters in the same time.

Furthermore, it allows the representation of non-linearities which are hard to include in pure mathematical programming methods; in fact, the existence of non-linearities

enhances the advantages of using GA against pure mathematical programming. These non-linearities arise not only from the non-linear character of objective functions and constraints but also from the discrete nature of many aspects of the distribution planning problem. These in some cases could lead to a non-convex domain, but GA are able to deal with such environments and can detect local minima or even “islands” of solutions.

The result of a GA are a generation of solutions filtered through the struggle for survival. Therefore, many interesting and valuable exercises on comparisons and trade offs may be executed, helping the planner to gain insight on the problem he is faced with and allowing field for better decisions to be taken.

Since GA program starts with randomly generated solutions and ad all other operations randomly, we can obtained several different results for the same problem. So can select the best among them.



 **APPENDIX A**
www.lib.mrt.ac.lk

[Faint circular stamp or mark]

MATPOWER

A MATLAB™ Power System Simulation Package

Version 2.0

December 24, 1997



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

User's Manual

Ray D. Zimmerman
rz10@cornell.edu

Deqiang (David) Gan
deqiang@ee.cornell.edu

© 1997 Power Systems Engineering Research Center (PSERC)
School of Electrical Engineering, Cornell University, Ithaca, NY 14853

1 Introduction

What is MATPOWER?

MATPOWER is a package of Matlab m-files for solving power flow and optimal power flow problems. It is intended as a simulation tool for researchers and educators which will be easy to use and modify. *MATPOWER* is designed to give the best performance possible while keeping the code simple to understand and modify. The *MATPOWER* home page can be found at:

<http://www.pserc.cornell.edu/matpower/matpower.html>

Where did it come from?

MATPOWER was developed by Ray Zimmerman and Deqiang Gan of PSERC at Cornell University (<http://www.pserc.cornell.edu/>) under the direction of Robert Thomas. The initial need for Matlab based power flow and optimal power flow code was born out of the computational requirements of the PowerWeb project (see <http://www.pserc.cornell.edu/powerweb/>).

Who can use it?

MATPOWER is free. Anyone may use it. Anyone may modify it for their own use as long as the original copyright notices remain in place. Please don't distribute modified versions of *MATPOWER* without written permission from us.

2 Getting Started

2.1 System Requirements

To use *MATPOWER* you will need a Mac, UNIX machine, or PC with:

- Matlab 4 or higher (available from The MathWorks)
- Matlab Optimization Toolbox (available from The MathWorks)

2.2 Installation

Step 1: Go to the *MATPOWER* home page² and follow the download instructions.

Step 2: Unpack the archive using the appropriate software for your machine (StuffIt Expander for Mac, gunzip and tar for UNIX, pkzip, WinZip, etc. for PC).

Step 3: Copy all of the m-files in the *MATPOWER* distribution to a location in your Matlab path.

¹ See <http://www.mathworks.com/>

² <http://www.pserc.cornell.edu/matpower/matpower.html>

2.3 Running a Power Flow

To run a simple Newton power flow on the default 9-bus system specified in the file `case.m`, with the default algorithm options, at the Matlab prompt, type:

```
>> runpf
```

To run a power flow on the 118-bus system whose data is in `case118.m`, type:

```
>> runpf('case118')
```

2.4 Running an Optimal Power Flow

To run an optimal power flow on the default 9-bus system specified in the file `case.m`, with the default algorithm options, at the Matlab prompt, type:

```
>> runopf
```

To run an optimal power flow on the 30-bus system whose data is in `case30.m`, type:

```
>> runopf('case30')
```

To run an optimal power flow on the same system, but with the option for *MATPOWER* to shut down (decommit) expensive generators, type:

```
>> runuopf('case30')
```

2.5 Getting Help

As with Matlab's built-in functions and toolbox routines, you can type `help` followed by the name of a command or m-file to get help on that particular function. Nearly all of *MATPOWER*'s m-files have such documentation. For example, the help for `runopf` looks like:

```
>> help runopf
RUNOPF Runs an optimal power flow.
[baseMVA, bus, gen, gencost, branch, f, success, et] = ...
runopf(casename, mpopt, fname)
```

Runs an optimal power flow where `casename` is the name of the m-file (without the `.m` extension) containing the opf data, and `mpopt` is a *MATPOWER* options vector (see 'help mpoption' for details). Uses default options if 2nd parameter is not given, and 'case' if 1st parameter is not given. The results may optionally be printed to a file (appended if the file exists) whose name is given in `fname` (in addition to printing to `STDOUT`). Optionally returns the final values of `baseMVA`, `bus`, `gen`, `gencost`, `branch`, `f`, `success`, and `et`.

MATPOWER also has many options which control the algorithms and the output. Type:

```
>> help mpooption
```

and see *Section 3.5* for more information on *MATPOWER's* options.

3 Technical Reference

3.1 Data File Format

The data files used by *MATPOWER* are simply Matlab m-files which define and return the variables *baseMVA*, *bus*, *branch*, *gen*, *area*, and *gencost*. The *bus*, *branch*, and *gen* variables are matrices. Each row in the matrix corresponds to a single bus, branch, or generator, respectively. The columns are similar to the columns in the standard IEEE and PTI formats. The details of the specification of the *MATPOWER* case file can be found in the help for *case.m*:

```
>> help case
```

```
CASE Defines the power flow data in a format similar to PTI.
```

```
[baseMVA, bus, gen, branch, area, gencost] = case
```

```
The format for the data is similar to PTI format except where noted.
```

```
An item marked with (+) indicates that it is included in this data but is not part of the PTI format. An item marked with (-) is one that is in the PTI format but is not included here.
```

Bus Data Format

```
1 bus number (1 to 29997)
2 bus type
   PQ bus = 1
   PV bus = 2
   reference bus = 3
   isolated bus = 4
3 Pd, real power demand (MW)
4 Qd, reactive power demand (MVAR)
5 Gs, shunt conductance (MW (demanded?) at V = 1.0 p.u.)
6 Bs, shunt susceptance (MVAR (injected?) at V = 1.0 p.u.)
7 area number, 1-100
8 Vm, voltage magnitude (p.u.)
9 Va, voltage angle (degrees)
(-) (bus name)
10 baseKV, base voltage (kV)
11 zone, loss zone (1-999)
(+) 12 maxVm, maximum voltage magnitude (p.u.)
(+) 13 minVm, minimum voltage magnitude (p.u.)
```

Generator Data Format

```
1 bus number
(-) (machine identifier, 0-9, A-Z)
2 Pg, real power output (MW)
3 Qg, reactive power output (MVAR)
4 Qmax, maximum reactive power output (MVAR)
5 Qmin, minimum reactive power output (MVAR)
6 Vg, voltage magnitude setpoint (p.u.)
(-) (remote controlled bus index)
```

- 7 mBase, total MVA base of this machine, defaults to baseMVA
- (-) (machine impedance, p.u. on mBase)
- (-) (step up transformer impedance, p.u. on mBase)
- (-) (step up transformer off nominal turns ratio)
- 8 status, 1 - machine in service, 0 - machine out of service
- (-) (% of total VARS to come from this gen in order to hold V at remote bus controlled by several generators)
- 9 Pmax, maximum real power output (MW)
- 10 Pmin, minimum real power output (MW)

Branch Data Format

- 1 f, from bus number
- 2 t, to bus number
- (-) (circuit identifier)
- 3 r, resistance (p.u.)
- 4 x, reactance (p.u.)
- 5 b, total line charging susceptance (p.u.)
- 6 rateA, MVA rating A (long term rating)
- 7 rateB, MVA rating B (short term rating)
- 8 rateC, MVA rating C (emergency rating)
- 9 ratio, transformer off nominal turns ratio (= 0 for lines)
(taps at 'from' bus, impedance at 'to' bus, i.e. ratio = Vf / Vt)
- 10 angle, transformer phase shift angle (degrees)
- (-) (Gf, shunt conductance at from bus p.u.)
- (-) (Bf, shunt susceptance at from bus p.u.)
- (-) (Gt, shunt conductance at to bus p.u.)
- (-) (Bt, shunt susceptance at to bus p.u.)
- 11 initial branch status, 1 - in service, 0 - out of service

(+) Area Data Format

- 1 i, area number
- 2 price_ref_bus, reference bus for that area

(+) Generator Cost Data Format

NOTE: If gen has n rows, then the first n rows of gencost contain the cost for active power produced by the corresponding generators. If gencost has 2*n rows then rows n+1 to 2*n contain the reactive power costs in the same format.

- 1 model, 1 - piecewise linear, 2 - polynomial
- 2 startup, startup cost in US dollars
- 3 shutdown, shutdown cost in US dollars
- 4 n, number of cost coefficients to follow for polynomial
(or data points for piecewise linear) total cost function
- 5 and following, cost data, piecewise linear data as:
x0, y0, x1, y1, x2, y2, ...
and polynomial data as, e.g.:
c2, c1, c0
where the polynomial is $c0 + c1*P + c2*P^2$

3.2 Power Flow

MATPOWER has three power flow solvers. The default power flow solver is based on a standard Newton's method [11] using a full Jacobian, updated at each iteration. This method is described in detail in many textbooks. The other two power flow solvers are

variations of the fast-decoupled method [9]. *MATPOWER* implements the XB and BX variations as described in [1]. Currently, *MATPOWER*'s power flow solvers do not include any transformer tap changing or feasibility checking capabilities.

Performance of the power flow solvers should be excellent even on very large-scale power systems, since the algorithms and implementation take advantage of Matlab's built-in sparse matrix handling. On a Sun Ultra 2200, *MATPOWER* solves a 9600-bus test case in about 10 seconds, and a 38400 bus case in about 50 seconds.

3.3 Optimal Power Flow

MATPOWER includes two solvers for the optimal power flow (OPF) problem. The first is based on the `constr` function included in Matlab's Optimization Toolbox, which uses a successive quadratic programming technique with a quasi-Newton approximation for the Hessian matrix. The second approach is based on linear programming. It can use the LP solver in the Optimization Toolbox or other Matlab LP solvers available from third parties.

The performance of *MATPOWER*'s OPF solvers depends on several factors. First, the `constr` function uses an algorithm which does not exploit or preserve sparsity, so it is inherently limited to small power systems. The LP-based algorithm, on the other hand, does preserve sparsity. However, the LP-solver included in the Optimization Toolbox does not exploit this sparsity. In fact, the LP-based method with the default LP solver performs worse than the `constr`-based method, even on small systems. Fortunately, there are LP-solvers available from third parties which do exploit sparsity. In general, these yield *much* higher performance. One in particular, called *bpmpd* [7] (actually a QP-solver), has proven to be robust and efficient.

It should be noted, however, that even with a good LP-solver, *MATPOWER*'s LP-based OPF solver, unlike its power flow solver, is not suitable for very-large scale problems. Substantial improvements in performance may still be possible, though they may require significantly more complicated coding and possibly a custom LP-solver. On a Sun Ultra 2200, the LP-based OPF solver using *bpmpd* solves a 30-bus system in under 4 seconds and a 118-bus case in under 25 seconds.

OPF Formulation

The OPF problem solved by *MATPOWER* is a "smooth" OPF with no discrete variables or controls. The objective function is the total cost of real and/or reactive generation. These costs may be defined as polynomials or as piecewise-linear functions of generator output. The problem is formulated as follows:

min

$$P_g, Q_g \quad \sum f_{1i}(P_{gi}) + f_{2i}(Q_{gi})$$

such that ...

$$P_{gi} - P_{Li} - P(V, \theta) = 0 \quad (\text{active power balance equations})$$

$$\begin{aligned}
Q_{gi} - Q_{Li} - Q(V, \theta) &= 0 && \text{(reactive power balance equations)} \\
S_{ij}^f &\leq S_{ij}^{\max} && \text{(apparent power flow limit of lines, from side)} \\
S_{ij}^t &\leq S_{ij}^{\max} && \text{(apparent power flow limit of lines, to side)} \\
V_i^{\min} &\leq V_i \leq V_i^{\max} && \text{(bus voltage limits)} \\
P_{gi}^{\min} &\leq P_{gi} \leq P_{gi}^{\max} && \text{(active power generation limits)} \\
Q_{gi}^{\min} &\leq Q_{gi} \leq Q_{gi}^{\max} && \text{(reactive power generation limits)}
\end{aligned}$$

Here f_i and f_{2i} are the costs of active and reactive power generation, respectively, for generator I at a given dispatch point. Both f_{1i} and f_{2i} are assumed to be a polynomial or piecewise-linear functions. The problem can be written more compactly in the following form:

$$\begin{aligned}
&\min_x f(x_2) \\
&\text{such that ...}
\end{aligned}$$

$$g(x) \leq 0$$

where f and g are non-linear functions.

Optimization Toolbox Based OPF Solver (constr)

The first of the two OPF solvers in *MATPOWER* is based on the `constr` non-linear constrained optimization function in Matlab's Optimization Toolbox. The `constr` function and the algorithms it uses are covered in the Optimization Toolbox manual [5]. *MATPOWER* provides `constr` with two m-files which it uses during for the optimization. One computes the objective function, f , and the constraint violations, g , at a given point, x , and the other computes their gradients $\partial f / \partial x$ and $\partial g / \partial x$.

MATPOWER has two versions of these m-files. One set is used to solve systems with polynomial cost functions. In this formulation, the cost functions are included in a straightforward way into the objective function. The other set is used to solve systems with piecewise-linear costs. Piecewise-linear cost functions are handled by introducing a cost variable for each piecewise-linear cost function. The objective function is simply the sum of these cost variables which are then constrained to lie above each of the linear functions which make up the piecewise-linear cost function. Clearly, this method works only for convex cost functions. In the *MATPOWER* documentation this will be referred to as a constrained cost variable (CCV) formulation.

The algorithm codes 100 and 200, respectively, are used to identify the `constr`-based solver for polynomial and piecewise-linear cost functions. If algorithm 200 is chosen for a system with polynomial cost function, the cost function will be approximated by a piecewise-linear function by evaluating the polynomial at a fixed number of points determined by the options vector (see Section 3.5 for more details on the *MATPOWER*

options). It should be noted that the `constr`-based method can also benefit from a superior QP-solver such as *bpmpd*. See Appendix A for more information on LP and QP-solvers.

LP-Based OPF Solver (LPconstr)

Linear programming based OPF methods are in wide use today in the industry. However, the LPbased algorithm included in *MATPOWER* is much simpler than the algorithms used in productiongrade software.

The LP-based methods in *MATPOWER* use the same problem formulation as the `constr`-based methods, including the CCV formulation for the case of piecewise-linear costs. The compact form of the OPF problem can be rewritten to partition g into equality and inequality constraints, and to partition the variable x as follows:

$$\min_x f(x_2)$$

such that ...

$$g_1(x_1, x_2) = 0 \text{ (equality constraints)}$$

$$g_2(x_1, x_2) \leq 0 \text{ (inequality constraints)}$$

where x_1 contains the system voltage magnitudes and angles, and x_2 contains the generator real and reactive power outputs (and corresponding cost variables for the CCV formulation). This is a general non-linear programming problem, with the additional assumption that the equality constraints can be used to solve for x_1 , given a value for x_2 . The LP-based OPF solver is implemented with a function `LPconstr`, which is similar to `constr` in that it uses the same m-files for computing the objective function, constraints, and their respective gradients. In addition, a third m-file (`lpeqslvr.m`) is needed to solve for x_1 from the equality constraints, given a value for x_2 . This architecture makes it relatively simple to modify the formulation of the problem and still be able to use both the `constr`-based and LP-based solvers.

The algorithm proceeds as follows, where the superscripts denote iteration number:

Step 0: Set iteration counter $k \leftarrow 0$ and choose an appropriate initial value, call it x_{20} , for x_2 .

Step 1: Solve the equality constraint (power flow) equations $g_1(x_{1k}, x_{2k}) = 0$ for x_{1k} .

Step 2: Linearize the problem around x_k , solve the resulting LP for Δx .

$$\min_{\Delta x} \left[\frac{\partial f}{\partial x} \Big|_{x=x^k} \right] \cdot \Delta x$$

such that ...

$$\min_{\Delta x} \left[\frac{\partial f}{\partial x} \right]_{x=x^k} \Delta x \leq -g(x^k)$$

$$-\Delta \leq \Delta x \leq \Delta$$

Step 3: Set $k \leftarrow k+1$, update current solution $x_k = x_{k-1} + \Delta x$.

Step 4: If x_k meets termination criteria, stop, otherwise go to step 5.

Step 5: Adjust step size limit Δ based on the trust region algorithm in [3], go to step 1.

The termination criteria is outlined below:

$$\frac{\partial L}{\partial x} = \frac{\partial f}{\partial x} + \lambda^T \cdot \frac{\partial g}{\partial x} \leq \text{tolerance}_1$$

$$g(x) \leq \text{tolerance}_2$$

$$\Delta x \leq \text{tolerance}_3$$

Here λ is the vector of Lagrange multipliers of the LP problem. The first condition pertains to the size of the gradient, the second to the violation of constraints, and the third to the step size. More detail can be found in [4].

Quite frequently, the value of x_k given by step 1 is infeasible and could result in an infeasible LP problem. In such cases, a slack variable is added for each violated constraint. These slack variables must be zero at the optimal solution.

The `LPconstr` function implements the following three methods:

- sparse formulation with full set of inequality constraints
- sparse formulation with relaxed constraints (ICS, Iterative Constraint Search)
- dense formulation with relaxed constraints (ICS) [10]

These three methods are specified using algorithm codes 160, 140, and 120, respectively, for systems with polynomial costs, and 260, 240, and 220, respectively, for systems with piecewise-linear costs. As with the `constr`-based method, selecting one of the `2xx` algorithms for a system with polynomial cost will cause the cost to be replaced by a piecewise-linear approximation.

In the dense formulation, some of the variables x_1 and the equality constraints g_1 are eliminated from the problem before posing the LP sub-problem. This procedure is outlined below. Suppose the LP sub-problem is given by:

$$\min c^T \cdot \Delta x$$

such that ...

$$A \cdot \Delta x \leq b$$

$$-\Delta \leq \Delta x \leq \Delta$$

If this is rewritten as:

$$\min c_1^T \cdot \Delta x_1 + c_2^T \cdot \Delta x_2$$

such that ...



$$A_{11} \cdot \Delta x_1 + A_{12} \cdot \Delta x_2 = b_1$$

$$A_{21} \cdot \Delta x_1 + A_{22} \cdot \Delta x_2 = b_2$$

$$-\Delta \leq \Delta x \leq \Delta$$

Where A_{11} is a square matrix, Δx_1 can be computed as:

$$\Delta x_1 = A_{11}^{-1} (b_1 - A_{12} \cdot \Delta x_2)$$

Substituting back in to the problem, yields a new LP problem:

$$\min(-c_1^T A_{11}^{-1} A_{12} + c_2^T) \cdot \Delta x_2$$

such that ...

$$A_{11} \cdot \Delta x_1 + A_{12} \cdot \Delta x_2 = b_1$$

$$A_{21} \cdot A_{11}^{-1} (b_1 - A_{12} \Delta x_2) + A_{22} \cdot \Delta x_2 \leq b_2$$

$$-\Delta_1 \leq A_{11}^{-1} (b_1 - A_{12} \Delta x_2) \leq \Delta_1$$

$$-\Delta_2 \leq \Delta x_2 \leq \Delta_2$$

This new LP problem is smaller than the original, but it is no longer sparse. As mentioned above, to realize the full potential of the LP-based OPF solvers, it will be necessary to obtain a good LP-solver, such as *bpmpd*. See Appendix A for more details.

3.4 Unit Decommittment Algorithm



Electronic Theses & Dissertations
www.lib.mrt.ac.lk

The standard OPF formulation described in the previous section has no mechanism for completely shutting down generators which are very expensive to operate. Instead they are simply dispatched at their minimum generation limits. *MATPOWER* includes a unit decommitment algorithm which allows it to shut down these expensive units. The algorithm is based on a simplified version of the decommitment technique proposed in [6].

The algorithm proceeds as follows:

Step 0: Assume all generators are on-line with all generator limits in place.

Step 1: Solve a normal OPF.

Step 2: If the OPF converged to a feasible solution and the objective function decreased from the previous iteration (or if this is the first iteration), go to step 3, otherwise go to step 4.

Step 3: Compute a decommitment index for each generator i as follows:

$$d_i = f_i(P_i) - \lambda_i \cdot P_i$$

where P_i is generator i 's dispatch computed by the OPF, f_i is the cost of operating at P_i , and λ_i is the Lagrange multiplier on the real power equality constraint at the bus where generator i is located. Continue with step 5.

Step 4: Return to the previous commitment and set d_k to zero (to eliminate it from consideration).

Step 5: Find the generator k with the smallest decommitment index. If d_k is negative, shut

Down generator k and return to step 1. If d_k is positive, stop.

3.5 MATPOWER Options

MATPOWER uses an options vector to control the many options available. It is similar to the options vector produced by the `foptions` function in Matlab's Optimization Toolbox. The primary difference is that modifications can be made by option name, as opposed to having to remember the index of each option. The default *MATPOWER* options vector is obtained by calling `mpoption` with no arguments. So, typing:

```
>> runopf('case30', mption)
```

is another way to run the OPF solver with the all of the default options.

The *MATPOWER* options vector controls the following:

- power flow algorithm
- power flow termination criterion
- OPF algorithm
- OPF default algorithms for different cost models
- OPF cost conversion parameters
- OPF termination criterion
- verbose level
- printing of results

The details are given below:

```
>> help mption
```

MPTION Used to set and retrieve a *MATPOWER* options vector.

```
opt = mption
      returns the default options vector
opt = mption(name1, value1, name2, value2, ...)
      returns the default options vector with new values for up to 7
      options, name# is the name of an option, and value# is the new
      value. Example: options = mption('PF_ALG', 2, 'PF_TOL', 1e-4)
opt = mption(opt, name1, value1, name2, value2, ...)
      same as above except it uses the options vector opt as a base
      instead of the default options vector.
      The currently defined options are as follows:
```

idx - NAME, default	description [options]
---	-----

power flow options

```

1 - PF_ALG, 1                power flow algorithm
    [ 1 - Newton's method                    ]
    [ 2 - Fast-Decoupled (XB version)         ]
    [ 3 - Fast-Decoupled (BX version)         ]
2 - PF_TOL, 1e-8            termination tolerance on per unit
                             P & Q mismatch
3 - PF_MAX_IT, 10           maximum number of iterations for
                             Newton's method
4 - PF_MAX_IT_FD, 30        maximum number of iterations for
                             fast decoupled method

```

OPF options

```

11 - OPF_ALG, 0             algorithm to use for OPF
    [ (see README for more info on formulations/algorithms) ]
    [ The algorithm code = F * 100 + S * 20, where ...       ]
    [ F specifies one of the following OPF formulations      ]
    [ 1 - standard (polynomial cost in obj fcn)             ]
    [ 2 - CCV (constrained cost variables)                  ]
    [ S specifies one of the following solvers                ]
    [ 0 - 'constr' from Optimization Toolbox                 ]
    [ 1 - Dense LP-based method                             ]
    [ 2 - Sparse LP-based method w/relaxed constraints]
    [ 3 - Sparse LP-based method w/full constraints         ]
    [ This yields the following 9 codes:                     ]
    [ 0 - choose appropriate default from OPF_ALG_POLY      ]
    [ or OPF_ALG_PWL                                        ]
    [ 100 - standard formulation, constr                    ]
    [ 120 - standard formulation, dense LP                  ]
    [ 140 - standard formulation, sparse LP (relaxed) ]
    [ 160 - standard formulation, sparse LP (full)         ]
    [ 200 - CCV formulation, constr                        ]
    [ 220 - CCV formulation, dense LP                      ]
    [ 240 - CCV formulation, sparse LP (relaxed)           ]
    [ 260 - CCV formulation, sparse LP (full)              ]
12 - OPF_ALG_POLY, 100     default OPF algorithm for use with
                             polynomial cost functions
13 - OPF_ALG_PWL, 200      default OPF algorithm for use with
                             piece-wise linear cost functions
14 - OPF_POLY2PWL_PTS, 10  number of evaluation points to use
                             when converting from polynomial to
                             piece-wise linear costs
15 - OPF_NEQ, 0            number of equality constraints
                             0 => 2*nb, set by program, not a
                             user option)
16 - OPF_VIOLATION, 5e-6   constraint violation tolerance
17 - CONSTR_TOL_X, 1e-4    termination tol on x for 'constr'
18 - CONSTR_TOL_F, 1e-4    termination tol on F for 'constr'
19 - CONSTR_MAX_IT, 0       max number of iterations for 'constr'
                             [ 0 => 2*nb + 150 ]
20 - LPC_TOL_GRAD, 3e-3     termination tolerance on gradient
                             for 'LPconstr'
21 - LPC_TOL_X, 5e-3        termination tolerance on x (min
                             step size) for 'LPconstr'
22 - LPC_MAX_IT, 1000       maximum number of iterations for
                             'LPconstr'

```


23 - LPC_MAX_RESTART, 5 maximum number of restarts for
'LPconstr'

output options

```
31 - VERBOSE, 1 amount of progress info printed
    [ 0 - print no progress info ]
    [ 1 - print a little progress info ]
    [ 2 - print a lot of progress info ]
    [ 3 - print all progress info ]
32 - OUT_ALL, -1 controls printing of results
    [ -1 - individual flags control what prints ]
    [ 0 - don't print anything ]
    [ (overrides individual flags, except OUT_RAW) ]
    [ 1 - print everything ]
    [ (overrides individual flags, except OUT_RAW) ]
33 - OUT_SYS_SUM, 1 print system summary [ 0 or 1 ]
34 - OUT_AREA_SUM, 0 print area summaries [ 0 or 1 ]
35 - OUT_BUS, 1 print bus detail [ 0 or 1 ]
36 - OUT_BRANCH, 1 print branch detail [ 0 or 1 ]
37 - OUT_GEN, 0 print generator detail [ 0 or 1 ]
    (OUT_BUS also includes gen info)
38 - OUT_ALL_LIM, -1 control constraint info output
    [ -1 - individual flags control what constraint info prints ]
    [ 0 - no constraint info (overrides individual flags) ]
    [ 1 - binding constraint info (overrides individual flags) ]
    [ 2 - all constraint info (overrides individual flags) ]
39 - OUT_V_LIM, 1 control output of voltage limit info
    [ 0 - don't print ]
    [ 1 - print binding constraints only ]
    [ 2 - print all constraints ]
    [ (same options for OUT_LINE_LIM, OUT_PG_LIM, OUT_QG_LIM) ]
40 - OUT_LINE_LIM, 1 control output of line limit info
41 - OUT_PG_LIM, 1 control output of gen P limit info
42 - OUT_QG_LIM, 1 control output of gen Q limit info
43 - OUT_RAW, 0 print raw data for Perl database
    interface code [ 0 or 1 ]
```

A typical usage of the options vector might be as follows:

Get the default options vector:

```
>> opt = mption;
```

Use the fast-decoupled method to solve power flow:

```
>> opt = mption(opt, 'PF_ALG', 2);
```

Display only system summary and generator info:

```
>> opt = mption(opt, 'OUT_BUS', 0, 'OUT_BRANCH', 0, 'OUT_GEN', 1);
```

Show all progress info:

```
>> opt = mption(opt, 'VERBOSE', 3);
```



Now, run a bunch of power flows using these settings:

```
>> runpf('case57', opt)
>> runpf('case118', opt)
>> runpf('case300', opt)
```

3.6 Summary of the Files

Documentation files:

```
README      - basic intro to MATPOWER
CHANGES    - modification history of MATPOWER
manual.pdf  - PDF version of the MATPOWER User's Manual
              (requires Adobe Acrobat Reader)
```

Input data files:

```
cdf2matp.m - a stand-alone m-file which reads IEEE CDF formatted
              data and outputs data in MATPOWER's case.m format
case.m      - same as case9.m
case9.m     - a 3 generator, 9 bus case
case30.m    - a 6 generator, 30 bus case
case57.m    - IEEE 57-Bus case
case118.m   - IEEE 118-Bus case
case300.m   - IEEE 300-Bus case
case9Q.m    - case9.m, with costs for reactive generation
case30Q.m   - case30.m, with costs for reactive generation
case30pwl.m - case30.m with a piece-wise linear cost function
```

Source files used by all algorithms: mrt.ac.lk

```
bustypes.m
dSbus_dV.m - computes partial derivatives for Jacobian
ext2int.m
idx_brch.m
idx_bus.m
idx_gen.m
int2ext.m
makeSbus.m
makeYbus.m - builds Ybus matrix
mpoption.m - sets MATPOWER options
printpf.m  - prints output
```

Other source files used by PF (Power Flow):

```
fdpf.m      - implements fast decoupled power flow
newtonpf.m  - implements Newton's method power flow
pfsoln.m
runpf.m     - main program for running a power flow
makeB.m
```

Other source files used by OPF (Optimal Power Flow):

```
dAbr_dV.m   - computes partial derivatives of apparent power flows
dSbr_dV.m   - computes partial derivatives of complex power flows
fg_names.m
```

fun_ccv.m - computes obj fcn and constraints for CCV formulation
 fun_std.m - computes obj fcn and constraints for standard formulation
 grad_ccv.m - computes gradients for standard formulation
 grad_std.m - computes gradients for standard formulation
 idx_area.m
 idx_cost.m
 opf.m - implements main OPF routine
 opfsoln.m
 opf_form.m
 opf_slvr.m
 poly2pwl.m
 pqcost.m
 runopf.m - main program for running an optimal power flow
 totcost.m - computes cost

The following are used only by the LP-based OPF algorithms:

LPconstr.m
 LPeqslvr.m
 LPrelax.m
 LPsetup.m

Other source files used by UOPF (Unit decommitment/OPF):

(all files from OPF, except runopf.m)
 uopf.m - implements decommitment heuristic
 runuopf.m - main program for running OPF with decommitment algorithm

Files for use with the  *bpmpd* LP/QP-solver: University of Moratuwa, Sri Lanka. Dissertations www.lib.mrt.ac.lk

bpmpd/lp.m-replacement for Optimization Toolbox lp.m
 bpmpd/qp.m-replacement for Optimization Toolbox qp.m (used by constr.m)

4 Acknowledgments

The authors would like to acknowledge contributions from several people. Thanks to Carlos Murillo-Sánchez for suggesting the CCV formulation for handling piecewise linear costs in the OPF, for his help on the decommitment algorithm, and for creating the Matlab MEX interface to the *bpmpd* LP and QP solver. Thanks to Chris DeMarco, one of our PSERC associates at the University of Wisconsin, for the technique for building the Jacobian matrix. Our appreciation to Bruce Wollenberg for all of his suggestions for improvements to version 1. The enhanced output functionality in version 2.0 are primarily due to his input. Thanks also to Andrew Ward for code which helped us verify and test the ability of the OPF to optimize reactive power costs. Last but not least, we'd like to acknowledge the input of Bob Thomas throughout the of development of *MATPOWER* here at PSERC Cornell.



Genetic Algorithms: Diophantine Equation Solver

This is a C++ program that solves a diophantine equation using genetic algorithms

CDiophantine

Firstly the class header (note for formatting reasons, a lot of the documentation is taken out):

```
#include <stdlib.h>
#include <time.h>

#define MAXPOP 25

struct gene {
    int alleles[4];
    int fitness;
    float likelihood;

    // Test for equality.
    operator==(gene gn) {
        for (int i=0;i<4;i++) {
            if (gn.alleles[i] != alleles[i]) return false;
        }
        return true;
    }
};

class CDiophantine {
public:
    CDiophantine(int, int, int, int, int);
    int Solve();

    // Returns a given gene.
    gene GetGene(int i) { return population[i];}

protected:
    int ca,cb,cc,cd;
    int result;
    gene population[MAXPOP];

    int Fitness(gene &);
    void GenerateLikelihoods();
    float MultInv();inverse.
    int CreateFitnesses();
```



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

```

void CreateNewPopulation();
    int GetIndex(float val);

    gene Breed(int p1, int p2);

};

```

Firstly you notice that there are two structures, the gene structure and the actual CDiophantine class. The gene structure is used to keep track of the different solution sets. The population generated is a population of genes. The gene structure keeps track of its own fitness and likelihood values itself

Fitness function

The fitness functions calculate the fitness of each gene. In our case the fitness function is the difference between the calculated value of the gene and the result we want. This class uses two functions, one that calculates all the fitnesses and another smaller one (you should probably make the function inline) to calculate it per gene.

```

int CDiophantine::Fitness(gene &gn) {
    int total = ca * gn.alleles[0] + cb * gn.alleles[1]
        + cc * gn.alleles[2] + cd * gn.alleles[3];

    return gn.fitness = abs(total - result);
}

int CDiophantine::CreateFitnesses() {
    float avgfit = 0;
    int fitness = 0;
    for(int i=0;i<MAXPOP;i++) {
        fitness = Fitness(population[i]);
        avgfit += fitness;
        if (fitness == 0) {
            return i;
        }
    }

    return 0;
}

```



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Note that if the fitness is zero, then a solution has been found - so return the index. After we calculate the fitnesses, we then have to calculate the likelihoods of the gene's being chosen.

Likelihood functions

We calculate the likelihood by first calculating the sum of the multiplicative inverses, then by dividing the inverse of the fitnesses by that value. When the likelihoods are

calculated though, the likelihoods are cumulative - this makes calculating the parents easy.

Chromosome	Likelihood
1	$(1/84)/0.135266 = 8.80\%$
2	$(1/24)/0.135266 = 30.8\%$
3	$(1/26)/0.135266 = 28.4\%$
4	$(1/133)/0.135266 = 5.56\%$
5	$(1/28)/0.135266 = 26.4\%$

In program though, with the same initial values, the likelihoods would be cumulatively added - imagine it like the percentage of a pie chart. The first gene is from 0 to 8.80%, the next goes to 39.6% (because it *starts* from 8.8). The likelihood table would look like this:

Chromosome	Likelihood (smi = 0.135266)
1	$(1/84)/smi = 8.80\%$
2	$(1/24)/smi = 39.6\% (30.6+8.8)$
3	$(1/26)/smi = 68\% (28.4+39.6)$
4	$(1/133)/smi = 73.56\% (5.56+68)$
5	$(1/28)/smi = 99.96\% (26.4+73.56)$

The last value will always be 100 (in the above example, it isn't due to rounding). Now with the theory of it, let's look at the code. The code is very simple - note though that a type cast to float is required on the fitness value so that integer division doesn't occur. There are two functions, one to calculate the smi, and another to generate all the likelihoods.

```
float CDiophantine::MultInv() {
    float sum = 0;

    for(int i=0;i<MAXPOP;i++) {
        sum += 1/((float)population[i].fitness);
    }

    return sum;
}

void CDiophantine::GenerateLikelihoods() {
    float multinv = MultInv();
```



```

float last = 0;
for(int i=0;i<MAXPOP;i++) {
    population[i].likelihood = last
    = last + ((1/((float)population[i].fitness) / multinv) * 100);
}
}

```

Ok, so now that we have the fitness and likelihoods values all set up, it's time to do some breeding!

Breeding Functions

The breeding functions are composed of three functions, one to get the gene index corresponding to a randomly generated number between 0 and 100, a function to actually calculate the crossover of two genes, and a main function to create the new population. We'll take the functions one at a time, seeing how they call each other. Here is the main breeding function:

```

void CDiophantine::CreateNewPopulation() {
    gene temppop[MAXPOP];

    for(int i=0;i<MAXPOP;i++) {
        int parent1 = 0, parent2 = 0, iterations = 0;
        while(parent1 == parent2 || population[parent1]
            == population[parent2]) {
            parent1 = GetIndex((float)(rand() % 101));
            parent2 = GetIndex((float)(rand() % 101));
            if (++iterations > (MAXPOP * MAXPOP)) break;
        }

        temppop[i] = Breed(parent1, parent2); // Create a child.
    }

    for(i=0;i<MAXPOP;i++) population[i] = temppop[i];
}

```

So, we firstly create a temporary population of genes. Then we loop through all the genes. Now, when choosing genes we don't want the genes to be the same (no point mating with yourself :), and we don't need the genes to be the same either (that is where the operator== from the gene structure comes in handy). In choosing a parent, we generate a random number, then call the GetIndex function. GetIndex uses the idea of the cumulative likelihoods and merely iterates through the genes until it find the gene that contains that number:

```

int CDiophantine::GetIndex(float val) {
    float last = 0;

```

```

    for(int i=0;i<MAXPOP;i++) {
        if (last <= val && val <= population[i].likelihood) return i;
        else last = population[i].likelihood;
    }
    return 4;
}

```

Returning to the CreateNewPopulation() function, you can also see that if the number of iterations exceeds MAXPOP squared, it will take any parents. After parents are chosen, we breed them, by passing the indices up to the Breed function. The Breed function returns a gene, which is put in the temporary population. Here's the code:

```

gene CDiophantine::Breed(int p1, int p2) {
    int crossover = rand() % 3+1;
    int first = rand() % 100;

    gene child = population[p1];

    int initial = 0, final = 3;
    if (first < 50) initial = crossover;
    else final = crossover+1;

    for(int i=initial;i<final;i++) {
        child.alleles[i] = population[p2].alleles[i];
        if (rand() % 101 < 5) child.alleles[i] = rand() % (result + 1);
    }

    return child;
}

```

Firstly we determine the crossover point. Now remember, we don't want the crossover to be the first or the last, because that entails copying over all or none of the second parent - pointless. We then create a random number that will determine when the first parents takes the initial crossover or not. The rest is self-explanatory - you can see that I've added a tiny mutation factor to the breeding. There's a 5% chance that a new number will occur.

And finally...

Now we can look at the Solve() function. It merely calls the above functions iteratively. Note that we test whether the function managed to find a result on the initial population - this is unlikely.

```

int CDiophantine::Solve() {
    int fitness = -1;

```



```

// Generate initial population.
srand((unsigned)time(NULL));

for(int i=0;i<MAXPOP;i++) {
    for (int j=0;j<4;j++) {
        population[i].alleles[j] = rand() % (result + 1);
    }
}

if (fitness = CreateFitnesses()) {
    return fitness;
}

int iterations = 0;
while (fitness != 0 || iterations < 50) {
    GenerateLikelihoods();
    CreateNewPopulation();
    if (fitness = CreateFitnesses()) {
        return fitness;
    }

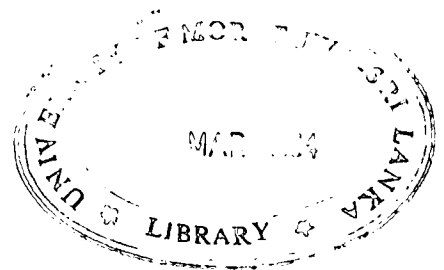
    iterations++;
}

return -1;
}

```



University of Moratuwa, Sri Lanka
 Electronic Theses & Dissertations
www.lib.mrt.ac.lk



REFERENCES

- [1] www.generation5.org/coevolution.shtml
- [2] www.generation5.org/biles.shtml
- [3] www.generation5.org/ga.shtml
- [4] www.generation5.org/ga_math.shtml
- [5] www.generation5.org/diophantine_ga.shtml  University of Moratuwa, Sri Lanka
www.lib.mrt.ac.lk
- [6] www.pmsi.fr/gafxmpa.htm
- [7] www.shef.ac.uk/uni/projects/gaipp/mogas.html
- [8] www.shef.ac.uk/uni/projects/gaipp/control1.html
- [9] www.shef.ac.uk/uni/projects/gaipp/sched.html
- [10] www.nd.com/products/genetic.htm



[11] www.nd.com/products/genetic/whatisga.htm

[12] www.nd.com/products/genetic/apps.htm

[13] www.cs.rochester.edu/users/faculty/leblance/csc173/genetic-algs/example.html

[14] <http://online.cen.uiuc.edu/webcourses/ge485/index.html?intro.html&2>

[15] www.lalena.com/ai/tsp/

[16] www.esatclear.ie/~rwallace/lithos.html

[17] <http://cs.felk.cvut.cz/~xobitko/ga/menu.html>



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations

[18] www.pserc.cornell.edu/matpower/matpower.html

[19] <http://www.isis.ecs.soton.ac.uk/isystems/evolutionary/evol/>

[20] www.pmsi.fr/gainita.htm

[21] Distributed GENESIS User's Guide – Version 1.0

[22] Ceylon Electricity Board – Long Term Transmission Development Studies 2001 – 2010.

[23] Interactive Transmission Network Planning Using a Least-Effort Criterion – IEEE Transaction on power Apparatus and Systems, Vol. PAS-101, No.10, October 1982.

[24] Transmission Network Planning Using Linear Programming – IEEE Transaction on power Apparatus and Systems, Vol. PAS-104, No.2, February 1985.

[25] Application of Sensitivity Analysis of Load Supplying Capability to Interactive Transmission Expansion Planning - IEEE Transaction on power Apparatus and Systems, Vol. PAS-104, No.2, February 1985.

[26] A Zero-One Implicit Enumeration Method for Optimizing Investments in Transmission Expansion Planning - IEEE Transaction on Power Systems, Vol. 9, No 3, August 1994.

 University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
[27] Large Scale Transmission Network Planning Using Optimization and Heuristic Techniques - IEEE Transaction on Power Systems, Vol. 10, No 4, August 1995.

[28] A Hierarchical Decomposition Approach for Transmission Network Expansion Planning - IEEE Transaction on Power Systems, Vol. 9, No 1, February 1994.

[29] Chopin, A Heuristic Model for Long Term Transmission Expansion Planning - IEEE Transaction on Power Systems, Vol. 9, No 4, November 1994.

[30] Hybrid Mathematical and Rule-based System for Transmission network Planning in Open Access Schemes – IEE Proceedings online no. 20010432.

[31] Genetic Algorithms in optimal Multistage Distribution Network Planning – IEEE Transactions on Power Systems, Vol. 9, No. 4, November 1994.

[32] A Practical Schema Theorem for Genetic Algorithm Design and Tuning – David E. Goldberg and Kunara Sastry – IlliGAL.

[33] The Practitioner's Role in Competent Search and Optimization Using Genetic Algorithms – Patrick M. Reed, Barbara S. Minsker and David E. Goldberg.

[34] Making Genetic Algorithms Work in the Real World: Guidelines from Competent GA Theory – Patrik Reed and Barbara Minsker.

[35] Designing a Competent Simple genetic Algorithm for Search and Optimization – Patrick Reed, Barbara Minsker and David E. Goldberg.

[36] Generator Maintenance Scheduling of Electric Power Systems Using Genetic Algorithms with Integer Representation – K.P. Dahal, J.R. McDonald – Centre for Electrical Power Engineering, University of Strathclyde, Glasgow, UK.

[37] Application of GA and Mathematical Models in Long Sea Outfall Designing and Concentration Predictions – B.N.S. Lankasena, N. Ratnayake and M. Indralingam.

[38] Optimization for Engineering Design, Algorithms and Examples – Kalyanmoy Deb