REFERENCE LIST

1. Yapa Nadeeka. Genetic algorithms in Induction Motor efficiency determination. University of Clarkson. (BSc thesis); 2004 May.
2. T. Phumiphak, C. Chat-uthai. Estimation of induction motor parameters based on field test coupled with genetic algorithm.
3. Ion boldia, Syed A Nasar. The induction machine handbook: CRC press. Chapter 22
4. IEEE Standard test procedure for Polyphase Induction Motors and Generators. IEEE Std 112-1996.
5. Ed. Richard, C. Dorf Boca Raton. The electrical engineering handbook: CRC press LLC; 2000.
6. Riccardo Poli, William B, Langdon, Nicolas F, McPhee, John R Koza. A field guide to Genetic programming; March 2008. P. 11-142.
7. Colin R Reeves, Jonathan E Rowe. Genetic algorithms, Principles and perspectives: Kluwer academic publishers; 2003. P. 1-44.
8. HitoshiIba, Topon kumar Paul, Yoshihiko Hasegawa. Applied genetic programming and machine learning; CRC press. P. 7-19.
9. Khalil Banan, Mohammad B.B Sharifian, Jafar Mohommadi. Induction motor efficiency using genetic algorithm. Proceedings of world academy of science, engineering and technology. Vol 3; 2005 Jan. ISSN 1307-6884. P 152-154.
10. Nagendrappa H, Prakash Bure. Energy audit and management of induction motor using field teat and genetic algorithm. 2009 may; 1(3): 16-20.
11. R.S Wimalendra. Determination of maximum possible fuel economy of HEV for known drive cycle Genetic algorithm based approach. (MSc thesis): University of Moratuwa; 2008. P. 33-39.
12. T. Phimiphak, C. Chat-uthai. Estimation of induction Motor Field parameters based on field test coupled with genetic algorithm.
13. T. Phimiphak, C. Chat-uthai. Effective estimation of induction motor field efficiency based on On-site measurements.
14. Ivan Kostov, Vasil spasov, vania rangelova. Application of genetic algorithms for determining the parameters of induction motors.

15. http://www.reliableplant.com/read/19531/harmonizing-international- motor-efficiency-standards, date Accessed:13/08/2010
16. http://en.wikipedia.org/wiki/Premium efficiency, date accessed: 13/08/2010
17. http://www.engineering tool box.com/nema-insulation-classes- d 734.html, date accessed:02/09/2010
18. Energy efficient electric motor selection hand book. Revision 3. Chapter 2, 8. 1993 Jan.
19. Premium Efficiency Motors, Technical fact sheet. Office of Energy Efficiency,Natural resource Canada, 2004
20. M. Benhaddadi, G. Oliver. Premium efficiency motors and market penetration policy.
21. Boteler, Brunner, De Almieda, Doppelbauer, Hoyt. Motor MEPS guide. 1st edition. Zurich; 2009.
22. Paul Waide, Conrad U. brunner. Energy efficiency policy opportunities for electric motor driven systems. International energy agency. International energy agency. 2011.
23. John S. Hsu, John D. Kueck, Michel Olszewski, Don A Casada, Pedro J Otaduy, Leon M. Tolbert. Comparison of Induction motor field efficiency evaluation methods.1998; 34(1)
24. Rotating electrical machines. Part 1- Rating and performance. IEC standard 60034-1: 2004(E). 11th edit; 2008 April.
25. Methods of determination of efficiency of rotating electrical machines; 1989 October. Bureau of Indian standards.
26. Energy efficient induction motors- three phase squirrel cage.IS 12615:2004 Bureau of Indian standards.
27. Energy Medium Energy management Guide for Selection and Use of Fixed Frequency Medium AC Squirrel – cage polyphase Induction Motors. NEMA Standard Publication MG 10-2001(R2007).
28. Electric Motors. Devki Energy Consultancy Pvt. Ltd: 2006. Prepared for Bureau of Energy Efficiency and Indian Renewable Energy Developmental Agency.

29. C. Grantham, D. Seyoum, D. Indyk, D. McKinnn. Calculation of the parameters and parameter variations of an induction motor and the effect of measurement error. School of Electrical Engineering and Telecommunications. The University of new south Wales.
30. Aldo Boglietti, Andrea Cavagnino, Mario Lazzari, Michele pastorelli. International Standard for the Induction Motor Efficiency Evaluation: A Critical Analysis of the Stray – Load Loss Determination. IEEE Transactions on industry Applications. 2004 sep/oct; 40(5).
31. S. Corino, E. Romero, L.F Matilla. How the efficiency of induction motor is measured? Department of Electrical Engineering and Energy.
32. World- class LV motors from a global leader. ABB Technical Brochure.
33. IEC Squirrel – Cage Motors. New efficiency classes and efficiencies according to IEC 60034-30:2008 and IEC 60034-2-1:2007.
34. J. D. Kueck, Development of a method for estimating motor efficiency and analysing motor condition. Oak Ridge National Laboratory
35. S.N.Sivanandam, S.N.Deepa, Introduction to Genetic Algorithms, Springer Berlin Heidelberg New York, 2008, ISBN 978-3-540-73189-4. P 70-108.

Appendix

Appendix- A: Summary of Data Collected and Estimated Efficiency

No	Factory	Tea/Rubber	Location	Motor Brand	Rated Power (kW)	Synchronous Speed	No of Poles	Installed Year	No of Re-Wounds	Phase Voltage (V)	Line Current (A)	Stator Resistance (Ω)	Input Power (W)	Power Factor	Slip	Efficiency (%)
1	Lavant	Rubber - S*	Mill - 01	LD&C	22.5	750	8	1968	1	388.15	23.5	1.3	23210	0.85	0.025	85.07
2	Lavant	Rubber - S*	Mill - 02	LD&C	22.5	750	8	1968	2	388.85	20.86	0.62	12896	0.53	0.120	78.97
3	Lavant	Rubber - S*	Mill - 03	LD&C	15	750	8	1968	2	388.67	8.54	1.12	5178	0.52	0.060	79.84
4	Lavant	Rubber - S*	Mill - 04	LD&C	15	750	8	1968	4	389.37	12.73	0.94	7137	0.48	0.052	76.45
5	Lavant	Rubber - S*	Mill - 05	LD&C	15	750	8	1968	1	389.88	13.6	0.82	8112	0.51	0.040	80.91
6	Lavant	Rubber - S*	Mill - 06	LD&C	15	750	8	1968	0	391.4	9.75	0.92	6640	0.58	0.050	84.28
7	Lavant	Rubber - S*	Mill - 07	LD&C	15	750	8	1968	3	389.71	8.78	0.8	5132	0.50	0.043	79.34
8	Lavant	Rubber - S*	Scrap Washer 1	NA	11	1000	6	NA	NA	390.4	13.8	0.92	8081	0.50	0.068	77.98
9	Lavant	Rubber - S*	Scrap Washer 2	LD&C	11	750	8	1968	2	390.92	12.3	0.98	7501	0.52	0.050	81.17
10	Panawatta	Rubber- SC**	Mill - 01	CG	22	1000	6	1982	4	418.91	16.3	1.85	10447	0.51	0.048	79.42
11	Panawatta	Rubber- SC**	Mill - 02	CG	22	750	8	1982	1	416.73	17.32	0.63	1692	0.54	0.048	83.01
12	Panawatta	Rubber- SC**	Mill - 03	CG	22	750	8	1982	0	412.74	22.71	0.72	19965	0.71	0.030	86.01
13	Panawatta	Rubber- SC**	Mill - 04	CG	30	750	8	1982	4	418.8	24.01	2.21	15083	0.50	0.060	74.60
14	Panawatta	Rubber- SC**	Mill - 05	Brook	22	750	8	1990	0	416.5	20.36	1.31	13483	0.53	0.083	79.72
15	Panawatta	Rubber- SC**	Mill - 06	CG	22	1000	6	1982	1	418.6	20.24	1.46	12963	0.51	0.046	79.87
16	Panawatta	Rubber- SC**	Mill - 07	Brook	22	750	8	1990	0	418.2	15.89	0.82	14353	0.72	0.036	86.85
17	Panawatta	Rubber- SC**	Mill - 08	CG	15	750	8	1982	4	415.3	13.32	0.63	8463	0.51	0.048	80.88

No	Factory	Tea/Rubber	Location	Motor Brand	Rated Power (kW)	Synchronous Speed	No of Poles	Installed Year	No of Re-Wounds	Phase Voltage (V)	Line Current (A)	Stator Resistance (Ω)	Input Power (W)	Power Factor	Slip	Efficiency (%)
18	Panawatta	Rubber- SC**	Mill - 09	CG	15	750	8	1982	5	417.2	12.84	0.73	7713	0.48	0.043	78.11
19	Panawatta	Rubber- SC**	Mill - 10	CG	22	750	8	1982	1	416.6	28.4	0.91	18457	0.52	0.054	81.02
20	Panawatta	Rubber- SC**	Mill - 11	NA	22	750	8	NA	NA	416.5	30.31	1.2	21590	0.57	0.063	84.01
21	Panawatta	Rubber- SC**	Mill - 12	NA	15	750	8	NA	NA	416.5	10.31	0.97	6183	0.48	0.054	76.11
22	Panawatta	Rubber- SC**	Mill - 13	Brook	15	750	8	1990	2	416.8	11.24	0.94	7027	0.50	0.061	78.65
23	Panawatta	Rubber- SC**	Mill - 14	CG	22	750	8	1982	5	416	18.2	1.3	10902	0.48	0.070	68.75
24	Panawatta	Rubber- SC**	Mill - 15	CG	22	750	8	1982	3	416.2	17.35	0.98	10614	0.49	0.063	76.42
25	Dewalakanda	Rubber- SC**	Mill - 01	CG	22	1000	6	1984	1	407.03	19.75	1.25	12540	0.52	0.084	78.43
26	Dewalakanda	Rubber- SC**	Mill - 02	CG	22	750	8	1984	0	407.38	20.48	0.98	13265	0.53	0.050	81.83
27	Dewalakanda	Rubber- SC**	Mill - 03	CG	22	750	8	1984	4	407.21	19.58	1.34	11720	0.49	0.048	77.60
28	Dewalakanda	Rubber- SC**	Mill - 04	CG	22	1000	6	1984	2	407.03	21.46	1.03	12578	0.48	0.059	77.47
29	Dewalakanda	Rubber- SC**	Mill - 05	CG	22	1000	6	1984	3	407.03	20.98	1.34	12553	0.49	0.060	75.76
30	Dewalakanda	Rubber- SC**	Mill - 06	Brook	22	1000	6	1987	1	409.11	21.4	1.08	13132	0.50	0.054	78.65
31	Dewalakanda	Rubber- SC**	Mill - 07	CG	22	750	8	1984	3	408.76	22.3	1.14	13946	0.51	0.058	79.14
32	Dewalakanda	Rubber- SC**	Mill - 08	CG	22	1000	6	1984	5	408.42	21.4	1.15	12848	0.50	0.060	77.24
33	Dewalakanda	Rubber- SC**	Mill - 09	Brook	22	750	8	1987	1	408.59	21.6	1.47	13238	0.50	0.061	76.64
34	Dewalakanda	Rubber- SC**	Mill - 10	CG	15	1000	6	1984	3	406.17	8.64	1.08	5263	0.50	0.053	78.42
35	Dewalakanda	Rubber- SC**	Mill - 11	CG	15	1000	6	1984	2	406.17	7.36	0.98	4394	0.49	0.059	76.02
36	Dewalakanda	Rubber- SC**	Mill - 12	CG	15	1000	6	1984	4	495.3	7.85	1.03	4390	0.46	0.052	71.05
37	Dewalakanda	Rubber- SC**	Mill - 13	CG	15	750	8	1984	3	405.47	8.34	1.26	4869	0.48	0.057	74.42
38	Dewalakanda	Rubber- SC**	Mill - 14	CG	22	1000	6	1984	0	407.21	21.6	1.06	13193	0.50	0.049	79.13

No	Factory	Tea/Rubber	Location	Motor Brand	Rated Power (kW)	Synchronous Speed	No of Poles	Installed Year	No of Re-Wounds	Phase Voltage (V)	Line Current (A)	Stator Resistance (Ω)	Input Power (W)	Power Factor	Slip	Efficiency (%)
39	Dewalakanda	Rubber- SC**	Mill - 15	Brook	22	1000	6	1987	5	407.21	22.6	1.48	13743	0.51	0.063	77.88
40	Dewalakanda	Rubber- SC**	Mill - 16	Brook	15	1000	6	1987	1	406.17	9.44	1.21	5751	0.50	0.054	78.66
41	Dewalakanda	Rubber- SC**	Mill - 17	NA	15	750	8	NA	NA	405.99	9.68	1.48	5777	0.49	0.059	76.15
42	Dewalakanda	Rubber- SC**	Mill - 18	Brook	15	1000	6	1987	4	407.03	9.57	1.32	5492	0.47	0.054	73.28
43	Dunedine	Rubber - Sk***	Mill - 01	BB	22.5	1000	6	NA	NA	388.67	17.85	1.48	10823	0.52	0.058	79.89
44	Dunedine	Rubber - Sk***	Mill - 02	BB	11	1000	6	NA	NA	389.02	6.45	0.94	3763	0.50	0.040	79.62
45	Dunedine	Rubber - Sk***	Mill - 03	CG	15	1500	4	NA	NA	388.85	8.36	1.26	4973	0.51	0.050	79.19
46	Dunedine	Rubber - Sk***	Mill - 04	CG	18.5	1500	4	NA	NA	387.98	10.95	1.26	6118	0.48	0.048	76.17
47	Dunedine	Rubber - Sk***	Mill - 05	CG	15	750	8	NA	NA	388.2	11.3	1.12	6579	0.50	0.051	76.67
48	Dunedine	Rubber - Sk***	Mill - 06	BB	15	1000	6	NA	NA	388.5	10.28	1.03	6110	0.51	0.049	79.89
49	Dunedine	Rubber - Sk***	Mill - 07	NA	15	750	8	NA	NA	388	10.47	0.93	5971	0.49	0.048	78.28
50	Dunedine	Rubber - Sk***	Mill - 08	NA	22	1000	6	NA	NA	389	16.3	1.27	9701	0.51	0.049	79.71
51	Dunedine	Rubber - Sk***	Mill - 09	NA	22	1000	6	NA	NA	388.7	17.2	1.4	10429	0.52	0.042	80.74
52	Dunedine	Rubber - Sk***	Mill - 10	CG	22.5	750	8	NA	NA	388.1	16.3	1.31	9489	0.50	0.040	79.14
53	Dunedine	Rubber - Sk***	Mill - 11	CG	22.5	750	8	NA	NA	388	16.8	1.08	9582	0.49	0.053	77.29
54	Dunedine	Rubber -	Mill - 12	CG	22.	750	8	NA	NA	388	19.7	1.31	11006	0.48	0.045	76.24

No	Factory	Tea/Rubber	Location	Motor Brand	Rated Power (kW)	Synchronous Speed	No of Poles	Installed Year	No of Re-Wounds	Phase Voltage (V)	Line Current (A)	Stator Resistance (Ω)	Input Power (W)	Power Factor	Slip	Efficiency (%)
		Sk***			5											
55	Edarapola	Tea	Roller - 01	NA	15	1500	4	NA	NA	405.3	8.2	0.94	4985	0.50	0.040	79.10
56	Edarapola	Tea	Roller - 02	NA	11	1500	4	NA	NA	405	9.3	0.89	5423	0.48	0.047	77.84
57	Edarapola	Tea	Roller - 03	EE	15	1500	4	1964	3	405	10.02	0.8	596	0.49	0.050	78.25
58	Edarapola	Tea	Roller - 04	EE	15	1500	4	1964	2	405.6	9.85	0.97	5992	0.50	0.046	79.40
59	Edarapola	Tea	Roller - 05	EE	7.5	1500	4	1964	4	404.9	6.2	0.85	3614	0.48	0.040	78.19
60	Edarapola	Tea	Roller - 06	Elektrim	11	1500	4	1970	1	405	8.3	0.9	4941	0.49	0.040	79.27
61	Edarapola	Tea	Roller - 07	Elektrim	11	1500	4	1970	4	405.2	10.3	0.94	5634	0.45	0.041	74.13
62	Edarapola	Tea	Roller - 08	EE	11	1500	4	1964	5	405.8	9.76	1.04	5584	0.47	0.051	75.17
63	Edarapola	Tea	Trough - 04	Elektrim	11	1000	6	1970	3	406	7.24	0.98	4409	0.50	0.040	79.87
64	Edarapola	Tea	Trough - 05	CG	11	1000	6	1976	5	405.4	10.2	0.85	5458	0.44	0.042	71.16
65	Edarapola	Tea	Trough - 06	CG	11	1000	6	1976	1	405	7.85	0.9	5722	0.60	0.040	84.86
66	Edarapola	Tea	Trough - 07	NA	11	1000	6	NA	NA	405.8	8.43	1	5233	0.51	0.048	80.56
67	Annfield	Tea	Roller - 01	HIGGS	11	1500	4	1970	2	402.1	7.31	1.34	4938	0.56	0.038	83.22
68	Annfield	Tea	Roller - 02	CG	15	1500	4	1986	5	403.3	8.85	1.32	5139	0.48	0.050	75.88
69	Annfield	Tea	Roller - 03	EE	11	1500	4	1970	3	402	9.45	1.02	5926	0.52	0.041	81.73
70	Annfield	Tea	Rotorwane - 1	NA	11	1500	4	NA	NA	402	12.85	0.96	7450	0.48	0.040	78.84
71	Annfield	Tea	Rotorwane - 2	CG	11	1500	4	1986	3	402	13.4	1.05	7924	0.49	0.040	79.78
72	Annfield	Tea	Rotorwane - 3	Kirloskar	11	1500	4	1970	6	402.7	13.85	1.34	7696	0.46	0.044	74.38
73	Annfield	Tea	Rotorwane - 4	Kirloskar	11	1500	4	1970	4	402.4	14	0.99	7943	0.47	0.040	77.51
74	Annfield	Tea	Rotorwane - 5	NA	11	1500	4	NA	NA	402	12.41	0.9	7601	0.46	0.057	68.19

No	Factory	Tea/Rubber	Location	Motor Brand	Rated Power (kW)	Synchronous Speed	No of Poles	Installed Year	No of Re-Wounds	Phase Voltage (V)	Line Current (A)	Stator Resistance (Ω)	Input Power (W)	Power Factor	Slip	Efficiency (%)
75	Annfield	Tea	Rotorwane - 6	NA	15	1500	4	NA	NA	402	15.4	0.84	8564	0.46	0.050	72.84
76	Annfield	Tea	Trough - 07	Elektrim	7.5	1000	6	1970	4	402.8	6.2	0.88	3596	0.48	0.040	78.30
77	Annfield	Tea	Trough - 08	Elektrim	7.5	1000	6	1970	5	402.8	6.71	0.93	3976	0.49	0.039	79.36
78	Annfield	Tea	Trough - 09	Elektrim	7.5	1000	6	1970	3	403	7.02	1.03	4243	0.50	0.041	79.96
79	Annfield	Tea	Trough - 10	Elektrim	7.5	1000	6	1970	5	402.6	6.22	1.14	3606	0.48	0.040	77.87
80	Fordyce	Tea	Roller - 01	HIGGS	15	1500	4	1967	3	400.3	9.23	0.98	5542	0.50	0.039	79.54
81	Fordyce	Tea	Roller - 02	HIGGS	15	1500	4	1967	6	401	8.74	1.09	4836	0.46	0.040	74.77
82	Fordyce	Tea	Roller - 03	NA	18.5	1500	4	NA	NA	401	10.28	1.28	5816	0.47	0.052	73.86
83	Fordyce	Tea	Rotorwane - 1	CG	15	1500	4	1982	4	400.8	8.75	1.09	4734	0.45	0.048	69.44
84	Fordyce	Tea	Rotorwane - 2	Kirloskar	11	1500	4	1967	3	400.5	10.16	1.11	5615	0.46	0.040	74.95
85	Fordyce	Tea	Rotorwane - 3	Kirloskar	11	1500	4	1967	5	401.1	10.39	1.35	5626	0.45	0.039	73.38
86	Fordyce	Tea	Rotorwane - 4	Kirloskar	11	1500	4	1967	3	400.6	11.25	0.93	6355	0.47	0.041	76.84
87	Fordyce	Tea	Rotorwane - 5	Kirloskar	11	1500	4	1967	6	400	9.78	0.85	5868	0.50	0.042	79.61
88	Fordyce	Tea	Rotorwane - 6	CG	15	1500	4	1982	3	401.2	13.92	0.91	8209	0.49	0.046	79.81
89	Fordyce	Tea	Trough - 02	EE	9.3	1000	6	1978	2	401	6.86	0.73	4126	0.50	0.037	80.26
90	Fordyce	Tea	Trough - 03	EE	9.3	1000	6	1978	4	401	6.92	0.81	3998	0.48	0.041	77.96
91	Fordyce	Tea	Trough - 04	EE	9.3	1000	6	1978	5	400.3	7.01	0.92	3872	0.46	0.042	75.41
92	Fordyce	Tea	Trough - 05	EE	9.3	1000	6	1978	4	400.7	6.54	1.21	3695	0.47	0.040	76.73
93	Fordyce	Tea	Trough - 06	EE	9.3	1000	6	1978	6	400	7.04	0.97	3886	0.46	0.051	72.84
94	Nuwaraeliya	Tea	Roller - 01	CG	15	1500	4	1972	3	413.1	10.03	0.71	6215	0.50	0.043	79.83
95	Nuwaraeliya	Tea	Roller - 02	WAPAK	18.	1500	4	NA	NA	413.5	9.74	0.95	5799	0.48	0.054	75.77

No	Factory	Tea/Rubber	Location	Motor Brand	Rated Power (kW)	Synchronous Speed	No of Poles	Installed Year	No of Re-Wounds	Phase Voltage (V)	Line Current (A)	Stator Resistance (Ω)	Input Power (W)	Power Factor	Slip	Efficiency (%)
					5											
96	Nuwaraeliya	Tea	Roller - 03	NA	15	1500	4	NA	NA	411	8.49	1.01	6071	0.58	0.040	83.68
97	Nuwaraeliya	Tea	Roller - 04	NA	15	1500	4	NA	NA	412.4	9.25	0.74	5378	0.47	0.046	76.01
98	Nuwaraeliya	Tea	Roller - 05	NA	15	1500	4	NA	NA	412.1	8.82	0.85	5343	0.49	0.040	78.82
99	Nuwaraeliya	Tea	Roller - 06	NA	15	1500	4	NA	NA	412.2	8.91	0.69	5288	0.48	0.042	77.49
100	Nuwaraeliya	Tea	Roller - 07	NA	15	1500	4	NA	NA	412	7.42	0.84	4585	0.50	0.043	79.08
101	Nuwaraeliya	Tea	Roller - 08	HIGGS	11	1500	4	NA	NA	412.3	6.89	0.79	4092	0.48	0.040	77.89
102	Nuwaraeliya	Tea	Rotorwane -1	Teco	11	1500	4	1974	3	412.3	10.41	0.92	6180	0.49	0.046	78.10
103	Nuwaraeliya	Tea	Trough - 01	CG	7.5	1000	6	1979	5	400.4	6.22	1.14	3586	0.48	0.043	77.80
104	Nuwaraeliya	Tea	Trough - 02	CG	7.5	1000	6	1979	4	400.8	7.06	0.74	3989	0.47	0.040	77.21
105	Nuwaraeliya	Tea	Trough - 03	CG	7.5	1000	6	1979	2	401	7.34	0.62	4326	0.49	0.039	79.68
106	Nuwaraeliya	Tea	Trough - 04	CG	7.5	1000	6	1979	5	400.4	6.71	0.8	3788	0.47	0.040	77.42
107	Nuwaraeliya	Tea	Trough - 05	CG	7.5	1000	6	1979	2	400.2	6.38	0.72	3829	0.50	0.041	79.89
108	Nuwaraeliya	Tea	Dryer	Elektrim	18.5	1500	4	1967	1	400	15.32	0.93	9375	0.51	0.043	80.89
109	Pedro	Tea	Roller - 01	CG	15	1500	4	1976	5	411.2	8.35	0.84	5047	0.49	0.038	78.64
110	Pedro	Tea	Roller - 02	CG	15	1500	4	1976	4	411.4	9.65	0.92	5955	0.50	0.041	79.67
111	Pedro	Tea	Roller - 03	CG	15	1500	4	1976	1	412	10.02	0.81	7554	0.61	0.039	84.98
112	Pedro	Tea	Trough - 01	WC	7.5	1000	6	1967	2	400.8	6.14	0.65	3469	0.47	0.042	76.94
113	Pedro	Tea	Trough - 02	WC	7.5	1000	6	1967	6	401.3	6.37	0.8	3450	0.45	0.040	74.01
114	Pedro	Tea	Trough - 03	WC	7.5	1000	6	1967	0	400.8	8.21	0.62	5725	0.58	0.038	84.68
115	Pedro	Tea	Trough - 04	WC	7.5	1000	6	1967	3	401.3	7.35	0.82	4778	0.54	0.040	82.77

No	Factory	Tea/Rubber	Location	Motor Brand	Rated Power (kW)	Synchronous Speed	No of Poles	Installed Year	No of Re-Wounds	Phase Voltage (V)	Line Current (A)	Stator Resistance (Ω)	Input Power (W)	Power Factor	Slip	Efficiency (%)
116	Pedro	Tea	Trough - 05	WC	7.5	1000	6	1967	2	400.7	9.46	0.64	5572	0.49	0.035	80.23
117	Pedro	Tea	Trough - 06	WC	11	1000	6	1967	1	400	10.35	0.68	6955	0.56	0.047	84.14
118	Pedro	Tea	Trough - 07	CG	11	1000	6	1976	5	401.5	10.98	0.85	6215	0.47	0.051	75.47
119	Pedro	Tea	Trough - 08	CG	11	1000	6	1976	5	402	9.27	1.02	5366	0.48	0.041	78.32
120	Pedro	Tea	Trough - 09	CG	11	1000	6	1976	4	401.4	10.04	0.93	5561	0.46	0.040	75.82

S – Scrap

SC – Sole Crepe

SK - Skim

NA – Data Not Available

CG – Crompton Graves

LD&C – Lancashire Dynamo & Crypto

WC – Woods of Colchester

EE – English Electric

BB – Brown Bover

Appendix- B : Potential Energy Saving and Simple Payback for Each IM

No	Factory	Location	Rated kW	No of Poles	Efficiency of Present Installation	Efficiency EFF1 (IE2)	Efficiency EFF2 (IE1)	Re-Winding Cost	Saving with EFF1	Simple Pay Back EFF1 (Months)	RE-wind Vs Replace EFF1 Payback	Saving with EFF2	Simple payback EFF2	RE-wind Vs Replace EFF2 Payback
83	Fordyce	Rotorwane - 1	15	4	69.44	91.8	89.4	36,500	110,097	14	10	100,918	**9**	5
74	Annfield	Rotorwane - 5	11	4	68.19	91.0	88.4	30,500	84,610	17	13	77,171	**10**	5
75	Annfield	Rotorwane - 6	15	4	72.84	91.8	89.4	36,500	88,998	18	13	79,820	**11**	6
82	Fordyce	Roller - 03	18.5	4	73.86	92.2	90.0	44,500	104,255	17	12	93,992	**12**	6
81	Fordyce	Roller - 02	15	4	74.77	91.8	89.4	36,500	77,876	20	14	68,697	13	7
95	Nuwaraeliya	Roller - 02	18.5	4	75.77	92.2	90.0	44,500	91,043	20	14	80,780	14	7
36	Dewalakanda	Mill - 12	15	6	71.05	90.5	88.5	38,000	94,943	20	15	87,105	14	9
46	Dunedine	Mill - 04	18.5	4	76.17	92.2	90.0	44,500	88,360	20	14	78,097	14	7
85	Fordyce	Rotorwane - 3	11	4	73.38	91.0	88.4	30,500	60,736	24	18	53,296	14	8
23	Panawatta	Mill - 14	22	8	68.75	90.2	88.5	64,500	159,234	21	16	149,431	14	9
68	Annfield	Roller - 02	15	4	75.88	91.8	89.4	36,500	71,735	22	16	62,556	15	8
97	Nuwaraeliya	Roller - 04	15	4	76.01	91.8	89.4	36,500	71,027	22	16	61,848	15	8
61	Edarapola	Roller - 07	11	4	74.13	91.0	88.4	30,500	57,562	25	19	50,123	15	8
72	Annfield	Rotorwane - 3	11	4	74.38	91.0	88.4	30,500	56,519	25	19	49,079	16	8
42	Dewalakanda	Mill - 18	15	6	73.28	90.5	88.5	38,000	81,500	23	17	73,662	16	10
84	Fordyce	Rotorwane - 2	11	4	74.95	91.0	88.4	30,500	54,165	27	20	46,726	16	9
62	Edarapola	Roller - 08	11	4	75.17	91.0	88.4	30,500	53,266	27	20	45,827	17	9
99	Nuwaraeliya	Roller - 06	15	4	77.49	91.8	89.4	36,500	63,141	25	18	53,962	17	9
64	Edarapola	Trough - 05	11	6	71.16	89.7	87.5	34,000	66,856	25	19	60,404	17	11
29	Dewalakanda	Mill - 05	22	6	75.76	91.8	90.0	57,500	106,172	27	21	96,143	18	11
57	Edarapola	Roller - 03	15	4	78.25	91.8	89.4	36,500	59,206	26	19	50,028	18	9
93	Fordyce	Trough - 06	9.3	6	72.84	89.3	87.0	30,500	49,244	26	18	43,483	19	10

No	Factory	Location	Rated kW	No of Poles	Efficiency of Present Installation	Efficiency EFF1 (IE2)	Efficiency EFF2 (IE1)	Re-Winding Cost	Saving with EFF1	Simple Pay Back EFF1 (Months)	RE-wind Vs Replace EFF1 Payback	Saving with EFF2	Simple payback EFF2	RE-wind Vs Replace EFF2 Payback
109	Pedro	Roller - 01	15	4	78.64	91.8	89.4	36,500	57,217	27	20	48,038	19	10
98	Nuwaraeliya	Roller - 05	15	4	78.82	91.8	89.4	36,500	56,306	28	20	47,127	19	10
86	Fordyce	Rotorwane - 4	11	4	76.84	91.0	88.4	30,500	46,611	31	23	39,172	20	10
100	Nuwaraeliya	Roller - 07	15	4	79.08	91.8	89.4	36,500	54,996	28	20	45,818	20	10
55	Edarapola	Roller - 01	15	4	79.10	91.8	89.4	36,500	54,896	28	20	45,717	20	10
45	Dunedine	Mill - 03	15	4	79.19	91.8	89.4	36,500	54,445	29	21	45,266	20	10
35	Dewalakanda	Mill - 11	15	6	76.02	90.5	88.5	38,000	66,062	28	21	58,224	21	13
58	Edarapola	Roller - 04	15	4	79.40	91.8	89.4	36,500	53,397	29	21	44,218	21	11
32	Dewalakanda	Mill - 08	22	6	77.24	91.8	90.0	57,500	94,529	30	23	84,499	21	13
80	Fordyce	Roller - 01	15	4	79.54	91.8	89.4	36,500	52,701	30	21	43,522	21	11
73	Annfield	Rotorwane - 4	11	4	77.51	91.0	88.4	30,500	44,022	33	24	36,583	21	11
28	Dewalakanda	Mill - 04	22	6	77.47	91.8	90.0	57,500	92,759	31	24	82,730	21	13
110	Pedro	Roller - 02	15	4	79.67	91.8	89.4	36,500	52,057	30	22	42,878	21	11
88	Fordyce	Rotorwane - 6	15	4	79.81	91.8	89.4	36,500	51,366	30	22	42,187	22	11
94	Nuwaraeliya	Roller - 01	15	4	79.83	91.8	89.4	36,500	51,267	30	22	42,089	22	11
56	Edarapola	Roller - 02	11	4	77.84	91.0	88.4	30,500	42,763	34	25	35,324	22	11
101	Nuwaraeliya	Roller - 08	11	4	77.89	91.0	88.4	30,500	42,573	34	25	35,134	22	11
39	Dewalakanda	Mill - 15	22	6	77.88	91.8	90.0	57,500	89,631	32	24	79,602	22	13
13	Panawatta	Mill - 04	22	8	74.60	90.2	88.5	64,500	106,725	31	24	96,922	22	14
102	Nuwaraeliya	Rotorwane -1	11	4	78.10	91.0	88.4	30,500	41,779	34	26	34,339	22	12
108	Nuwaraeliya	Dryer	18.5	4	80.89	92.2	90.0	44,500	58,705	31	22	48,442	23	12
25	Dewalakanda	Mill - 01	22	6	78.43	91.8	90.0	57,500	85,486	34	26	75,457	23	14
37	Dewalakanda	Mill - 13	15	8	74.42	89.0	87.0	41,500	69,093	32	25	60,986	24	15
30	Dewalakanda	Mill - 06	22	6	78.65	91.8	90.0	57,500	83,844	34	26	73,815	24	14
91	Fordyce	Trough - 04	9.3	6	75.41	89.3	87.0	30,500	40,139	31	22	34,378	24	13
70	Annfield	Rotorwane - 1	11	4	78.84	91.0	88.4	30,500	39,012	37	28	31,573	24	13
118	Pedro	Trough - 07	11	6	75.47	89.7	87.5	34,000	48,383	35	26	41,932	25	15

No	Factory	Location	Rated kW	No of Poles	Efficiency of Present Installation	Efficiency EFF1 (IE2)	Efficiency EFF2 (IE1)	Re-Winding Cost	Saving with EFF1	Simple Pay Back EFF1 (Months)	RE-wind Vs Replace EFF1 Payback	Saving with EFF2	Simple payback EFF2	RE-wind Vs Replace EFF2 Payback
38	Dewalakanda	Mill - 14	22	6	79.13	91.8	90.0	57,500	80,294	36	27	70,264	25	15
60	Edarapola	Roller - 06	11	4	79.27	91.0	88.4	30,500	37,429	38	29	29,989	26	13
10	Panawatta	Mill - 01	22	6	79.42	91.8	90.0	57,500	78,169	37	28	68,140	26	16
120	Pedro	Trough - 09	11	6	75.82	89.7	87.5	34,000	46,975	36	27	40,524	26	16
54	Dunedine	Mill - 12	22	8	76.24	90.2	88.5	64,500	93,451	35	27	83,647	26	17
59	Edarapola	Roller - 05	7.5	4	78.19	90.1	87.0	22,500	26,532	27	17	20,325	26	13
24	Panawatta	Mill - 15	22	8	76.42	90.2	88.5	64,500	92,029	36	27	82,225	26	17
34	Dewalakanda	Mill - 10	15	6	78.42	90.5	88.5	38,000	53,425	35	26	45,588	26	16
50	Dunedine	Mill - 08	22	6	79.71	91.8	90.0	57,500	76,060	38	29	66,031	27	16
87	Fordyce	Rotorwane - 5	11	4	79.61	91.0	88.4	30,500	36,189	40	30	28,749	27	14
113	Pedro	Trough - 02	7.5	6	74.01	88.1	86.0	27,000	33,913	32	22	29,564	27	16
33	Dewalakanda	Mill - 09	22	8	76.64	90.2	88.5	64,500	90,300	37	28	80,496	27	17
15	Panawatta	Mill - 06	22	6	79.87	91.8	90.0	57,500	74,904	38	29	64,874	27	16
40	Dewalakanda	Mill - 16	15	6	78.66	90.5	88.5	38,000	52,204	36	27	44,366	27	17
43	Dunedine	Mill - 01	22	6	79.89	91.8	90.0	57,500	74,759	39	29	64,730	27	16
92	Fordyce	Trough - 05	9.3	6	76.73	89.3	87.0	30,500	35,700	35	25	29,939	27	15
71	Annfield	Rotorwane - 2	11	4	79.78	91.0	88.4	30,500	35,573	40	30	28,133	27	14
21	Panawatta	Mill - 12	15	8	76.11	89.0	87.0	41,500	59,728	37	29	51,621	28	18
41	Dewalakanda	Mill - 17	15	8	76.15	89.0	87.0	41,500	59,511	37	29	51,404	28	18
53	Dunedine	Mill - 11	22	8	77.29	90.2	88.5	64,500	85,248	39	30	75,444	29	18
4	Lavant	Mill - 04	15	8	76.45	89.0	87.0	41,500	57,894	38	30	49,787	29	19
27	Dewalakanda	Mill - 03	22	8	77.60	90.2	88.5	64,500	82,869	40	30	73,065	30	19
47	Dunedine	Mill - 05	15	8	76.67	89.0	87.0	41,500	56,716	39	30	48,609	30	19
51	Dunedine	Mill - 09	22	6	80.74	91.8	90.0	57,500	68,693	42	32	58,663	30	18
48	Dunedine	Mill - 06	15	6	79.89	90.5	88.5	38,000	46,061	40	30	38,223	31	19
90	Fordyce	Trough - 03	9.3	6	77.96	89.3	87.0	30,500	31,698	40	28	25,937	31	17
8	Lavant	Scrap Washer	11	6	77.98	89.7	87.5	34,000	38,566	44	33	32,115	33	20

No	Factory	Location	Rated kW	No of Poles	Efficiency of Present Installation	Efficiency EFF1 (IE2)	Efficiency EFF2 (IE1)	Re-Winding Cost	Saving with EFF1	Simple Pay Back EFF1 (Months)	RE-wind Vs Replace EFF1 Payback	Saving with EFF2	Simple payback EFF2	RE-wind Vs Replace EFF2 Payback
		1												
119	Pedro	Trough - 08	11	6	78.32	89.7	87.5	34,000	37,285	45	34	30,833	34	21
2	Lavant	Mill - 02	22	8	78.97	90.2	88.5	64,500	72,577	45	35	62,773	34	22
18	Panawatta	Mill - 09	15	8	78.11	89.0	87.0	41,500	49,169	45	35	41,061	35	23
31	Dewalakanda	Mill - 07	22	8	79.14	90.2	88.5	64,500	71,325	46	35	61,521	35	23
52	Dunedine	Mill - 10	22	8	79.14	90.2	88.5	64,500	71,325	46	35	61,521	35	23
49	Dunedine	Mill - 07	15	8	78.28	89.0	87.0	41,500	48,296	46	36	40,189	36	23
69	Annfield	Roller - 03	11	4	81.73	91.0	88.4	30,500	28,689	50	37	21,250	36	19
112	Pedro	Trough - 01	7.5	6	76.94	88.1	86.0	27,000	25,838	42	29	21,488	37	22
22	Panawatta	Mill - 13	15	8	78.65	89.0	87.0	41,500	46,410	48	37	38,302	38	25
14	Panawatta	Mill - 05	22	8	79.72	90.2	88.5	64,500	67,093	49	38	57,289	38	24
96	Nuwaraeliya	Roller - 03	15	4	83.68	91.8	89.4	36,500	33,178	47	34	23,999	38	20
104	Nuwaraeliya	Trough - 02	7.5	6	77.21	88.1	86.0	27,000	25,125	43	30	20,775	38	23
106	Nuwaraeliya	Trough - 04	7.5	6	77.42	88.1	86.0	27,000	24,574	44	31	20,224	39	23
44	Dunedine	Mill - 02	11	6	79.62	89.7	87.5	34,000	32,487	52	39	26,035	40	24
103	Nuwaraeliya	Trough - 01	7.5	6	77.80	88.1	86.0	27,000	23,584	46	32	19,234	41	24
7	Lavant	Mill - 07	15	8	79.34	89.0	87.0	41,500	42,939	52	40	34,832	41	27
63	Edarapola	Trough - 04	11	6	79.87	89.7	87.5	34,000	31,582	53	40	25,130	42	25
79	Annfield	Trough - 10	7.5	6	77.87	88.1	86.0	27,000	23,402	46	32	19,052	42	25
89	Fordyce	Trough - 02	9.3	6	80.26	89.3	87.0	30,500	24,545	51	36	18,784	43	24
76	Annfield	Trough - 07	7.5	6	78.30	88.1	86.0	27,000	22,295	48	34	17,946	44	26
3	Lavant	Mill - 03	15	8	79.84	89.0	87.0	41,500	40,461	55	43	32,354	45	29
19	Panawatta	Mill - 10	22	8	81.02	90.2	88.5	64,500	57,827	57	44	48,024	45	29
66	Edarapola	Trough - 07	11	6	80.56	89.7	87.5	34,000	29,113	58	44	22,662	46	28
67	Annfield	Roller - 01	11	4	83.22	91.0	88.4	30,500	23,647	61	45	16,207	47	25
111	Pedro	Roller - 03	15	4	84.98	91.8	89.4	36,500	27,440	57	41	18,261	**50**	26
26	Dewalakanda	Mill - 02	22	8	81.83	90.2	88.5	64,500	52,203	63	48	42,399	**51**	33

No	Factory	Location	Rated kW	No of Poles	Efficiency of Present Installation	Efficiency EFF1 (IE2)	Efficiency EFF2 (IE1)	Re-Winding Cost	Saving with EFF1	Simple Pay Back EFF1 (Months)	RE-wind Vs Replace EFF1 Payback	Saving with EFF2	Simple payback EFF2	RE-wind Vs Replace EFF2 Payback
77	Annfield	Trough - 08	7.5	6	79.36	88.1	86.0	27,000	19,618	55	39	15,268	**52**	31
17	Panawatta	Mill - 08	15	8	80.88	89.0	87.0	41,500	35,406	63	49	27,299	**53**	35
5	Lavant	Mill - 05	15	8	80.91	89.0	87.0	41,500	35,262	63	49	27,155	**53**	35
105	Nuwaraeliya	Trough - 03	7.5	6	79.68	88.1	86.0	27,000	18,824	57	40	14,474	**55**	32
107	Nuwaraeliya	Trough - 05	7.5	6	79.89	88.1	86.0	27,000	18,306	59	41	13,957	**57**	34
78	Annfield	Trough - 09	7.5	6	79.96	88.1	86.0	27,000	18,134	60	42	13,785	**57**	34
116	Pedro	Trough - 05	7.5	6	80.23	88.1	86.0	27,000	17,474	62	43	13,124	**60**	36
11	Panawatta	Mill - 02	22	8	83.01	90.2	88.5	64,500	44,206	75	57	34,402	**63**	40
9	Lavant	Scrap Washer 2	11	8	81.17	88.1	86.0	38,000	22,306	91	71	15,926	**75**	47
20	Panawatta	Mill - 11	22	8	84.01	90.2	88.5	64,500	37,605	88	67	27,801	**78**	50
117	Pedro	Trough - 06	11	6	84.14	89.7	87.5	34,000	16,957	99	75	10,505	**99**	61
1	Lavant	Mill - 01	22	8	85.07	90.2	88.5	64,500	30,777	107	82	20,973	**103**	66
115	Pedro	Trough - 04	7.5	6	82.77	88.1	86.0	27,000	11,471	94	66	7,121	**111**	66
6	Lavant	Mill - 06	15	8	84.28	89.0	87.0	41,500	19,751	112	87	11,643	**124**	81
65	Edarapola	Trough - 06	11	6	84.86	89.7	87.5	34,000	14,636	115	87	8,184	**128**	78
12	Panawatta	Mill - 03	22	8	86.01	90.2	88.5	64,500	24,863	133	102	15,059	**143**	92
16	Panawatta	Mill - 07	22	8	86.85	90.2	88.5	64,500	19,686	168	128	9,882	**219**	140
114	Pedro	Trough - 03	7.5	6	84.68	88.1	86.0	27,000	7,194	150	105	2,845	**278**	165

Appendix- C: Java Program Developed for Efficiency Estimation

GA2.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Properties;
import java.util.Random;


public class GA2
{

	public static InputParamInfo[] inputInfos = new InputParamInfo[4];


	public static void main(String[] args)
	{
		GA2.doIt();

	}

	// Static info
	static int			chromoLen	= 4;
	static double	crossRate	= .8;
	static double	mutRate			= .002;
	static Random rand		= new Random();
	static int			poolSize	= 40;

	// input parameters;

	public static Properties inputData;

	public static InputParamInfo getInputParamInfo ( int inputParamType )
```

```
{
        for (int i = 0; i < inputInfos.length; i++)
        {
                InputParamInfo inputParam = inputInfos[i];

                if ( inputParam.getParam() == inputParamType )
                {
                        return inputParam;
                }
        }

        return null;
}

public static double getInputData ( String key )
{
        Object valStr = inputData.get(key);
        if ( valStr == null )
        {
                return -1.0;
        }
        else
        {
                try
                {

                        return Double.parseDouble(valStr.toString());
                }
                catch (Exception e)
                {
                        return -1.0;
                }
        }

}

private static void doIt()
{

        inputInfos[0] = new InputParamInfo(InputParamInfo.X1, 1, 500, 100.0,
9 );
        inputInfos[1] = new InputParamInfo(InputParamInfo.XM, 5, 255, 1.0,
8 );
```

```
		inputInfos[2] = new InputParamInfo(InputParamInfo.RM, 50, 1000,
1.0, 10 );
		inputInfos[3] = new InputParamInfo(InputParamInfo.R2, 20, 1000,
100.0, 10 );

		loadInputData();

		HashMap<Integer,Chromosome> solutions = new HashMap<Integer,
Chromosome>();

		int gen = 0;

		// Create the pool
		ArrayList pool = new ArrayList(poolSize);
		ArrayList newPool = new ArrayList(pool.size());

		System.out.println("Generating Population ...");
		// Generate unique cromosomes in the pool
		for (int x = 0; x < poolSize; x++)
		{
			Chromosome chromosome = new Chromosome();
			System.out.println(" Initial Population : " + chromosone);
			pool.add(chromosone);
		}

		System.out.println("Population generated");

		// Loop until solution is found
		while (true)
		{
			// Clear the new pool
			newPool.clear();

			// Add to the generations
			gen++;

			// Loop until the pool has been processed
			for (int x = pool.size() - 1; x >= 0; x -= 2)
			{
				// Select two members
				Chromosome n1 = selectMember(pool);
				Chromosome n2 = selectMember(pool);
```

```
                                        // Cross over and mutate
                                        n1.crossOver(n2);
                                        n1.mutate();
                                        n2.mutate();

                                        // Rescore the nodes
                                        n1.scoreChromo();
                                        n2.scoreChromo();

                                        System.out.println("Chromosome   generated.   =>   "   +
n1.toString());
                                        System.out.println("Chromosome   generated.   =>   "   +
n2.toString());


                                        if (n1.score < 100 && n1.isValid())
                                        {
                                                System.out.println("Soution Found in  " + gen +
" Generations\n"
                                                                        + n1 );
                                                try
                                                {
                                                        solutions.put(gen,(Chromosome)
n1.clone());
                                                }
                                                catch (CloneNotSupportedException e)
                                                {
                                                        // TODO Auto-generated catch block
                                                        e.printStackTrace();
                                                }
//                                              return;
                                        }
                                        if (n2.score < 100 && n2.isValid())
                                        {
                                                System.out.println("Soution Found in  " + gen +
" Generations\n"
                                                                        + n2 );
                                                try
                                                {
                                                        solutions.put(gen,(Chromosome)
n2.clone());
                                                }
                                                catch (CloneNotSupportedException e)
```

```
                                        {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
//                                      return;
                                }

                                if ( solutions.size() > 10 )
                                {
                                        printSolutions ( solutions );
                                        return;
                                }

                                // Add to the new pool
                                newPool.add(n1);
                                newPool.add(n2);
                        }

                        // Add the newPool back to the old pool
                        pool.addAll(newPool);
                }
        }

        private static void printSolutions(HashMap<Integer,Chromosome> solutions)
        {
                System.out.println("\n\n-----------------------------------------------------------
----");
                System.out.println("SOLUTIONS FOUND");

                for (Iterator<Integer> iterator = solutions.keySet().iterator();
iterator.hasNext();)
                {
                        Integer gen = iterator.next();

                        Chromosome ch = solutions.get(gen);
                        System.out.println(gen + " Generations\t=> " + ch );


                }
```

```
		System.out.println("\n\n--------------------------------------------------------
----");
	}

	private static void loadInputData() {
		inputData = new Properties();
		FileInputStream fin = null;
		try
		{
			fin = new FileInputStream (System.getProperty("user.dir") +
File.separator + "input.txt");
			inputData.load( fin );
		}
		catch (FileNotFoundException e)
		{
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		catch (IOException e)
		{
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		finally
		{
			if ( fin != null )
			{
				try
				{
					fin.close();
				} catch (IOException e)
				{
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
		}
	}

	// ---- Chomosone Class -----
	private static Chromosome selectMember(ArrayList<Chromosome>
population)
	{
```

```
			return selectMemberRoulette(population);
//			return selectMemberRankMethod(population);
		}

		private static Chromosome selectMemberRankMethod (
ArrayList<Chromosome> population )
		{
			Collections.sort(population, new Comparator<Chromosome>() {

				public int compare(Chromosome o1, Chromosome o2) {
					// TODO Auto-generated method stub
					return new Double(1 / o1.score).compareTo(1 /
o2.score);
				}

			});

			int total = 0;
			for (int i = 1; i <= population.size(); i++)
			{
				total += i;
```



```
			}
			double slice = total * rand.nextDouble();

			double ttot = 0.0;
			for (int i = 1; i <= population.size(); i++)
			{
				ttot += i;

				if ( ttot >= slice )
				{
					Chromosome node = population.remove(i - 1);
					return node;
				}
			}

			return population.remove(population.size() - 1);

		}

		private static Chromosome selectMemberRoulette(ArrayList<Chromosome>
population) {
```

```
		double tot = 0.0;
		for (int x = population.size() - 1; x >= 0; x--)
		{
			double score = 1.0/((Chromosome) population.get(x)).score;
			tot += score;
		}
		double slice = tot * rand.nextDouble();


		double ttot = 0.0;
		for (int x = population.size() - 1; x >= 0; x--)
		{
			Chromosome node = (Chromosome) population.get(x);
			ttot += 1.0/node.score;
			if (ttot >= slice)
			{
				population.remove(x);
				return node;
			}
		}

		return (Chromosome) population.remove(population.size() - 1);
	}

}
```


Chromosome.java

```
	// Genetic Algorithm Node
	public class Chromosome implements Cloneable
	{
		// The chromo

		@Override
		protected Object clone() throws CloneNotSupportedException
		{
			Chromosome n = (Chromosome) super.clone();
			n.chromo = new StringBuffer ( this.chromo.toString() );
```

```
        return n;
    }

    public static final int   BIT_VALUE_LENGTH       = 10;

    StringBuffer                  chromo                          =
new StringBuffer(

            GA2.chromoLen

                    * BIT_VALUE_LENGTH);
    public double                 score;

    public double                 v1                              =
GA2.getInputData("v1");
    public double                 i1                              =
GA2.getInputData("i1");
    public double                 r1                              =
GA2.getInputData("r1");
    public double                 pinp                       =
GA2.getInputData("pinp");
    public double                 pf                              =
GA2.getInputData("pf");
    public double                 s                               =
GA2.getInputData("s");
    public double                 pw                              =
GA2.getInputData("pw");
    public double                 k                               =
GA2.getInputData("k");



    public double efficiency;

    public Chromosome()
    {

        int[] genes = new int[4];

        for (int i = 0; i < GA2.inputInfos.length; i++)
        {
            InputParamInfo inputInfo = GA2.inputInfos[i];
```

```
                                        genes[i] = inputInfo.getMin() + GA2.rand.nextInt( 
inputInfo.getMax() - inputInfo.getMin() + 1 );

                                        String binString = Integer.toBinaryString(genes[i]);

                                        int fillLen = inputInfo.getBitLength() - 
binString.length();

                                        for (int x = 0; x < fillLen; x++)
                                        {
                                                chromo.append('0');
                                        }

                                        chromo.append(binString);
                                }

                                // Score the new cromo
                                 scoreChromo();
                        }

                        public String toString()
                        {
                                double genesDec[] = decode();
                                boolean valid = isValid(genesDec);

                                if ( valid )
                                {
                                return chromo.toString() + " => "
                                                + " : ff = " + score + " : efficiency = " + 
efficiency;
                                }
                                else
                                {
                                        return chromo.toString() + " => Invalid";
                                }
                        }

                        public Chromosome(StringBuffer chromo)
                        {
                                this.chromo = chromo;
                        }

                        public double[] decode()
                        {
```

```
            double[] genes = new double[4];
            int j = 0;

            int startIndex = 0;

            for (int i = 0; i < GA2.inputInfos.length; i++)
            {
                InputParamInfo inputInfo = GA2.inputInfos[i];

                String valStr = chromo.substring(startIndex, startIndex
+ inputInfo.getBitLength() );
                double val = Integer.parseInt(valStr, 2 );

                genes[i] = val / inputInfo.getFactor();

            }

            return genes;
        }

        // Scores this chromo
        public final void scoreChromo()
        {

            double genes[] = decode();

            if (isValid(genes))
            {
                double x1 = genes[0];
                double xm = genes[1];
                double rm = genes[2];
                double r2 = genes[3];

                double x2 = x1 / k;

                double A1 = Math.pow(
                        (((1.0 / rm) * (Math.pow((r2 / s), 2.0) +
Math.pow(x2,
                                2.0))) + (r2 / s)), 2.0);
                double A2 = Math.pow(
                        (((1.0 / xm) * (Math.pow((r2 / s), 2.0) +
Math.pow(x2,
                                2))) + x2), 2.0);
```

```
double A = A1 + A2;

double B = (Math.pow((r2 / s), 2.0) + Math.pow(x2,
2.0))
        * (((1.0 / rm) * (Math.pow((r2 / s), 2.0) +
Math.pow(
                x2, 2.0))) + (r2 / s));

double C = (Math.pow((r2 / s), 2.0) + Math.pow(x2,
2.0))
        * (((1.0 / xm) * (Math.pow((r2 / s), 2.0) +
Math.pow(
                x2, 2.0))) + x2);

double m1 = ((r1 * A) + B);
double m2 = ((k * x2 * A) + C);

double m3 = (v1 * A) / (Math.pow(m1, 2.0) +
Math.pow(m2, 2.0));

double alpha = m3 * m1;
double beta = m3 * m2;

double m4 = Math.sqrt(Math.pow(alpha, 2.0)
        + Math.pow(beta, 2.0));

double i1est = m4;
double pfest = alpha / m4;

double pin_est = 3 * v1 * alpha;

double f1 = ((i1est - i1) * 100) / i1;
double f2 = ((pin_est - pinp) * 100) / pinp;
double f3 = ((pfest - pf) * 100) / pf;

double ff = Math.pow(f1, 2.0) + Math.pow(f2, 2.0)
        + Math.pow(f3, 2.0);

score = Math.abs(ff - 7942);


double pscl = 3 * r1 * ( Math.pow(alpha, 2.0) +
Math.pow(beta, 2.0) );
```

```
                    double Pm = ( 3.0 / rm ) *( Math.pow(( v1 - (alpha * r1)
- ( beta * x1) ),2.0) +  Math.pow( ((alpha * x1 ) - ( beta * r1)), 2.0) );

                    double m6 =  (( v1 - (alpha * r1) - ( beta * x1) )/rm) - (
((alpha * x1 ) - ( beta * r1))/xm );

                    double prcl = 3 * r2 * (  Math.pow((alpha - m6),2.0) +
Math.pow ( (beta + m6), 2.0) );

                    double Pout = pinp - pscl - Pm - prcl - pw;

                    efficiency = ( Pout / pinp ) * 100;

               }
               else
               {
                    score = Integer.MAX_VALUE;
               }


          }
```



```
          // Crossover bits
          public final void crossOver(Chromosome other)
          {

               // Should we cross over?
               if (GA2.rand.nextDouble() > GA2.crossRate)
                    return;

               // Generate a random position
               int pos = GA2.rand.nextInt(chromo.length());

               // Swap all chars after that position
               for (int x = pos; x < chromo.length(); x++)
               {
                    // Get our character
                    char tmp = chromo.charAt(x);

                    // Swap the chars
                    chromo.setCharAt(x, other.chromo.charAt(x));
                    other.chromo.setCharAt(x, tmp);
               }
          }
```

```
		// Mutation
		public final void mutate()
		{
			for (int x = 0; x < chromo.length(); x++)
			{
				if (GA2.rand.nextDouble() <= GA2.mutRate)
					chromo.setCharAt(x, (chromo.charAt(x) == '0' ?
'1' : '0'));
			}
		}

		// Add up the contents of the decoded chromo

		public final boolean isValid()
		{
			double[] genes = decode();

			return isValid(genes);
		}

		public final boolean isValid(double[] genes)
		{
			for (int i = 0; i < genes.length; i++)
			{
				double val = genes[i];

				InputParamInfo inpuParamInfo = null;

				switch (i)
				{
					case 0:
						inpuParamInfo =
GA2.getInputParamInfo(InputParamInfo.X1);

						break;
					case 1:
						inpuParamInfo =
GA2.getInputParamInfo(InputParamInfo.XM);

						break;
					case 2:
						inpuParamInfo =
GA2.getInputParamInfo(InputParamInfo.RM);
```

```
                    break;
                case 3:
                    inpuParamInfo =
GA2.getInputParamInfo(InputParamInfo.R2);

                    break;
            }

            if ( ! ((val * inpuParamInfo.getFactor()) >=
inpuParamInfo.getMin() && (val*inpuParamInfo.getFactor()) <=
inpuParamInfo.getMax() ) )
            {
                return false;
            }
        }

        return true;
    }
}
```