Published in *Control and Cybernetics*, vol. 30, No2, 2001, pp. 159-176, ISSN: 0324-8569

Supplementary Crossover Operator for Genetic Algorithms based on the Center-of-Gravity Paradigm

Plamen Angelov

Department of Civil and Building Engineering,
University of Loughborough
Loughborough, Leicestershire LE11 3TU, UK
tel.: +44 (1509) 223 774; fax: +44 (1509) 223 981; e-mail: P.P.Angelov @Lboro.ac.UK

Abstract A supplementary crossover operator for genetic algorithms (GA) is proposed in the paper. It performs specific breeding between the two fittest parental chromosomes. The new child chromosome is based on the center of gravity (CoG) paradigm, taking into account both the parental weight (measured by their fitness) and their actual value. It is designed to be used in combination with other crossover and mutation operators (it applies to the best fitted two parental chromosomes only) both in binary and real-valued (evolutionary) GA. Analytical proof of its ability to improve the result is provided for the simplest case of one variable and when *elitist* selection strategy is used. The new operator is validated with a number of usually used numerical test functions as well as with a practical example of supply air temperature and flow rate scheduling in a hollow core ventilated slab thermal storage system. The tests indicate that it improves results (the speed of convergence as well as the final result) without significant increasing computational expenses.

Key words: Genetic algorithms; crossover, mutation, selection operators; center of gravity.

1 Introduction

Recently, GA have been widely applied to different control and optimization problems due to their robustness, success in dealing with multi-modal and complex problems (Pal and Wang, 1996; Angelov and Guthke, 1997, Onnen et.al., 1997). The main specific of the GA as an optimization method is their implicit parallelism, which is a result of the evolution and the hereditary-like process (Michalewicz, 1996). GA is, in fact, a *driven stochastic search* technique, which combine stochastic (represented by *mutation* operator) and 'logical' search (represented by *crossover* of parental chromosomes and survival of the fittest by appropriate *selection*). These characteristics of GA offers possibilities for their improvement by appropriate balance between *exploration* (possible because of the diversity in the population) and *exploitation* (due to the

preservation of the search logic). Initially, improvements of GA has been sought in the optimal proportion and adaptation of the main parameters of the GA, namely probability of mutation, probability of crossover, population size (Davis, 1989) – (Grefenstette, 1986). More recently, the attention has been shifted to the breeding (process of forming new trial chromosomes at each epoch) (Muhlenbein and Schlierkamp-Voosen, 1993).

In this paper a supplementary crossover operator is introduced, which is more *informative* than mutation and more *innovative* than crossover itself. It increases diversity by creating a new chromosome different to the previous population elements and in the same time preserves the 'search logic' by accumulating weighted information about parental population. It is designed to be used in addition (in combination) with other crossover and mutation operators both in binary and real-valued GA. Supplementary crossover operator is applied to the best fitted two parental chromosomes only. The rest of the chromosomes in the population are produced by applying any other mutation and crossover operators as usual. The new child chromosome, very often have better fitness as it represents a CoG of the two best parental chromosomes and, thus, improves the search capability of the whole algorithm.

The supplementary crossover operator has been tested with a number of commonly used in the literature test functions. A practical problem of scheduling of the supply air temperature and flow rate to a ventilated slab thermal storage system is also presented. The results demonstrate its superiority compared with the case when it is not used.

2 Basic concepts of GA

GA mimics the process of natural selection where "the fittest survives". The basic difference between the GA and all other optimization techniques is its implicit parallelism, a result of the evolution and hereditary. The GA explore a set of trial points (*chromosomes*) forming a *population* at each iteration (*epoch*). A *gene* (g) in the chromosome represents binary encoded problem variable (x) in the 'standard' binary-coded GA (Goldberg, 1989) or directly its value in the real-value coded GA (Michalewicz, 1996).

chromosome$_1$				chromosome$_2$				…	chromosome$_{ps}$			
g_1^1	g_2^1	⋯	g_n^1	g_1^2	g_2^2	⋯	g_n^2	⋯	g_1^{ps}	g_2^{ps}	⋯	g_n^{ps}

Table 1: Population of Individual Chromosomes

In binary coded GA each gene is represented by a number of bits, which have value 0 or 1. For example:

$g_i = \{1;0;1;1;1;0;0\};$ $\quad i=1,2,...,n$

In real-value coded GA each gene is a real value representing certain variable $g_i = x_i$.

The measure of the quality of a chromosome (candidate solution) is its fitness function (F), which should be maximized. All individual chromosomes are evaluated (by calculating their fitness) at each epoch. A part of the chromosomes from the current epoch (parental chromosomes) are *selected* for mating and reproduction, based on their fitness values.

Crossover and *mutation* are then applied for producing new (child) trial points (chromosomes). An example of a two-point crossover is represented as:

$$chrom_{child}^1 = \{ g_1^{\bullet}; g_2^{\bullet};...; g_{xover_1-1}^{\bullet}; g_{xover_1}^{\circ}; g_{xover_1+1}^{\circ};....; g_{xover_2-1}^{\circ}; g_{xover_2}^{\bullet}; g_{xover_2+1}^{\bullet};...; g_{n-1}^{\bullet}; g_n^{\bullet} \}$$

$$chrom_{child}^2 = \{ g_1^{\circ}; g_2^{\circ};...; g_{xover_1-1}^{\circ}; g_{xover_1}^{\bullet}; g_{xover_1+1}^{\bullet};...; g_{xover_2-1}^{\bullet}; g_{xover_2}^{\circ}; g_{xover_2+1}^{\circ};...; g_{n-1}^{\circ}; g_n^{\circ} \}$$

where $^{\bullet}$ denotes the first parent;

$^{\circ}$ denotes the second parent;

$xover_1$ denotes the first crossover point

$xover_2$ denotes the second crossover point

Mutation in binary-coded GA is a triggering from 0 to 1 and vice versa. In real-coded GA, generally, uniform and non-uniform mutation exist (Michalewicz, 1996) with alteration of the mutated gene to a random value (in the range of feasible values) in the former one and alteration of the mutated gene with a certain value (added or subtracted depending on the 0 or 1 value of a random number) in the later one.

A two-point mutation operator, generally, could be represented as:

$$chrom_{child} = \{g_1; g_2; ... ; g_{i_1-1}; g_{i_1}^{mut_1}; g_{i_1+1};...; g_{i_2-1}; g_{i_2}^{mut_2}; g_{i_2+1};...;g_n\}$$

where $g_{i_1}^{mut_1}$ *and* $g_{i_2}^{mut_2}$ are mutated genes.

A GA could be represented by the following pseudo-code:

```
Program GA
Begin
    Number_of_epochs = 0;
    Set the initial population Π0
    Determine fitness function value F;
    While (Number_of_epochs < Maximal_number_of_epochs) do
        Number_of_epochs = Number_of_epochs + 1;
        Selection;
        Crossover;
        Mutation;
        Fitness evaluation;
   end
End.
```

In this general scheme the main objects (Selection, Crossover and Mutation) could vary depending on the specific type chosen (Michalewicz, 1996).

3 New supplementary crossover operator

3.1 New operator - definition

The operator proposed in the paper is designed to be used in combination with (in addition to) the usually used crossover operators both in binary- and real-coded GA. It considers one of the child chromosomes to be produced by a special breeding of the two best fitted parental chromosomes (called $chrom^{prime}$ and $chrom^{second}$), while all other (ps-1) child chromosomes are produced in an usual way. The last place in the population is preserved for this special chromosome, which represents the center of gravity of $chrom^{prime}$ and $chrom^{second}$ from the previous population (Table 2):

chromosome $_1^i$	chromosome $_2^i$	…	CoG^{i-1}

Table 2 Population Π_i

For the fittest chromosome we have:

$$F(chrom^{prime}) \geq F(chrom_j);\ j=1,...,ps$$

If it take the last place in the population, then the second best chromosome ($chrom^{second}$) could be determined as:

$F(chrom^{second}) \geq F(chrom_j);$ $j=1,...,ps-1$

Using these notations, the genes of the child chromosome are determined as CoG of the parental ones:

$$g_i^{CoG} = \frac{g_i^{prime} * F(chrom^{prime}) + g_i^{second} * F(chrom^{second})}{F(chrom^{prime}) + F(chrom^{second})}; \quad i=1,\ldots,n$$

where g^{prime} denotes a gene from the $chrom^{prime}$;
g^{second} denotes a gene from the $chrom^{second}$.

In the case of binary GA the supplementary crossover operator is applied after selection and decoding of the genes values from binary into decimal numbers together with the main crossover operator. Each gene of the resulting child (CoG-based) chromosome is then encoded into binary bits in a similar way as the rest (ps-1) child chromosomes (see Table 2) and, finally, mutation and reproduction are applied to the whole new ($\prod^{i+1}$) population.

3.2 New operator - how it works

Let us consider a simple example to illustrate the new operator. The example population consists of chromosomes having 4 real-coded genes each. Let the best chromosome at the *i-th* epoch be:

$chrom^{prime} = \{1; 3; 4; 2\}$ Let its fitness is $F(chrom^{prime}) = 0.85$.

Further, let the $chrom^{second}$ and its fitness be respectively:

$chrom^{second}=\{8; 6; 5; 3\};$ $F(chrom^{second}) = 0.75$

Then the CoG-based child chromosome will be:

$CoG=\{4.28; 4.41; 4.47; 2.47\}$, because

$$g_1^{CoG} = \frac{1*0.85 + 8*0.75}{0.85 + 0.75} = 4.28;$$

$$g_2^{CoG} = \frac{3*0.85 + 6*0.75}{0.85 + 0.75} = 4.41,$$

etc.

3.3 New operator - why it works

Though, it is difficult to prove strongly that some new operator in GA is better even for some class of problems because of the probabilistic nature of the GA (Michalewicz, 1996), we can expect that in many cases CoG chromosome will have better fitness. This could easily be illustrated with the very simple example of one variable function $F(x)= e^{-0.1(\frac{x^2-11}{x})^2}$ (Fig.1).

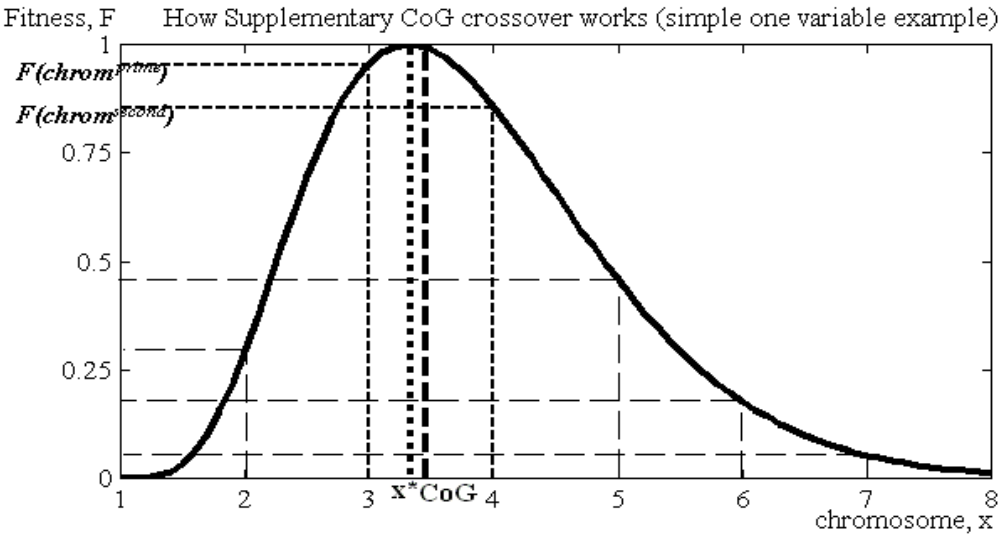

Figure 1: How Supplementary CoG Crossover Works - simple one variable example

Let the i-th population be $\prod^i=\{1; 2; 3; 4; 5; 6; 7; 8\}$. The $chrom^{prime}$ *and* $chrom^{second}$ will obviously be respectively:

$\boldsymbol{chrom^{prime} = 3;}$ $\quad F(chrom^{prime}) = e^{-0.0444} = 0.9565;$

$chrom^{second} = 4;$ $\quad F(chrom^{second}) = e^{-0.1563} = 0.8553;$

The CoG-based new child chromosome/gene (in the case of one variable the chromosome is equivalent to the gene) then will be:

$$\boldsymbol{CoG^i} = \frac{3\exp(-0.0444) + 4\exp(-0.1563)}{\exp(-0.0444) + \exp(-0.1563)} = \boldsymbol{3.4721}; \quad F(3.4721) = 0.9908$$

It is easy to see that it is much closer to the real maximum ($\sqrt{11}$). In the next population *CoG*-based child chromosome is considered. Note that the last gene (3) is due to the *elitist* strategy applied in addition to CoG in this case:

$\prod^{i+1}=\{ gene_1^{i+1}; gene_2^{i+1}; ...; gene_6^{i+1}; \boldsymbol{3.4721; 3}\}$

The other 6 chromosomes are produced by crossover and mutation of the parental chromosomes from the previous epoch (generation) as usual.

Analytically it is possible to prove that improvements will occur for the simplest case with one variable and convex in the interval $(x_-;x_+)$ fitness function (F) when:

$$x_- \le x^* \le x_+$$

where $x^* = \{x \mid F(x^*) = \max(F)\}$

$x_- = \min(crom^{prime}, chrom^{second})$

$x_+ = \max(crom^{prime}, chrom^{second})$

Real situations, however, are more complex, but as the test results indicate improvements often occur. This could be explained with the fact that CoG-based child chromosome is produced by the two best parent individuals incorporating also information about their fitness. By differ from the simple hill climbing it determines the new (often better) value of variables (x) directly (without using an estimation of the gradient and a step, which is usually computationally expensive, problem-dependent and a source of subjectivity).

4 Test examples

The new supplementary CoG-based crossover operator has been applied to five different commonly used numerical test problems (Michalewicz, 1996). Each test is performed with the same GA parameters (probabilities of crossover, mutation, population size). For consistency of the results, the same random number sequence is also used in both cases (with and without applying the new supplementary crossover operator).

4.1 Numerical test functions

Two series of 30 runs has been carried out with all of the five test functions:

i) search stops after a value of the objective function (J^*) close to the theoretical optimum ($J^{optimal}$) is reached ($J^* \approx J^{optimal}$). The number of epochs needed is recorded for both cases:

- using CoG-based supplementary crossover (N_{COG});
- without using CoG-based supplementary crossover (N_{conv}).

They make possible to calculate the rate of convergence ($rate_{conv} = \frac{N_{CoG}}{N_{conv}}$) which illustrates the effect of the new operator in saving computational time. Number of floating point operations for both cases is also registered and respective rate is calculated. In addition the number of improvements (N_{impr}) is registered as the number of epochs in which $CoG^i > F(chrom_j^{i+1})$; $j=1,2,\ldots ps$; $i=1,2,\ldots,N_{epochs}$. This indicates the number of cases in which the improvements in the fitness function are due to the new operator (in all other cases normally used crossover and mutation leads to the fitness improvement);

ii) search stops after a pre-specified number of epochs (max_N_{epoch}). Then objective function values and the number of improvements (N_{impr}) are recorded. The rate of objective is calculated ($rate_{obj} = \frac{J_{CoG}}{J_{conv}}$), which illustrates improvements in precision.

4.1.1 De Jong's function

The simplest test function is the so-called De Jong's function. It is continuos, convex and unimodal function:

$$f_i(x) = \sum_{i=1}^{n} x_i^2$$

The global minimum ($J^{optimal}$) of $f(x) = 0$ is at $x_i = 0$. The test with n = 30 variables and parameters of GA $p_c = 0.6$, $p_m = 0.03$, ps = 30 has been carried out for 60 different runs. The results are shown in Table 3 and a typical plot of the convergence is given on Fig. 2 (In this as well as in all next diagrams solid line represents the case when the new operator is used, while the dash-dotted line - the case when it is not used).

From the results one can conclude that the new operator allows to find the same solution ($J^*=0.1$) for almost 25% less epochs in average. In 15.8% of epochs improvement of the fitness is due to the new operator. Similarly, for a fixed number (3000) of epochs the result is closer to the global minimum of the objective function

Run	Results after **$J^* \leq 0.1$ is** reached	Results after **$max\_N_{epochs}=3000$**

No	N_{CoG}	N_{conv}	N_{impr}	$rate_{conv}$	$rate_{flops}$	J^{CoG}	J^{conv}	$rate_{obj}$	N_{impr}
1	2308	2750	285	0.8393	0.8424	0.0189	0.0726	0.2603	407
2	1218	2056	215	0.5924	0.594	0.0161	0.1079	0.1492	442
3	1817	2138	213	0.8499	0.8531	0.0168	0.0803	0.2092	590
4	1357	2332	119	0.5819	0.5838	0.0275	0.0769	0.3576	343
5	2053	2331	506	0.8807	0.8842	0.0250	0.0810	0.3086	265
6	1718	1912	135	0.8985	0.9022	0.0130	0.0331	0.3927	641
7	1775	3000	199	0.5917	0.5935	0.0423	0.0424	0.9976	667
8	1704	2041	271	0.8349	0.8373	0.0326	0.0995	0.3276	315
9	1469	2395	152	0.6134	0.6149	0.0106	0.0188	0.5638	245
10	1776	2057	283	0.8634	0.8668	0.0374	0.0482	0.7759	356
11	2552	2647	248	0.9641	0.9681	0.0146	0.0347	0.4207	562
12	2159	3000	474	0.7197	0.7221	0.0608	0.0906	0.6711	352
13	2526	3000	311	0.842	0.8453	0.0553	0.0748	0.7393	677
14	1378	2761	296	0.4991	0.501	0.0335	0.0492	0.6809	311
15	1539	2570	227	0.5988	0.6008	0.0541	0.0600	0.9017	350
16	1831	2246	227	0.8152	0.8181	0.0395	0.1629	0.2425	596
17	1834	2617	151	0.7008	0.7034	0.0190	0.0765	0.2484	370
18	1447	1715	251	0.8437	0.8464	0.0278	0.0921	0.3018	379
19	1957	2282	210	0.8576	0.8609	0.0450	0.0587	0.7666	360
20	1558	2663	212	0.5851	0.5871	0.0185	0.0476	0.3887	333
21	1715	2112	264	0.8121	0.8142	0.0211	0.0387	0.5452	406
22	1750	2050	357	0.8537	0.8559	0.0899	0.0954	0.9423	364
23	1530	2152	312	0.7109	0.7109	0.0385	0.0405	0.9506	268
24	1659	2578	288	0.6435	0.6435	0.0834	0.1228	0.6792	575
25	1724	2350	477	0.7336	0.7352	0.0312	0.0614	0.5081	389
26	1453	2467	366	0.5889	0.5889	0.0268	0.1070	0.2505	444
27	2002	2225	515	0.8998	0.9023	0.0372	0.0394	0.9442	341
28	1889	2716	298	0.6955	0.6955	0.0976	0.1458	0.6694	492
29	2112	2235	302	0.9449	0.9449	0.0388	0.0506	0.7668	397
30	1618	2186	273	0.7401	0.7401	0.0266	0.0343	0.7755	391
Avrg	**1781**	**2386**	**281**	**0.7532**	**0.7552**	**0.0300**	**0.0689**	**0.5681**	**427**

Table 3 DeJong's function. Two series of 30 runs

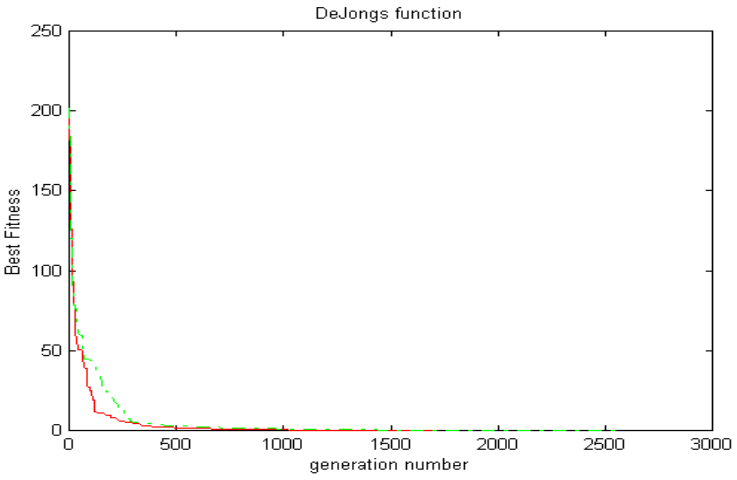

Fig. 2 A typical convergence in DeJong's function minimization

($J^{optimal}=0$) - the value of the objective function is almost 2 times less in average with improvements in 14.2% of epochs due to the new operator. It should be mention that the rate of elementary floating point operations is practically the same as the rate of convergence, which means that the additional computational effort due to the application of the new operator is negligible.

4.1.2 Rastrigin's function

Rastrigin's function has many local minima as it uses cosine modulation. Although, the test function is highly multi-modal, the minima are regularly distributed.

$$f_i(x)=10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x))$$

The global minimum ($J^{optimal}$) of $f(x)=0$ is at $x_i=0$. GA parameters used were p_m=0.005, p_c=0.8, number of bits = 10, population size =30. Test results for n=30 variables for two series of 30 runs are given in Table 4. A typical convergence in both cases (with and without using the new supplementary CoG-based crossover) are shown on Fig.3. Similar conclusions could be made that 3 times faster convergence take place to reach $J^*=100$ with practically no additional computation effort; in 22% of cases improvement is due to the new operator and 1.54 times better result is achieved for the fixed number (10000) of epochs and improvements occur in the same proportion (21.9%) of epochs.

4.1.3 Sum of different powers

The sum of different powers is usually used in unimodal tests:

$$f_i(x)=\sum_{i=1}^{n}|x|_i^{i+1}$$

The global minimum ($J^{optimal}$) of $f(x)=0$ is at $x_i=0$. The test results for n=30 variables and the following GA parameters (p_m=0.01, p_c=0.6, ps=60, bits=10) are given in Table 5 and a typical convergence is depicted on Fig.4. In this test the advantage of the addition of the new operator is most obvious: while $J^*=0.01$ is reached for 18 epochs in average when it is used, 2781 epochs are necessary for the case when it is not used, which means

Run No	Results after **J* ≤ 100 is** reached					Results after **max_N$_{epochs}$ = 3000**			
	N_{CoG}	N_{conv}	N_{impr}	$rate_{conv}$	$rate_{flops}$	J^{CoG}	J^{conv}	$rate_{obj}$	N_{impr}
1	1432	4169	348	0.3435	0.3441	82.903	119.72	0.6924	700
2	738	10000	214	0.0738	0.0739	63.052	81.856	0.7703	744
3	802	6494	156	0.1235	0.1237	88.878	167.90	0.5293	269
4	2419	10000	333	0.2419	0.2423	71.121	112.57	0.6318	726
5	1939	9528	150	0.2035	0.2039	76.592	78.459	0.9762	743
6	1702	2122	299	0.8021	0.8036	60.040	179.97	0.3336	901
7	3986	4323	882	0.9220	0.9238	64.056	172.59	0.3711	1023
8	754	2824	128	0.2670	0.2674	88.375	130.09	0.6793	743
9	655	3040	159	0.2155	0.2158	43.319	159.33	0.2719	902
10	4112	9906	1002	0.4151	0.4159	50.529	91.858	0.5501	849
11	3148	3866	646	0.8143	0.8158	76.961	90.826	0.8473	381
12	1204	4415	333	0.2727	0.2732	74.444	119.87	0.6210	424
13	1093	5912	213	0.1849	0.1852	75.388	148.96	0.5061	304
14	854	3969	176	0.2152	0.2155	58.065	119.20	0.4871	639
15	1195	2123	252	0.5629	0.5639	51.118	153.48	0.3330	1117
16	1467	3201	382	0.4583	0.4591	78.084	104.75	0.7454	852
17	1240	3873	397	0.3202	0.3207	68.007	109.52	0.6209	696
18	2467	3000	436	0.8223	0.8239	68.309	116.99	0.5839	218
19	846	7241	193	0.1168	0.1170	125.16	132.95	0.9414	627
20	1527	2762	212	0.5529	0.5538	103.84	107.34	0.9674	276
21	640	10000	122	0.0640	0.0641	79.832	137.29	0.5815	1292
22	2257	7353	571	0.3069	0.3075	113.90	140.51	0.8106	382
23	974	1358	337	0.7172	0.7185	76.928	89.272	0.8617	398
24	1316	8651	208	0.1521	0.1524	57.077	81.241	0.7026	902
25	2877	5412	758	0.5316	0.5326	72.125	142.74	0.5053	407
26	843	4135	162	0.2039	0.2042	114.70	116.09	0.988	624
27	2290	6354	422	0.3604	0.3610	78.067	149.76	0.5212	571
28	1148	4288	298	0.2677	0.2682	81.932	128.93	0.6355	789
29	865	7823	182	0.1106	0.1107	89.497	115.46	0.7751	669
30	3647	7383	1018	0.4940	0.4949	100.00	130.59	0.7658	517
Av.	**1681**	**5518**	**366**	**0.3712**	**0.3719**	**76.977**	**124.34**	**0.6497**	**656**

Table 4 Rastrigin's function. Two series of 30 runs

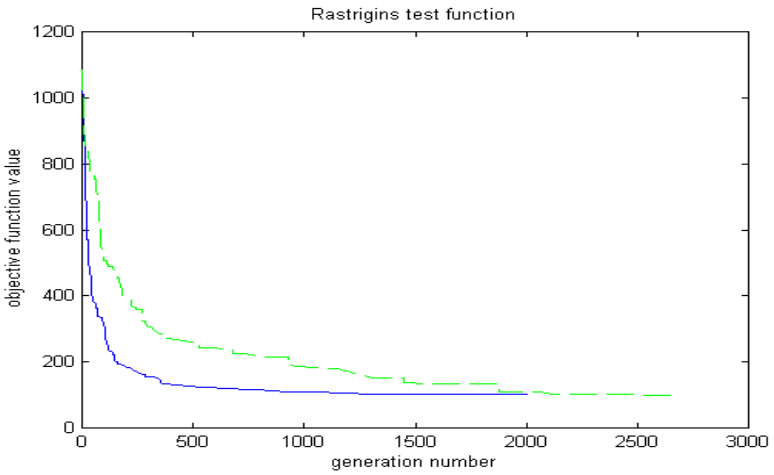

Fig. 3: Rastrigin's Function Minimization

Run	Results after **J* ≤ 0.01 is** reached					Results **after 3000 epochs**			
						***10^{-10}**			
No	N_{CoG}	N_{conv}	N_{impr}	$rate_{conv}$	$rate_{flops}$	$rate_{obj}$	J^{CoG}	J^{conv}	N_{impr}
1	4	3000	2	0.0013	0.0012	0.4130	0.0091	0.0221	458
2	10	2591	4	0.0039	0.0037	0.7090	0.0091	0.0129	1520
3	14	3000	6	0.0047	0.0045	0.2070	0.0091	0.0044	436
4	3	3000	2	0.0010	0.0008	0.3070	0.0091	0.0296	405
5	3	3000	2	0.0010	0.0008	0.1640	0.0091	0.0556	369
6	148	3000	33	0.0493	0.0494	0.2980	0.0091	0.0305	360
7	4	3000	1	0.0013	0.0012	0.7490	0.0091	0.0122	305
8	28	3000	12	0.0093	0.0092	0.4350	0.0091	0.0209	333
9	8	3000	4	0.0027	0.0025	0.2420	0.0091	0.0376	1557
10	3	3000	1	0.0010	0.0008	0.7620	0.0091	0.0119	411
11	14	3000	11	0.0047	0.0045	0.3750	0.0091	0.0243	1680
12	3	3000	2	0.0010	0.0008	0.5480	0.0097	0.0177	1547
13	5	3000	3	0.0017	0.0015	0.4940	0.0092	0.0186	518
14	3	3000	0	0.0010	0.0008	0.5750	0.0091	0.0158	320
15	3	3000	1	0.0010	0.0008	0.3870	0.0096	0.0249	423
16	9	1395	2	0.0065	0.0061	0.3590	0.0104	0.0291	403
17	12	3000	4	0.0040	0.0039	0.3150	0.0091	0.0289	1724
18	3	3000	2	0.0010	0.0008	0.2920	0.0094	0.0322	1552
19	3	3000	2	0.0010	0.0008	0.3220	0.0092	0.0284	285
20	3	3000	1	0.0010	0.0008	1.2200	0.0173	0.0142	341
21	3	3000	1	0.0010	0.0008	1.6060	0.0369	0.0023	341
22	7	3000	2	0.0023	0.0022	0.3990	0.0094	0.0235	1456
23	43	3000	1	0.0143	0.0142	3.2120	0.0824	0.0257	1684
24	3	344	2	0.0087	0.0074	0.3890	0.0091	0.0234	1400
25	38	3000	13	0.0127	0.0126	0.3700	0.0114	0.0308	267
26	2	3000	1	0.0007	0.0005	0.6630	0.0182	0.0275	490
27	18	3000	2	0.0060	0.0059	0.2850	0.0092	0.0321	1725
28	3	1111	1	0.0027	0.0023	0.8380	0.0183	0.0218	1776
29	117	3000	6	0.0390	0.0390	3.4400	0.0277	0.0081	1542
30	8	3000	3	0.0027	0.0025	0.6110	0.0092	0.0151	411
Av.	**18**	**2781**	**4**	**0.0063**	**0.0060**	**0.6995**	**0.0142**	**0.0247**	**868**

Table 5 Different powers function. Two series of 30 runs

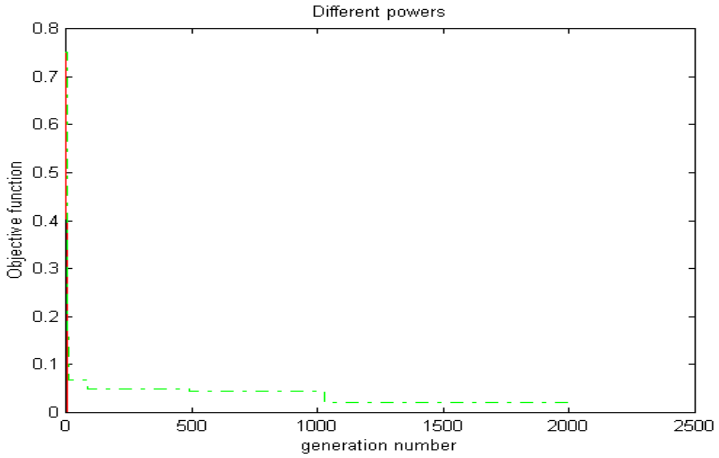

Fig. 4: Different powers function minimization

150 times more epochs (!). In each forth epoch an improvement occur due to the addition of the new operator. 14 billion times (!!!) better result is achieved for the same fixed number of epochs (3000).

4.1.4 Schwefel's function

Schwefel's function (Schwefel, 1981) determines a geometrically distant minimum (at x_i = 420.9687; $J^{optimal}$ = -418.9829*n) from the next best local minima. Therefore, the search algorithms often converge in a wrong direction.

$$f_i(x)=10n+\sum_{i=1}^{n} x_i - \sin(\sqrt{|x_i|})$$

The test results for n=20 variables and p_m=0.01, p_c=0.6, ps=30, bits=10 are represented in Table 6 and a typical convergence is given in Fig.5. The results are quite obvious: the GA which does not uses the new operator much more often falls into a local extremum. In average 5 times less epochs are necessary to reach *J*=-6000* and value of the objective function better with 56**%** in average is reached for the same fixed number of epochs (1000). Improvements occur in 88**%** of epochs (!) due to the new operator. It is interesting to mention that the standard deviation in results is 5 times higher for the case when the new operator is not used than when it is used. This could be explained by the fact that in the case when it is not used the algorithm relatively often falls into a local extremum.

4.1.5 Griewangk's function

Griewangk's function is similar to Rastrigin's function and has many regularly distributed local minimuma which are spread over the search space. The global one ($J^{optimal}$) is at $x_i=0$; $f(x)=0$.

$$f_i(x)=\sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$$

The test results for n=30, p_m=0.01, p_c=0.7, ps=30, bits=20 variables are given in Table 7 and a typical convergence is represented on Fig.6. Again, 2 times faster convergence with practically no additional computation effort has place. In 24% of the epochs improvements are due to the application of the new operator. For a fixed number of

Run No	Results after **J* ≤ -6000 is** reached					Results **after 1000 epochs**			
	N_{CoG}	N_{conv}	N_{impr}	$rate_{conv}$	$rate_{flops}$	J^{CoG}	J^{conv}	$rate_{obj}$	N_{impr}
1	91	4776	78	0.0191	0.0191	-6820	-5626	1.212	942
2	99	6904	96	0.0143	0.0144	-7237	-4055	1.785	942
3	83	783	58	0.1060	0.1061	-7021	-4671	1.503	976
4	100	4764	78	0.0210	0.0210	-7329	-5472	1.339	938
5	84	2264	83	0.0371	0.0371	-7438	-4444	1.674	919
6	182	6267	161	0.0290	0.0291	-7381	-4574	1.614	943
7	140	7246	78	0.0193	0.0193	-7512	-5504	1.365	958
8	151	3648	140	0.0414	0.0416	-7225	-6491	1.113	917
9	83	7542	75	0.0110	0.0112	-7773	-3931	1.977	842
10	67	2729	65	0.0246	0.0246	-7656	-4760	1.608	937
11	81	151	77	0.5364	0.5398	-7277	-3837	1.897	924
12	107	517	99	0.2070	0.2072	-6872	-4072	1.688	960
13	53	158	51	0.3354	0.3361	-7416	-3905	1.899	892
14	220	366	215	0.6011	0.6051	-7330	-3843	1.907	891
15	83	1097	63	0.0757	0.0759	-6735	-5540	1.216	869
16	76	1422	63	0.0534	0.0535	-7507	4209	1.784	959
17	57	158	56	0.3608	0.3616	-6899	-4116	1.676	916
18	36	53	35	0.6792	0.6816	-7717	-3805	2.028	927
19	65	176	59	0.3693	0.3705	-7508	-4537	1.655	928
20	68	1286	67	0.0529	0.0529	-7642	-4155	1.839	927
21	105	870	97	0.1207	0.1211	-7436	-4863	1.529	931
22	44	93	43	0.4731	0.4741	-7398	-4377	1.690	928
23	60	2465	31	0.0243	0.0243	-7527	-4707	1.599	934
24	57	704	54	0.0810	0.0810	-7615	-4161	1.830	883
25	78	620	77	0.1258	0.1261	-7435	-6746	1.102	903
26	56	349	45	0.1605	0.1606	-7047	-3896	1.809	952
27	96	132	95	0.7273	0.7318	-7182	-4502	1.595	901
28	93	2011	82	0.0462	0.0464	-7311	-4393	1.664	934
29	112	1141	78	0.0982	0.0985	-7658	-7266	1.054	970
30	120	554	116	0.2166	0.2176	-7434	-4585	1.621	927
Av.	**92**	**2042**	**81**	**0.1889**	**0.1896**	**-7345**	**-4700**	**1.563**	**926**

Table 6 Schwefel's function. Two series of 30 runs

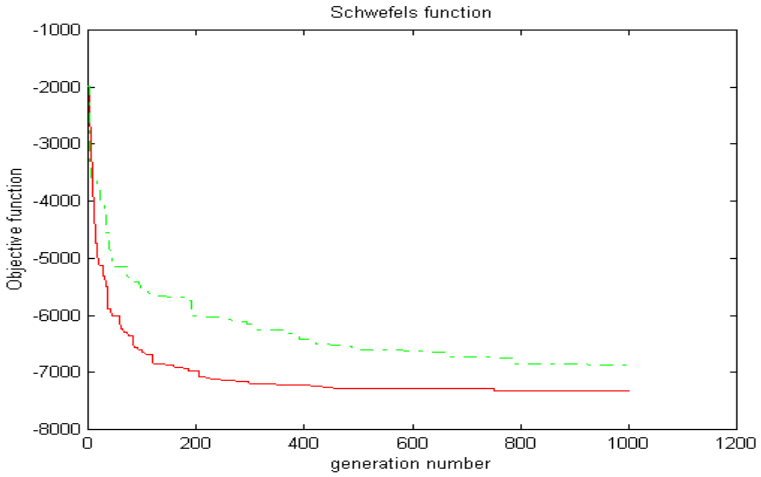

Fig. 5: Schwefel's Function Minimization

Run No	Results after **J* ≤ 1000 is** reached					Results **after 3000 epochs**				
	N_{CoG}	N_{conv}	N_{impr}	$Rate_{con}$	$rate_{flops}$	J^{CoG}	J^{conv}	$rate_{obj}$	$rate_{flops}$	N_{impr}
1	733	1383	187	0.5300	0.5307	0.0712	0.1055	0.6755	1.0031	482
2	709	1417	142	0.5004	0.501	0.0181	0.0746	0.2428	1.003	476
3	604	1507	150	0.4008	0.4013	0.0298	0.0749	0.3973	1.0031	495
4	711	1029	137	0.6910	0.692	0.0369	0.073	0.5056	1.0029	453
5	711	1020	230	0.6971	0.6981	0.0406	0.0883	0.4605	1.0028	522
6	824	1253	161	0.6576	0.6586	0.0647	0.0666	0.9722	1.003	396
7	476	1103	119	0.4316	0.432	0.0363	0.3295	0.1102	1.0029	538
8	481	1149	128	0.4186	0.4191	0.0312	0.2225	0.1402	1.003	423
9	613	1597	184	0.3838	0.3843	0.0256	0.0597	0.4292	1.003	494
10	812	1080	198	0.7519	0.753	0.0804	0.0863	0.9319	1.0029	549
11	694	987	151	0.7031	0.7042	0.0738	0.1466	0.5032	1.0028	358
12	766	1045	192	0.7330	0.7341	0.0702	0.1165	0.6026	1.003	713
13	930	1448	198	0.6423	0.6432	0.0272	0.0572	0.476	1.0031	468
14	559	1204	167	0.4643	0.4648	0.0956	0.1079	0.8859	1.0029	486
15	686	1268	142	0.5410	0.5417	0.0376	0.1904	0.1976	1.003	536
16	769	1537	176	0.5003	0.501	0.0385	0.1275	0.3016	1.0028	507
17	787	1052	171	0.7481	0.7492	0.0373	0.1183	0.3149	1.0028	618
18	744	1790	176	0.4156	0.4162	0.0546	0.1277	0.4274	1.0028	703
19	848	994	265	0.8531	0.8545	0.0307	0.0465	0.6608	1.003	633
20	603	1146	152	0.5262	0.5269	0.0474	0.1182	0.4008	1.0029	369
21	717	1114	204	0.6436	0.6445	0.037	0.061	0.6072	1.0031	502
22	627	1422	131	0.4409	0.4415	0.0384	0.2698	0.1422	0.9559	589
23	863	1651	182	0.5227	0.5234	0.0258	0.0877	0.2942	1.0028	550
24	789	1175	174	0.6715	0.6725	0.03	0.0511	0.5878	1.0029	394
25	602	1521	166	0.3958	0.3963	0.0493	0.1024	0.481	1.0029	398
26	730	1508	157	0.4841	0.4847	0.0516	0.0581	0.8882	1.0028	378
27	865	1877	193	0.4608	0.4615	0.0257	0.0561	0.4572	1.0029	400
28	663	1318	105	0.5030	0.5037	0.0471	0.0688	0.6852	1.0029	503
29	543	1588	168	0.3419	0.3423	0.0291	0.1432	0.2032	1.0031	591
30	677	1323	198	0.5117	0.5124	0.0309	0.0666	0.4634	1.0029	437
Av.	**706**	**1317**	**170**	**0.5522**	**0.5530**	**0.0438**	**0.1101**	**0.4815**	**1.0014**	**499**

Table 7 Griewangk's function. Two series of 30 runs

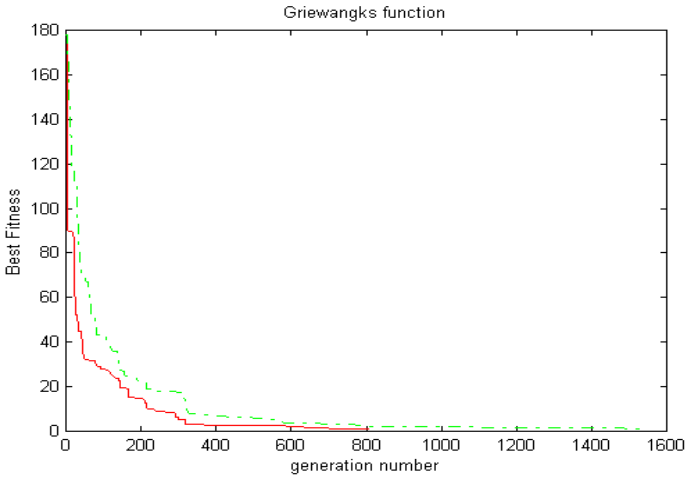

Fig. 6: Griewangk's Function Minimization

epochs (3000) 2 times better result (in average) is registered and in 17% of epochs improvements are due to the new operator. It is interesting to mention that the standard deviation of the results of the 30 runs with the new operator are more than 3 times smaller (0.019 instead of 0.066) which means that the role of mutation and the randomness is less important in this case than when it is not used.

4.2 Air-conditioning system optimization

The last example represents an practical optimization problem: to minimize the energy costs in a hollow core ventilated slab system used as a thermal storage during the night and off-peak electricity tariff periods such that not to compromise the comfort of the occupants (Ren, 1997). The results of application of the new CoG-based supplementary crossover operator together with the GA, which does not uses it are depicted on Fig.7-Fig.10 (all other parameters, including random generator are the same).

Supplementary CoG-based crossover improves significantly the convergence (Fig.7) as well as the final result: while the thermal comfort is practically not changed, the more effective (with 6%) solution is achieved (normalized value of the costs is 0.90886 instead of 0.95622). The optimal profiles of the fan power, supply air temperature and flow rate are given on Fig.8 - Fig.10 respectively.

The effect is achieved by lowering the supply air flow rate during the morning pre-cooling (Fig.10) while fan is switched on an hour earlier (Fig.9) with lower power. Supply air-temperature during the morning pre-cooling and after working evening hours is lower (Fig.9).

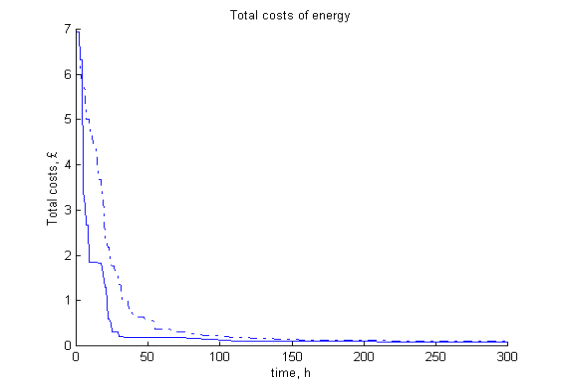

Fig. 7 Total costs of energy used

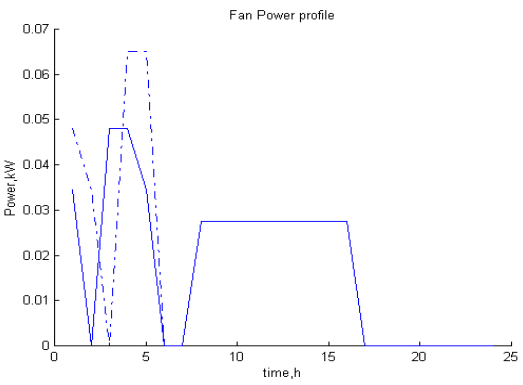

Fig. 8 Fan Power Profile

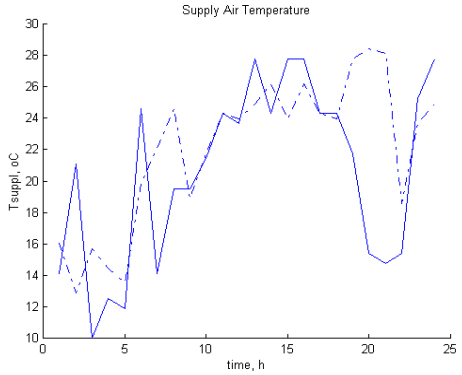

Fig. 9 Supply Air Temperature

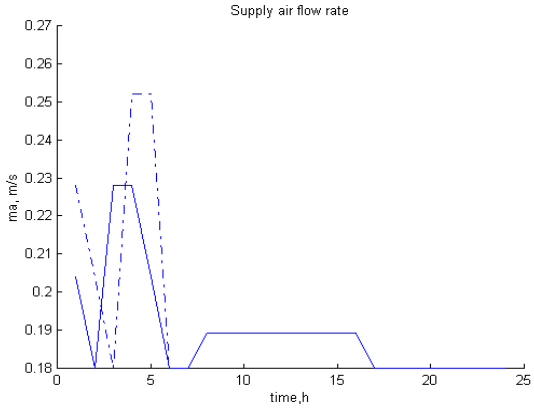

Fig. 10 Supply Air Flow Rate

All final results are given in Table 8 with GA parameters used. They illustrate efficiency of the proposed new CoG-based supplementary crossover operator.

Test functions	Improvements effect				GA parameters		
	$rate_{conv}$	$N_{improve}$	$rate_{obj}$	$N_{improve}$	p_m	p_c	ps
De Jong's	**0.7532**	**15.8%**	**0.5681**	**14.2%**	0.01	0.6	30
Rastrigin's	**0.3712**	**21.8%**	**0.6497**	**21.9%**	0.005	0.8	30
Schwefel's	**0.1889**	**88.0%**	**1.5628**	**92.6%**	0.01	0.6	60
Griewangk's	**0.5522**	**24.1%**	**0.4815**	**16.6%**	0.01	0.7	30
different powers	**0.0063**	**22.2%**	**$7*10^{-11}$**	**28.9%**	0.01	0.6	60
Air conditioning	**$Cost_{CoG}$=0.90896**		**$Cost_{conv}$=0.9562**		0.001	1	80

Table 8 Improvements effect (summarizing averages from all tests)

5 Conclusions

The new supplementary crossover operator is proposed and tested in the paper. It performs specific breeding between the two fittest parental chromosomes producing a new child chromosome, which is based on the center of gravity of the parental ones. The test results indicate that it leads to 2-3 times better results than when it is not used. Without significant increasing computational expenses the speed of convergence as well as the final result in tests is significantly improved. A limited proof of its efficiency is

provided for the case of one variable and using *elitists* selection strategy in combination with the new operator. The new operator could be used both in binary as well as in real-coded GA. A number of numerical tests as well as a practical example of supply air temperature and flow rate scheduling in a ventilated slab thermal storage system are presented, which proves the viability of the proposed new operator.

5 Acknowledgements

The author would like to acknowledge the support of the EPSRC (by grant GR/M97299) as well as the helpful comments of Dr. J.A.Wright and Dr. R. Farmani.

6 References

1. P.P.Angelov, R. Guthke, A GA-based Approach to Optimization of Bioprocesses Described by Fuzzy Rules, *Journal of Bioprocess Engineering*(1997) **v.16**, pp.299-301
2. L.Davis, Adapting Operator Probabilities in Genetic Algorithms, *Proc. of the International Conference on Genetic Algorithms ICGA89* (1989) 61-69
3. T. C. Fogarty, Varying the Probability of Mutation in The Genetic Algorithm, *Proc. of the International Conference on Genetic Algorithms ICGA89* (1989) 104-109
4. D.E.Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA (1989)
5. J.J.Grefenstette, Optimization of Control Parameters for Genetic Algorithms, *IEEE Transactions. on Systems Man and Cybernetics*, **16** (1) (1986) 122-128
6. P.Hajela, J.Yoo, Constraint Handling in Genetic Search - A Comparative Study, *Proc. of the American Institute of Aeronautics and Astronautics* (1995) 2176-2186
7. Matlab, High Performance Numeric Computation and Visualization Software, MathWorks Inc. (1994)
8. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, Berlin , 2nd edition (1996)
9. H.Muhlenbein, D.Schlierkamp-Voosen, Predictive Models for the Breeder Genetic Algorithm. I. Continuous Parameter Optimization, *Evolutionary Computation*, **1** (1) (1993) 25-49
10. L.M.Patanik, S.Mandavali, Adaptation in Genetic Algorithms, *In: Genetic Algorithms for Pattern recognition, S.K.Pal, P.P.Wang, Eds.*,CRC Press, Boca Raton, FL (1996), 45-64

11. S.K.Pal, P.P.Wang,*Genetic Algorithms for Pattern recognition*, CRC Press, Boca Raton, FL (1996)
12. C.Onnen et.al., Genetic algorithms for Optimization in Predictive Control, *Control Engineering Practice*, **5** (10) (1997) 1363-1372
13. M.J.Ren, *Optimal Predictive Supervisory Control of Fabric Thermal Storage Systems*, Ph.D. Thesis, Loughborough University, Loughborough, UK (1997)
14. Schwefel, H.-P., Numerical Optimization of Computer Models, John Wiley and Sons, Chichester (1981)
15. R.Yager, D. Filev, *Essentials of Fuzzy Modeling and Control*, John Willey and Sons, NY (1994)