Counter-Niching for Constructive Population Diversity

Maumita Bhattacharya, *Member, IEEE*

***Abstract*—Maintaining a desired level of diversity in the Evolutionary Algorithm (EA) population is a requirement to ensure that progress of the EA search is unhindered by premature convergence to suboptimal solutions. Loss of diversity in the EA population pushes the search to a state where the genetic operators can no longer produce superior or even different offspring required to escape the local optimum. Besides diversity's contribution to avoid premature convergence, it is also useful to locate multiple optima where there is more than one solution available. This paper presents a counter-niching technique [8] to introduce and maintain constructive diversity in the EA population. The proposed technique presented here uses informed genetic operations to reach promising, but un-explored or under-explored areas of the search space, while discouraging premature local convergence. Elitism is used at a different level aiming at convergence. The proposed technique's improved performance in terms solution accuracy and computation time is observed through simulation runs on a number of standard benchmark test functions with a genetic algorithm (GA) implementation.**

I. Background

In the context of evolutionary algorithm diversity means genetic variation in the population members. Diversity plays two major roles in an EA: firstly it delays convergence allowing increased exploration to find better single solution. Secondly, diversity helps to locate multiple solutions where there is more than one optimum. The significance of diversity and premature convergence in the context of evolutionary algorithms (specifically genetic algorithm) is quite different from what it is in its metaphor, natural evolution. Life forms have evolved in nature unhindered by being trapped in some local minima. Stagnation in evolutionary terms occurs only if it suits the specific species at a given period of time or if it is being forced to extinction. Again, redundant species often act as biological insurance against changes due to change in environmental conditions. Also, different ecotypes of the same species often exploit different niches. These ecotypes have similar enough genotypes to mate, but different phenotypes that maintain diversity in the breeding pool. This satisfies the prerequisite for avoiding premature convergence.

In case of evolutionary algorithm progress in the search process to create new solutions relies on the variation or diversity in the population. This variation is induced by the variation operators such as crossover by mating among individuals and mutation by introducing random changes in the individual's genome. However, high gene flow among the population members and high selection pressure leads to rapid loss in diversity resulting in low exploration. As the evolutionary process progresses, population diversity declines, as a canonical EA tends to concentrate more and more of its search effort near the already discovered "peaks" or "attractors", converging towards similar points or even single points in the search space, gradually reaching a near homogeneous state. High gene flow on the other hand, pushes the population towards homogeneous state by spreading genetic material across the population by means of unrestricted mating or crossover. This may in turn lead to premature convergence as the population looses the variation required to escape from local optima. Besides these roles of diversity regarding premature convergence in static optimization problems and locating multiple solutions, diversity also seems to be beneficial in problems involving non-stationary or dynamic environments. If the population converges towards similar or single points in the search space, all the future individuals in subsequent runs will be trapped in that single point even though the optimum may have moved to a new location in the fitness landscape. Presence of sufficient diversity in the population will mean the variation operators will be able to continue to generate new individuals, making it possible to find new optima.

Since the standard EA has problems maintaining population diversity, several EA models have been proposed that try to maintain or introduce diversity. The major researches can be broadly categorized as one of the following [10]:

a) Complex population structures to control gene flow, e.g., the diffusion model [14], the island model [14], the multinational EA [11] and the religion model [12].
b) Specialized operators to control and assist the selection procedure, e.g., crowding [7], deterministic crowding [13], and sharing [2] are believed to maintain diversity in the population.
c) Reintroduction of genetic material, e.g., random immigrants [5], mass extinction models [12], [3], and [15] are aimed at reintroduction of diversity in the population.
d) Dynamic parameter encoding (DPE), which dynamically resizes the available range of each parameter by expanding [9] or reducing the search window.
e) Diversity guided or controlled genetic algorithms that use a diversity measure to assess and control the survival probability of individuals and the process of exploration and exploitation [10].

This work was supported in part by a research grant of Charles Sturt University, Australia.

Maumita Bhattacharya is with the School of Business and Information Technology, Charles Sturt University, NSW 2640 AUSTRALIA (corresponding author to provide phone: 61 2 60519619; e-mail: maumita.bhattacharya@ ieee.org).

978-1-4244-1823-7/08/$25.00©2008 IEEE

Again, majority of the earlier methods to combat premature convergence, such as Crowding, Sharing, restricted mating, spatial mating and so on, involve additional computation. Methods, which reward parsimony or try to combine two measures, may result in individual solutions trading off parts of their fitness function. Further information on these techniques may be found in [1].

The COMMUNITY_GA technique [8], presented in this paper has a focus on intelligent introduction of diversity, while trying to avoid entrapment in non-optimal solutions. The proposed technique, first identifies EA's tendency to converge locally, and then introduces diversity by replacing redundant less-fit members of the local clusters or communities by promising samples from un-explored areas of the decision space. At the same time, exploitation is maintained by preserving representative members of the local clusters and also by elitism.

The rest of the paper is organized as follows: Section II presents the major functional aspects of the COMMUNITY_GA technique for introduction of constructive diversity. Experimental observations and discussions are presented in Section III. Finally, some conclusions are drawn in Section IV.

II. Counter-Niching for Constructive Diversity

The proposed technique first extracts information about the population landscape before deciding on introduction of diversity through informed mutation [8]. The aim is to identify locally converging regions or *donor* communities in the landscape that could spare less fit individuals those could be replaced by more promising members sampled in un-explored or under-explored sections of the decision space. The existence of such communities is purely based on the position and spread of individuals in the decision space at a given point in time. Once such regions are identified, random sampling is done on yet to be explored sections of the landscape. Best representatives replace the worst members of the identified regions. The term *counter-niching* is use to explain this specific characteristic of the algorithm, that counter the process of niche formation by introducing superior *aliens* into the community or cluster. Regular mutation and recombination takes place in the population as a whole. The basic functional components of the framework are as described in subsequent subsections.

A. COMMUNITY_GA: The Basic Framework

This algorithm probabilistically and randomly sample the global search space and explores for promising regions while concentrating search on the hyperplanes that are likely to contain good solutions. The COMMUNITY_GA algorithm is as explained below in Fig. 1.

Algorithm 1: *Procedure COMMUNITY_GA*

1: **begin**
2: $t = 0$
3: Initialize population $P(t)$
4: Evaluate population $P(t)$
5: **while** (not<termination condition>)
6: **begin**
7: $t = t + 1$
(* Perform pseudo-niching of the population*)
8: Call Procedure GRID_NICHING
(* Perform informed genetic operations *)
9: Call Procedure INFORMED_OP
10: Create new population using an elitist selection mechanism
11: Evaluate $P(t)$
14: **end while**
15: **end**

Fig. 1. The basic structure of the COMMUNITY_GA algorithm.

B. GRID_Niching: A Pseudo-niching Scheme to Extract Landscape Information

Here, we have used the term *niching*, simply to identify *environments* of individuals in the population, based on their spatial information. In other words, we try to identify the gross individual clusters in the decision space based on their genotypic proximity. This method organizes the space around the patterns and not the patterns themselves. It finds the spatial distribution information of the individuals in the decision space. First, in each generation, the population members are placed on a multidimensional grid data structure. This block partitioning of the decision space is done to identify initial signs of cluster formation. Once adequate number of individuals is found in any block, it is considered to be part of a potential cluster. Bounds of the identified blocks are expanded in all N dimensions in order to find a community or cluster.

The pseudo-niching procedure GID_NICHING is as described in Fig. 2.

Algorithm 2: *Procedure GRID_NICHING*

1: **begin**
(* Create grid structure *)
2: Obtain genotypically sorted population $P(t)$
3: Create the grid structure
4: Compute grid block densities D_{BI}, where, $D_{BI} = \frac{N_{BI}}{N_{POP}}$; N_{BI} and N_{POP} being the number of individuals in block (B_I) and the population size, respectively .
(* Identify potential community hub or cluster center *)
5: **for** each grid block B_I
6: **begin**
7: **if** ($D_{BI} >$ Threshold α) **then**
8: Mark block as potential community hub $B_{I\text{-}HUB}$
9: **end for**
(* Look for possible cluster around each marked block $B_{I\text{-}HUB}$ *)

10: **for** each marked block, $B_{I\text{-}HUB}$
11: **begin**
12: **while** (Number of new members found < Threshold N_{thr}) AND (not < boundary overlap with any marked cluster CLUST_K >)
13: **begin**
14: Expand the region $\Re \rightarrow R^{Block}$ in all dimensions by a predefined step size ∂
15: Find members that belong to the expanded region or new non-uniform hypercube $\Re$
16: **end**
17: Mark the new found community, CLUST_K where, K is the cluster number
18: **end while**
19: **end for**
20: Return community chart, consisting of list of CLUST_K
21: **end**

Fig. 2. The basic structure of the GRID_NICHING algorithm.

The procedure GRID-NICHING thus returns information about community or cluster formation in the population, for the current generation. The next section presents the INFORMED_OP algorithm, which sketches how this information is used to guide the mutation operation.

C. INFORMED_OP: The Informed Operator Algorithm to Guide Genetic Operations

The INFORMED_OP algorithm identifies locally converging communities with redundant members of similar fitness. The idea is to explore greater part of the solution space at the expense of these *extra* members. However, INFORMED_OP operates on selected communities only. Regular mutation and recombination is performed as usual on the entire population. This algorithm (see Fig. 3) uses the pseudo-niching information obtained from GRID-NICHING to guide the genetic operators.

Algorithm 3: *Procedure INFORMED_OP*

1: **begin**
2: Get community chart, consisting of list of CLUST_K
(* Perform Cluster analysis and Informed mutation *)
3: **for** each cluster CLUST_K,
4: **begin**
5: **if** (Fitness standard deviation of cluster, $\sigma_{CLUST\_K} \leq$ average fitness standard deviation of all clusters) AND cluster density, $D_{CI} \geq$ average cluster density of all clusters) **then**
6: **begin**
7: Randomly sample N individuals in *virgin zones* (an archive of unexplored blocks) and replace M worst representatives from the identified cluster, by M best-sampled individuals.
8: Update *virgin zone* archive
9: **end if**
10: **else if** (Average cluster fitness $f_{CLUST\_AVG}(K)>$ Average fitness of all clusters) **then**
11: **begin**
12: Mark best individual of the cluster as an *elite* to be transferred to the new population unchanged, irrespective of the individual's fitness ranking when the entire population is considered.
13: **end else if**
14: **end for**
(* Apply regular mutation and recombination to the entire population *)
15: Mutate P(t)
16: Recombine P(t)
17: Return P(t)
18: **end**

Fig. 3. The basic structure of the INFORMED_OP algorithm.

III. EXPERIMENTS

Simulations were carried out to track the proposed algorithm's performance in terms of solution accuracy and population diversity. For comparison purposes results reported by Ursem [10] has been used.

A. Experiment Setup

The test suite for our simulations comprises of the following popular benchmark function optimisation (minimization) problems:

a) Ackley's Path function ($F1$)
b) Griewangk's function ($F2$)
c) Rastrigin's function ($F3$)
d) Rosenbrock's valley ($F4$)

Descriptions of the above functions are as given below.

$$\text{Ackley } F1(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi \cdot x_i)\right) \text{ where } -30 \leq x_i \leq 30 \quad (1)$$

$$\text{Griewank } F2(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1 \text{ where } -600 \leq x_i \leq 600 \quad (2)$$

$$\text{Rastrigin } F3(x) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \text{ where } -5.12 \leq x_i \leq 5.12 \quad (3)$$

$$\text{Rosenbrock } F4(x) = \sum_{i=1}^{n-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$$

where $-100 \le x_i \le 100$ (4)

All the functions were considered in 20, 50 and 100 dimensions. Simulations were carried out with the following parameter settings for the COMMUNITY_GA algorithm: real valued encoding, binary tournament selection, N (*population size*) =200, Gaussian mutation operator with p_m (*mutation probability*) =0.01 and a hybrid recombination operator with p_r (*recombination probability*) =0.9. Reported results were averaged over 30 independent runs, maximum number of generations in each run being only 500, as against 1000 generations in [10] for the same set of test cases for the 20 dimensional scenarios. The comparison algorithms use 50 times the dimensionality of the test problems as the terminating generation number in general, while the COMMUNITY_GA uses 500, 1000 and 2000 generations for the 20, 50 and 100 dimensional problem variants respectively.

The observed results have been compared with the following techniques as reported by Ursem in [10]: (i) SEA (Standard EA), (ii) SOCEA (Self-organized criticality EA), (iii) CEA (The Cellular EA), and (iv) DGEA (Diversity guided EA). The parameter settings for these algorithms are as follows: real valued encoding, population size of N (*population size*) =400, variants of the Gaussian mutation operator with p_m (*mutation probability*) =0.75 and arithmetic recombination operator with p_r (*recombination probability*) =0.9. CEA uses a 20x20 grid with wrapped edges, where the grid size corresponds to the population size of 400 individuals as used in the other algorithms. Further details are given in [10]. All the simulation processes were executed using a Pentium® 4, 2.4GHz CPU processor.

B. Results and Discussions

Simulation results obtained with COMMUNITY_GA in comparison to SEA, SOCEA, CEA, and DGEA are presented in Table I, II and III. These simulation results ascertain COMMUNITY_GA's superior performance as regards solution precision in all four test cases. This may be attributed to COMMUNITY_GA's ability to strike a better balance between exploration and exploitation. Table IV depicts the runtimes for the tested algorithms for the 100 dimensional scenarios of the four test cases. Considering the structures of the algorithms, a tradeoff between solution accuracy and computational time can be expected for COMMUNITY_GA. On the other hand, DGEA, which is designed to skip certain genetic operations depending on the level of population diversity, would be a clear winner in terms of computation time if all the algorithms are executed for the same number of generations in each run. For the reported results as shown in Table I, II, III and Table IV, the 100 dimensional scenarios of the test problems used 5000 generations for each of the compared algorithm, namely, SEA, SOCEA, CEA and DGEA [10]. On the other hand, COMMUNITY_GA used only 2000 generations to reach the reported results. Hence, for comparison purposes it is only fair to consider the computation time required by the different methods to reach comparable results. As can be observed from Table IV, despite its relatively complex algorithmic structure, COMMUNITY_GA requires less computation time to reach better or comparable solution accuracy. We have also extended the simulation runs beyond the fixed number of generations and to the *stagnation point*. Here, *stagnation point* is defined by the generation with 500 successive generations of no fitness improvement preceding it. Table V summarizes the results for DGEA and COMMUNITY_GA with fixed run and at stagnation. Both DGEA and COMMUNITY_GA show some improvement

TABLE I
AVERAGE FITNESS COMPARISON FOR SEA, SOCEA, THE CEA, DGEA AND COMMUNITY_GA*. DIMENSION OF EACH FUNCTION IN THIS CASE IS 20.

	$F1$	$F2$	$F3$	$F4$
SEA	2.494	1.171	11.12	8292.32
SOCEA	0.633	0.930	2.875	406.490
CEA	0.239	0.642	1.250	149.056
DGEA	8.05E-4	7.02E-4	2.21E-5	96.007
C_GA*	1.08E-6	4.6E-10	1.21E-5	1.0E-10

TABLE II
AVERAGE FITNESS COMPARISON FOR SEA, SOCEA, THE CEA, DGEA AND COMMUNITY_GA*. DIMENSION OF EACH FUNCTION IN THIS CASE IS 50.

	$F1$	$F2$	$F3$	$F4$
SEA	2.870	1.616	44.674	41425.674
SOCEA	1.525	1.147	22.460	4783.246
CEA	0.651	1.032	14.224	1160.078
DGEA	4.61E-3	4.40E-3	0.01664	315.395
C_GA*	1.07E-5	3.31E-6	7.5E-5	0.01E-3

TABLE III
AVERAGE FITNESS COMPARISON FOR SEA, SOCEA, THE CEA, DGEA AND COMMUNITY_GA*. DIMENSION OF EACH FUNCTION IN THIS CASE IS 100.

	$F1$	$F2$	$F3$	$F4$
SEA	2.893	2.250	106.212	91251.300
SOCEA	2.220	1.629	86.364	30427.63
CEA	1.140	1.179	58.380	6053.870
DGEA	0.01329	0.01238	0.15665	1161.550
C_GA*	1.08E-4	6.71E-4	9.99E-5	1.032

TABLE IV
AVERAGE RUNTIME IN MILLISECONDS FOR SEA, SOCEA, THE CEA, DGEA AND COMMUNITY_GA* FOR THE 100 DIMENSIONAL SCENARIOS. (AVERAGE OF 100 RUNS WITH 2000 GENERATIONS FOR COMMUNITY_GA* AND 5000 GENERATIONS FOR OTHER ALGORITHMS.)

	$F1$	$F2$	$F3$	$F4$
SEA	1128405	1171301	1124925	1087615

SOCEA	1528864	1562931	1513691	1496164
CEA	2951963	3656724	2897793	2183283
DGEA	864316	969683	819691	883811
C_GA*	418489	521800	491411	510266

over the results obtained with fixed number of generations in most cases. COMMUNITY_GA still outperforms DGEA. Also, COMMUNITY_GA has arrived at these superior results in much fewer generations. However, no significant improvement was observed in case of all three different dimensional cases of the Rosenbrock function, in case of COMMUNITY_GA.

TABLE V
AVERAGE FITNESS COMPARISON FOR DGEA AND COMMUNITY_GA*. DIMENSION OF EACH FUNCTION IN THIS CASE IS 100. BOTH ALGORITHMS WERE EXECUTED TILL STAGNATION.

Test Function	DGEA (Fixed Run)	DGEA (Stagnation)	C_GA* (Fixed Run)	C_GA* (Stagnation)
$F1_{20\ D}$	8.05E-4	3.36e-5	1.08E-6	1.01E-8
$F1_{50\ D}$	4.61E-3	2.52E-4	1.07E-5	0.19E-6
$F1_{100\ D}$	0.01329	9.80E-4	1.08E-4	0.09E-4
$F2_{20\ D}$	7.02E-4	7.88E-8	4.6E-10	0.7E-10
$F2_{50\ D}$	4.40E-3	1.19E-3	3.31E-6	2.11E-6
$F2_{100\ D}$	0.01238	3.24E-3	6.71E-4	1.01E-5
$F3_{20\ D}$	2.21E-5	3.37E-8	1.21E-5	0.02E-7
$F3_{50\ D}$	0.01664	1.97E-6	7.5E-5	0.15E-6
$F3_{100\ D}$	0.15665	6.56E-5	9.99E-5	0.11E-5
$F4_{20\ D}$	96.007	8.127	1.0E-10	1.0E-10
$F4_{50\ D}$	315.395	59.789	0.01E-3	0.01E-3
$F4_{100\ D}$	1161.550	880.324	1.032	1.000

In the second phase of our experiments, we have investigated COMMUNITY_GA's performance in terms of maintaining constructive diversity. There are various measures of diversity available. The "distance-to-average-point" measure used in [10] is relatively robust with respect to population size, dimensionality of problem and the search range of each variable. Hence, we have used this measure of diversity in our investigation. The "distance-to-average-point" measure for N dimensional numerical problems can be described as below [10].

$$diversity(P) = \frac{1}{|L| \cdot |P|} \cdot \sum_{i=1}^{|P|} \sqrt{\sum_{j=1}^{N} \left(s_{ij} - \bar{s}_j\right)^2} \qquad (5)$$

where, $|L|$ is the length of the diagonal or range in the search space $S \subseteq \Re^N$, P is the population, $|P|$ is the population size, N is the dimensionality of the problem, s_{ij} is the j'th value of the i'th individual, and $\bar{s}_j$ is the j'th value of the average point $\bar{s}$. It is assumed that each search variable s_k is in a finite range, $s_{k\_\min} \le s_k \le s_{k\_\max}$. Table VI depicts the average diversity for the four test problems with COMMUNITY_GA simulation runs.

The average fitness reported in Table VI, averages the value of the diversity measure in equation (5) calculated at each generation where there has been an improvement in average fitness over 500, 1000 and 2000 generations for the 20, 50 and 100 dimensional cases respectively. Final values were averaged over 100 runs. To eliminate the noise in the initial generations of a run, diversity calculation does not start until the generation, since which a relatively steady improvement in fitness has been observed. Table VI shows

TABLE VI
AVERAGE POPULATION DIVERSITY COMPARISON FOR COMMUNITY_GA. AN AVERAGE OF 100 RUNS HAVE BEEN REPORTED IN EACH CASE.

Test Function	COMMUNITY_GA (Fixed Run)		
	20 D Test Case	50 D Test Case	100 D Test Case
$F1$	0.000350	0.000811	0.001001
$F2$	0.000290	0.000725	0.001099
$F3$	0.002000	0.002550	0.003015
$F4$	0.000718	0.001025	0.001989

that the COMMUNITY_GA do not necessarily maintain very high average population diversity. However, EA's requirement is not to maintain very high average population diversity but to maintain an optimal level of population diversity. The high solution accuracy obtained by COMMUNITY_GA proves that the algorithm is successful in this respect.

IV. CONCLUSIONS

This paper presents a counter-niching based novel genetic algorithm framework to introduce constructive diversity with the primary aim to curb premature convergence. The COMMUNITY_GA algorithm [8] presented in this paper basically incorporates two key processes. Firstly, the population's spatial information is obtained with a computationally relatively inexpensive GRID_NICHING algorithm. Secondly, the information is used to identify potential local convergence and cluster formations. Then diversity is intelligently introduced with informed genetic operations, aiming at the following two objectives: (a) Introduction of promising samples from unexplored regions replacing redundant less fit members of over-populated

clusters. (b) While local entrapment is discouraged, representative members of the clusters are still preserved to encourage exploitation. While the current focus of the research was to introduce and maintain population diversity to avoid local entrapment, this community-based algorithm can also be adapted to serve as a low cost alternative for niching GA, to identify multiple solutions in multimodal problems as well as to suit the diversity requirements of a dynamic environment. Future research will investigate COMMUNITY_GA's suitability in more complex problem domains such as problems involving uncertain environments. Further investigation is also required to establish the exact nature of correlation between introduction of diversity and improvement in average fitness.

REFERENCES

[1] C. Ryan, "Reducing Premature Convergence in Evolutionary In Evolutionary Algorithms", PhD. Thesis, University College, Cork, 1996.

[2] D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization", *Genetic Algorithms and their Applications* (ICGA'87), Grefenstette, J.J. (ed.), Lawrence Erlbaum Associates, Publishers, 1987, PP. 41–49.

[3] G. W. Greenwood, G. B. Fogel and M. Ciobanu, "Emphasizing Extinction in Evolutionary Programming", *Proceedings of the Congress of Evolutionary Computation*, Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A. (eds.), Vol. 1., 1999, pp. 666–671.

[4] H. B. Amor, A. Rettinger, "Intelligent exploration for genetic algorithms: using self-organizing maps in evolutionary computation", *Proceedings of GECCO 2005*, pp. 1531-1538.

[5] H. G. Cobb and J. F. Grefenstette, "Genetic Algorithms for Tracking Changing Environments", *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993, pp. 523–530.

[6] J. Hu, E.Goodman, K.Seo, Z. Fan, R. Rosenberg, "The Hierarchical Fair Competition (HFC) Framework for Sustainable Evolutionary Algorithms". *Evolutionary Computation*, 13(1), 2005 (in press).

[7] K. A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", PhD thesis, University of Michigan, Ann Arbor, MI, Dissertation Abstracts International 36(10), 5140B, University Microfilms Number 76-9381, 1975.

[8] M. Bhattacharya, "An Informed Operator Approach to Tackle Diversity Constraints in Evolutionary Search", *Proceedings of The International Conference on Information Technology, ITCC 2004*, Vol. 2, IEEE Computer Society Press, ISBN 0-7695-2108-8,pp. 326-330.

[9] N. N. Schraudolph and R. K. Belew, "Dynamic parameter encoding for genetic algorithms", *Machine Learning*, 9(1), 1992, pp. 9—21.

[10] R. K. Ursem, "Diversity-Guided Evolutionary Algorithms", *Proceedings of Parallel Problem Solving from Nature VII (PPSN-2002)*, 2002, pp. 462-471.

[11] R. K. Ursem, "Multinational Evolutionary Algorithms", *Proceedings of the Congress of Evolutionary Computation (CEC-99)*, Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A. (eds.), Vol. 3. , 1999, pp. 1633–1640.

[12] R. Thomsen and P. Rickers, "Introducing Spatial Agent-Based Models and Self-Organised Criticality to Evolutionary Algorithms" Master's thesis, University of Aarhus, Denmark, 2000.

[13] S. Mahfoud, "Crowding and preselection revisited.", Technical Report 92004, Illinois Genetic Algorithms Laboratory (IlliGAL),, 1992.

[14] T. B¨ack, D. B. Fogel, Z. Michalewicz, and others, (eds.), *Handbook on Evolutionary Computation*, IOP Publishing Ltd and Oxford University Press, 1997.

[15] T. Krink, R. Thomsen and P. Rickers, "Applying Self-Organised Criticality to Evolutionary Algorithms", *Parallel Problem Solving from Nature – PPSN VI*, Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., and Schwefel, H.P. (eds.), Vol. 1. , 2000, pp. 375–384.