The LifeCycle model: Combining Particle Swarm Optimisation, Genetic Algorithms and HillClimbers

Thiemo Krink and Morten Løvbjerg

EVALife Group, Department of Computer Science
Ny Munkegade, Bldg. 540, University of Aarhus
DK-8000 Aarhus C, Denmark
{krink, lovbjerg}@daimi.au.dk
http://www.evalife.dk

Abstract. Adaptive search heuristics are known to be valuable in approximating solutions to hard search problems. However, these techniques are problem dependent. Inspired by the idea of life cycle stages found in nature, we introduce a hybrid approach called the LifeCycle model that simultaneously applies genetic algorithms (GAs), particle swarm optimisation (PSOs), and stochastic hill climbing to create a generally well-performing search heuristics. In the LifeCycle model, we consider candidate solutions and their fitness as individuals, which, based on their recent search progress, can decide to become either a GA individual, a particle of a PSO, or a single stochastic hill climber. First results from a comparison of our new approach with the single search algorithms indicate a generally good performance in numerical optimization.

1 Introduction

In biology, the term life cycle refers to the various phases an individual passes through from birth to maturity and reproduction [1]. This process often leads to drastic transformations of the individual with stage specific adaptations to a particular environment. This phenomenon is particularly amazing considering that the genome remains the same within each cell and life stage, whereas the morphology and behaviour of the phenotype can change drastically in accordance to the requirements of the life stage niche. Some life cycle changes in nature are one-time events such as sexual maturity. Other changes are re-occurring, such as mating seasons. These stages are genetically determined and the individuals have little or no influence on the change of the life cycle stage. The transition between life cycle changes are often triggered by environmental factors. Environmental changes often determine transitions from one life cycle stage to another. Some animals are able to sense and predict these changes and can actively decide to alter their life cycle stage. A particularly interesting animal that has this capability is the microscopically small *Symbion pandora*, which lives as a symbiont on its much larger host - the Norway lobster. The host provides food,

substrate, and transportation for the symbionts which only inhabit the mouth parts of the host. This rich and diverse environment poses a great challenge to the symbiont because the individual need to evacuate and recolonize the lobster with each lobster moulting. The life stages of *Symbion pandora* include feeding and non-feeding, sedentary and free swimming, as well as sexual and asexual reproduction stages [2].

The ability of an individual to actively decide about its kind of life form in response to its success in its current environment inspired us to the study presented in this paper. The idea behind our LifeCycle model is to create a self-adaptive search heuristic in which each individual (containing the candidate solution) can decide whether it would prefer to belong to a population of a genetic algorithm (GA), a particle swarm optimization (PSO), or become a solitary stochastic hill climber (HC). The decision of the individual depends on its success in searching the fitness landscape. Our motivation for this hybrid approach was that each of these search techniques on its own has its specific problem dependent strengths and weaknesses.

GAs, for instance, are widely applicable, and particularly powerful when domain knowledge can be incorporated in the operator design (see e.g. [3]). However, particle swarm optimisation (PSO) [4] can achieve clearly superior results in many instances of numerical optimization, but there is no general superiority compared to GAs (e.g. [5–9]). Hill climbers, in contrast, are good for local search with a high probability of finding the closest optimum. However, for multimodal functions, their performance is highly dependent on their starting position and hill-climbing techniques often convergence prematurely at local optima. Their main weakness compared to population based approaches, such as GAs and PSOs, is that candidate solutions neither compete nor cooperate [10].

The goal of our LifeCycle model is to make a self-adaptive approach towards a problem invariant search technique that can further take advantage of the changing search requirements during the optimization, such as initial exploration and local fine-tuning towards the end of the run.

2 The LifeCycle Model

The LifeCycle model consists of individuals starting out as PSO particles, which can turn into GA individuals, then hill climbers, then back to particles and so on. The structure of the LifeCycle model is illustrated in fig. 1. In all these heuristics, we use one fitness evaluation per individual per iteration. A LifeCycle individual switches its stage when it has made no fitness improvement for more than 50 iterations.

2.1 The PSO model

The PSO model used in the LifeCycle model is similar to the traditional PSO model described in [4]. The model consists of a number of particles moving around in the search space, where the position of each particle represents a candidate solution to a numerical problem. Each particle has a position vector $\boldsymbol{x}_i$,

```
program LifeCycle_Model
begin
    initialise
    while (not terminate-condition) do
        begin
            for (all individuals)
                evaluate fitness
                switch LifeCycle stage if no recent improvement
            for (PSO particles)
                calculate new velocity vectors
                move
            for (GA individuals)
                select new population
                recombine population
                mutate population
            for (HillClimbers)
                find possible new neighbouring solution
                evaluate fitness for the new solution
                shift to new solution with probability p
        end
end
```

Fig. 1. Structure of the LifeCycle model.

a velocity vector $\boldsymbol{v}_i$ and the position of the best candidate solution encountered by the particle $\boldsymbol{p}_i$. The PSO also stores the overall best found point $\boldsymbol{p}_g$. The memorized positions are used to attract particles to search space areas with known good solutions.

In each iteration the velocity of each particle is updated in the following way

$$\boldsymbol{v}_i = \chi(w\boldsymbol{v}_i + \boldsymbol{\varphi}_{1i}(\boldsymbol{p}_i - \boldsymbol{x}_i) + \boldsymbol{\varphi}_{2i}(\boldsymbol{p}_g - \boldsymbol{x}_i))$$

where χ is known as the *constriction coefficient* described in [11] and w is the *inertia weight* described in [6]. $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ are random values, which are different for each particle and for each dimension. The velocity $\boldsymbol{v}_i$ of each particle is limited by an upper threshold v_{max}. The position of each particle is updated in each iteration by adding the velocity vector to the position vector, such that, $\boldsymbol{x}_i = \boldsymbol{x}_i + \boldsymbol{v}_i$. The particles have no neighbourhood restrictions, meaning that each particle can affect all other particles. This neighbourhood is of type *star* (fully connected network), which has been shown to be a good topology ([5]).

2.2 The GA model

A classic genetic algorithm consists of a population of individuals refining their candidate solutions through interaction and adaptation. Each individual represents a candidate solution to the given problem. After initialisation the GA enters a loop, in which the population is evaluated, a new population is selected and this new population is altered (p. 151 [10]). The LifeCycle GA model uses binary tournament selection (p. 61 [12]) to generate a new population and elitism

to ensure the survival of the individual with the best fitness. The LifeCycle GA model alters the population by crossover and mutation. The crossover operator used in the LifeCycle GA model is the so-called arithmetic crossover. This operator replaces two parent individuals selected for crossover with two child individuals as follows:

$$\boldsymbol{x}_{child1} = w * \boldsymbol{x}_{parent1} + (1 - w) * \boldsymbol{x}_{parent2}$$
$$\boldsymbol{x}_{child2} = w * \boldsymbol{x}_{parent2} + (1 - w) * \boldsymbol{x}_{parent1}$$

where w is a random value between zero and one. The crossover probability PC determines the probability of an individual to be selected for crossover. For each dimension the probability of mutation PM determines whether or not to mutate. The mutation scheme used in the LifeCycle GA model is the non-uniform mutation described on page 103 in [12]. Entry j of an individual is mutated according to:

$$\Delta x_j = \begin{cases} +(Max - x_j)(1 - r^{(1-t/T)^b}) \\ -(x_j - Min)(1 - r^{(1-t/T)^b}) \end{cases}$$

with a 50% chance each. Max is the search space maximum, Min is the minimum, r is a random number in [0..1], t is the current iteration, T is the total number of iterations and b is a parameter determining the degree of iteration number dependency. Hence, the effect of mutation decreases over the course of the iterations with this scheme.

2.3 The HillClimber

HillClimbers are individuals that refine their candidate solution independently of other individuals by examining the local neighbourhood. The hill-climbing method used in the LifeCycle model is a stochastic hill-climber as described in [10] pp. 118-120. Each HillClimber consists of a solution $\boldsymbol{x}_c$. For each iteration, a new candidate solution $\boldsymbol{x}_n$ is selected within the neighbourhood of $\boldsymbol{x}_c$. $\boldsymbol{x}_c$ is replaced by $\boldsymbol{x}_n$ with probability p given by

$$p = 1/(1 + exp(\frac{eval(\boldsymbol{x}_n) - eval(\boldsymbol{x}_c)}{T})) \qquad (minimisation)$$

where T is a parameter determining the influence of the relative merit (difference in fitness performance). In this paper, we experimented with a constant T=10, which is different from simulated annealing. The better the fitness of the neighbouring point, the higher the chance of replacement.

3 Experimental settings

In our experiments, we compared the performance of the standard PSO, the standard GA, HillClimbers and the LifeCycle model on five numerical benchmark problems (all minimisation) (see table 1). The first two are unimodal while the latter three are multimodal with many local minima.

The initial population of GAs and PSOs is usually uniformly distributed over the entire search space. According to Angeline [8], this can give false indications

Table 1. Test functions

Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$
Rosenbrock	$f_2(x) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n} cos(\frac{x_i - 100}{\sqrt{i}}) + 1$
Rastrigin	$f_4(x) = \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) + 10)$
Ackley	$f_5(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)}$

Table 2. Search space and Initialisation ranges

Function	**Search space**	**Initialisation range**
f_1 *Sphere*	$-100 \leq x_i \leq 100$	$50 \leq x_i \leq 100$
f_2 *Rosenbrock*	$-100 \leq x_i \leq 100$	$15 \leq x_i \leq 30$
f_3 *Griewank*	$-600 \leq x_i \leq 600$	$300 \leq x_i \leq 600$
f_4 *Rastrigin*	$-10 \leq x_i \leq 10$	$2.56 \leq x_i \leq 5.12$
f_5 *Ackley*	$-32.768 \leq x_i \leq 32.768$	$16.384 \leq x_i \leq 32.768$

of relative performance, especially if the search space is symmetric around the origin where many test functions have their global optimum such as in the classic benchmarks that we used in this paper. In order to prevent this, we used the asymmetric initialisation method by Angeline [8] for all experiments. Search space and initialisation ranges for the experiments are shown in table 2.

In all experiments the population of the LifeCycle model was fixed at 150 individuals. These were all initialised as PSO particles.

Table 3. Crossover and mutation probability used in the GA.

Function	f_1 *Sphere*	f_2 *Rosenbrock*	f_3 *Griewank*	f_4 *Rastrigin*	f_5 *Ackley*
Crossover prob.	0.60	0.50	0.50	0.20	0.50
Mutation prob.	0.30	0.30	0.40	0.02	0.30

3.1 Settings for the PSO

In the PSO model the upper limits for $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ were set to 2.0. The inertia weight w was linearly decreased from 0.7 to 0.4 and the constriction coefficient χ was set to 1. The maximum velocity v_{max} of each particle was set to half the length of the search space for each dimension (e.g. $v_{max} = 100$ for f_1 and f_2). Previous research by Shi [6] regarding scalability of the standard PSO showed that the performance of the standard PSO is not sensitive to the population size. This is also our experience from earlier work ([9, 13]). In all PSO experiments, we used a fixed population size of 20 particles.

3.2 Settings for the GA

For the GA, we used the crossover and mutation probabilities shown in table 3. Neither selection nor crossover were performed for population sizes smaller than

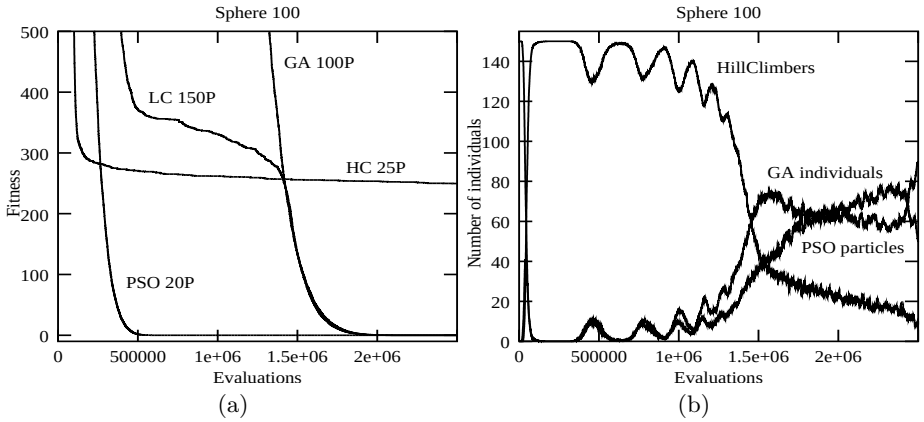

Fig. 2. Sphere function: (a) Performance of the standard GA with population size 100, the standard PSO with population size 20, 25 HillClimbers and the LifeCycle model with population size 150. (b) Composition of LifeCycle individuals.

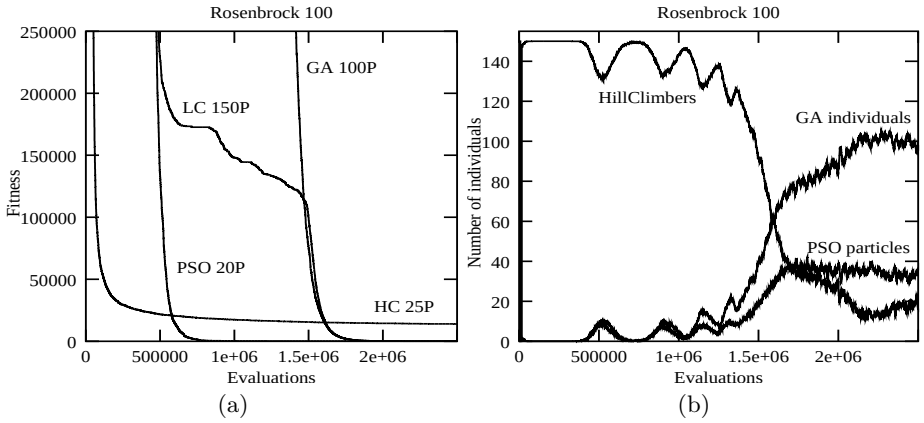

Fig. 3. Rosenbrock function: (a) Performance. (b) Composition of LifeCycle individuals

three. Moreover, we set the mutation loop dependency b to 5 and used a fixed population size of 100 individuals.

3.3 Settings for the HillClimbers

In our HillClimber experiments, we used a constant temperature parameter T = 10. The size of the search neighbourhood linearly decreased from 1 to 0.001 percent of the search space for each dimension to allow fine-tuning towards the end of the run. In all HillClimbing experiments we used 25 HillClimbers simultaneously and indepedently from each other.

4 Experimental results

The test functions that we used in our experiments were all 100 dimensional. All experiments were running for 2.500.000 evaluations. Table 4 and fig. 2 to 6 show

the fitness vs. the number of evaluations regarding the five benchmark problems. The graphs illustrate the mean values of 50 repetitions for each experiment.

Table 4. Experimental results

Function	PSO	GA	HC	LC
f_1 *Sphere*	$1.7401E-106$	$7.3762E-3$	249.5553	$4.9468E-8$
f_2 *Rosenbrock*	62.4291	99.9857	13908.6493	94.7642
f_3 *Griewank*	$1.0003E-2$	175.1830	269.9571	$5.0235E-3$
f_4 *Rastrigin*	154.4770	0.5387	725.8330	83.4788
f_5 *Ackley*	19.8617	$1.3563E-2$	21.2318	$2.0350E-11$

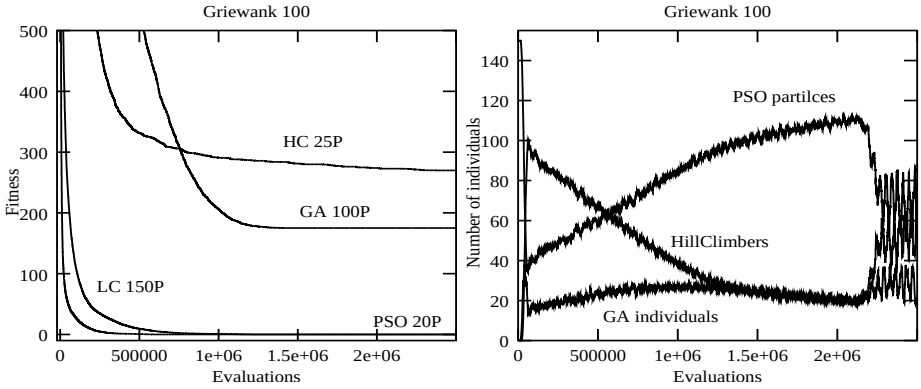

Fig. 4. Griewank function: (a) Performance. (b) Composition of LifeCycle individuals.

4.1 Performance

Figure 2(a) to 6(a) show the performance of the LifeCycle (LC) model compared to the standard PSO (PSO), the standard GA (GA) and the HillClimbing (HC) algorithm described in section 2.

For the Sphere (fig. 2(a)) and Rosenbrock (fig. 3(a)) function the standard PSO turned out to have the best performance. In both cases HillClimbers yielded the fastest fitness improvements, but failed to find an exact solution. Both the LifeCycle and GA models converged slower than the PSO. The LifeCycle model converged faster and to a better value than the standard GA. Regarding the Griewank function (fig. 4(a)) the standard PSO outperformed the standard GA and the HillClimbing algorithm by far. Here, the search improvements of the LifeCycle model were a bit slower but eventually better compared to the standard PSO. For the Rastrigin function (fig. 5(a)) the standard GA found the best solution, but required more time than the LifeCycle model and the PSO. Here, the PSO achieved the fastest initial search improvements, but was eventually

outperformed by the LifeCycle model towards the end of the run. HillClimbers turned out to be ineffective. Finally, regarding the Ackley function (fig. 6(a)), we found that the PSO and the HillClimbing algorithms were clearly outperformed by the standard GA and the LifeCycle model, where the latter outperformed all other heuristics significantly.

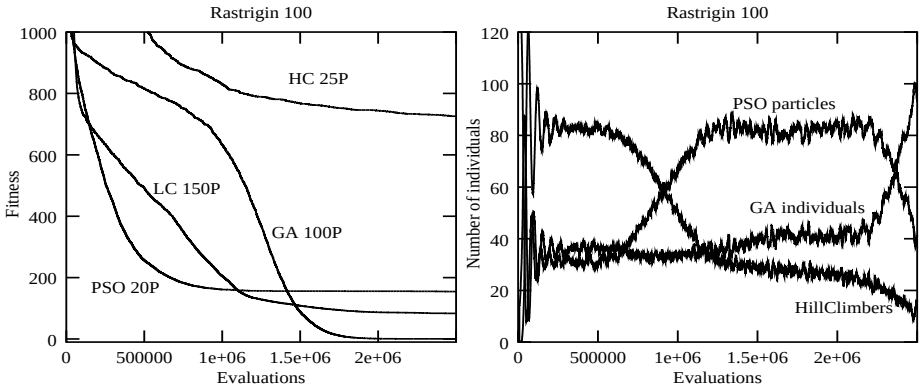

Fig. 5. Rastrigin function: (a) Performance. (b) Composition of LifeCycle individuals.

4.2 Composition of the LifeCycle individuals

Figures 2(b) to 6(b) show the frequency of the life cycle stages over time. All LifeCycle individuals were set to start out as PSO particles. For the Sphere (fig. 2(b)) and Rosenbrock (fig. 3(b)) functions all individuals prefered to become HillClimbers in the beginning of the run. After approximately 50% of the evaluations, a majority of the individuals turned into GA individuals. Moreover, the frequency of the PSO individuals increased towards the end of the run. Regarding the Griewank (fig. 4(b)) and Rastrigin (fig. 5(b)) functions, we found

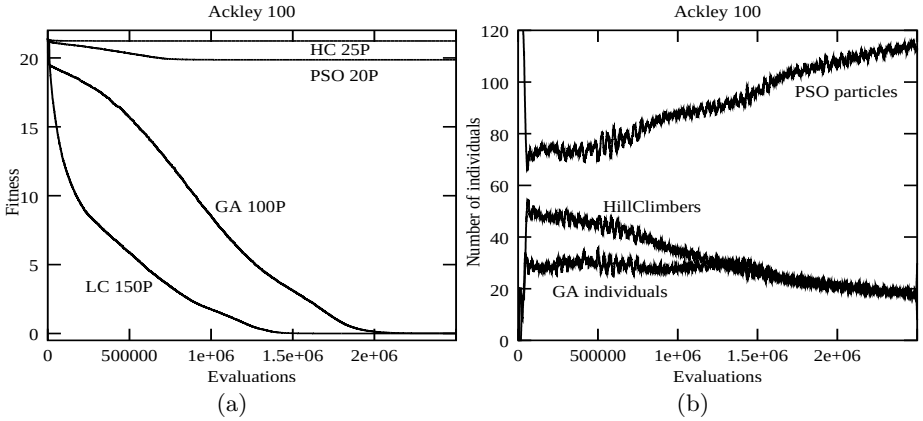

Fig. 6. Ackley function: (a) Performance. (b) Composition of LifeCycle individuals.

a similar self-adaptation pattern with a high frequency of HillClimbers in the beginning of the run. However, in contrast to the other two test functions, most HillClimbers turned into PSO particles during the run. At the end of the run, we found that GA individuals increasingly took the place of the PSO particles. In case of the Ackley function (fig. 6(b)), we found that the LifeCycle population consisted of a majority of PSO particles, which further increased over time.

5 Discussion and future work

Our approach of combining three standard adaptive optimisation algorithms into one self-adaptive hybrid approach turned out to be an improvement over the individual algorithms. Our results show that the LifeCycle heuristic has a generally good performance on all benchmark problems that we used in this study in contrast to the single adaptive algorithms, which have a highly problem dependent performance. For the Sphere (fig. 2) and Rosenbrock (fig. 3) functions, our results suggest that the best strategy is to start with a large number of HillClimbers, which are later turned into PSO particles. On first sight this seems to be counter-intuitive, because of the fine-tuning abilities of the HillClimbers. However, the Stochastic HillClimber has the problem that the acceptance probability of a new candidate solution approaches 0.5 in case of very small fitness differences between neighboured candidate solutions. Furthermore, the neighbourhood step size in continuous optimization has a strong impact on the quality of the results. A too low step size slows down the hill climber unnecessarily whereas a too large step size can result in missing the global optimum. Judging from our experimental results the LifeCycle model helps the standard models to achieve superior results for multimodal testfunctions. This can be seen in figures 4 to 6 for the Griewank, Rastrigin and Ackley functions. For all these three functions the LifeCycle model works best with many PSO particles. However, the pure PSO model is clearly outperformed, especially for the Ackley function (fig. 6(a)). In the present study we focused on the most simple and classic versions of GAs, PSOs, and HillClimbers. An interesting future extension would be to use the life cycle heuristic based on more advanced heuristics, such as spatially extended PSOs [13], SOC EAs [14], and simulated annealing [12]. Another idea would be to introduce tabu search as a fourth search heuristic to the LifeCycle model. Other future work could be to investigate other transition schemes. Furthermore, one could try to exchange one PSO particle with several GA individuals and vice versa, since GAs work better with much larger population sizes than PSOs.

6 Acknowledgments

The authors would like to thank EVALife colleagues Peter Funch and Matthias Obst of the Dept. of Zoology, Univ. of Aarhus, for the biological inspiration and Rasmus K. Ursem, EVALife, Dept. of Computer Science, Univ. of Aarhus, for reviewing the manuscript. This research was supported by the Danish Natural Science Research Council.

References

1. Lawrence, E. (ed.): Henderson's Dictionary of Biological Terms. Longman (1996).
2. Funch, P., Kristensen, R. M.: Cyclicophora is a new phylum with affinities to entoprocta and ectoprocta. In: Nature 378. (1995) 711–714.
3. Filipič, B., Štrancar, J.: Genetic optimization of the EPR spectral parameters: Algorithm implementation and preliminary results. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., Schwefel, H.-P. (eds.): Parallel Problem Solving from Nature – PPSN VI. Springer, Berlin (2000) 693–701.
4. Kennedy, J., Eberhart, R. C.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Vol. 4. (1995) 1942–1948.
5. Kennedy, J.: Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In: Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A. (eds.): Proceedings of the Congress of Evolutionary Computation, Vol. 3. IEEE Press (1999) 1931–1938.
6. Shi, Y., Eberhart, R. C.: Parameter selection in particle swarm optimization. In: Porto, V. W., Saravanan, N., Waagen, D., Eiben, A. E. (eds.): Evolutionary Programming VII. Lecture Notes in Computer Science 1447. Springer, Berlin (1998) 591–600.
7. Suganthan, P.: Particle swarm optimiser with neighbourhood operator. In: Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A. (eds.): Proceedings of the Congress of Evolutionary Computation, Vol. 3. IEEE Press (1999) 1958–1962.
8. Angeline, P. J.: Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In: Porto, V. W., Saravanan, N., Waagen, D., Eiben, A. E. (eds.): Evolutionary Programming VII. Lecture Notes in Computer Science, Vol. 1447. Springer, Berlin (1998) 601–610.
9. Løvbjerg, M., Rasmussen, T. K., Krink, T.: Hybrid particle swarm optimiser with breeding and subpopulations. In: Spector, L., Goodman, E. D., Wu, A., Langdon, W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., Burke, E. (eds.): Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001). Morgan Kaufmann (2001) 469–476.
10. Michalewicz, Z., Fogel, D. B.: How to Solve It: Modern Heuristics. Springer, Berlin (2000).
11. Clerc, M.: The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In: Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A. (eds.): Proceedings of the Congress of Evolutionary Computation, Vol. 3. IEEE Press (1999) 1951–1957.
12. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin (1992).
13. Krink, T., Vesterstrøm, J. S., Riget, J.: Particle Swarm Optimisation with Spatial Particle Extension. In: Fogel, D. B., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M. (eds.): Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002). (2002).
14. Krink, T., Thomsen, R., Rickers, P.: Applying Self-Organised Criticality to Evolutionary Algorithms. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., Schwefel, H.-P. (eds.): Parallel Problem Solving from Nature – PPSN VI. Springer, Berlin (2000) 375–384.