Feeding the Fish – Weight Update Strategies for the Fish School Search Algorithm

Andreas Janecek and Ying Tan

Key Laboratory of Machine Perception (MOE), Peking University
Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, China
andreas.janecek@univie.ac.at, ytan@pku.edu.cn

Abstract. Choosing optimal parameter settings and update strategies is a key issue for almost all population based optimization algorithms based on swarm intelligence. For state-of-the-art optimization algorithms the optimal parameter settings and update strategies for different problem sizes are well known.

In this paper we investigate and compare different newly developed weight update strategies for the recently developed Fish School Search (FSS) algorithm. For this algorithm the optimal update strategies have not been investigated so far. We introduce a new dilation multiplier as well as different weight update steps where fish in poor regions loose weight more quickly than fish in regions with a lot of food. Moreover, we show how a simple non-linear decrease of the individual and volitive step parameters is able to significantly speed up the convergence of FSS.

1 Introduction

The Fish School Search (FSS) algorithm [1, 2, 3] is a recently developed swarm intelligence algorithm based on the social behavior of schools of fish. By living in swarms, the fish improve survivability of the whole group due to mutual protection against enemies. Moreover, the fish perform collective tasks in order to achieve synergy (e.g. finding locations with lots of food). Comparable to real fish that swim in the aquarium in order to find food, the artificial fish search (*swim*) the search space (*aquarium*) for the best candidate solutions (locations with most *food*). The location of each fish represents a possible solution to the problem – comparable to locations of particles in Particle Swarm Optimization (PSO, [4]). The individual success of a fish is measured by its weight – consequently, promising areas can be inferred from regions where bigger ensembles of fish are located. As for other heuristic search algorithms we consider the problem of finding a "good" (ideally the global) solution of an optimization problem with bound constraints in the form: $\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})$, where $f : \mathbb{R}^N \rightarrow \mathbb{R}$ is a non-linear objective function and $\boldsymbol{x}$ is the feasible region. Since we do not assume that f is convex, f may possess many local minima. Solving such tasks for high dimensional real world problems may be expensive in terms of runtime if exact algorithms were used. Various nature inspired algorithms have shown to be able

Y. Tan et al. (Eds.): ICSI 2011, Part II, LNCS 6729, pp. 553–562, 2011.

to preform well with these difficulties. Even though if these algorithms are only meta-heuristics, i.e. there is no proof that they reach the global optimum of the solution, these techniques often achieve a reasonably good solution for the given task at hand in a reasonable amount of time.

Related work. The FSS algorithm was introduced to the scientific community in 2008 [1]. This paper was extended to a book chapter [2] where FSS has been evaluated and compared to different variants of PSO. Results indicate that FSS is able to achieve better results as PSO on several benchmark functions, especially on multimodal functions with several local minima. In another study [3] the same authors analyzed the importance of the swimming operators of FSS and showed that all operators have strong influences on the results. Although for some benchmarks the individual operator alone sometimes produced better results than all operators together, the results using only the individual operator are highly sensitive to the initial and also final values of $step_{ind}$ and $step_{vol}$. Moreover, it was shown that a rather large initial value for $step_{ind}$ ($step_{ind\,initial} = 10\%$) generally achieved the best results. In a very recent study FSS has been used successfully to initialize the factors of the non-negative matrix factorization (NMF) [5].

In this work we aim at investigating the influence of newly developed weight update strategies for FSS as well as the influence of a non-linear decrease of the step-size parameters $step_{ind}$ and $step_{vol}$. We introduce and compare weight update strategies based on a linear decrease of weights, as well as a fitness based weight decrease strategy. Moreover, we introduce a combination of (*i*) this fitness based weight decrease strategy, (*ii*) the non-linear decrease of the step-size parameters, and (*iii*) a newly introduced dilation multiplier which breaks the symmetry between contraction and dilation but can be useful in some situations to escape from local minima. Experimental evaluation performed on five benchmark functions shows that especially the non-linear decrease of the step-size parameters is an effective and efficient way to significantly speed up the convergence of FSS and also to achieve better fitness per iteration results.

2 The Fish School Search Algorithm

FSS is based on four operators which can be grouped into two classes: feeding and swimming. *Feeding* represents updating the weight of the fish based on the successfulness of the current movement. The *swimming* operators (individual movement, collective instinctive movement, and collective volitive movement) move the fish according to the feeding operator. FSS is closely related to PSO and other population based algorithms such as Genetic Algorithms [6], Differential Evolution [7], and the Firework Algorithm [8]. The main difference compared to PSO is that no global variables need to be logged in FSS. Some similarities and differences of FSS to other population based algorithms are given in [2].

FSS operators. In the following we briefly review the basic operators of the Fish School Search algorithm as presented in [3]. A pseudo code of the FSS algorithm can also be found in [3]. The algorithm starts with all fish initialized at random positions and equal weight $w_i(0)$ set to 1.

A. *Individual movement:* In each iteration, each fish randomly chooses a new position which is determined by adding to each dimension j of the current position a random number multiplied by a predetermined step ($step_{ind}$).

$$n_j(t) = x_j(t) + randu(-1, 1) * step_{ind}, \tag{1}$$

where $randu(-1, 1)$ is a random number from a uniform distribution in the interval $[-1, 1]$. The movement *only* occurs if the new position $\boldsymbol{n}$ has a better fitness than the current position $\boldsymbol{x}$, and if $\boldsymbol{n}$ lies within the aquarium boundaries. Fitness difference (Δf) and displacement ($\Delta \boldsymbol{x}$) are evaluated according to

$$\Delta f = f(\boldsymbol{n}) - f(\boldsymbol{x}), \tag{2}$$

$$\Delta \boldsymbol{x} = \boldsymbol{n} - \boldsymbol{x}. \tag{3}$$

If no individual movement occurs $\Delta f = 0$ and $\Delta \boldsymbol{x} = \boldsymbol{0}$. The parameter $step_{ind}$ decreases linearly during the iterations

$$step_{ind}(t+1) = step_{ind}(t) - \frac{step_{ind\,initial} - step_{ind\,final}}{number\,of\,iterations}. \tag{4}$$

B. *Feeding:* Fish can increase their weight depending on the success of the individual movement according to

$$w_i(t+1) = w_i(t) + \Delta f(i)/max(\Delta f), \tag{5}$$

where $w_i(t)$ is the weight of fish i, $\Delta f(i)$ is the difference of the fitness at current and new location, and $\max(\Delta f)$ is the maximum Δf of all fish. An additional parameter w_{scale} limits the weight of a fish ($1 <= w_i <= w_{scale}$).

C. *Collective instinctive movement:* After all fish have moved individually, a weighted average of individual movements based on the instantaneous success of all fish is computed. All fish that successfully performed individual movements influence the resulting direction of the school movement (i.e. only fish whose $\Delta \boldsymbol{x} != \boldsymbol{0}$ influence the direction). The resulting direction $\boldsymbol{m}(t)$ is evaluated by

$$\boldsymbol{m}(t) = \frac{\sum_{i=1}^{N} \Delta \boldsymbol{x}_i \Delta f_i}{\sum_{i=1}^{N} \Delta f_i}. \tag{6}$$

Then, all fish of the school update their positions according to $\boldsymbol{m}(t)$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{m}(t). \tag{7}$$

D. *Collective volitive movement:* This movement is either a contraction of the swarm towards the barycenter of all fish, or a dilation of the swarm away from the barycenter, depending on the overall success rate of the whole school of fish. If the overall weight increased after the individual movement step, the radius of the fish school is contracted in order to increase the exploitation ability, else the

radius of the fish school is dilated in order to cover a bigger area of the search space. First, the barycenter $\boldsymbol{b}$ (center of mass/gravity) needs to be calculated

$$\boldsymbol{b}(t) = \frac{\sum_{i=1}^{N} \boldsymbol{x}_i w_i(t)}{\sum_{i=1}^{N} w_i(t)}. \tag{8}$$

When the total weight of the school *increased* in the current iteration, all fish must update their location according to

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) - step_{vol} randu(0,1) \frac{(\boldsymbol{x}(t) - \boldsymbol{b}(t))}{distance(\boldsymbol{x}(t), \boldsymbol{b}(t))}, \tag{9}$$

when the total weight *decreased* in the current iteration the update is

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + step_{vol} randu(0,1) \frac{(\boldsymbol{x}(t) - \boldsymbol{b}(t))}{distance(\boldsymbol{x}(t), \boldsymbol{b}(t))}, \tag{10}$$

where $distance()$ is a function which returns the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{b}$, and $step_{vol}$ is a predetermined step used to control the displacement from/to the barycenter. As suggested in [3] we set $step_{vol} = 2 * step_{ind}$.

3 New Update Strategies

On the one hand, we apply new weight update strategies that aim at adjusting the weight of each fish in each iteration (***S1, S2***). On the other hand, we introduce a non-linear update to the step-size parameters $step_{ind}$ and $step_{vol}$ (***S3***). Finally, ***S4*** is a combination of ***S2***, ***S3***, and an additional parameter.

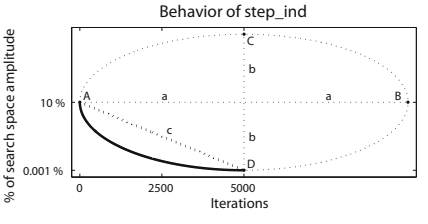

Fig. 1. Linear and non-linear decrease of *step_ind* and *step_vol*

- ***S1*** *(weight update) - linear decrease of weights:* Here, the weights of all fish are decreased linearly in each iteration by a pre-defined factor Δ_{lin} such that after the weight update in Eqn. (5) the weight of all fish is reduced by $w_i = w_i - \Delta_{lin}$, and all weights smaller than 1 are set to 1.
- ***S2*** *(weight update) - fitness based decrease of weights:* Here, not all fish will have their weights diminished by the same factor, instead fish in poor regions will loose weight more quickly. If $f(\boldsymbol{x})$ is a vector containing the fitness values of all fish at their current location, the weight of the fish will be decreased by

$\Delta \boldsymbol{f}_{fit\,based} = normalize(f(\boldsymbol{x}))$, where *normalize()* is a function that scales $f(\boldsymbol{x})$ in the range $[0, 1]$. Experiments showed that in order to get good results $\Delta \boldsymbol{f}_{fit\,based}$ needs to be scaled by a constant c_{fit} (between 3 and 5), which is done by $\Delta \boldsymbol{f}_{fit\,based} = (\Delta \boldsymbol{f}_{fit\,based}.^2)/c_{fit}$. Finally the weights are updated by (11) and weights smaller than 1 are set to 1.

$$w_i = w_i - \Delta \boldsymbol{f}_{fit\,based} \tag{11}$$

- ***S3*** *(step size update) - non-linear decrease of* $step_{ind}$ *and* $step_{vol}$*:* ***S3*** implements a non-linear decrease of the step size parameters which is based on the shape of an ellipse (see Fig. 1). The motivation for this non-linear decrease is that the algorithm is forced to converge earlier to the (ideally global) minimum and has more iterations to search the area around the optimum in more detail. The bold curve in Fig. 1 shows the new non-linear step parameter $step_{ind\,nonlin}$, while the dotted line ("c") shows the behavior when $step_{ind}$ is decreased linearly. Remember that $step_{vol} = 2 * step_{ind}$.
  Let a be the number of iterations, let b be the distance between $step_{ind\,initial}$ and $step_{ind\,final}$, and let t be the number of the current iteration. In each iteration $step_{ind\,nonlin}(t)$ is calculated by

$$step_{ind\,nonlin}(t) = step_{ind\,initial} - sqrt\left[(1 - t^2/a^2) * b^2\right], \tag{12}$$

  which is derived from the canonical ellipse equation $x^2/a^2 + y^2/b^2 = 1$ (where x is replaced with t, and y is replaced with $step_{step_{ind\,nonlin}(t)}$.
- ***S4*** *- combination of* **S2**, **S3** *and a dilation multiplier:* This strategy combines ***S2*** and ***S3*** with a newly introduced dilation multiplier c_{dil} that allows to cover a bigger area of the search space when a dilation occurs in the collective volitive movement (i.e. when the total weight of the school decreased in the current iteration). The general idea behind the dilation multiplier is to help the algorithm to jump out of local minima. ***S2*** and ***S3*** are applied in every iteration, and, moreover, in case of a dilation all weights are reset to their initial weight ($\boldsymbol{w}(t) = 1$). A pseudo code of ***S4*** follows

while *stop criterion is not met* **do**
  apply (in this order) Eqns. (1) (2) (3) **(11) (12)** (6) (7) (8);
  if $(w(t) > w(t-1))$ **then**
    Eqn. (9);
  else
    $\boldsymbol{w}(t) = 1$;
    $\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + c_{dil} * step_{vol} * randu(0,1) * \frac{(\boldsymbol{x}(t) - \boldsymbol{b}(t))}{distance(\boldsymbol{x}(t), \boldsymbol{b}(t))}$
  end
end

4 Experimental Setup

Table 1 shows the benchmark functions used for minimization in this paper, as well as the search space and the optimum point for each function. The initialization subspace was chosen to be in the interval $[up/2, up]$, where up is the upper

Table 1. Benchmark test functions and function parameters

Name	Equation	Search space	Optimum
$F_{Ackley}(\boldsymbol{x})$	$= -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	$-32 \leq x_i \leq 32$	$0.0D$
$F_{Griewank}(\boldsymbol{x})$	$= \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(x_i/\sqrt{i}\right) + 1$	$-600 \leq x_i \leq 600$	$0.0D$
$F_{Rastrigin}(\boldsymbol{x})$	$= 10D + \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i)\right]$	$-5.12 \leq x_i \leq 5.12$	$0.0D$
$F_{Rosenbrock}(\boldsymbol{x})$	$= \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right]$	$-30 \leq x_i \leq 30$	$1.0D$
$F_{Sphere}(\boldsymbol{x})$	$= \sum_{i=1}^{D} x_i^2$	$-100 \leq x_i \leq 100$	$0.0D$

limit of the search space for each function (similar to [3]). We used the same settings as in [3]: 5 000 iterations, 30 dimension, and 30 fishes, leading to 300 000 functions evaluations. We performed 15 trials per function. For all experiments $step_{ind\,initial}$ was set to 10% of *up*, and $step_{ind\,final}$ was set to 0.001% of *up*.

5 Evaluation

In this section we evaluate the update strategies introduced in Section 3. First, we discuss each strategy separately and focus especially on fitness per iteration aspects, i.e. how many iterations are needed in order to achieve a given fitness. Later we compare the best results achieved by all update strategies to each other and discuss the increase of computational cost caused by the update strategies. In all figures *basic FSS* refers to the basic FSS algorithm as presented in [3].

S1* - linear decrease of weights.** The results for strategy ***S1 are show in Fig. 2. The results are shown for four different values of Δ_{lin} (abbreviated as "Δ lin 0.0XXX" in the figure) ranging from 0.0125 to 0.075. Subplots (B) to (F) show the fitness per iteration for the five benchmark functions, and Subplot (A) shows the average (mean) weight of all fish per iteration, which decreases with increasing Δ_{lin}. Obviously, in most cases ***S1*** is not able to improve the results, but for the Rastrigin function the final result after 5 000 iterations can by clearly improved when Δ_{lin} is set to 0.075 (and partly also for 0.05). However, generally this update strategy is neither able to improve the final results after 5 000 iterations nor to achieve better results after a given number of iterations.

S2* - fitness based decrease of weights.** The results for strategy ***S2 are show in Fig. 3. Subplot (A) shows again the average (mean) weight of all fish per iteration – the average weight is very similar to Subplot (A) of Fig. 2. The parameter c_{fit} that scales the decrease of the weights is abbreviated as "c_{fit} X". The results for the Rastrigin function are even better than for the strategy ***S1***

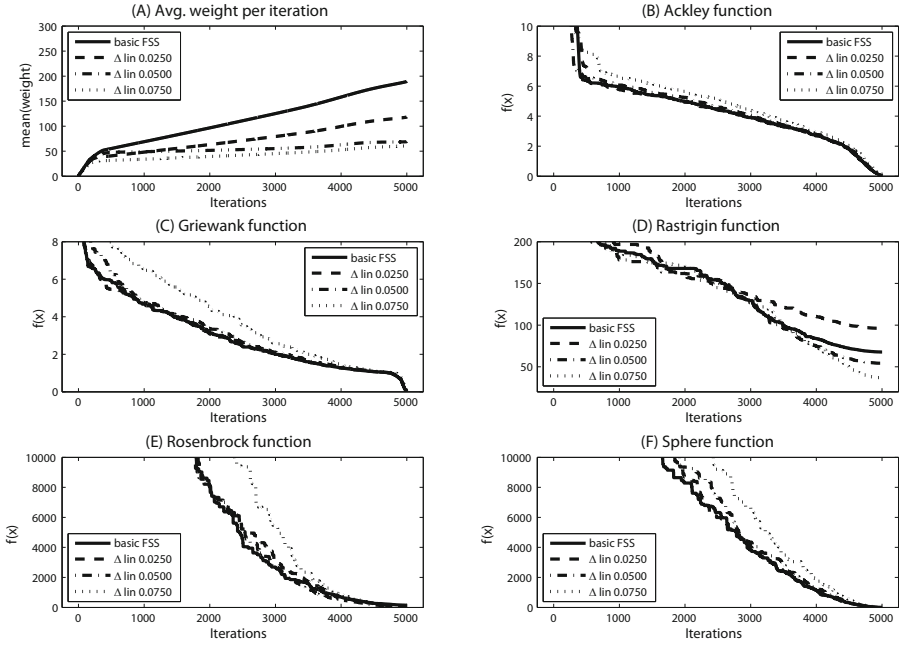

Fig. 2. ***S1*** - linear decrease of weights

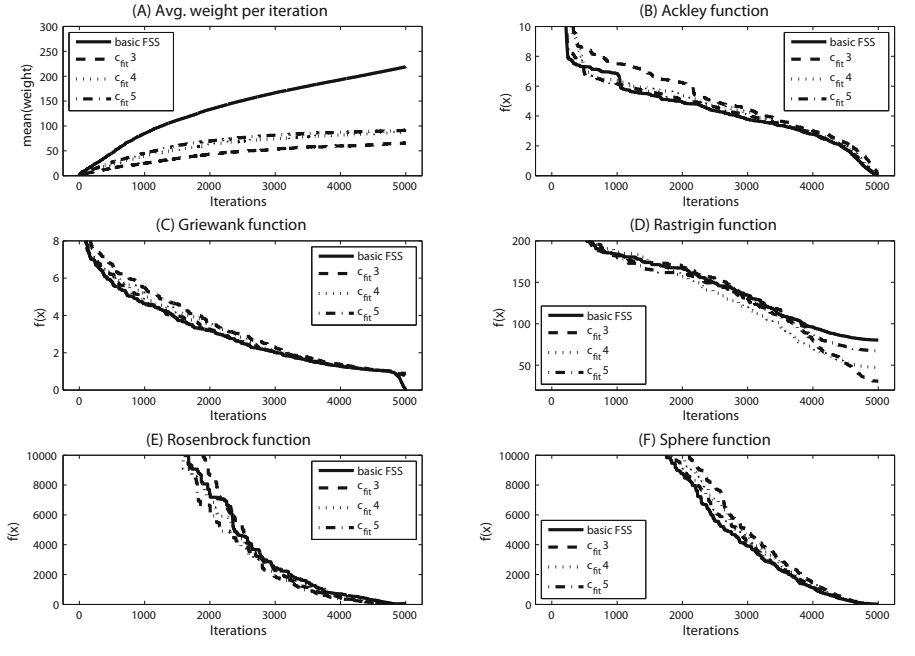

Fig. 3. ***S2*** - fitness based decrease of weights

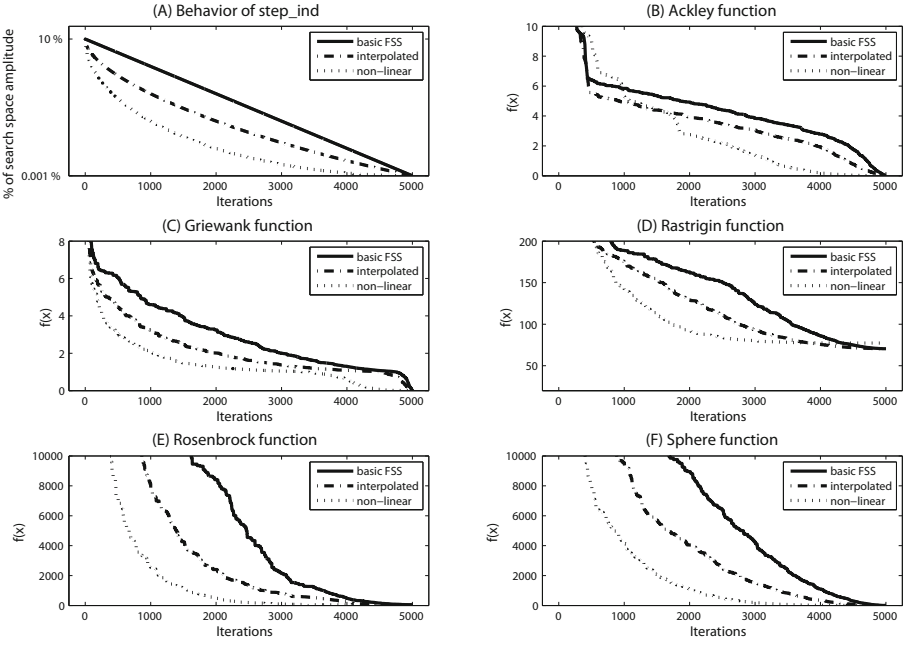

Fig. 4. ***S3*** - non-linear decrease of $step_{ind}$ and $step_{vol}$

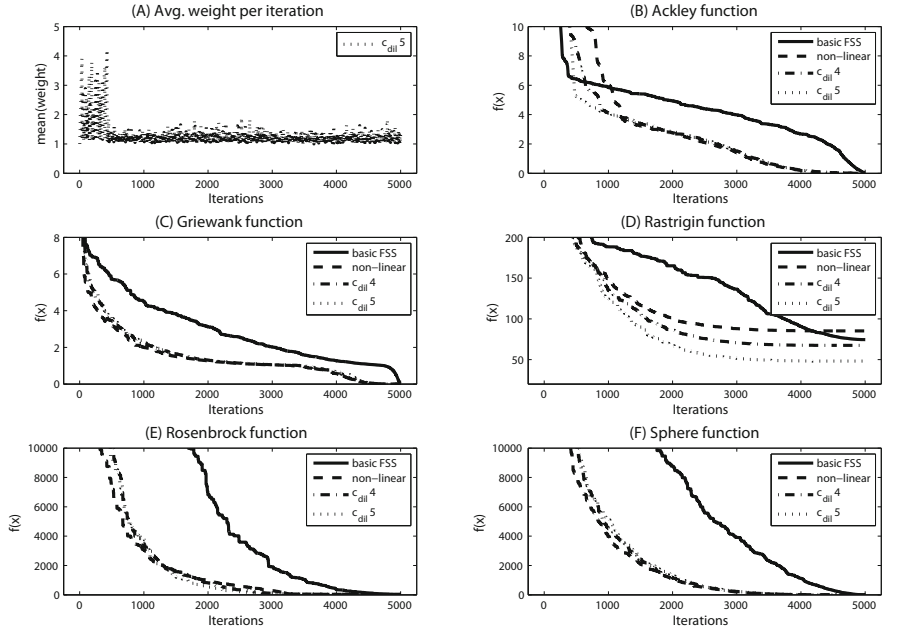

Fig. 5. ***S4*** - combination of ***S1***, ***S2*** and dilation multiplier

Table 2. Comparison of mean value and standard deviation (in small font under mean vaule) for 15 trials after 5 000 iterations for the five benchmarks functions. The best results are highlighted in bold. Last row: computational cost.

Function	basic FSS	***S1***	***S2***	***S3***	***S4***
$F_{Ackley}(\boldsymbol{x})$	0.0100 *0.0019*	0.0100 *0.0023*	0.1270 *0.0043*	**0.0007** *6.7e-05*	**0.0007** *5.3e-05*
$F_{Griewank}(\boldsymbol{x})$	0.0233 *0.0098*	0.0172 *0.0061*	0.7501 *0.1393*	0.0058 *0.0048*	**3.2e-05** *5.7e-06*
$F_{Rastrigin}(\boldsymbol{x})$	70.443 *19.465*	36.879 *8.0181*	**30.745** *11.801*	67.126 *15.834*	48.156 *13.780*
$F_{Rosenbrock}(\boldsymbol{x})$	27.574 *1.2501*	28.498 *1.3876*	26.277 *2.3244*	**22.775** *2.5801*	23.718 *2.5353*
$F_{Sphere}(\boldsymbol{x})$	0.0699 *0.0183*	0.0649 *0.0237*	0.1165 *0.0615*	**3.4e-04** *7.26e-05*	**3.4e-04** *6.59e-05*
Runtime	1.0	× 1.0017	× 1.0042	× 1.0142	× 1.0238

and also the results for the Rosenbrock function could be improved slightly. As Table 2 indicates, this strategy achieves the best final result for the Rastrigin function of all strategies after 5 000 iterations. For the other benchmark functions this strategy perform equally or worse than basic FSS.

S3* - non-linear decrease of $step_{ind}$ and $step_{vol}$. *S3 results are show in Fig. 4, where $step_{nonlinear}(t)$ is abbreviated as "non-linear". "Interpolated" shows the results using an interpolation of "basic FSS" and "non-linear", i.e. $step_{interpol}(t)$ = $step_{ind}(t)$ - $[step_{ind}(t) - step_{nonlinear}(t)]/2$. Subplot (A) shows the behavior of $step_{ind}$ and should be compared to Fig. 1. The results indicate that this non-linear decrease of the step-size parameters significantly improves the fitness per iteration for all five benchmark functions. Generally, "non-linear" achieves the best results, followed by "interpolated". For some functions, such as (D) or (E), this strategy needs only about half as many iterations as basic FSS to achieve almost the same results as basic FSS after 5 000 iterations.

S4* - combination of *S2*, *S3* and dilation multiplier.** The results for strategy ***S4 are show in Fig. 5 and are compared to basic FSS and "non-linear" from strategy ***S3***. The dilation multiplier c_{dil} is abbreviated as "c_{dil} X". Since the weight of all fish is reset to 1 if a dilation occurs, the average (mean) weight per iteration is relatively low (see Subplot (A)). Generally, this strategy achieves similar results as strategy ***S3***, but clearly improves the results for the Rastrigin function and achieves a better final result after 5 000 iterations and also better fitness per iteration for "c_{dil} 5".

Comparison of final results. Table 2 shows a comparison of the mean values and the standard deviations after 5 000 iterations. As can be seen, the results for all five benchmark functions could be improved. Overall, strategy ***S4*** achieves the best results followed by ***S3***. ***S1*** and ***S2*** are better or equal than basic FSS for 4 out of 5 benchmark functions.

Computational cost. The last row of Table 2 shows the increase in computational cost caused by the additional computations of the update steps. Example: the runtime for ***S1*** is 1.0017 times as long as the runtime for basic FSS. This indicates that the increase in runtime is only marginal and further motivates the utilization of the presented update steps.

6 Conclusion

In this paper we presented new update strategies for the Fish School Swarm algorithm. We investigated the influence of newly developed weight update strategies as well as the influence of a non-linear decrease of the step-size parameters $step_{ind}$ and $step_{vol}$. Results indicate that strategies ***S3*** and ***S4*** are able to significantly improve the fitness per iteration for all benchmark functions and also achieve better final results after 5 000 iterations when compared to the basic implementation of FSS. The results motivate for further research on update strategies for FSS and also for adapting the non-linear decrease of the step size parameters for other search heuristics.

Acknowledgments. This work was supported by National Natural Science Foundation of China (NSFC), Grant No. 60875080. Andreas wants to thank the *Erasmus Mundus External Coop. Window*, Lot 14 (2009-1650/001-001-ECW).

References

[1] Bastos Filho, C., Lima Neto, F., Lins, A., Nascimento, A.I.S., Lima, M.: A novel search algorithm based on fish school behavior. In: IEEE International Conference on Systems, Man and Cybernetics, SMC 2008, pp. 2646–2651 (2008)
[2] Bastos Filho, C., Lima Neto, F., Lins, A., Nascimento, A.I.S., Lima, M.: Fish school search: An overview. In: Chiong, R. (ed.) Nature-Inspired Algorithms for Optimisation. SCI, vol. 193, pp. 261–277. Springer, Heidelberg (2009)
[3] Bastos Filho, C., Lima Neto, F., Sousa, M., Pontes, M., Madeiro, S.: On the influence of the swimming operators in the fish school search algorithm. In: Int. Conference on Systems, Man and Cybernetics, pp. 5012–5017 (2009)
[4] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
[5] Janecek, A.G., Tan, Y.: Using population based algorithms for initializing nonnegative matrix factorization. In: ICSI 2011: Second International Conference on Swarm Intelligence (to appear, 2011)
[6] Goldberg, D.E.: Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman, Boston (1989)
[7] Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341–359 (1997)
[8] Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 355–364. Springer, Heidelberg (2010)