

# Artificial Fish School Algorithm Applied in a Combinatorial Optimization Problem

Yun Cai

Wuhan University of Science and Technology/Country College of Machinery and Automation, Wuhan, China  
Email: yuncaicai@126.com

**Abstract**—An improved artificial fish swarm algorithm (AFSA) for solving a combinatorial optimization problem—a berth allocation problem (BAP), which was formulated. Its objective is to minimize the turnaround time of vessels at container terminals so as to improve operation efficiency customer satisfaction. An adaptive artificial fish swarm algorithm was proposed to solve it. Firstly, the basic principle and the algorithm design of the AFSA were introduced. Then, for a test case, computational experiments explored the effect of algorithm parameters on the convergence of the algorithm. Experimental results verified the validity and feasibility of the proposed algorithm with rational parameters, and show that the algorithm has better convergence performance than genetic algorithm (GA) and ant colony optimization (ACO).

**Index Terms**—combinatorial optimization, berth allocation, scheduling, artificial fish swarm algorithm

## I. INTRODUCTION

A combinatorial optimization problem  $P=(S, f)$  can be defined by:

- a set of variables  $X=\{x_1, \dots, x_n\}$ ;
- variable domains  $D_1, \dots, D_n$ ;
- constraints among variables;
- an objective function  $f$  to be minimized,

where  $f: D_1 \times \dots \times D_n \rightarrow IR^+$ ; The set of all possible feasible assignments is  $S=\{s=\{(x_1, v_1), \dots, (x_n, v_n)\} \mid v_i \in D_i, s \text{ satisfies all the constraints}\}$ .  $S$  is usually called a search (or solution) space, as each element of the set can be seen as a candidate solution. To solve a combinatorial optimization problem one has to find a solution  $s^* \in S$  with minimum objective function value, that is,  $f(s^*) \leq f(s) \forall s \in S$ .  $s^*$  is called a globally optimal solution of  $(S, f)$  and the set  $S^* \subseteq S$  is called the set of globally optimal solutions.

Combinatorial optimization problems exist widely in the fields of economic management, transportation, communication network and other fields. Its main purpose is to find optimal scheduling, grouping, order or filtering of discrete events. Many combinatorial optimization problems of both practical and theoretical importance are known to be NP-hard, such as the Knapsack problem, the Traveling Salesman Problem (TSP), and Timetabling and Scheduling problems.

Since exact algorithms are not feasible in such cases, heuristics are the main approach to tackle these problems. In the last two decades, swarm intelligence techniques,

such as Genetic Algorithm [1], Particle Swarm Optimization (PSO) [1], Bee Colony Optimization (BCO) [2], or Ant Colony Optimization [3], have made some progress in combinatorial optimization problems. They are population-based methods that make use of the global behavior that emerges from the local interaction of individuals with one another and with their environment.

AFSA, which was presented by X. L. Li [4], is a new swarm intelligence optimization method by simulating fish swarm behavior. It is becoming a prospective method because of its good performances in solving traveling salesman problem [5], routing optimization problem [6], complex function optimization problem [7], and so on.

An efficient schedule will improve customer satisfaction, increase port cargo throughput, and enhance port revenues. The BAP involves the determination of berthing times and locations of vessels that are to arrive at a port within a planning horizon, while satisfying a number of spatial and temporal constraints, to optimize operations. Berth planning has been shown to be a NP-hard problem by relating it to the set partitioning problem, the single machine scheduling problem with release dates and the two dimensional cutting stock problem [8]. In recent years, many scholars have studied the BAP using heuristic methods. For example, C. Tong [3] used ant colony optimization to find solutions with good mean costs in a static BAP. K. H. Kim [9] formulated a mixed integer linear programming (MIP) model considering the cost. Experimental results showed that a simulated annealing algorithm obtains solutions that are similar to the optimal solutions found by a MIP model. E. Nishimura [10] developed a heuristic procedure based on GA for determining a dynamic berth assignment to ships in the public berth system, and computational experiments showed that the proposed algorithm is adaptable to real world applications. Many scholars applied their algorithms to different models and used real world data or random numbers in test cases under different computational environments. So it is difficult for us to tell whose algorithm is better.

In this paper, an adaptive AFSA was applied to a BAP. The focus of the paper is on verifying the validity and feasibility of the method and on comparing the convergence performance of the AFSA with that of GA and ACO in a numerical experiment.

The rest of the paper is organized as follows. Section II introduces the model formulation. Section III proposes the adaptive AFSA for the BAP. Section IV presents

computational experiments and the final section consists of concluding remarks.

## II. PROBLEM FORMULATION

The discussed BAP was a discrete and dynamic berth allocation problem. A number of assumptions were made in the following model as follows:

- Each vessel should be serviced once and exactly once at any berth.
- Vessels should be serviced after the arrival of vessels.
- Vessels should be serviced at any berth with acceptable physical conditions such as water depth and quay length.
- A berth section handled at most one vessel at a time.

The following notations were used in the mathematical model.

$B$ : set of berths.

$V$ : set of vessels.

$i$ :  $i=(1, \dots, I) \in B$ .

$j$ :  $j=(1, \dots, J) \in V$ .

$b_j$ : the starting time of the service for the  $j^{\text{th}}$  vessel.

$A_j$ : the arrival time of the  $j^{\text{th}}$  vessel.

$C_{ij}$ : the service time of the  $j^{\text{th}}$  vessel at the  $i^{\text{th}}$  berth.

$X_{ij}$ : indicated that if the  $j^{\text{th}}$  vessel was serviced at the  $i^{\text{th}}$  berth,  $X_{ij}$  was equal to 1, otherwise it was equal to 0.

$S_i$ : the number of vessels at the  $i^{\text{th}}$  berth.

$S$ : the total number of arrival vessels.

$W_i$ : the water depth of the  $i^{\text{th}}$  berth.

$D_j$ : the draft of the  $j^{\text{th}}$  vessel including the safety vertical distance for berthing.

$P_i$ : the quay length of the  $i^{\text{th}}$  berth.

$L_j$ : the length of the  $j^{\text{th}}$  vessel including the horizontal safety length.

$$\text{Objective: Minimize } \sum_{i \in B} \sum_{j \in V} (b_j - A_j + C_{ij}) X_{ij} \quad (1)$$

$$\text{Subject to: } \sum_{i \in B} S_i = S \quad (2)$$

$$\sum_{i \in B} X_{ij} = 1 \quad (3)$$

$$b_j - A_j \geq 0 \quad (4)$$

$$(W_i - D_j) X_{ij} \geq 0 \quad (5)$$

$$(P_i - L_j) X_{ij} \geq 0 \quad (6)$$

$$X_{ij} \in \{0, 1\} \quad (7)$$

Equation (1) minimized the total of flow time incurred by vessels. Equation (2) ensured that the total number of arrival vessels was equal to the sum of vessels at berths. Equation (3) ensured that each vessel should be serviced once and exactly once at any berth. Equation (4) ensured that vessels should be serviced after their arrivals. Equation (5) guaranteed that the draft of the  $j^{\text{th}}$  vessel did not exceed the water depth of the assigned berth. Equation (6) ensured that the length of the  $j^{\text{th}}$  vessel did not exceed the quay length of the  $i^{\text{th}}$  berth [7] [11].

## III. DESIGNING AN ADAPTIVE AFSA

### A. Basic Principle of AFSA

In water areas, following other fish, a fish can always find food at a place where there are a lots of food, hence generally the more food, the more fish. According to this phenomenon, AFSA builds some artificial fish (AF), which search an optimal solution in solution space (the environment in which AF live) by imitating fish swarm behavior. Three basic behaviors of AF are defined as follows [4]:

- Prey*: The fish perceives the concentration of food in water to determine the movement by vision or sense and then chooses the tendency.
- Swarm*: The fish will assemble in groups naturally in the moving process, which is a kind of living habits in order to guarantee the existence of the colony and avoid dangers.
- Follow*: In the moving process of the fish swarm, when a single fish or several fish find food, the neighborhood partners will trail and reach the food quickly.

The three behaviors' pseudo-code can be seen in [4].

### B. Algorithm Design for the BAP

1) *Symbol definition*: The following notations were used in the algorithm.

$X$ : the current status of an artificial fish swarm.  $X = (X_1, X_2, \dots, X_n)$ , where  $X_i$  ( $i=1, \dots, n$ ) represented the current status of the  $i^{\text{th}}$  AF, i.e., the searched solution. It was the berthing sequence assigned to vessels. For example,  $X_i = (1,3,4,2)$  expressed that vessels  $S_1, S_3$  were assigned to berth1 in the order of 1,3; vessels  $S_2, S_4$  to berth2 in the order of 4,2.

$Y_i$ : the value of function (1) corresponding to  $X_i$ , i.e., the value of fitness function  $Y_i = f(X_i)$ .

$D_{ij}$ : the distance between the  $i^{\text{th}}$  AF and the  $j^{\text{th}}$  AF. It was the number of different elements between  $X_i$  and  $X_j$ . For example, if  $X_i = (1,3,2,5,6,4)$  and  $X_j = (1,3,6,2,5,4)$ , then  $D_{ij} = 3$ .

*Visual*: the visual range of the AF.

$N(X_i, \text{Visual})$ : set of the visual-neighbors of the  $i^{\text{th}}$  AF.

$$N(X_i, \text{Visual}) = \{X_j \mid D_{ij} \leq \text{Visual}\}.$$

$\delta$ : congestion degree,  $0 < \delta < 1$ .

*FishNum*: the population size of an artificial fish swarm.

*Maxgen*: maximum iterations.

*Trynumber*: the number of prey iteration.

2) *Behavior description of AF*: Let us assume that  $X_i$  was the status of the  $i^{\text{th}}$  AF at present.

a) *Follow behavior*

Step1: For  $X_i$ , selected the best status  $X_j$  with the minimum  $Y_j$  within  $N(X_i, \text{Visual})$ . That is to say, the  $j^{\text{th}}$  AF had the minimum value of objective function among visual-neighbors of the  $i^{\text{th}}$  AF.

Step2: Calculated  $nf$ , the number of visual-neighbors of the  $i^{\text{th}}$  AF. If  $Y_j < Y_i$  and  $nf / \text{FishNum} < \delta$  (there was additional space for other fish), then replaced  $X_i$  with  $X_j$ ;

otherwise executed prey behavior. Pseudo code description of the process is presented as follows:

```
Artificial_fish_follow()
{
   $Y_j = \text{Min}(f(X_j)), X_j \in N(X_i, \text{Visual});$ 
   $nf = N(X_i, \text{Visual});$ 
  if ( $nf / \text{FishNum} < \delta$  and  $Y_j < Y_i$ )
  then  $X_i = X_j;$ 
  else  $AF\text{-prey}();$ 
  end
}
```

#### b) Prey behavior

Step1: For  $X_i$ , randomly selected a status  $X_j$  from visual-neighbors of the  $i^{\text{th}}$  AF.

Step2: If  $Y_j < Y_i$ , then substituted  $X_j$  for  $X_i$ , i.e.,  $X_j$  became the new current status of the  $i^{\text{th}}$  AF; otherwise went to Step1.

If the above operation executed *Trynumber* times and could not find a better status, then set the last random status as the new current status. Pseudo code description of the process is presented as follows:

```
Artificial_fish_pre()
{
  for  $i=0$  to trynumber
   $X_j = \text{Random}(N(X_i, \text{Visual}));$ 
  if ( $Y_j < Y_i$ )
  then  $X_i = X_j;$ 
  end
end
}
```

#### c) Swarm behavior

Step1: For  $X_i$ , calculated  $nf$ , the number of visual-neighbors of the  $i^{\text{th}}$  AF.

Step2: If  $nf \neq 0$  and  $nf / \text{FishNum} < \delta$ , determined the central position  $X_c$  of all artificial fish, in other words, selected the mutual status of the neighborhood partners of  $X_i$ . Calculated the fitness function  $Y_c = f(X_c)$ . If  $Y_c < Y_i$ , which meant that the food concentration of  $X_c$  was high and the surroundings was not very crowded, then replaced  $X_i$  with  $X_c$ ; otherwise executed move behavior.

Step3: If  $nf = 0$ , executed another behavior. Pseudo code description of the process is presented as follows:

```
Artificial_fish_swarm()
{
   $nf = N(X_i, \text{Visual});$ 
  if ( $nf \neq 0$  and  $nf / \text{FishNum} < \delta$ )
  then
     $X_c = \text{Center}(N(X_i, \text{Visual}));$ 
     $Y_c = f(X_c);$ 
    if  $Y_c < Y_i$ 
    then  $X_i = X_c;$ 
    else  $AF\text{-move}();$ 
    end
  else
     $AF\text{-move}();$ 
  end
}
```

#### d) Move behavior

If the optimal solution was not improved in the above optimization progress, generated another status randomly. If the function value of the new status was better than  $X_i$ , then adopted it as the current status of the  $i^{\text{th}}$  AF; otherwise executed another behavior.

This behavior could help the solution to escape from local optimum, meanwhile reduced calculation workload and improved algorithm efficiency.

In this paper, four behaviors (follow, prey, swarm and move) were executed in turn, and the latter was used until the former failed. For example, the AF executed follow behavior first, if its status could not be improved, then executed prey behavior.

3) *Constraint strategy*: When a vessel moors to a berth, it should meet the requirement of the berth, such as water depth and quay length. The vessel sequence was generated randomly in the algorithm. To exclude infeasible solutions, a large number was appointed to the objective values of infeasible solutions in the process of evaluating solutions.

4) *Stopping Criteria*: AFSA is an iterative algorithm, which gradually converges into the global optimal solution. For problems having a known solution, common stopping criterion is maximum iterations. There is another method that checks the variation of the fitness value. If there is very little variation after some iteration, then terminate the calculation. This paper adopted the former. A bulletin board recorded optimal solutions obtained at each iteration and output the best found finally.

5) *Algorithm Procedure*: The algorithm procedure of the BAP was described as follows.

Step1: Initialization. Set *FishNum*, *Maxgen*, *Trynumber*, *Visual*,  $\delta$ ; input the expected arrival time and service time of vessels; randomly generated an artificial fish swarm with the population size *FishNum*, i.e., initialize  $X = (X_1, X_2, \dots, X_i, \dots, X_{\text{FishNum}})$ .  $NC = 0$ .

Step2: For each  $X_i$ , executed follow behavior; if found a better solution  $X_j$ , replaced  $X_i$  with  $X_j$  then went to Step7; otherwise went to Step3.

Step3: Rounded  $\text{Visual} \times (1 - NC / \text{Maxgen})$  to the closest integer *Visual2*; if *Visual2*  $> 0$ , went to Step 4 using *Visual2*; otherwise went to Step5.

Step4: For  $X_i$ , executed prey behavior; if found a better solution  $X_j$ , replaced  $X_i$  with  $X_j$  then went to Step7; otherwise went to Step5.

Step5: For  $X_i$ , executed swarm behavior; if found a better solution  $X_j$ , replaced  $X_i$  with  $X_j$  then went to Step 7; otherwise went to Step6.

Step6: For  $X_i$ , executed move behavior. If found a better solution  $X_j$ , replaced  $X_i$  with  $X_j$  then went to Step7; otherwise went to Step8.

Step7:  $NC = NC + 1$ ; updated the optimal solution on the bulletin board.

Step8: If  $NC = \text{Maxgen}$ , output the current optimal solution, the end; otherwise, went to Step2.

IV. COMPUTATIONAL EXPERIMENTS

A. A Test Case

A test case used in Ref. [7] and [11] was selected to compare the convergence performance of the AFSA with that of GA and ACO. There are two berths in a container terminal. Seven vessels will arrive in a period of time. Assume that the number of quays at every berth is equal and vessels meet constraint (5) and (6) in Part II. Berth1 is idle at 0:00; berth2 is idle at 06:00. The arrival time and the service time of vessels, see Table I.

According to Ref. [7] and [11], there are two optimal solutions:  $X_1=(1,5,7,6;3,4,2)$  and  $X_2=(1,5,4;3,2,7,6)$ . The function values of them both are 73[9]. Table II shows partial results obtained by the AFSA.

TABLE I. VESSEL INFORMATION

Vessel name	Information	
	Arrival time(hour)	Service time(hour)
Ship1	00:00	12
Ship2	04:00	10
Ship3	06:00	3
Ship4	09:00	8
Ship5	11:00	5
Ship6	18:00	12
Ship7	19:00	4

TABLE II. PARTIAL RESULTS OBTAINED BY THE AFSA

Iterations	Current best solution	Objective value
1	1,5,3,7;2,4,6	82
2	1,2,5,7;3,4,6	81
3	1,5,4,7;3,2,6	75
4	1,5,7,2;3,4,6	74
5	1,5,7,6;3,4,2	73

B. Parameter Selection Experiments

The method of parameters selection in the algorithm has no strict theoretical basis, the optimal parameters of the algorithm was also a problem need to be solved. So in parameters experiments only one parameter was changed and other parameters are fixed to find its influence on the performance of the algorithm.

The range of parameters in selection experiments was:  $FishNum = 2\sim 20$ ,  $Maxgen = 5\sim 50$ ,  $Trynumber = 5\sim 100$ ,  $Visual = 2\sim 10$  and  $\delta = 0.2\sim 1.0$ . We have found the random initial value has little effect on convergence performance when parameters do not change. Partial simulation results are shown in Table III~ Table VII .

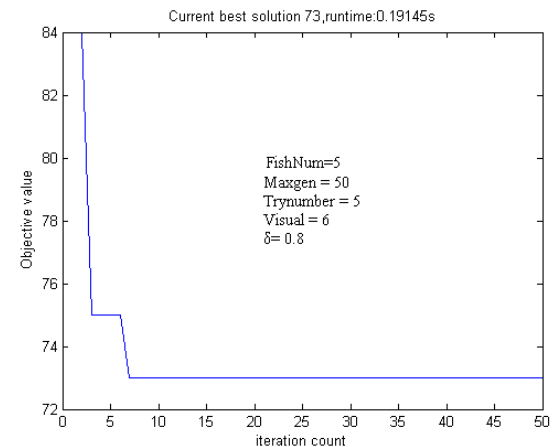
1) The influence of  $FishNum$

With the increase of population size  $FishNum$ , iterations with the optimal solution decreases generally and the runtime of the AFSA becomes longer. The increased AF can help the AFSA flee from the local extreme value. For instance, If  $FishNum < 5$ ,  $Trynumber < 35$ ,  $Visual = 6$  and  $\delta = 0.8$ , sometimes the AFSA can not converge after 20 iterations. Fig.1 shows the bigger

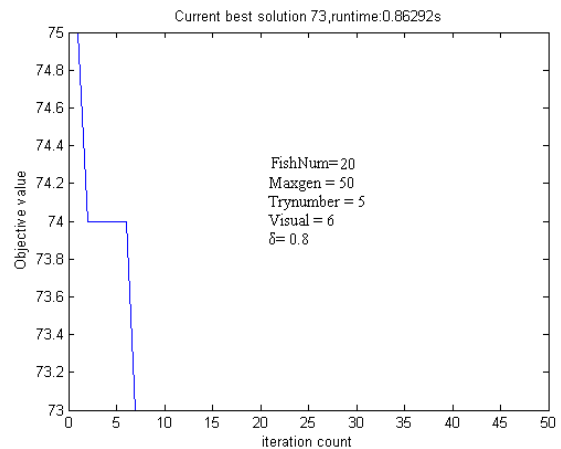
$FishNum$  leads to the smaller optimal solution of the first generation.

TABLE III. THE INFLUENCE OF  $FishNum$

Maxgen = 50, Trynumber = 5, Visual = 6 and $\delta = 0.8$				
FishNum	5	10	15	20
Time(s)	0.9938	1.0114	1.5968	2.2703
iteration	7	4	2	2



a)  $FishNum=5$



b)  $FishNum=20$

Figure1. The influence of  $FishNum$ .

2) The influence of  $Maxgen$

When parameters (except  $Maxgen$ ) are small, the AFSA usually converges after 20~30 iterations. It means  $Maxgen$  must be set as at least 20~30 for obtaining the optimal solution. With the increasing of  $Maxgen$ , the runtime of the AFSA becomes longer

TABLE IV.  
THE INFLUENCE OF *Maxgen*

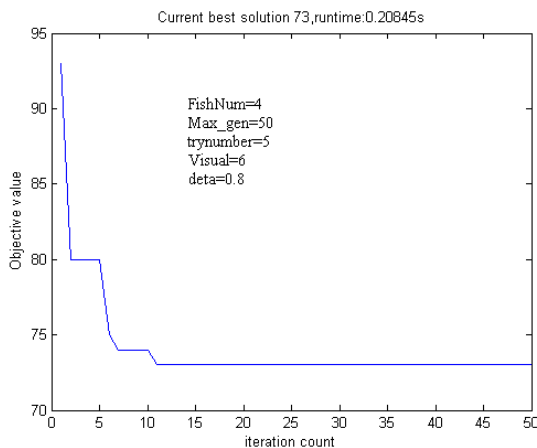
FishNum = 10, Trynumber = 40, Visual = 6 and $\delta = 0.8$				
Maxgen	20	40	60	80
Time(s)	0.1701	0.376	0.584	0.957
iteration	5	6	6	3

3) The influence of *Trynumber*

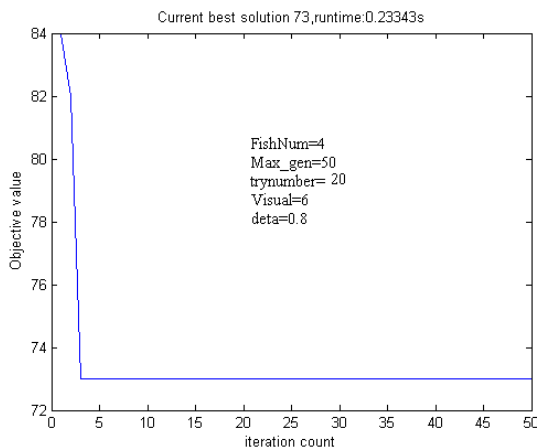
In general, the increase of *Trynumber* makes the AFSA converge quickly on condition that *FishNum* is greater than 4 and *Trynumber* is greater than 9. But big *Trynumber* will bring about the increasing of runtime, see Fig.2.

TABLE V.  
THE INFLUENCE OF *Trynumber*

FishNum = 10, Maxgen = 100, Visual = 6 and $\delta = 0.8$				
Trynumber	10	20	30	40
Time(s)	0.651	0.770	0.890	0.9944
iteration	2	4	4	5



a) *Trynumber* = 5



b) *Trynumber* = 20

Figure2. The influence of *Trynumber*.

4) The influence of *Visual*

The greater *Visual* is, the more easily the AFSA finds the global extreme value and converges. The value of *Visual* must match with the population size of a fish swarm, in general, for 10 artificial fish, only if  $3 < Visual < 7$ , the AFSA can converge. Fig.3 shows a simulation result when *Visual* was 2. Besides, the greater *Visual* will increase runtime generally.

From the above can be concluded, *Visual* is more important, it reflects the relation of neighborhood fish. It is easier to fall into local extremum using smaller *Visual*, finally be unable to converge to the global optimal solution. Big *Visual* will increase searching range, and greatly reduce the algorithm convergence speed; therefore, it is most important that selecting appropriate *Visual*, where we suggest let *Visual* be 5.

TABLE VI.  
THE INFLUENCE OF *Visual*

FishNum = 10, Maxgen = 100, Trynumber = 40 and $\delta = 0.8$				
Visual	4	5	7	8
Time(s)	0.843	0.914	not converge	not converge
iteration	6	5	not converge	not converge

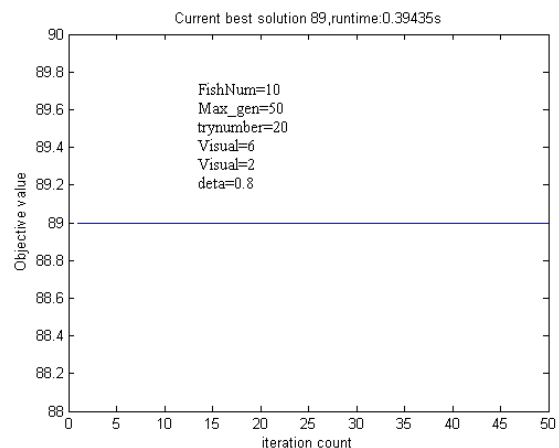
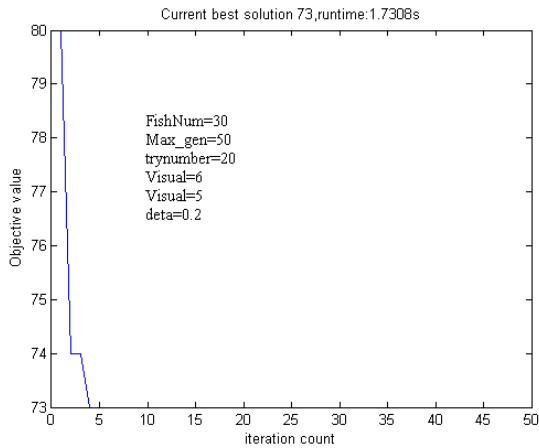


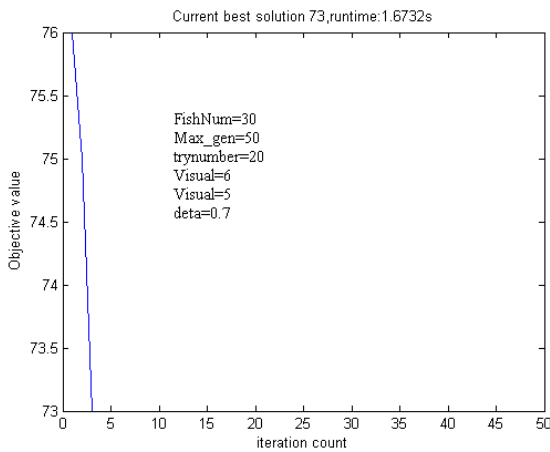
Figure3. The influence of *Visual*

5) The influence of  $\delta$

Fig.4 shows the greater the number of *FishNum* is, the stronger the influence of  $\delta$  is, the bigger  $\delta$  leads to the smaller optimal solution of the first generation. A rational value of  $\delta$  can prevent the AFSA from falling into local extreme value. Besides, we should consider the value of *Visual* when select the value of  $\delta$ . Let  $\delta$  be a large number when *Visual* is a small number. For example, if *Visual* = 4, let  $\delta = 1$ .



a)  $\delta=0.2$



b)  $\delta=0.7$

Figure4. The influence of  $\delta$

TABLE VII.  
THE INFLUENCE OF  $\delta$

<i>FishNum = 10, Maxgen = 100, Visual = 6 and Trynumber = 40</i>				
$\delta$	0.2	0.4	0.6	1
Time(s)	1.014	1.034	1.136	not converge
iteration	3	2	2	not converge

C. Comparing the AFSA with Other Heuristic Methods

Fig.5 shows iteration times for finding  $X_i$  and the number of relative tests (after 50 tests). It clearly shows that the AFSA often converges after about 3 iterations, which are much smaller than that of the GA (32 times in Ref. [7]), hybrid Algorithm of GA and Simulated Annealing (SA), and the ACO (8 times in Ref. [11]). This is to say, the AFSA converges more quickly than the GA and the ACO, see Fig.6. In terms of the computational time, because Ref. [7] and [11] have not introduced their computational environments, we can not compare the runtime, which is about 31~62ms in our tests without plot output (Intel Core 2 Duo E6550, 2.0GB RAM, MatLab) and is about 60ms in Ref. [7].

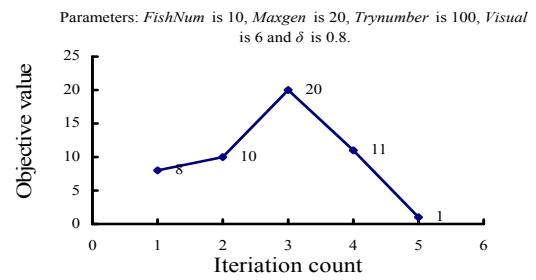


Figure5. Test count and iteration count (for finding  $X_i$ ).

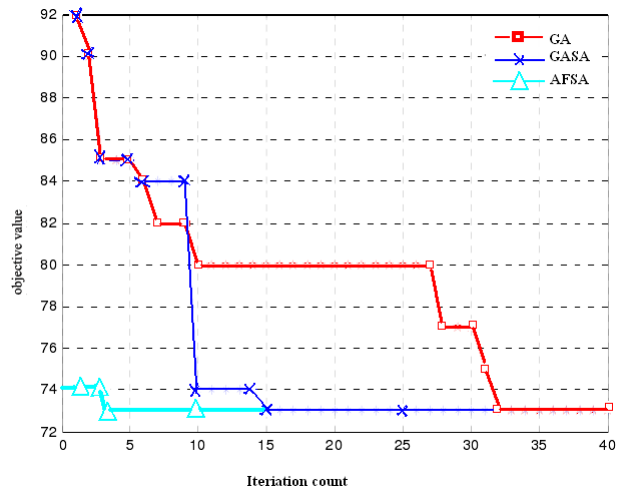


Figure6. Comparing the AFSA with other heuristic methods

V. CONCLUSIONS

In conclusion, the convergence speed of the AFSA is related to the parameter selection. With rational parameter values, the AFSA can approach and obtain the optimal or sub-optimal solution. In order to make AFSA more useful in a BAP and to compare its performance with that of other heuristic methods, we improved original AFSA for the BAP. The experimental results show that the model is capable of allocating berths efficiently and the algorithm can lead to satisfactory results in allowable CPU time. The AFSA can adjust the searching range adaptively by four search behaviors. The computational time and the quality of solutions depend on parameter selection. Optimizing parameters could further improve convergence performance. The improved AFSA converges more quickly than the GA in Ref. [7] and the ACO in Ref. [11]. The algorithm has improved the efficiency of the berth scheduling and has some practical significance.

Although the AFSA provided a new idea to solve the BAP, in practice, the schedule of quay cranes is considered in the BAP sometimes, because the service time of a vessel varies with the number of cranes assigned to it. So the issue of crane scheduling will be incorporated into the next model for further study. In addition, there are some other objectives should be considered, such as the cost, the traffic, etc.

#### ACKNOWLEDGMENT

This research was supported in part by the project fund of the Hubei Province key laboratory of mechanical transmission and manufacturing engineering Wuhan University of Science and Technology (2007A08).

#### REFERENCES

- [1] P. Surekha, P.R.A. Mohana Raajan, and S.Sumathi, "Genetic Algorithm and Particle Swarm Optimization approaches to solve combinatorial job shop scheduling problems," 2010 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, pp. 202-206, December, 2010.
- [2] D. Teodorovic, and M. Dell'Orco, Bee colony optimization –A cooperative learning approach to complex transportation problems, In *Advanced OR and AI Methods in Transportation*, pp.51-60, 2005.
- [3] C Tong, H Lau, A Lim, "Ant colony optimization for the ship berthing problem," in *Advances in Computing Science - ASIAN'99*, P.S. Thiagarajan and R. Yap, Eds. Thailand: LNCS1742, 1999, pp. 359-370.
- [4] X. L. Li, Z. J. Shao, and J. X. Qian, "An optimizing method based on autonomous animats: Fish-swarm Algorithm," *System Engineering Theory and Practice*, vol. 22, pp.32-38, November 2003.
- [5] Y. Q. Zhou, Z. C. Xie, "Improved artificial fish-school swarm algorithm for solving TSP," *Systems Engineering and Electronics*, vol. 31, pp. 1458-1461, June 2009.
- [6] X. J. Shan, M. Y. Jiang, "The routing optimization based on improved artificial fish swarm algorithm," *Proc. of IEEE the 6th World Congress on Intelligent Control and Automation*, Dalian China, pp.3658-3662, October 2006.
- [7] P. Li, *Modelling and Optimization of Berth Allocation and Quay Scheduling System*. Dissertation, Tianjin, China: Tianjin university, 2007.
- [8] C. Bierwirth, and F. Meisel, "A survey of berth allocation and quay crane scheduling problems in container terminals," *European Journal of Operational Research*, vol. 202, pp. 615-627, May 2010.
- [9] K. H. Kim, K. C. Moon, "Berth scheduling by simulated annealing," *Transportation Research Part B*, vol. 37, pp.541-560, July 2003.
- [10] E. Nishimura, A. Imai, and S. Papadimitriou, "Berth allocation planning in the public berth system by genetic algorithms," *European Journal of Operational Research*, vol. 131, pp. 282-292, June 2001.
- [11] L. P. Ouyang, X. H. Wang and J. M. Xiao, "Berth scheduling problem based on ant colony algorithm," *Control Engineering of China*, vol. 16, S1, pp.106-109, July 2009.



**Yun Cai** was born in Urumchi, China in 1970. She received Bachelor's degree of Mechanical Technology from Dalian railway institute, Dalian, China in 1991, Master's degree of Mechatronic Engineering from Xinjiang university, Xinjiang, China in 2002, and Doctor's degree of Mechanical Design and

Theory from Wuhan university of technology, Wuhan, China in 2005.

She has over five years experience teaching at Wuhan university of science and technology as an associate professor, Her major field of study is optimization and simulation technology. Currently she is engaged in solving scheduling problem in logistics systems.