Solving 0–1 Quadratic Knapsack Problems with a Population-based Artificial Fish Swarm Algorithm

Md. Abul Kalam Azad[1], **Ana Maria A.C. Rocha**[1,2] **and Edite M.G.P. Fernandes**[1]

[1] *Algoritmi R&D Centre, University of Minho, 4710-057 Braga, Portugal*
akazad@dps.uminho.pt; emgpf@dps.uminho.pt
[2] *Department of Production and Systems, University of Minho, 4710-057 Braga, Portugal* arocha@dps.uminho.pt

Extended Abstract

1 Introduction

The 0–1 Quadratic Knapsack Problem (QKP) consists in maximizing a quadratic objective function subject to a linear capacity constraint. This problem has been introduced by Gallo et al. [2] and may be expressed as follows:

$$\begin{aligned} &\text{maximize } f(\boldsymbol{x}) \equiv \sum_{i=1}^{n} p_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{ij} x_i x_j \\ &\text{subject to } \quad \sum_{i=1}^{n} a_i x_i \leq b \\ &\qquad\qquad\quad x_i \in \{0,1\}, \quad i = 1,2,\ldots,n, \end{aligned} \tag{1}$$

where the coefficients $p_i, a_i (i = 1,2,\ldots,n)$ and $p_{ij} (i = 1,2,\ldots,n-1, j = i+1,\ldots,n)$ are positive integers and b is an integer such that $\max\{a_i : i = 1,2,\ldots,n\} \leq b < \sum_{i=1}^{n} a_i$. Here p_i is a profit achieved if item i is selected and p_{ij} is a profit achieved if both items i and j $(j > i)$ are selected. The goal is to find a subset of n items that yields maximum profit f without exceeding capacity b.

The QKP arises in a variety of real world applications including finance, VLSI design, compiler construction, telecommunication, flexible manufacturing systems, locations, hydrological studies. Classical graph and hypergraph partitioning problems can also be formulated as the QKP. Several deterministic [1, 2] as well as stochastic solution methods [4, 8] have been proposed to solve (1).

Recently, a population-based artificial fish swarm algorithm that simulates the behavior of the fish swarm inside water was proposed [3, 6]. Applying to the optimization problem, generally a ‘fish’ represents an individual point in a population. Fishes desire to stay close to the swarm, to protect themselves from predators and to look for food, and to avoid collisions within the group. In this paper, we propose a binary version of the artificial fish swarm algorithm for solving (1).

2 A Binary Artificial Fish Swarm Algorithm

Here we will present the proposed binary version of the artificial fish swarm algorithm for solving (1) and simply denote it by bAFSA.

Initialization

The binary artificial fish swarm algorithm uses a population of N individual points (the fish swarm) $\boldsymbol{x}^i, i = 1, 2, \ldots, N$ to identify promising regions looking for a optimal solution. In bAFSA, N individual points represented by binary 0/1 string of length n are randomly initialized. An example is shown in Figure 1.

$$\boldsymbol{x}^{i,1} = \boxed{1\,|\,0\,|\,0\,|\,0\,|\,1\,|\,1\,|\,0\,|\,1\,|\,0\,|\,1\,|\,1\,|\,0\,|\,1\,|\,1\,|\,0}$$

Figure 1: Individual representation in bAFSA

However, the randomly initialized point $\boldsymbol{x}^i$ may not be feasible since the problem (1) has a constraint. The widely used approach to deal with constrained optimization problem is based on penalty functions. The performance of penalty-type method is not always satisfactory due to the choice of an appropriate penalty parameter, hence alternative techniques have been proposed. In bAFSA, the decoding algorithm proposed by Sakawa and Kato [7] is used in order to make $\boldsymbol{x}^i$ feasible.

The visual

After initializing N feasible individual points, the crucial issue of bAFSA is the visual of each individual point $\boldsymbol{x}^i$ that helps us to create a corresponding trial point $\boldsymbol{y}^i$. This represents a closed neighborhood of $\boldsymbol{x}^i$ with a radius equal to a positive quantity ν. A point is to be considered inside the visual of $\boldsymbol{x}^i$ if the distance of that point to $\boldsymbol{x}^i$ is within ν. The Hamming distance between two points is used to define the visual. Here we take $\nu = \delta \times n$, where $\delta \in (0, 1)$ and n (items) can be the maximum Hamming distance between two points.

Let np^i be the number of points inside the visual of point $\boldsymbol{x}^i$. Depending on the relative positions of the points in the population, three possible scenarios may occur: (i) the visual is empty if $np^i = 0$; (ii) the visual is not crowded if $np^i/N \leq \theta$; and (iii) the visual is crowded, otherwise. Here $\theta \in (0, 1)$ is the crowding parameter.

Fish Behavior

Depending on the crowding scenario of the visual, the point $\boldsymbol{x}^i$ performs different behavior. In bAFSA, the fish (point) behavior that create the trial points are outlined as follows.

Chasing behavior: If the visual is not crowded, the point $\boldsymbol{x}^i$ performs the chasing behavior. This behavior is related with a movement towards the point $\boldsymbol{x}^{\text{best}}$ inside the visual that has the best objective function value. The point $\boldsymbol{x}^i$ performs the chasing behavior if $f(\boldsymbol{x}^{\text{best}}) > f(\boldsymbol{x}^i)$. In chasing, the uniform crossover between $\boldsymbol{x}^i$ and $\boldsymbol{x}^{\text{best}}$ is performed to create the trial point $\boldsymbol{y}^i$.

Swarming behavior: This behavior of the point $\boldsymbol{x}^i$ is related with a movement towards the central point $\boldsymbol{x}^c$ inside the visual if it is not crowded and $f(\boldsymbol{x}^{\text{best}}) \leq$

$f(\boldsymbol{x}^i)$. In bAFSA, an individual point is represented by binary string of 0/1 bits, so the central point $\boldsymbol{x}^c$ inside the visual is calculated according to [5]. Then if $f(\boldsymbol{x}^c) > f(\boldsymbol{x}^i)$, the point $\boldsymbol{x}^i$ performs the swarming behavior. In swarming, the uniform crossover between $\boldsymbol{x}^i$ and $\boldsymbol{x}^c$ is performed to create the trial point $\boldsymbol{y}^i$.

Searching behavior: When the visual is crowded, or it is not crowded but the point $\boldsymbol{x}^i$ did not perform the chasing or the swarming behavior, the point performs the searching behavior. Here, a point $\boldsymbol{x}^{\text{rand}}$ inside the visual is randomly selected and the point $\boldsymbol{x}^i$ moves towards it if the condition $f(\boldsymbol{x}^{\text{rand}}) > f(\boldsymbol{x}^i)$ holds. In searching, the uniform crossover between $\boldsymbol{x}^i$ and $\boldsymbol{x}^{\text{rand}}$ is also performed to create the trial point $\boldsymbol{y}^i$.

Random behavior: When the visual is empty, or the other fish behavior are not performed, the point $\boldsymbol{x}^i$ performs the random behavior. This behavior is related with a random movement for a better region, and the trial point $\boldsymbol{y}^i$ is created by randomly setting binary 0/1 bits of length n.

Selection

After creating the N trial points $\boldsymbol{y}^{i,t+1}$, $i = 1, 2, \ldots, N$, the decoding algorithm is performed to make them feasible. In order to decide whether or not they should become members of the population in the next iteration $t+1$, the trial point $\boldsymbol{y}^{i,t+1}$ is compared to the current point $\boldsymbol{x}^{i,t}$ using the following greedy criterion:

$$\boldsymbol{x}^{i,t+1} = \begin{cases} \boldsymbol{y}^{i,t+1} & \text{if } f(\boldsymbol{y}^{i,t+1}) \geq f(\boldsymbol{x}^{i,t}) \\ \boldsymbol{x}^{i,t} & \text{otherwise} \end{cases}, \quad i = 1, 2, \ldots, N. \qquad (2)$$

Termination Condition

Let $T_{\max}$ be the maximum number of iterations. If $f_{\max}$ is the maximum objective function value attained at t and if f_{opt} is the known optimal value, then bAFSA terminates if $(t > T_{\max}$ or $(|f_{\max} - f_{\text{opt}}|) \leq \epsilon)$, for a small positive number ϵ.

The bAFSA

The algorithm of the herein proposed binary version of the artificial fish swarm algorithm for solving (1) is outlined.

Step 1: Set parameter values.
Step 2: Set $t = 1$. Randomly initialize $\boldsymbol{x}^{i,1}, i = 1, 2, \ldots, N$.
Step 3: Perform decoding and evaluate f. Identify $\boldsymbol{x}_{\max}$ and $f_{\max}$.
Step 4: If termination condition is met, stop.
Step 5: For all $\boldsymbol{x}^{i,t}$,
  Define visual and identify crowding scenario;
  Perform fish behavior to create trial point $\boldsymbol{y}^{i,t+1}$;
  Perform decoding to make the trial point feasible.
Step 6: Perform selection according to (2) to create new current points.
Step 7: Evaluate f and identify $\boldsymbol{x}_{\max}$ and $f_{\max}$.
Step 8: Set $t = t + 1$ and go to Step 4.

3 Preliminary Results

We code bAFSA in C and compile with Microsoft Visual Studio 9.0 compiler in a PC having 2.5 GHz Intel Core 2 Duo processor and 4 GB RAM. We set $N = 100$, $\delta = 0.5$, $\theta = 0.8$ and $\epsilon = 10^{-4}$. We also set $T_{\max} = 1000$ for $n \leq 50$, otherwise $T_{\max} = 2000$. Firstly, 11 QKP are considered to test the performance criteria of the proposed bAFSA. The data were generated randomly according to [1]. The profit coefficients of objective function were generated with density, $d = 1.00$. The density means the percentage of non-zeros in the profit coefficients. We solved these problems using MINLP[1] and compared the results with our solutions. The results obtained by MINLP and bAFSA are shown in Table 1. Thirty independent runs

Table 1: Results of 11 test problems obtained by MINLP and bAFSA

Prob.	n	MINLP solver		bAFSA								
				successful runs					among 30 runs			
		f_{opt}	FE	$f_{\max}$	AITsr	AFEsr	ATsr	Nsr	f_{avg}	AIT	AFE	AT
1	5	282	82	282	1	100	0.00	30	282.00	1	100	0.00
2	10	1224	162	1224	1	114	0.00	30	1224.00	1	114	0.00
3	20	5272	5793	5272	32	3212	0.07	30	5272.00	32	3212	0.07
4	30	1629	46	1629	6	577	0.02	30	1629.00	6	577	0.02
5	40	31083	3759	31083	47	4688	0.23	29	31082.00	79	7865	0.35
6	50	58588	2319	58588	40	4004	0.24	30	58588.00	40	4004	0.24
7	60	66286	8964	66286	550	54997	4.08	11	65706.40	1468	146852	10.74
8	70	91144	3184	91144	36	3554	0.34	26	91035.73	297	29750	2.69
9	80	85783	1408	85783	67	6661	0.80	20	85730.77	711	71116	7.97
10	90	183475	20298	183475	102	10218	1.16	24	183419.93	482	48179	5.10
11	100	99759	258400	99759	149	14902	2.05	11	99406.20	1321	132143	15.96

were carried out for each problem using bAFSA. In a run if the algorithm finds the optimal solution (or near optimal with a tolerance) of a test problem, then the run is considered to be a successful run. In the table,'f_{opt}' and 'FE', the number of function evaluations are obtained with MINLP solver. The performance criteria of bAFSA are: (i) for the successful runs – '$f_{\max}$'; the average number of iterations, 'AITsr'; the average number of function evaluations, 'AFEsr'; the average computational time (in seconds), 'ATsr' and the number of successful runs, 'Nsr'; (ii) among the 30 runs – the average of best objective function values obtained, f_{avg}; 'AIT', 'AFE' and 'AT' bear the same previously defined meanings but among the 30 runs. We remark that 'AFE' was computed based on the entire population.

Secondly, 20 benchmark QKP test problems[2] are considered. The results obtained by MINLP and bAFSA are shown in Table 2.

We may conclude from the results in Tables 1 and 2, that the herein proposed bAFSA is capable of solving 0–1 quadratic knapsack problems although some improvement in efficiency is still required.

[1] available at http://neos-server.org/neos/

[2] available at http://cedric.cnam.fr/∼soutif/QKP/

Table 2: Results of 20 test problems obtained by MINLP and bAFSA

Prob.	n	MINLP solver		bAFSA								
				successful runs					among 30 runs			
	(d)	f_{opt}	FE	$f_{\max}$	AITsr	AFEsr	ATsr	Nsr	f_{avg}	AIT	AFE	AT
1	100	18558	8364	18558	142	14183	2.51	6	18485.60	1628	162852	27.41
2	(0.25)	56525	706	56525	111	11062	1.47	18	56407.63	866	86645	10.14
3		3752	1713	3752	101	10050	1.30	6	3688.43	1620	162025	18.96
4		50382	61	50382	183	18301	2.47	12	50153.10	1273	127332	15.09
5		61494	5130	61494	39	3912	0.57	26	61466.40	301	30059	4.54
6	100	83742	7062	83742	137	13716	1.90	22	83643.43	634	63397	7.73
7	(0.50)	104856	10071	104856	139	13851	1.88	2	104604.03	1876	187608	22.23
8		34006	17786	34006	146	14564	2.05	11	33942.83	1320	132018	15.74
9		105996	157	105996	59	5937	0.91	27	105859.50	253	25345	3.86
10		56464	7478	56464	78	7815	1.35	15	56447.57	1039	103918	16.37
11	100	189137	7	189137	5	490	0.07	30	189137.00	5	490	0.07
12	(0.75)	95074	173495	95074	206	20643	2.81	5	94984.87	1701	170124	20.26
13		62098	3447	62098	84	8432	1.56	19	62042.80	787	78681	12.93
14		72245	301913	72245	104	10426	1.85	8	72116.53	1494	149461	24.30
15		27616	14023	27616	103	10288	1.39	16	27456.83	988	98829	11.88
16	100	81978	38198	81978	124	12391	1.75	11	81835.10	1312	131221	15.82
17	(1.00)	190424	37144	190424	121	12133	1.90	6	188728.63	1624	162442	24.81
18		225434	1590	225434	210	20968	2.66	6	223298.73	1642	164209	19.40
19		230076	1370	230076	247	24659	3.10	19	229244.37	890	88958	10.74
20		74358	13569	74358	143	14255	2.01	22	74256.80	638	63791	7.71

4 Conclusion

In this paper, a binary version of the artificial fish swarm algorithm for solving 0–1 quadratic knapsack problem has been presented. In this method a point is represented by a binary string of 0/1 bits. The visual of a point is defined using the Hamming distance. Depending on the number of points inside the visual, a point can perform either chasing, swarming, searching or random behavior. Crossover and mutation are implemented to create trial points. In order to make points feasible a decoding algorithm is also implemented. A greedy selection criterion is used to decide whether or not the trial points should be included in the population of the next iteration.

A set of 0–1 QKP were considered to test the performance of bAFSA. It has been shown that the proposed method is capable of solving those problems. Future development will focus on improving the algorithm efficiency and the comparison of the proposed method with the other solution methods available in literature.

Acknowledgments. The first author acknowledges Ciência 2007 of FCT – Fundação para a Ciência e a Tecnologia (Foundation for Science and Technology), Portugal for the financial support under fellowship grant: C2007-UMINHO-ALGORITMI-04. The second and third authors acknowledge FEDER COMPETE – Programa Operacional Fatores de Competitividade (Operational Programme Thematic Factors of Competitiveness) and FCT for the financial support under project grant: FCOMP-01-0124-FEDER-022674.

References

[1] A. Billionnet, Éric Soutif, An exact method based on Lagrangian decomposition for the 0–1 quadratic knapsack problem, *Eur. J. Oper. Res.*, **157**, pp. 565–575 (2004).

[2] G. Gallo, P. L. Hammer, B. Simeone, Quadratic knapsack problems, *Math. Program. Study*, **12**, pp. 132–149 (1980)

[3] M. Jiang, Y. Wang, S. Pfletschinger, M. A. Lagunas, D. Yuan, Optimal multiuser detection with artificial fish swarm algorithm, in: D.-S. Huang, L. Heutte, M. Loog (eds.), *Advanced Intelligent Computing Theories and Applications–ICIC 2007, Part 22, CCIS vol. 2*, pp. 1084–1093. Springer-Verlag (2007).

[4] B. A. Julstrom, Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem, in: *Proceedings of the GECCO'05*, pp. 607–614 (2005).

[5] A. Moraglio, C. G. Johnson, Geometric generalization of the Nedler-Mead algorithm, in: P. I. Cowling et al. (eds.) *EvoCOP2010, LNCS, vol. 6022*, pp. 190–201. Springer-Verlag (2010).

[6] A. M. A. C. Rocha, T. F. M. C. Martins, E. M. G. P. Fernandes, An augmented Lagrangian fish swarm based method for global optimization, *J. Comput. Appl. Math.*, **235**, pp. 4611–4620 (2011).

[7] M. Sakawa, K. Kato, Genetic algorithms with double strings for 0–1 programming problems, *Eur. J. Oper. Res.*, **144**, pp. 581–597 (2003).

[8] X. -F. Xie, J. Liu, A Mini-swarm for the quadratic knapsack problem, in: *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 190–197 (2007).