

# Learning about *FANS* behaviour: Knapsack Problems as a test case

**Armando Blanco**  
Depto. de C.C. e I.A  
Universidad de Granada  
18071 Granada, Spain  
armando@ugr.es

**David A. Pelta**  
Depto. de C.C. e I.A  
Universidad de Granada  
18071 Granada, Spain  
dpelta@ugr.es

**José-L. Verdegay**  
Depto. de C.C. e I.A  
Universidad de Granada  
18071 Granada, Spain  
verdegay@ugr.es

## Abstract

The Fuzzy Adaptive Neighborhood Search (*FANS*) method was previously presented and successfully compared against a genetic algorithm and simulated annealing doing minimization of functions in  $\mathcal{R}^n$ . Here we propose to evaluate *FANS* on instances of knapsack with multiple restrictions problems, making comparison against the cited algorithms in order to gain knowledge about the behaviour of our method.

## 1 Introduction

Within the framework of Decision Support Systems, simple general purpose optimization tools are key elements to decision makers because they enable them to obtain initial solutions with minimal knowledge of the problem being solved. In this way, such initial solutions may serve as a guide for further improvements.

Among those simple approaches are neighborhood or local search based heuristic techniques.

In [4, 3], we presented the main ideas of a Fuzzy Adaptive Neighborhood Search (*FANS*) heuristic. Its main motivation is to provide a general purpose optimization tool, suitable to be embedded in a decision support system, which is easy to tailor to specific problems by means of appropriate definition of its components.

*FANS* is termed Fuzzy, because the solutions are qualified in terms of *fuzzy valuations*, and

*adaptive*, because its behaviour is adapted as a response to the state of the search.

By means of a fuzzy valuation, represented in *FANS* by fuzzy sets, a fuzzy measure of the generated solutions is obtained. Fuzzy valuations may represent concepts like “Acceptability”, thus a degree of acceptability is calculated for each solution and such degrees are used by *FANS* at the decision stages.

The use of schedulers enable *FANS* to modify its behaviour as a function of the search state. In particular, the operator used to generate solutions is changed when the search seems trapped, leading to an intermediate escaping mechanism. When this mechanism fails, a classical restart operator is applied.

Being a heuristic, *FANS* needs some parameters and components to be defined. Suitable component definitions and parameters, lead *FANS* to reflect the behaviour of traditional techniques like HillClimbing, Random Walks, etc.

In [4, 3], we showed how *FANS* outperformed a genetic algorithm (*GA*) and simulated annealing (*SA*) over a set of real function minimization problems when the three algorithms were given a fixed amount of cost function evaluations. In those problems the only restriction was the range available for each variable, so it was easy to ensure that all the generated solutions were feasible.

Now, we propose to test *FANS* over Knapsack Problems with multiple restrictions in order to confirm the potential of *FANS* as a general purpose optimization technique. In this case, infeasible solutions will exist and the algorithm

must deal with them in some way.

We will not describe here the basics of fuzzy sets; the interested reader is referred to [5].

The paper is organized as follows: in Section 2 test problems are presented. Then in Section 3 the main components of *FANS* are defined, and the characteristics of the *GA* and *SA* implemented are shown. The experiments and results obtained are in Section 4. Finally, Section 5 is devoted to conclusions and further work.

## 2 Test Instances

Knapsack Problem with Multiple Restrictions is the more general class of knapsack problems. It is an integer linear problem with the following formulation:

$$\begin{aligned} \max \sum_{i=1}^n p_i * x_i \\ \text{s.t. } \sum_{i=1}^n w_{ij} * x_i \leq C_j \text{ with } j = 1..m \end{aligned}$$

where  $n$  is the number of items,  $m$  the number of restrictions,  $x_i \in \{0, 1\}$  indicates if the  $i$  item is included or not in the knapsack,  $p_j$  is the profit associated with the item  $i$ , and  $w_{ij} \in [0, \dots, r]$  is the weight of item  $i$  with respect to constraint  $j$ .

Also  $w_{ij} < C_j \forall i, j$  (every item alone fits in the knapsack) and  $\sum_{i=1}^n w_{ij} > C_j \forall j$  (the whole set of items don't fit in the knapsack).

For our experiments we select 9 instances from a standard set of 55 problems available from [1]. The instances are named *pb5*, *pb7*, *Weing1*, *Weing3*, *Weing7*, *Weish10*, *Weish14*, *Weish18*, *Weish27*. The number of variables range from 20 to 105 with 2 to 30 restrictions.

## 3 Algorithms

In this section we present the main components of *FANS*, together with their interaction. Also, design decisions for the *SA* and *GA* implementations are later shown.

Knapsack solutions are represented by binary vectors  $X$ , where position  $i$  represents the variable  $x_i$ . This representation is used in *FANS*, *SA* and *GA*.

In our experiments, infeasible solutions will not be taken into account and they will be discarded. No reparation procedure will be used.

Initial solutions for each method contains just a unique position in 1.

### 3.1 *FANS* Components

*FANS* operation relies on four main components: an operator, to construct new solutions; a fuzzy valuation, to qualify them; an operator scheduler, to adapt the operator's behaviour; and a neighborhood scheduler, to generate and select a new solution.

In order to apply *FANS* to a particular problem, these components must be defined. As usual, when more problem-dependant definitions are used, the better the performance. We decide to use simple definitions in order to test the quality of the search strategy induced by *FANS*. Below, we describe the definition proposed for each component.

**Modification Operator *k*-BitFlip:** randomly chooses  $k$  positions and flips the associated bit value. "Back mutation" is not allowed.

**Fuzzy Valuation: Acceptable:** the generated solutions will be qualified in terms of "acceptability", a concept reflecting the following idea: with a solution at hand, those generated solutions improving the current cost, will have a higher degree of acceptability than those with lower cost. Solutions diminishing the cost a little, will be considered "acceptable" but with lower degree. Finally those solutions demeliorating too much the current cost will not be considered as acceptable.

So, given the objective function  $f$ , the current solution  $s$ ,  $q$  a neighbor solution, and  $\beta$  the limit for what is considered as acceptable, the following definition comprise those ideas:

$$\mu_a(q, s, \beta) = \begin{cases} 0.0 & \text{if } f(q) < \beta \\ \frac{(f(q)-\beta)}{(f(s)-\beta)} & \text{if } \beta \leq f(q) \leq f(s) \\ 1.0 & \text{if } f(q) > f(s) \end{cases}$$

We used  $\beta = f(s) * (1 - \gamma)$ , and  $\gamma \in [0..1]$ . For our experiments, we set  $\gamma = 0.05$ .

**Operator Scheduler:** the  $k$ -BitFlip operator will be adapted through changes on the  $k$  parameter. The used scheme is rather simple: being  $k_t$  the actual value, then  $k_{t+1}$  will be a random integer value in  $[1, 2 * k_t]$ . Also, if  $k_{t+1} > top = \frac{n}{10}$ , then  $k_{t+1} = top$ .

**Neighborhood Scheduler:** given the cost function  $f$ , the operator  $\mathcal{O}$ , the fuzzy valuation  $\mu$  and the current solution  $s$ , we define the operational neighborhood of  $s$  as  $N(s) = \{\hat{x} | \hat{x} = \mathcal{O}(s)\}$  and the “semantic” neighborhood of  $s$  as  $\hat{N}(s) = \{\hat{x} | \mu(f(\hat{x})) > \lambda; \hat{x} \in N(s)\}$

Taking into account both definitions, we provide two neighborhood schedulers.

**Quality Based Grouping Scheme  $R|S|T$ :** it tries to generate  $R$  “Acceptable” neighborhood solutions in  $maxTrials$  trials, then those solutions are grouped into  $S$  sets on the basis of their acceptability degree, and finally  $T$  solutions are returned [4].

We used a 5|3|1 scheme with  $maxTrials = 12$ .

The  $S = 3$  sets or clusters are represented by overlapped triangular membership functions with boundaries adjusted to fit the range  $[\lambda, 1.0]$ , being  $\lambda = 0.99$  the minimum level of acceptability required. The sets represent the terms *Low*, *Medium*, *High* for the linguistic variable “Quality”. At the end of the process,  $T = 1$  solution must be returned. Here, we choose to return any solution of the highest quality available.

**First Found Scheme:** at most  $maxTrials = 12$  trials are available to obtain a solution  $\hat{x} \in \hat{N}(x)$ . The first one found is returned.

### 3.1.1 Components Interaction

Fig. 1 shows *FANS* pseudo code. The iterations end when some external condition holds. The neighborhood scheduler *NS* is called at the beginning of each iteration, with parameters: current solution  $S_{cur}$ ; fuzzy valuation  $\mu()$  and modification operator  $\mathcal{O}$ . Two situations may occur: an “acceptable” neighborhood solution  $S_{new}$  was found or not.

In the first case  $S_{new}$  is taken as the current solution and  $\mu()$  parameters are adapted. In this

```

Procedure FANS:
Begin
  While ( not-finalization ) Do
    NS->Run( $\mathcal{O}, \mu(), S_{cur}, S_{new}, ok$ );
    If (ok) Then
       $S_{cur} = S_{new}$ ;
      adaptFuzzyVal( $\mu(), S_{cur}$ );
    Else
      OS->Run(Oper);
    Fi
    If (trappedCondition()) Then
      doRestart();
    Fi
  Od
End.

```

Figure 1: *FANS* Pseudocode

way, we are varying our notion of “Acceptability” as a function of the context.

If *NS* could not return any acceptable solution, an exception condition ( $OK = false$ ) is raised. No solutions were acceptable in the neighborhood induced by the operator. In this case, the operator scheduler *OS* is executed, returning a modified version of  $\mathcal{O}$ . The next time *NS* will have a modified operator to search for solutions.

The *trappedCondition()* exception is raised when *Top* iterations were done without improvements in the best solution found. In this case, the *doRestart()* procedure is executed applying a perturbation operation over the current solution:  $\frac{1}{3}$  of randomly chosen variables in one are set to zero. Then, the cost of the current solution is reevaluated and  $\mu()$  is adapted. Then the process is restarted.

## 3.2 Genetic Algorithm and Simulated Annealing

The *GA* used may be regarded as “traditional”.

Mutation is applied to all individuals with certain probability. As mutation operator, the  $k$ -BitFlip operator is used with  $k = 2$ .

If the solution obtained after mutation is infeasible, it is discarded and at most four more trials are done to obtain a feasible one. If no feasible solution was obtained, the original one is kept.

As crossover operator, two classical ones are

implemented: 1 point and uniform crossover. In this way we obtain 2 algorithms: *GAop* y *GAux* respectively. Elitism is also used in both versions.

Other parameters are population size  $PopSize = 100$ ; crossover and mutation probabilities  $P_{crossover} = 0.8, P_{mut} = 0.2$ , and Tournament Selection with tournament size  $q = 2$  within a  $(\mu = 50 + \lambda = 75)$  scheme.

Our implementation of *SA* is simple and follows the guidelines presented in [2]. The *k*-BitFlip operator is also used with  $k = 2$ . The initial temperature was  $T_0 = 5$  and proportional cooling is used with  $T_{k+1} = T_k * \alpha$  with  $\alpha = 0.9$ . Temperature is adapted when 15 neighbors were accepted, or when  $2 * n$  neighbors were generated, being  $n$  the dimension of the problem.

The values for the parameters were empirically determined after a reduced set of experiments.

## 4 Experiments and Results

In order to analyze the performance of *FANS*, we conduct a set of experiments and made comparisons among the following five algorithms:

$F_{rst}$ , is *FANS* with scheduler *R|S|T*;  $F_{ff}$ , is *FANS* with scheduler *FirstFound*; *GAux*, is the *GA* with uniform crossover; *GAop*, is the *GA* with one-point crossover; and *SA*, the Simulated Annealing implementation.

We want to compare the performances of the algorithms under none or minimal knowledge of the problem (reflected by the use of very simple operators), and when they are given a fixed number of resources (i.e cost function evaluations and number of generated solutions). In this way, we can assume the results are consequence of the search strategies and not of additional knowledge.

For each problem and algorithm 30 runs were made; each one ending when  $maxEvals = 15000$  cost function evaluations were done or when  $maxEvals * 4$  solutions were generated. This limit is needed because only feasible solutions are evaluated.

The results are analyzed in terms of the error for each problem and globally over the whole test set. The first results are presented on Table 1, where

part (a) shows the mean of the errors over 30 runs for each algorithm on each problem. The error is calculated as:

$$error = 100 * \frac{Optimum - ObtainedValue}{Optimum}$$

The Table shows that both versions of *FANS* achieved the lower values, except for problems Weing3, Weing7 y Weish27. *SA* achieved the better value on Weing7 and *GAux* did it on the other two problems.

On part (b) of Table 1, an *X* indicate if the algorithm on the column reached the optimum of the problem on the row in any of the 30 runs. We can see that  $F_{rst}$ ,  $F_{ff}$  and *GAux* obtained the optimum on 6 of 9 problems, while *GAop* on 5 of 9 and *SA* just in 1 of 9 (this optimum was reached by all algorithms).

It is hard to determine why *FANS* failed to achieve the optimums for those problems, because it is not clear what makes an instance easy or hard. One aspect is the correlation between profits and weights, which is well established for classical knapsack problems but not for multiple restrictions knapsack problems. Other elements are needed but this discussion is out of the aim of this work.

The last row (# Opt) in Table 1 (b), indicates the number of executions ending at the optimum for a total of  $9 * 30 = 270$  runs.  $F_{rst}$  y  $F_{ff}$  achieves the higher values followed by *GAux*. *SA* and *GAop* are quite ineffective from this point of view.

To conclude, taking the mean and variance of the errors over the whole set of problems (results not shown) it was clear that both version of *FANS* achieved the lowest values, followed by *GAux*. The mean error in *GAux* y *GAop* was almost twice of that in  $F_{rst}$  y  $F_{ff}$ ; and *SA* values were 5 times higher.

In order to confirm if mean error differences were of statistical significance, t-tests were done with a confidence level of 95%. The results enabled us to confirm that both versions of *FANS* outperformed *GAux*, *GAop* y *SA*. Both *GA* outperformed *SA*, and no significative differences were found among them, in spite of the apparent superiority of *GAux*.

Table 1: Results for each problem. On (a), mean error, for each problem and algorithm. On (b), an  $x$  means the algorithm of the column reached the optimum of the problem on the row on any run.

	$F_{rst}$	$F_{ff}$	$SA$	$GAop$	$GAux$
pb5	1.04	0.92	6.52	3.37	3.07
pb7	0.94	1.19	4.13	3.83	4.23
weing1	0.20	0.19	8.07	0.92	1.37
weing3	1.54	1.32	22.04	1.85	0.91
weing7	0.50	0.51	0.48	1.13	0.93
weish10	0.27	0.14	1.22	1.34	1.18
weish14	0.85	0.78	1.93	1.85	0.91
weish18	0.73	0.71	1.39	0.95	0.89
weish27	3.02	2.89	2.85	3.21	1.18

(a)

	$F_{rst}$	$F_{ff}$	$SA$	$GAop$	$GAux$
pb5	x	x			
pb7	x	x			
weing1	x	x		x	x
weing3				x	x
weing7					
weish10	x	x	x	x	x
weish14	x	x		x	x
weish18	x	x		x	x
weish27					x
# Opt.	39	39	9	17	31

(b)

## Acknowledgements

Research supported in part by Projects PB98-1305 and TIC 99-0563. David Pelta is a grant holder from Consejo Nac. de Invest. Cientificas y Técnicas (Argentina).

## 5 Conclusions

In this work we gained evidence about the suitability of *FANS* as a general purpose optimization tool.

Although our intention was not to develop a tool to deal with knapsack problems but to use these problems to learn about *FANS* behaviour, the results show the good performance of *FANS* over this problem in spite of the simple component definitions used.

It seems clear that using specific problem elements, like a repairing scheme for infeasible solutions, the current results may be improved, making *FANS* a valid tool to deal with multiple restriction knapsack problems.

Experiments are being done in order to apply *FANS* to problems arising in computational molecular biology, where the operator's knowledge is very important to assess the quality of the results.

In order to make *FANS* available, a distribution version is in preparation. A preliminary version may be requested from *dpelta@ugr.es*.

## References

- [1] J. Beasley. The or-library: a collection of test data sets. Technical report, Management School, Imperial College, London SW7 2AZ., 1997. <http://mscmga.ms.ic.ac.uk/info.html>.
- [2] A. Diaz, F. Glover, H. Ghaziri, J. Gonzalez, M. Laguna, P. Moscato, and F. Tseng. *Heuristic Optimization and Neural Nets*. Ed. Paraninfo, 1996. In Spanish.
- [3] D. Pelta, A. Blanco, and J. L. Verdegay. A fuzzy adaptive neighborhood search for function optimization. In *4th Int. Conf. on Knowledge-Based Intelligent Engineering Systems & Allied Technologies, KES 2000*, vol 2, pages 594–597, 2000.
- [4] D. Pelta, A. Blanco, and J. L. Verdegay. Introducing fans: a fuzzy adaptive neighborhood search. In *8th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU 2000*, vol 3, pages 1349–1355, 2000.
- [5] H. J. Zimmermann. *Fuzzy Sets Theory and Its Applications*. Kluwer Academic, 1996.