An Experimental Study on Ant Colony Optimization Hyper-heuristics for Solving the Knapsack Problem

Bronson Duhart, Fernando Camarena, José Carlos Ortiz-Bayliss, Ivan Amaya and Hugo Terashima-Marín

Tecnologico de Monterrey
Escuela de Ingeniería y Ciencias
b_duhart@live.com.mx, fernando@camarenat.com,
{jcobayliss, iamaya2, terashima}@itesm.mx

Abstract. The knapsack problem is a fundamental problem that has been extensively studied in combinatorial optimization. The reason is that such a problem has many practical applications. Several solution techniques have been proposed in the past, but their performance is usually limited by the complexity of the problem. Hence, this paper studies a novel hyper-heuristic approach based on the ant colony optimization algorithm to solve the knapsack problem. The hyper-heuristic is used to produce rules that decide which heuristic to apply given the current problem state of the instance being solved. We test the hyper-heuristic model on sets with a variety of knapsack problem instances. Our resulting data seems promising.

1 Introduction

Imagine you are given a bag in a store. The owner says you are free to grab as many items as you wish, as long as they all fit together in the bag. Each product in the store has a specific value and weight. But, the bag has limited capacity (represented by the maximum total weight it can hold). Obviously, you would want to pack the products that maximize the overall monetary value. How would you decide such a combination of items? This is a very simple example of what the literature refers to as the knapsack problem.

The knapsack problem is nothing but a simplification of more complicated real-life optimization problems. But, they all have a distinctive feature: Select an item subset that maximizes the total profit. Of course, there is an imposed constraint on the items that can be selected. Also, each item within the set has an associated weight and profit.

Given a set of n items, 2^n possible subsets can be formed. Therefore, an enumerating approach is unpractical for most cases. In fact, this is one of the features that make the knapsack problem a well known NP-complete problem. This means that no exact solving method has been found that runs in polynomial time. Moreover, should one ever be found, it could also be used to solve any other NP problem [1, 2]. Hence, its importance.

Exact methods guarantee finding the optimal solution. But, they require that a solution exists and that enough run time is given. Dynamic programming and branch and bound are included in this category. Unfortunately, these methods can only solve small instances because of the exponential growth in the solution space.

Aside from exact methods, literature also describes approximated ones. These are known as heuristics. With them, finding the optimal solution is not guaranteed. But, it is feasible to find one acceptable enough, according to some specific performance metric. Heuristics select the next item to pack following the best evaluation of one or more criterion. Despite their success, some major drawbacks prevent heuristics from becoming the definite solver. First, they behave inconsistently across a wide range of instances. Thus, a heuristic that performs exceptionally well on some instances can be easily outperformed by others on different ones. Hence, the performance of heuristics changes from instance to instance. Because of that, there is no single heuristic that best solves all instances of the knapsack problem.

A potential solution to this problem is to try and learn the patterns that characterize specific 'classes' of instances. This knowledge can then be applied for automatically switching heuristics as the search progresses, attempting to improve the solution process. Such an approach is known as a hyper-heuristic. The idea is to find a set of rules for deciding when to apply each particular heuristic. Hence, they are usually referred to as "heuristics to choose heuristics" [3]. In this work, we propose using the ant colony optimization (ACO) algorithm to search the space of heuristics. We strive to find a combination that remains steadily competent on a wider range of instances.

The remainder of this document is organized as follows. Section 2 presents a review of some important concepts related to this work. The solution model proposed herein is described in Sect. 3. Section 4 presents the experiments we conducted, their results and analysis. Finally, Sect. 5 presents the conclusion and future work derived from this investigation.

2 Background and Related Work

In this work, we aim at exploring how ACO can be used to produce hyper-heuristics for solving the knapsack problem. The ACO algorithm is a meta-heuristic based on swarm intelligence. Such swarm intelligence relates to the fact that there is no centralized control of the search. Instead, the algorithm discovers the best solutions in a way similar to ants finding the most convenient paths to food sources in nature [4]. In the past, ACO has been already used to solve the knapsack problem in its different variants [5–7, 1, 8, 9]. Some works have even combined ACO with other techniques, such as fuzzy logic [10].

Regarding hyper-heuristics, ACO was used as a hyper-heuristic on the traveling salesman problem (TSP) and compared against other seven meta-heuristics that operated at the problem level [11]. The results presented in that work show that ACO, when used within a hyper-heuristic, generally achieves a bet-

ter performance than isolated heuristics. Other problem domains where ACO has been used with a hyper-heuristic include the traveling tournament problem (TTP) [12], the set covering problem (SCP) [13] and the 2D bin packing problem (2DBPP) [14]. In all the cases, these hyper-heuristics have obtained competent results.

2.1 Heuristics for the Knapsack Problem

In this work, we have taken four commonly used heuristic from literature. Each one of them selects an item following a particular criterion, whilst only three focus on a greedy strategy. They can be briefly described as:

Default (DF). Items are packed in the same order they are initially presented in the problem.

Maximum profit (MP). Items are sorted decreasingly by their profit. Afterwards, MP begins packing objects in this order as long as there is space for them. This strategy attempts to gain as much profit as possible and as quickly as it can, by going for the most valuable elements first.

Minimum weight (MW). Items are sorted increasingly by their weight. Then, items are selected one by one until filling the knapsack. The rationale behind this heuristic is that taking more items results in higher profits.

Maximum profit per weight (MPW). Items are sorted decreasingly by the quotient of profit over weight.

3 Solution model

In this section, we proceed to explain our solution. First, we present how we defined a suitable representation for the problem state, which makes it possible to extract knowledge about the best heuristics for a particular instance. Afterwards, we describe the ACO algorithm that we applied for the generation of hyper-heuristics in the form of state-action rules.

3.1 Problem Representation

We closely followed the selection of features used in [15], due to the similarity of the problems. Our representation describes the state of the items available to pick from, as well as the state of the knapsack, since some problems may share the same set of items and only differ in their knapsacks. The representation we define also takes into account the progress of a solution.

In this way, the eight features considered for our proposal are the following: normalized median weight of the objects ($\tilde{w}/\max w_i$), normalized mean weight of the objects ($\bar{w}/\max w_i$), normalized standard deviation of the weight of the objects ($\sigma_w/\max w_i$), normalized median profit of the objects ($\tilde{p}/\max p_i$), normalized mean profit of the objects ($\bar{p}/\max p_i$), normalized standard deviation of the profit of the objects ($\sigma_p/\max p_i$), normalized pearson correlation between

the profit and the weight of the objects ($\text{corr}(w,p)/2 + 0.5$) and normalized remaining capacity of the knapsack ($\frac{C-\sum w_i x_i}{C}$), where $\tilde{w}$, $\bar{w}$, $\tilde{p}$ and $\bar{p}$ denote the mean weight, median weight, mean profit and median profit, respectively. All features were normalized to the interval $[0, 1]$ to set a uniform scale for the training phase.

3.2 The Hyper-heuristic Generation Algorithm

The solution we developed is based on the original ACO algorithm [4]. The general idea is to gradually build a population of candidate hyper-heuristics (the 'solutions' found by the ants) and select the best among them. At the beginning of each iteration, each ant is assigned a problem instance from the training set. Then, each ant selects one of the available heuristics and applies such a heuristic to solve its respective problem instance, which results in a new point in the problem space according to our representation (defined by the eight features described in Sect. 3.1). Each iteration ends when all of the ants have finished solving their respective problem instances. When one iteration ends, the pheromone trace is updated, proportional to the quality of the solution found by each ant. In this way, each ant constructs a set of state-heuristic pairs by the end of each iteration. Such set of rules constitutes a potential hyper-heuristic. Thus, the result of the process, when the termination criterion is met, is the hyper-heuristic that performed the best across all the iterations, among all ants. This hyper-heuristic can then be applied to both seen and unseen instances.

The complete hyper-heuristic generation algorithm is shown in Alg. 1. It should be noted that the sampling of problems for initializing the ants is done with repetition and that we discretized the problem space through multiplying every feature by 10 and dropping the fractional part. In order to measure the quality of the solutions obtained by the ants, we employed the normalized profit of the knapsack, relative to the original total profit of the items: $\hat{P} = \sum p_i x_i / \sum p_i$.

Regarding the specific parameters for the ACO algorithm, we can mention the following. The probability of selecting a heuristic H to apply in state s is given by Eq. 1.

$$p(s, H) = \frac{(\tau_{s,H})^\alpha (\eta_{s,H})^\beta}{\sum_h (\tau_{s,h})^\alpha (\eta_{s,h})^\beta} \tag{1}$$

where $\tau_{s,h}$ is the pheromone level of the edge (s, h), and $\eta_{s,h}$ is the attractiveness of this edge, that corresponds to applying heuristic h to the problem at state s. The attractiveness is a user defined function, which in our case was chosen to be the same as the solution evaluation, i.e. the normalized profit that we would obtain if the problem was solved by only using h to solve it from state s onward. If the result returned by some heuristic at a given state is that no object can be packed, then we set $\eta_{s,h} = 0$, which gives zero probability of selecting it.

For parameters α and β, which balance the importance given to exploration and exploitation of solutions, we tried several different values but the ones suggested in literature provided the best results [16, 17]. We also found $Q = 10$

repeat
  Randomly assign a problem from the training set to k ants
  $i \leftarrow 0$
  while some ant can pack more items **do**
    for all ants **do**
      $s_i \leftarrow$ state representation of the problem
      Solve the problem with each heuristic and compute $\eta_{s_i,H}$
      Compute $p(s_i, H)$ for all heuristics
      $h^* \leftarrow$ heuristic with the highest probability $p(s_i, H)$
      Add the edge (s_i, h^*) to the ant's path and apply h^* to the problem
    end for
  end while
  $i \leftarrow i + 1$
  Update pheromone levels of all edges (s, h) accordingly
  Save the hyper-heuristic (path) with the highest profit
until a termination criterion is met
return the best hyper-heuristic (best path)

Fig. 1. Hyper-heuristic generation process through ACO.

and $\tau_{s,h}(0) = 1$ to be suitable values for the purpose of generating a hyper-heuristic with the desired features. The number of ants was selected as to have a probability of selecting a problem for solution construction equal to 80%.

The pheromone for the next iteration is updated with the expression: $\tau_{s,h}(i+1) = \rho\tau_{s,h}(i) + \Delta\tau_{s,h}$, where $\Delta\tau_{s,h}$ is the total amount of pheromone left by all ants that walked over edge (s, h). The pheromone update of ant k is obtained from a pheromone supply Q and the normalized profit $\hat{P}$ according to Eq. 2

$$\Delta\tau^k_{s,h} = \begin{cases} \hat{P}Q & (s,h) \text{ is in the ant's path} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

4 Experiments and Results

In order to evaluate the performance of our proposed strategies, we have classified a total of 1000 problem instances under four different instance sets. The sets S1 and S2 correspond to synthetic instances, while problems in P20 and P50 correspond to a subset of hard instances described by Pisinger [18]. S1, S2 and P50 contain instances with 50 items, while P20 contains instances with 20 items. The knapsack capacity for sets S1 and S2 is fixed to 50, while in P20 and P50 the capacity varies from instance to instance. In the case of synthetic problems —S1 and S2—, the sets were assembled in such a way that they contained a balanced number of favorable instances for each individual heuristic. Therefore, DF was the best performer on 22.25% of the instances; MP, on 25%; MPW, on 28.5%, and MW, on 25.25%.

For training, we designed two different schemes. In the first of them, we used set S1 as training set for the hyper-heuristics. On the second scheme, we

also wanted to evaluate the effect of extending training with instances from other sources. Therefore, we added 60% of P20 and of P50 to the training set. In both schemes, the test set was composed by S2 and the remaining 40% of P20 and P50. Thus, two hyper-heuristics were produced and tested: ACO-HHS and ACO-HHSP (hyper-heuristics trained with the first and the second scheme, respectively).

Since the ACO-based hyper-heuristic incorporates stochastic components in the decision phase, we averaged the profits of the model over various runs. Because the number of different outcomes was finite and small, with little variance among the respective profits, we used 10 repetitions of the testing phase to provide an adequate trade off between uncertainty and calculation time. In the following section, we only report the results that correspond to the best performing ACO-HH and ACO-HHSP out of the five trained for each training scheme.

4.1 Analysis of Hyper-heuristics

The first part of our analysis involves the evaluation of the quality of the solutions obtained by all the methods considered for this investigation. To compare the methods, we define four metrics. The first two metrics are related to the profit error (the difference between the method's profit and a profit reference value). We used two different reference values to obtain the first two metrics: (1) E_B, the difference between the solution of the method and the best known solution from any of the methods under study (four heuristics and two hyper-heuristics) and E_O, (2) the difference between the solution of the method and the optimal solution (obtained in this case by using dynamic programming). The third and fourth metrics, the success rates, are defined as the percentage of instances where each method obtains a profit error equal to zero. Following the rationale behind the first two metrics, two versions are also obtained for the success rate: (1) SR_B, the percentage of instances where the method is as good as the best known solution from any of the methods under study and SR_O, (2) the percentage of instances where the method finds the optimal solution. The results of this comparison are shown in Table 1.

Based on the results from Table 1 we can observe that the individual results of the heuristics degrade due to the results obtained by the hyper-heuristics. We also observe that ACO-HHSP obtains better success rates than any heuristic. Regarding the learning strategies, we can state two important claims. First, training with a mixture of instances improves the performance of ACO-based hyper-heuristics, as we can observe in Table 1.

4.2 Selection of Heuristics throughout the Search

Aiming at a better understanding of the behavior of the ACO-based hyper-heuristics proposed, we analyzed the frequency of use of each particular heuristic across different stages of the solution process. This information is shown in Fig. 2, 3 and 4. Based on these results, we can explain why ACO-HHS behaved so poorly in comparison to ACO-HHSP. In the three sets, it indistinctly selected

Table 1. Analysis of heuristics and hyper-heuristics with respect to the optimal and best solutions. Best results for each set and metric are highlighted in bold.

Set	Method	SR_B (%)	$\bar{E_B}$	SR_O (%)	$\bar{E_O}$
S2	DF	19.25	593.41	5.75	598.81
	MP	24.75	336.44	22.25	341.84
	MPW	27.75	**28.25**	19.75	**33.65**
	MW	25.00	255.75	12.50	261.15
	ACO-HHS	18.25	59.05	11.25	64.45
	ACO-HHSP	**34.50**	111.92	**28.25**	117.32
P20	DF	16.66	333.60	2.50	424.45
	MP	27.08	536.28	**14.16**	627.13
	MPW	**36.25**	158.64	12.50	249.49
	MW	15.00	420.85	3.33	511.70
	ACO-HHS	26.25	196.49	9.58	287.34
	ACO-HHSP	35.00	**157.19**	12.08	**248.04**
P50	DF	9.58	803.95	1.66	913.96
	MP	18.75	1658.30	9.58	1768.32
	MPW	49.16	108.98	**22.08**	219.00
	MW	16.66	574.68	2.08	684.69
	ACO-HHS	26.25	430.20	15.41	540.21
	ACO-HHSP	**51.25**	**108.24**	20.00	**218.25**

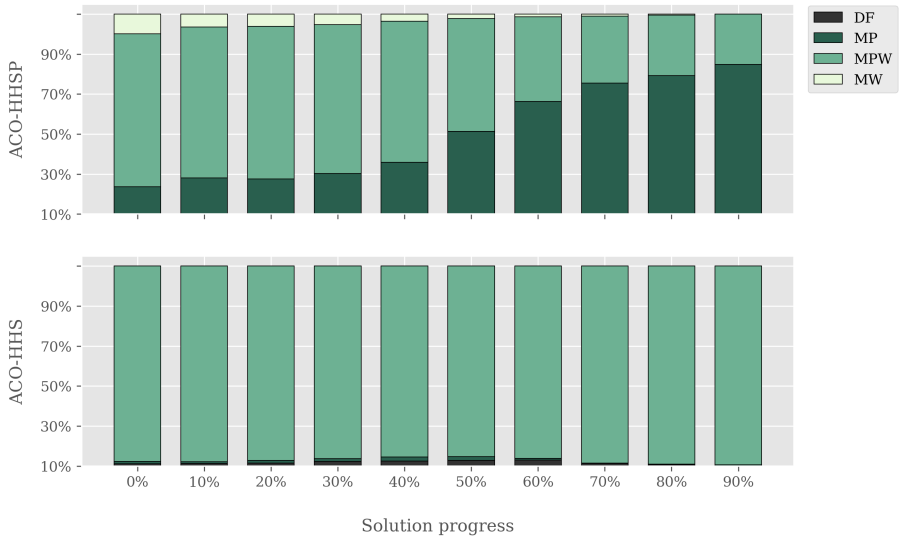

Fig. 2. Average heuristic selection through the different stages of solutions for set S2.

MPW, with a slight preference for MP. ACO-HHSP shows a peculiar behavior. In set S2, it starts using MPW most of the times and it progressively moves

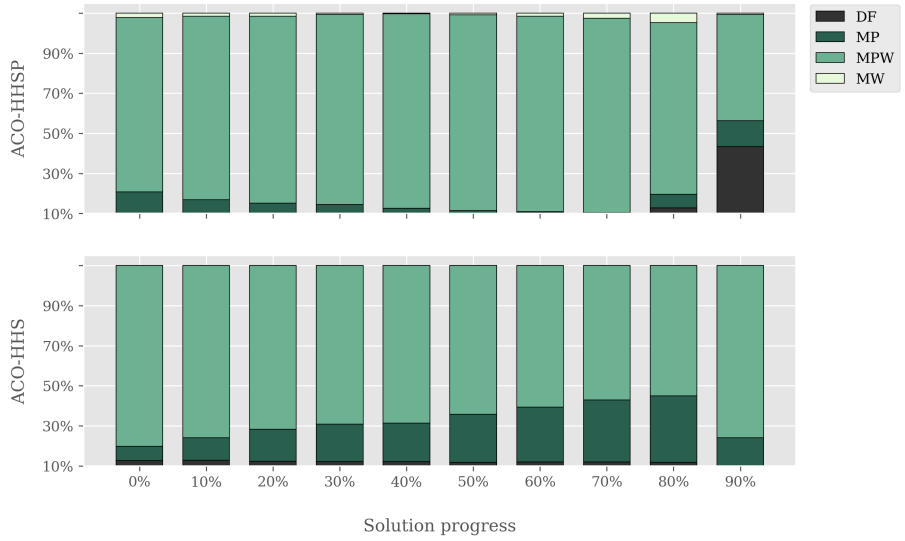

Fig. 3. Average heuristic selection through the different stages of solutions for set P20.

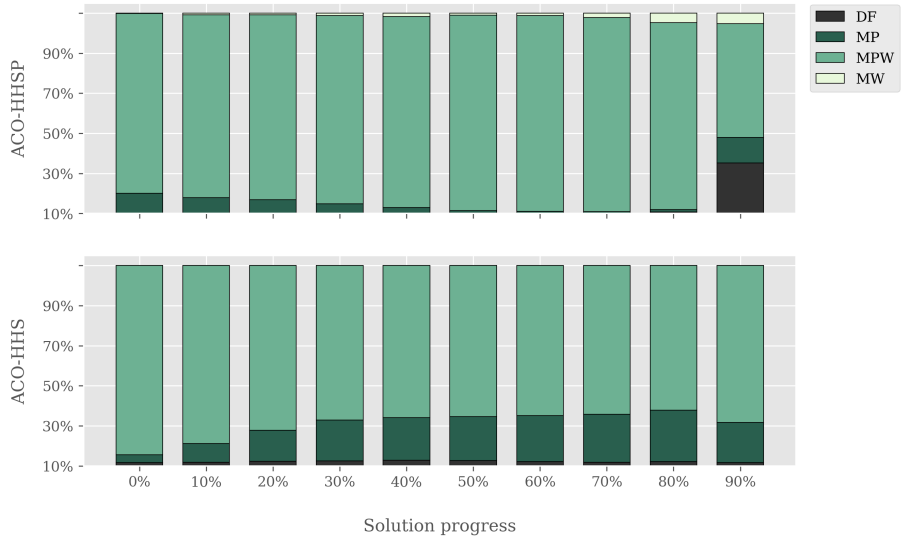

Fig. 4. Average heuristic selection through the different stages of solutions for set P50.

towards using MP on a regular basis. Then, ACO-HHSP is greedy on the profit only at the end of the search. Interestingly, this strategy changed for sets P20 anf P50. In these sets, ACO-HHSP was very consistent through different stages, by selecting MPW for packing most of the items. From the analysis of individual

heuristics we observed that MPW outperforms the other three heuristics by a large margin, especially in set P50. There is, however, a noticeable change in the strategy of ACO-HHSP in the last steps of the search, since it applies DF with more frequency than in previous stages. This is something that really amazes us, since we have already stated how naïvely DF behaves. Nonetheless, it appears that this sudden variety introduced by ACO-HHSP helped it become the overall best strategy for set P50.

5 Conclusion and Future Work

Throughout this work we used the Ant Colony Optimization (ACO) algorithm to propose a hyper-heuristic model that overcomes the drawbacks of using a single heuristic. By analyzing its performance on the knapsack problem, we found it to be a good approach. The main reason for its effectiveness comes from the varied performance of heuristics. The behavior of heuristics is linked to specific classes of instances, defined by the problem inherent features. By deriving a state representation of the problem, we were able to identify distinctive features of the problem. This allowed the hyper-heuristic to discern between different states of a problem, so that it selected the method that best suited the current problem. Hence, an ACO-trained hyper-heuristic generalizes better than simple heuristics. Even so, the MPW heuristic is very difficult to outperform since it yields near optimal results. But, there are some instances where it fails. Our data show that using hyper-heuristics is still better than using MPW. This is specially true for the ACO-trained hyper-heuristic, since it is fairly stable, and performs equally, or even better, than the heuristics.

There still remains paths to explore. For example, ACO tends to mimic MPW and does not tend to explore. Moreover, we consider that a more careful choice of features could yield better state representations. Such idea could be explored through statistical analysis. It would also be interesting to explore different hyper-heuristic models. Finally, the sensitivity of ACO parameters should also be analyzed.

Acknowledgements

This research was partially supported by CONACyT Basic Science Projects under grants 241461, 221551 and 287479, and ITESM Research Group with Strategic Focus in Intelligent Systems.

References

1. Schiff, K.: Ant colony optimization algorithm for the 0-1 knapsack problem. Technical Transactions. Automatic Control **R 110**(AC-3) (2013) 39–52
2. Sahni, S.: Approximate algorithms for the 0/1 knapsack problem. J. ACM **22**(1) (1975) 115–124

3. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society **64**(12) (Dec 2013) 1695–1724
4. Dorigo, M., Stützle, T.: Ant Colony Optimization. Bradford Company, Scituate, MA, USA (2004)
5. Zhang, J.: Comparative study of several intelligent algorithms for knapsack problem. Procedia Environmental Sciences **11** (2011) 163 – 168
6. Ke, L., Feng, Z., Ren, Z., Wei, X.: An ant colony optimization approach for the multidimensional knapsack problem. Journal of Heuristics **16**(1) (2010) 65–83
7. Ren, Z., Feng, Z.: An ant colony optimization approach to the multiple-choice multidimensional knapsack problem. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. GECCO '10, ACM (2010) 281–288
8. Du, D.p., Zu, Y.r.: Greedy strategy based self-adaption ant colony algorithm for 0/1 knapsack problem. In Park, J.J.J.H., Pan, Y., Chao, H.C., Yi, G., eds.: Ubiquitous Computing Application and Wireless Sensor, Springer Netherlands (2015) 663–670
9. He, L., Huang, Y.: Research of ant colony algorithm and the application of 0/1 knapsack. In: 2011 6th International Conference on Computer Science Education (ICCSE). (2011) 464–467
10. Changdar, C., Mahapatra, G.S., Pal, R.K.: An ant colony optimization approach for binary knapsack problem under fuzziness. Applied Mathematics and Computation **223** (2013)
11. Aziz, Z.A.: Ant colony hyper-heuristics for travelling salesman problem. Procedia Computer Science **76** (2015) 534 – 538
12. Chen, P.C., Kendall, G., Berghe, G.V.: An ant based hyper-heuristic for the travelling tournament problem. In: 2007 IEEE Symposium on Computational Intelligence in Scheduling. (2007) 19–26
13. Ferreira, A.S., Pozo, A., Gonçalves, R.A.: An ant colony based hyper-heuristic approach for the set covering problem. Advances in Distributed Computing and Artificial Intelligence Journal (2015)
14. Cuesta-Cañada, A., Garrido, L., Terashima-Marín, H.: Building hyper-heuristics through ant colony optimization for the 2d bin packing problem. In Khosla, R., Howlett, R.J., Jain, L.C., eds.: Knowledge-Based Intelligent Information and Engineering Systems, Berlin, Heidelberg, Springer Berlin Heidelberg (2005) 654–660
15. López-Camacho, E., hugo Marin, H., Ross, P., Ochoa, G.: A unified hyper-heuristic framework for solving bin packing problems. Expert Syst. Appl. **41**(15) (2014) 6876–6889
16. Gaertner, D., Clark, K.: On optimal parameters for ant colony optimization algorithms. In: Proceedings of the International Conference on Artificial Intelligence 2005, CSREA Press (2005) 83–89
17. Wei, X.: Parameters analysis for basic ant colony optimization algorithm in tsp. International Journal of u-and e-Service, Science and Technology **7**(4) (2014) 159–170
18. Pisinger, D.: Where are the hard knapsack problems? Computers & Operations Research **32**(9) (2005) 2271–2284