The Ant Colony Metaphor for Multiple Knapsack Problem

Cena, M.; Crespo M.L.; Kavka, C; Leguizamón, G.•

Abstract

This paper presents an *Ant Colony Optimisation (ACO)* model for the *Multiple Knapsack Problem (MKP)*. The ACO Algorithms, as well as other evolutionary metaphors, are being applied successfully to diverse heavily constrained problems: Travelling Salesman Problem, Quadratic Assignment Problem and Bin Packing Problem. An Ant System, the first ACO algorithm that we presented in this paper, is also considered a class of multiagent distributed algorithm for combinatorial optimisation. The principle of an ACO algorithm is adapted to the MKP. We present some results regarding its performance against known optimum for different instances of MKP. The obtained results show the potential power of this particular evolutionary approach for optimisation problems.

Keywords: *Nature Based Metaheuristics, Ant Colony Optimisation, Subset Problems, Multiple knapsack Problem.*

•

Members of the UNSL 338403 project (**Grupo de Interés en Sistemas de Computación)**
Departamento de Informática Universidad Nacional de San Luis. Ejército de los Andes 950 - Local 106
(5700) San Luis – Argentina. e-mail: {legui, mcena, mcrespo, ckavka,@unsl.edu.ar. Fax: +54 652 30224

1. Introduction

The Zero/One Multiple Knapsack Problem and Zero/One Knapsack Problem have been intensively studied using different traditional methods, such as Branch-and-Bound and Dynamic Programming [11,12]. Also, modern heuristic techniques [9,10,13] like Genetic Algorithms (GA) and Simulated Annealing (SA) have shown to be able to produce high-quality solutions for these types of highly constrained NP-complete problems.

In this paper we propose to solve MKP by using the Ant Colony Metaphor [4,5], an approach based on the result of low-level interaction among many co-operating simple agents that are not aware of their co-operative behaviour. Each simple agent is called 'ant' and the Ant System (a distributed algorithm) is a set of ants co-operating in a common problem solving activity.

Previous works in this area have inspired our work [2,3,5,6]. Different versions of ACO algorithms have been used in order to solve Travelling Salesman Problem (TSP), Bin packing and continuous space optimisation problems. Problems like TSP and Bin Packing can be represented as a sequence on n items (*n cities* to be visited or *n objects* to be packed), where the actual order of the sequence determines a particular solution to the problem. Thus in general, the search space consists of all $n!$ permutations.

Bin Packing and TSP are properly suitable to be solved by using an Ant System [2]. In this paper, we adapt the original Ant System [4] so it can be used to solve non-ordering problems like the MKP. We also present experimental results that show that the adapted Ant System can solve MKP in an efficient way. In this adapted Ant System, the ants (agents) are not concerned with discovering the best ordering (or tours), they just look for a subset of n items, such that the total profit is maximised and all constrains are satisfied.

The adapted Ant System uses Ant-cycle [5], a particular instance of the algorithms belonging to Ant system class (other instances proposed in [5] are Ant-density and Ant-quantity). In the Ant-cycle algorithm every ant changes the system-shared memory using a quantity (called *trail*) that is proportional to its global behaviour. On the other hand, Ant-density and Ant-quantity algorithms use strictly local information.

The primary purpose of our study is to evaluate the Ant System performance in relation with the known results for different instances of MKP. The MKP instances considered are taken from [1].

We also present some results regarding its performance when varying the values of the parameters that control the probabilities for item selection.

The remainder of the paper is organised in the following way. In the next sections the MKP formulation and the classical and adapted model of an Ant Colony System are given. Next, the experiments performed, the results obtained and the statistical analysis are shown. Finally, the conclusions are exposed.

2. MKP formulation

MKP can be formulated [1,9,11] as:

$$maximise \quad \sum_{j=}^{n} p_j x_j$$

$$subject \quad to \quad \sum_{j=1}^{n} r_{ij} x_j \leq c_i \qquad i=1,...,m \qquad (2.2)$$

$$x_j \in \qquad\qquad j \qquad n$$

m constrains described in equation (2.2) is called a knapsack constrain, so the MKP is also called the *m-dimensional Knapsack Problem*.

Let $I=\{1,...,m\}$ and $J=\{1,...,n\}$, with $c_i \geq 0$ for all $i \in I$, $j \in J$. A *well-stated* MKP assumes that $p_j > 0$ and $r_{ij} \leq c_i < \sum_{j=1}^{n} r_{ij}$ for all $i \in I$, $j \in J$, since any violation of these conditions will result in some x_j's being fixed to zero and/or some constrains being eliminated. Note that the $(r_{ij})_{mxn}$ matrix and $(c_i)_m$ vector are both non-negative which distinguishes this problem from general 0-1 linear integer programming problem.

Many practical problems can be formulated as a MKP, for example, the capital budgeting problem where project j has profit p_j and consumes r_{ij} units of resource i. The goal is to find a subset of the n projects such that the total profit is maximised and all resource constrains are satisfied.

3. The Ant System for an Order Based Problem

In this section, we describe our Ant System applied to the Travelling Salesman Problem.

Given a set of n cities, the Travelling Salesman Problem [11,12] is to find a closed path that visits every city exactly once (tour) with minimal total length. i.e.

$$minimize\, COST(i_1,\ldots,i_n) = \sum_{j=1}^{n-1} d(C_{i_j}, C_{i_{j+1}}) + d(C_{i_n}, C_{i_1})$$

where $d(C_k, C_l)$ is the distance between city k and city l.

Let $b_i(t)$ $(i=1,\ldots,n)$ be the number of ants in city i at time t and let $N_a = \sum_{i=1}^{n} b_i(t)$, the total number of ants.

Let $\tau_{ij}(t+n)$ be the *intensity of trial* on $path_{ij}$ at time $t+n$, given by

$$\tau_{ij}(t+n) = \rho\, \tau_{ij}(t) + \Delta\tau_{ij}(t, t+n) \qquad \textbf{(3.1)}$$

The amount ρ is such $(1-\rho)$ is the *coefficient of evaporation* $(0<\rho<1)$.

$\Delta\tau_{ij}(t,t+n) = \sum_{k=1}^{N_a} \Delta\tau^k_{ij}(t, t+n)$, where $\Delta\tau^k_{ij}(t, t+n)$ is the quantity per unit of length of trial substance (pheromone in real ants) laid on $path_{ij}$ by the *k-th* ant between time t and $t+n$ and is given by the following formula:

$$\Delta\tau^k_{ij}(t+n) = \begin{cases} \dfrac{Q}{L^k} & \text{if k -- th ant uses edge } (i, j) \text{ in its tour} \\ 0 & otherwise \end{cases} \qquad \textbf{(3.2)}$$

where Q is a constant and L^k is the tour length of the k-th ant.

The intensity of trial at time 0, $\tau_{ij}(0)$, is set to a randomly chosen value.

During the next $(t+n)$ tour the probability to visit city j when being at city i is

$$P_{ij}(t,k) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h\in allowed_k}[\tau_{ih}(t)]^{\alpha}[\eta_{ih}]^{\beta}} & j \in allowed_k \\ 0 & otherwise \end{cases} \qquad \textbf{(3.3)}$$

The set $allowed_k$ represents the cities still not visited for that particular tour and η_{ij} is a local heuristic. For the TSP the parameter η_{ij}, called *visibility,* is $1/d(Ci,Cj)$ [5].

The parameters α and β allow a user control on the relative importance of trail versus visibility. Hence, the transition probability is a trade-off between visibility, which says that close cities should be chosen with high probability, and trail intensity, that says that if on $path_{ij}$ there is a lot of traffic then is it highly profitable.

A data structure, called *tabu list*, is associated to each ant in order to avoid that ants visit a city more than once, i.e. the $\textbf{\textit{tabu}}_k$ list maintain a set of visited cities up to time *t* by the k-th ant. Therefore the $allowed_k$ set can be defined as follow: $allowed_k=\{j\ /\ j \notin \textbf{\textit{tabu}}_k\}$. When a tour is completed the *tabu list* is emptied and every ant is free again to choose an alternative way.

By using the above definitions, we describe the Ant-cycle algorithm:

The Ant-cycle algorithm

```
1  Initialise:
   Set t:=0                                               {t is the time counter}
     For every edge (i,j) set an initial value τij(t)
For every edge (i,j) set Δτij(t,t+n):=0
     Place bi(t) ants on every node i                     {bi(t) is the number of ants on node i at time t}
   Set s:=1                                               {s is the tabu list index}
     For i:=1 to n do
       For k:= 1 to bi(t) do
         tabuk(s):=i                                      {starting town is the first element of the tabu
                                                            list of the k-th ant}

2  Repeat until tabu list is full                         {this step will be repeated (n-1) times}
   Set s:=s+1
   For i:=1 to n do                                       {for every town}
      For k:=1 to bi(t) do                        {for every k-th ant on town i still not moved}
         Choose the town j to moved to, with
         probability Pij(t,k) given by equation (3.3)
         Move the k-th ant to j                           {this instruction creates the new values
                                                          bj(t+1)}
         Insert node j in tabuk(s)

   3.    For k:= 1 to n do
           Compute L^k                                    {it results from the tabu list}
         For s:= 1 to n-1 do                              {scan the tabu list of the k-th ant}
           Set (h,l):=(tabuk(s), tabuk(s+1))              {(h,l) is the edge connecting town s y s+1 in
                                                          the tabu list of ant k}
           Δτhl (t+n):= Δτhl (t+n) + Q/L^k

   4     For every edge (i,j) compute τij(t+n) according to equation (3.1)
         Set t:=t+n
         For every edge (i,j) set Δτij(t+n):=0

   5    Memorise the shortest tour found so far
         if (NC < NC_MAX) or (not all the ants choose the same tour)
                                                  {NC is the number of algorithms cycles; in NC
                                                   cycles are tested NC * Na solutions}
         then
                 Empty all tabu list
                 Goto step 1.2
         Else
                 Print shortest tour and Stop
```

4. The Ant System for MKP

In order to solve MKP, it is necessary to adapt the Ant System in some way. The purpose of the ants in MKP is not to get a tour with minimum cost like in TSP; but they look for a subset of n items or projects (see MKP formulation) such that the total profit is maximised and all resource constrains are satisfied.

Let b_i $(i=1,...,n)$ be the number of ants incorporating in the solution the item i at time $t=0$ and let $N_a = \sum_{i=1}^{n} b_i$; the total number of ants.

Since in MKP there are not *paths,* the *intensity of trial* and *visibility* are computed in a slightly different way.

Let $\tau_i(t+N_{max})$ be the *intensity of trial* on item i at time $t+N_{max}$, given by

$$\tau_i(t+N_{\max}) = \rho\ \tau_i(t) + \Delta\tau_i(t,\ t+N_{max}) \qquad \textbf{(4.1)}$$

As in *Eq. 3.1,* ρ is such $(1-\rho)$ is the *coefficient of evaporation* and N_{max} is the maximum number of items qualified to be added to some solution by some ant.

$\Delta\tau_i(t,t+N_{max}) = \sum_{k=1}^{N_a} \Delta\tau_i^k(t,\ t+N_{max})$, where $\Delta\tau_i^k(t,\ t+N_{max})$ is the quantity per unit of length of trial substance (pheromone in real ants) laid on item i by the *k-th* ant between time t and $t+N_{max}$ and is given by the following formula:

$$\Delta\tau_i^k(t+N_{max}) = \begin{cases} \dfrac{L^k}{Q} & \text{if } k\text{-th ant incorporates item } i \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(4.2)}$$

where Q is a constant and L^k is the profit *(Eq. 2.1)* obtained by the k-th ant.

The intensity of trial at time 0, $\tau_i(0)$, is set to a randomly chosen value.

During the next $(t+N_{max})$ item incorporation the probability to select the item i by the k^{th} ant, in order to complete the $solution_k$ is :

$$P_i(t,k) = \begin{cases} \dfrac{[\tau_i(t)]^{\alpha}[\eta_i(k,t)]^{\beta}}{\sum_{j\in allowed_k}[\tau_j(t)]^{\alpha}[\eta_j(k,t)]^{\beta}} & i \in allowed_k \\ 0 & otherwise \end{cases} \qquad \textbf{(4.3)}$$

where $allowed_k$ is a set of items still not considered by the k^{th} ant and the $solution_k$ satisfies all constraints if some of them are added. The parameter η_i, called *pseudo-utility,* is a local heuristic. We chose η_i as follows:

$$\eta_i(k,t) = \frac{p_i}{\overline{\delta}_{ij}(k)} \quad ; \quad \overline{\delta}_{ij}(k) = \frac{\sum_{j=}^{m} \delta_{ij}(k)}{m}$$

(4.4)

$$\delta_{ij}(k) = \frac{r_{ji}}{(c_j - u_j(k))} \quad ; \quad u_j(k) = \sum_{l \in solution_k} r_{jl}$$

Where $(c_j - u_j(k))$ is the remaining amount to reach the boundary of the constraint j, $r_{ji} \le (c_j - u_j(k))$ and $\delta_{ij}(k) \in (0,1]$, is the *tightness* of item *i* on constraint *j* when item *i* is added to $solution_k$. Consequently the *pseudo-utility* $\eta_i(t,k)$ turns larger as $\overline{\partial}_{ij}(k)$ (*tightness average)* turns smaller.

The parameters α and β, as for TSP, allow a user control on relative importance of *trail* versus the heuristic *(pseudo-utility for MKP)*. Hence, the transition probability is a trade-off between *pseudo-utility*, which says that more profitable items that uses less resources should be chosen with high probability, and trail intensity, that says that if item *I* is part of a lot of solutions, then is it highly desirable.

A data structure, called *tabu list*, is also associated to each ant in order to avoid that ants choice a item more than once, i.e. the ***tabu***$_{\mathbf{k}}$ list maintain the set of added items up to time *t* by the *k-th* ant. This list also maintains u_j(k) *(j=1..m)* in order to reduce the required computational time.

The $allowed_k$ set can be defined as follows: $allowed_k = \{j \,/\, j \notin \textbf{tabu}_k$ *and the* $solution_k$ *with item j added satisfies all constraints*$\}$. When all ants add to the solutions as many items as they can, an item h_k is selected from the k^{th} *tabu list*. Then the k^{th} *tabu list* is emptied and the k^{th} ant is free again to choose starting with the item h_k as its initial solution.

The outline of the adapted Ant-cycle algorithm follows:

The Adapted Ant-cycle algorithm

1 Initialise:
  Set t:=0 {t is the time counter}
  For every item (i) set an initial value $\tau_i(t)$
  For every item (i) set $\Delta\tau_i(t,t+N_{max})$:=0
  b_i ants choose the item i {b_i is the number of ants choosing item i at time 0}

  Set s:=1 {s is the tabu list index}
  For i:=1 to n do
    For k:= 1 to b_i do
      $\textbf{tabu}_k$(s):=i {initial item of the tabu list of the k-th ant}

2 For k:=1 to Na do
  s:=2
  Repeat until some constraint is no satisfied by k-th ant
  Choose the item i, with probability $P_i(t,k)$ given by equation (4.3)
  $\textbf{tabu}_k$(s):=i
  s:=s+1

3	For k:= 1 to Na do
	  Compute L^k
	For s:= 1 to Number of items in **tabu**$_k$ do
	  h = **tabu**$_k$(s)
	  $\Delta\tau_h (t,t+ N_{max}) := \Delta\tau_h (t, t+ N_{max}) + L^k/Q$
		
		{it results from the tabu list}
		{scan the tabu list of the k-th ant}

4	For every item (i) compute $\tau_i(t+ N_{max})$ according to equation (4.1) Set $t:=t+ N_{max}$ For every item (i) set $\Delta\tau_i (t,t+N_{max}):=0$	

5	Memorise the best solution found so far
	if ($NC < NC_{MAX}$) or (not all the ants find the same solution)
	
	then
	  h_k=item randomly select from **tabu**$_k$ (k=1..Na)
	  Empty all tabu list
	  tabu$_k$(1)=h_k (k=1..Na)
	  Goto step 2
	else
	Print best solution and Stop
		
		{NC is the number of algorithms cycles; in NC cycles are tested NC.Na solutions}

5. Experimental results

Several parameters were considered for the experiments. Next each one and its values for different running are presented below.

# cycles:	***20***
# ants:	***# items of the instance to be applied***
Q (used in Eq. 4.2):	$\sum_{j=}^{n} p_j$ ***(Eq. 2.1)***
Coefficient of evaporation (ρ):	***0.3, 0.7***
Trade-off between trail and pseudo-utility (α,β):	***(0,1), (1,0), (1,1), (1,5), (5,1)***

The above values were selected regarding the results achieved in previous works [2,3,4,5,6]. The five combinations of values corresponding to α and β parameters have influence on the probability values *Pi(t,k)* as follows:

a) (α=0, β=1) - *only the heuristic is important (no co-operation between agents)*
b) (α=1, β=0) - *only the trial is important (no problem knowledge tailored)*
c) (α=1, β=1) - *heuristic and trial are equally proportional*
d) (α=1, β=5) - *the heuristic is more important*
e) (α=5, β=1) - *the trial is more important*

Table I shows for each test case the following data:

- *# of instance*
- *Size of instance <n,m>*
- *Known optimum (from [1])*
- *The best result obtained out of 5 runs by the Ant System using some combination of parameter values*

#instance	Instance	Known Optimum	Best Value by AC
1	<15,10>	4015	4015
2	<20,10>	6120	6120
3	<28,10>	12400	12400
4	<39,5>	10618	10570
5	<50,5>	16537	*16470*
6	<60,30>	7772	7772
7	<60,30>	8722	*8722*
8	<105,2>	1095445	*1095382*
9	<105,2>	624319	624319
10	<28,2>	141278	141278
11	<28,2>	130883	130883
12	<28,2>	95677	95677
13	<28,2>	119337	119337
14	<28,2>	98796	98796
15	<28,2>	130623	130623
16	<30,5>	4554	4554
17	<6,10>	3800	3800
18	<10,10>	8706.1	8706.1
19	<100,5>	24382	24308

***Table I**. Best results obtained by the Ant System*

The shadow cells in Table I indicate suboptimal results which are just 4 out of 19 instances of the MKP considered. For example, the best value obtained for the instance #8 is the same that was reported by Khuri et al. [9] where a GA was used. On the other hand, for test case #19, the value corresponding to "known Optimum" column is the best value reported by Paul Chu et al. [13] again, by using a GA. An improvement on the results showed in Table I and additional experiments on hard and bigger instances of MKP, also taken from [1,13] are reported in [14].

It is important to remark that for some instances, the Ant System was able to find the best solution (or suboptimal) when the first cycle was completed, meaning that the greedy heuristic was sufficient, excluding any co-operation. However, the best results in general were obtained by using ($\alpha=1,\ \beta=1$) *or* ($\alpha=1,\ \beta=5$) combinations, i.e. the significance of the heuristic influence was at least as important as the trail influence. Table II shows the relative importance of the trail and heuristic effect on the final result, for instances #8, #9, #10 and #15 (from Table I).

#instance	Known Optimum	α=0,β=1	α=1,β=0	α=1,β=1	α=1,β=5	α=5,β=1
8	1095445	1093831	1073356	1095382	1095382	1095232
9	624319	611140	559648	624116	624319	620872
10	141278	141098	135717	141278	140778	140618
15	130623	130623	118379	130233	130623	127523

***Table II.** Relative importance of trial versus pseudo-utility*

By the ($\alpha=1,\ \beta=0$) combination, the Ant System achieved the worst results, because it explores the solution search space by using only co-operation which alone is not a very effective strategy. On the other hand, by ($\alpha=5,\ \beta=1$) combination, the Ant System showed for all instances a stable behaviour, however the achieved results were not the best ones. In Figure 1 the best results found in each cycle by using two different seeds for instance #11 are plotted. It can be seen that the Ant System stuck prematurely in a local optimum due to the considerable pressure on the trail *(α=5)*.

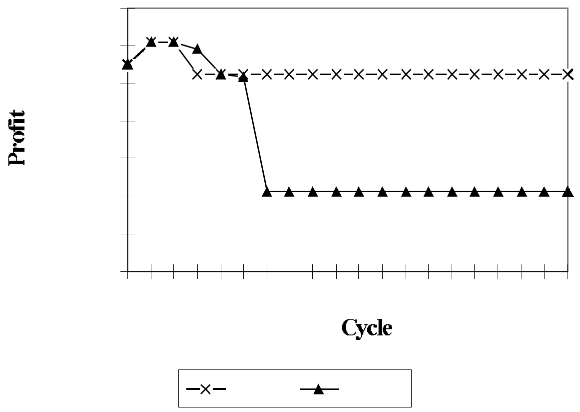

Figure 1. *The Ant System stuck in a local optimum*

Figure 2 shows the changes on the amount of trail on each item for instance #9, along the running of the Ant System. It can be seen that as the running proceeds, the amount of trail turns larger on those items which are part of the solution. On the other hand, the amount of trail is not necessarily zero or near zero for that items which are not part of the solution. At the bottom of the Figure 2, it is possible to note the correspondence between the best solution in relation to the amount of trial after 20 cycles.

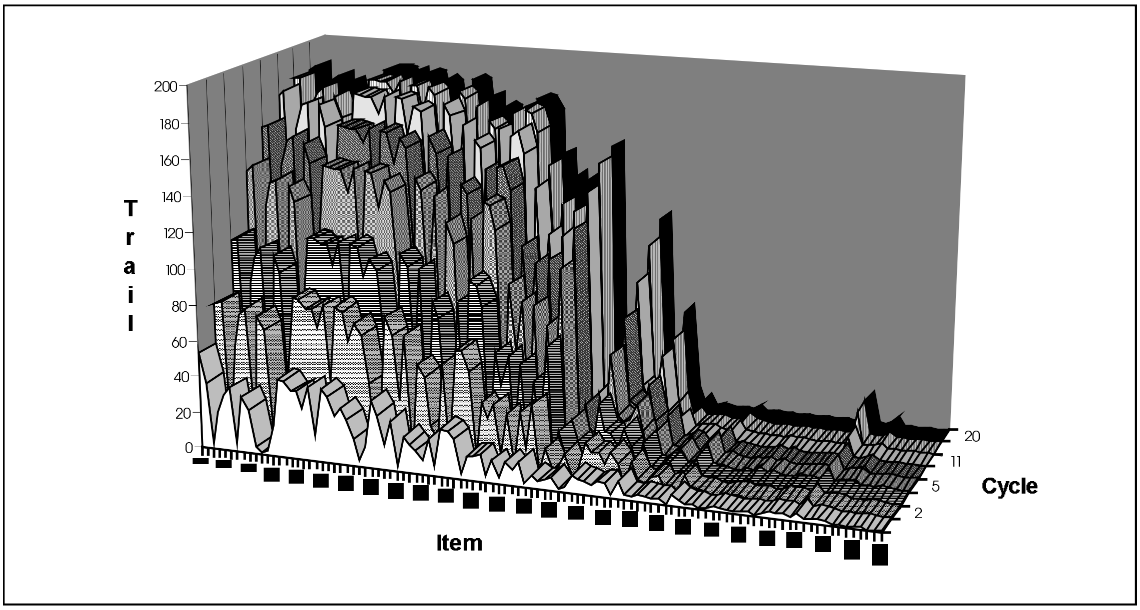

Figure 2. *Amount of trail along the Ant System running*

Figure 3, for the same instance, shows other perspective of the amount of the trial on each item by plotting that amount for the first (randomly chosen) and the last cycle (when the Ant System converged to the solution).

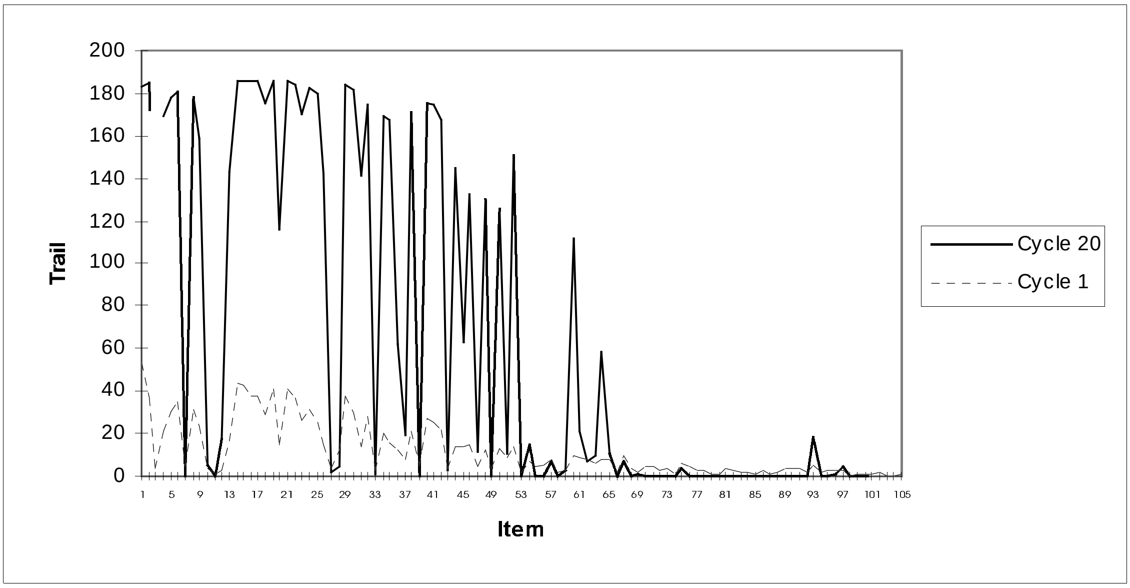

Figure 3. *Amount of trail at cycle 1 and 20.*

Finally, it is worth to note that the variation on the *evaporation coefficient* (ρ parameter) did not influence strongly the Ant System performance.

6. Conclusions

We have shown an alternative approach to solve the Multiple Knapsack Problem. The original conception applied to Travelling Salesman Problem, Bin Packing and continuous space optimisation problems, was adapted to obtain an Ant System able to obtain goods solutions for MKP.
Although in some cases the greedy heuristic (local search) was enough, the best performance of the Ant System was achieved when the trial and heuristic guide the search in conjunction. However, the weight of the heuristic owned a primary role when the probabilities of the item selection were calculated.

Current research on Ant Systems involves an extensive study regarding the effect on the performance of the Ant System according different values of its main parameters in order to test harder instances of MKP [14]. Also, a study considering alternative heuristics, in order to decrease the computational time of the Ant System, is reported in [15].

Acknowledgement

The authors wish to acknowledge Prof. Raúl Gallard, Director of the research group in which our work is included, UNSL for providing us necessary resources and Prof. Zbigniew Michalewicz for his very useful suggestions.

References

[1] Beasley, J. - "OR-Library: Distributing Test Problems by Electronic Mail" - e-mail: o.rlibrary@ic.ac.uk

[2] Bilchev, G. (1996). "Evolutionary Metaphors for the Bin Packing Problem" - Proceedings of the Fifth Annual Conference on Evolutionary Programming. San Diego, California – USA.

[3] Bilchev, G.; COMPLETAR!! (1995). "The Ant Colony Metaphor for Searching Continuous Design Spaces" - Published in Evolutionary Computation, selected papers from AISB Workshop, Sheffield UK. Edited by T.C. Fogarty.

[4] Dorigo, M. COMPLETAR!! (1991). "Distributed Optimisation by Ant Colonies" - Proceedings of ECAL91. Elsevier Publishing, pp 134-142.

[5] Dorigo, M.; COMPLETAR!! (1992). "An investigation of some properties of an Ant algorithm" - Proceedings oh the Parallel Problem Solving from Nature Conference. Elsevier Publishing.

[6] Dorigo, M.; COMPLETAR!! (1996). "A study of some properties of ANT-Q" - Published in Proceedings of PPSN IV. Springer-Verlag.

[5] Fogel, D. COMPLETAR!! (1995). "Evolutionary Computation, Toward a New Philosophy of Machine Learning" - IEEE Press.

[9] Khuri, S; Bäck, T and Heitkötter, J. (1994). "The Zero/One Multiple Knapsack Problem and Genetic Algorithms". Proceedings of the 1994 ACM Symposium of Applied Computation. ACM Press, New York.

[10] Michalewicz, Z. (1994). "Genetic Algorithms + Data Structures = Evolution Programs". Springer Verlag.

[11] Nemhauser, G. COMPLETAR!! (1988). "Integer and Combinatorial Optimisation". John Wiley & Sons, Inc.

[12] Papadimitriou, C. COMPLETAR!! (1982). "Combinatorial Optimisation: Algorithms and Complexity". Prentice Hall.

[13] Chu, P. and Beasley, J. (1998). "A Genetic Algorithm for the Multi-constraint Knapsack Problem". Journal of Heuristics, Vol 4.

[14] Leguizamón, G. and Michalewicz, Z. (1999). “An New Version of the Ant System for Subset Problems”. Submitted and accepted in the International Conference on Evolutionary Computation – Special Session on Ant Colony Optimisation. Washington – USA.

[15] Leguizamón, G.; Crespo, M.L.; Kavka, C. and Cena, M. (1998). “A Study of Performance of an Ant Colony System applied to the Multiple Knapsack Problem”. Proceedings of EIS-98, E. Apayd (ed), ICSC Academic.