Ant Colony Optimization for Model Checking*

Enrique Alba and Francisco Chicano

GISUM Group, Departamento de Lenguajes y Ciencias de la Computación
University of Málaga, SPAIN
eat@lcc.uma.es chicano@lcc.uma.es

Extended Abstract

Model Checking is a well-known and fully automatic technique for checking software properties, usually given as temporal logic formulas on the program variables. Most of model checkers found in the literature use exact deterministic algorithms to check the properties. These algorithms usually require huge amounts of computational resources if the checked model is large. We propose here the use of Ant Colony Optimization (ACO) to check safety properties of concurrent systems.

ACO algorithms are stochastic techniques belonging to the class of metaheuristic algorithms [1] and inspired in the foraging behavior of the real ants. The main idea consists in using *artificial ants* which simulate this behavior in a graph. Artificial ants are placed on initial nodes of the graph and they jump stochastically from one node to another in order to search the shortest path from the initial node to an objective one. The cooperation among the artificial ants is a key factor in the search. This cooperation is performed indirectly by means of the *pheromone trails*, which is a model of the chemicals that real ants use for their communication. For more details on ACO algorithms see [2].

We use in this work Max-Min Ant System ($\mathcal{MM}$AS), a kind of ACO algorithm in which the pheromone values are bounded to a given range. For the experiments we use five faulty concurrent systems modelled in PROMELA: `basic_call2`, `garp`, `giop12`, `marriers4`, and `phi8`. We propose two different versions of the $\mathcal{MM}$AS algorithm. In the first one, called $\mathcal{MM}$AS-b, there is no heuristic information for guiding the ants in the construction phase. We compare this version against Depth-First Search (DFS) and Bread-First Search (BFS) algorithms, which perform also a blind search on the exploration graph and are the most used algorithms in current model checkers. In our second $\mathcal{MM}$AS algorithm, called $\mathcal{MM}$AS-h, the number of enabled transitions in each state is used as a heuristic information for guiding the ants. We compare this version against the A* algorithm using the same heuristic (a novel proposal by Edelkampf, Leue and Lluch-Lafuente [3]). We perform 100 independent runs of the $\mathcal{MM}$AS algorithms to get statistical confidence. In all the cases $\mathcal{MM}$AS algorithms found

* This work has been partially funded by the Ministry of Education and Science and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project). Francisco Chicano is supported by a grant (BOJA 68/2003) from the Junta de Andalucía.

out a deadlock state in the concurrent systems. In Table 1 we show the average length of the error trail found, the average CPU time used, and the average total amount of memory used for all the algorithms.

Table 1. Results obtained with all the algorithms.

Concurrent System		DFS	BFS	$\mathcal{MM}$AS-b	A*	$\mathcal{MM}$AS-h
`basic_call2`	Len.	82.00	**26.00**	30.24	**26.00**	31.30
Basic call protocol	CPU (ms)	**10.00**	80.00	41.70	110.00	**49.40**
with 2 users	Mem. (KB)	**2713.00**	16384.00	4219.44	17408.00	**4278.44**
`garp`	Len.	65.00	**17.00**	24.70	**17.00**	24.00
Generic Attribute	CPU (ms)	10.00	53180.00	**6.00**	2820.00	**9.50**
Registration Protocol	Mem. (KB)	3357.00	480256.00	**2532.28**	122880.00	**2634.44**
`giop12`	Len.	48.00	**43.00**	43.45	**43.00**	**43.00**
CORBA General Inter-ORB	CPU (ms)	**10.00**	350.00	41.20	490.00	**26.80**
Protocol (1 client, 2 servers)	Mem. (KB)	**2901.00**	49152.00	4085.12	37888.00	**3202.24**
`marriers4`	Len.	-	-	**85.79**	-	**88.76**
Stable marriage problem	CPU (ms)	-	-	**148.60**	-	**91.90**
with 4 suitors	Mem. (KB)	-	-	**15388.39**	-	**10625.80**
`phi8`	Len.	1338.00	**10.00**	**10.00**	**10.00**	**10.00**
Dining philosophers	CPU (ms)	70.00	60.00	**21.90**	**0.00**	35.20
with 8 philosophers	Mem. (KB)	29696.00	17408.00	**4333.92**	**2105.00**	4986.68

We can observe in the results that our $\mathcal{MM}$AS algorithms are the unique ones in finding out a deadlock state in `marriers4` model. The exact algorithms (DFS, BFS, and A*) were stopped by the operative system because of their memory consumption and thus were unable of discovering the deadlock state. Among the algorithms without heuristic information, $\mathcal{MM}$AS-b obtains error trails of a similar length as BFS but using much less resources (memory and CPU time). The length of the error trails found by DFS are much longer (bad quality) than those of the other two algorithms (BFS and $\mathcal{MM}$AS-b).

Comparing the algorithms using heuristic information we can observe that $\mathcal{MM}$AS-h gets near optimal error trails with important savings of resources with respect to A*. Furthermore, in the `giop12` model, $\mathcal{MM}$AS-h obtains in the 100 independent runs an error trail with the minimum length using one tenth of the memory and one twentieth of the time that A* requires for the same goal.

The results above state that ACO algorithms find optimal or near optimal error trails in faulty concurrent systems with a reduced amount of resources, outperforming algorithms that are the state of the art in model checking. This fact makes them suitable for checking safety properties in large concurrent systems, in which traditional techniques fail to find errors because of the model size.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys **35**(3) (2003) 268–308
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press (2004)
3. Edelkampf, S., Leue, S., Lluch-Lafuente, A.: Directed explicit-state model checking in the validation of communication protocols. International Journal of Software Tools for Technology Transfer **5** (2004) 247–267