College of Engineering

Parallel $\mathcal{MAX} - \mathcal{MIN}$ Ant System

for combinatorial constraint-satisfaction problems

David Vulakh[1], Raphael Finkel[2]

[1]*Math, Science, and Technology Center, Paul Laurence Dunbar High School, Lexington, KY 40513*

[2]*Department of Computer Science, University of Kentucky, Lexington, KY 40506*

THE MATH, SCIENCE, AND TECHNOLOGY CENTER at Paul Laurence Dunbar High School

INTRODUCTION

Many mathematical and practical problems require elements of a set to be arranged in a way that satisfies specific conditions. Familiar examples include Sudoku, timetabling, scheduling, and resource distribution (Russell & Norvig, 2009). Such problems, called constraint-satisfaction problems (CSPs), are solved when values from a domain are assigned to variables in a way that does not violate a set of constraints. Because some CSPs are NP-complete (Bulatov, Krokhin, & Jeavons, 2000), no polynomial-time algorithm to solve the general constraint-satisfaction problem is known. In order to reduce the time required to solve large instances, an Ant Colony Optimization (ACO) algorithm called the $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) (Stützle & Hoos, 2000) was applied to combinatorial CSPs with focus on a specific CSP (the Costas-array problem, which remains open for most sizes $M \geq 30$).

BACKGROUND

Definition 1. A ***constraint-satisfaction problem*** (CSP) (Bulatov et al., 2000) is an ordered triple $\langle V, D, C \rangle$, where

- V is the set of ***variables***
- D is the ***domain***
- C is the set of ***constraints***

Each constraint C_i is of the form $\langle s_i, \rho_i \rangle$, where s_i describes a tuple of variables $s_i : \{1, \ldots, m_i\} \to V$ and ρ_i is an m_i-ary Boolean relation on D, $\rho_i \subseteq D^{m_i}$

A ***solution*** of the constraint-satisfaction problem $\langle V, D, C \rangle$ is a function $f : V \to D$ such that for all constraints $C_i = \langle s_i, \rho_i \rangle$, $f \circ s_i \in \rho_i$

Definition 2. A ***Costas array*** (Drakakis, 2011) of size M is a permutation of the integers from 1 through M such that its permutation matrix contains no equal displacement vectors between distinct pairs of distinct elements.

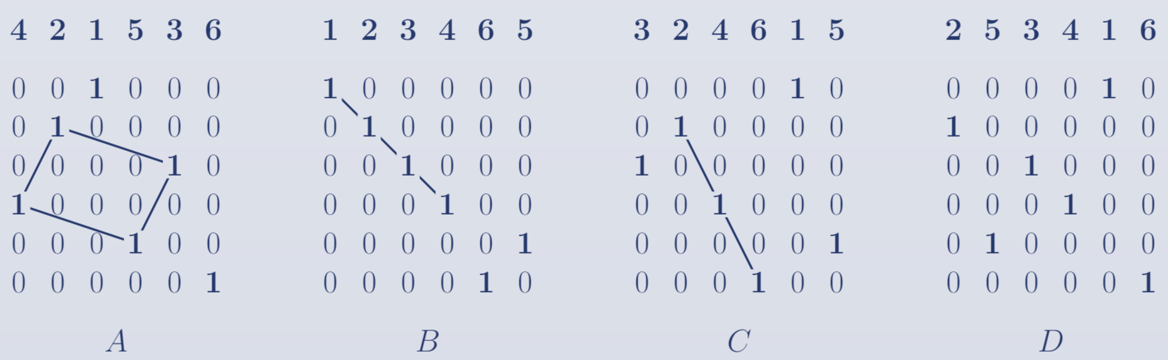

Figure 1: Three permutations (A, B, and C) that violate the Costas-array property and one (D) that satisfies it.

The constraint-satisfaction problem of searching for Costas arrays of size M can be formally defined as

$$\begin{aligned}&\langle\ \{x_1, \ldots, x_M\}, \{1, \ldots, M\},\\ &\quad \{\ \langle i \mapsto x_i, \{t : \forall i, j\, (t(i) = t(j) \to i = j)\}\rangle\,,\\ &\qquad \langle i \mapsto x_i, \{t : \forall i, j, k, l\, (i \neq j \wedge i \neq k \to j - i \neq l - k \vee t(j) - t(i) \neq t(l) - t(k))\}\rangle\\ &\quad \}\ \rangle\end{aligned}$$

For $M \geq 30$, it is not in general known whether a solution to the Costas-array problem exists; the problem for a given size M is solved if a solution is found.

Definition 3. If a parallel program with N workers solves a problem in T time and the same program with one worker solves the problem in T_0 time, the ***speedup*** S and ***efficiency*** E of the program for that problem are

$$S = \frac{T_0}{T} \qquad E = \frac{S}{N} = \frac{T_0}{N \cdot T}$$

DESIGN GOAL AND METHODS

Because general CSP solvers run in exponential time, the runtime of programs solving combinatorial CSPs grows rapidly. The goal of the project was to create a parallelized and heuristic-accelerated $\mathcal{MM}$AS framework that:

- Distributes work to local processors to solve the Costas-array problem
- Maintains high efficiency ($E > 0.5$) at high numbers of workers (up to $N = 250$)
- Exhibits lower average time-to-first solution with the new optimizations than without.

A detailed description of the research process can be found in the accompanying journal.

The design process started with the implementation of a single-threaded $\mathcal{MM}$AS solver for Costas arrays. Over the course of several months, this solver was parallelized and new optimizations, such as map-based pheromone storage, were developed. Two languages were used:

- C++ for the $\mathcal{MM}$AS framework
- Java for the visualizer program

No libraries outside the C++17 STL and Java standard were used.

The ant system was tested on a machine with 256 identical Xeon Phi Knight's Landing processors.

A discussion of the techniques used to apply $\mathcal{MM}$AS to the Costas-array problem and the new optimizations added follows.

FRAMEWORK DESIGN

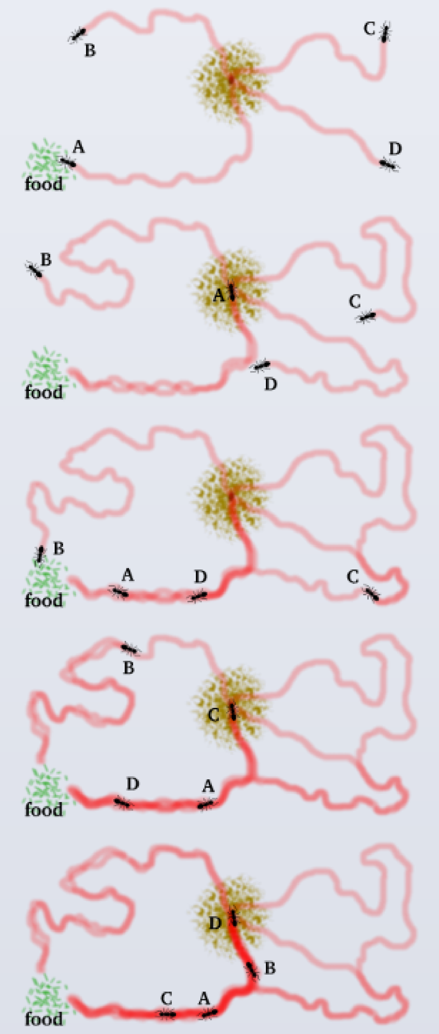

Figure 2: A diagram of ant-foraging behavior. Initially, ants follow random paths. Over time, better paths acquire higher concentrations of pheromone, and all ants are drawn to a single optimal path.

Image taken from http://mute-net.sourceforge.net/howAnts.shtml

ACO algorithms (Dorigo & Di Caro, 1999) are based on the behavior of foraging ants, which indirectly communicate with each other by secreting and detecting chemicals called pheromones (Fig. 2).

- In order to apply ACO to an optimization problem, the problem is represented as a graph connecting domain values called the ***construction graph***.
- Processes called ants stochastically traverse this graph, altering the properties that govern ant behavior.

$\mathcal{MM}$AS was applied to the problem of Costas arrays as follows:

- Paths through the construction graph represent permutations.
- The ***cost*** of a path is the number of Costas-array-property violations.
- The ***heuristic value*** η_{si} associated with adding i to permutation s is inversely proportional to the cost incurred.
- The ***pheromone value*** τ_{si} is based on the learned favorability of following s with i.
- The probability that an ant will add i to permutation s is based on a function of η_{si} and τ_{si}, the ***ant-routing table*** $\mathcal{A}_{si}$.

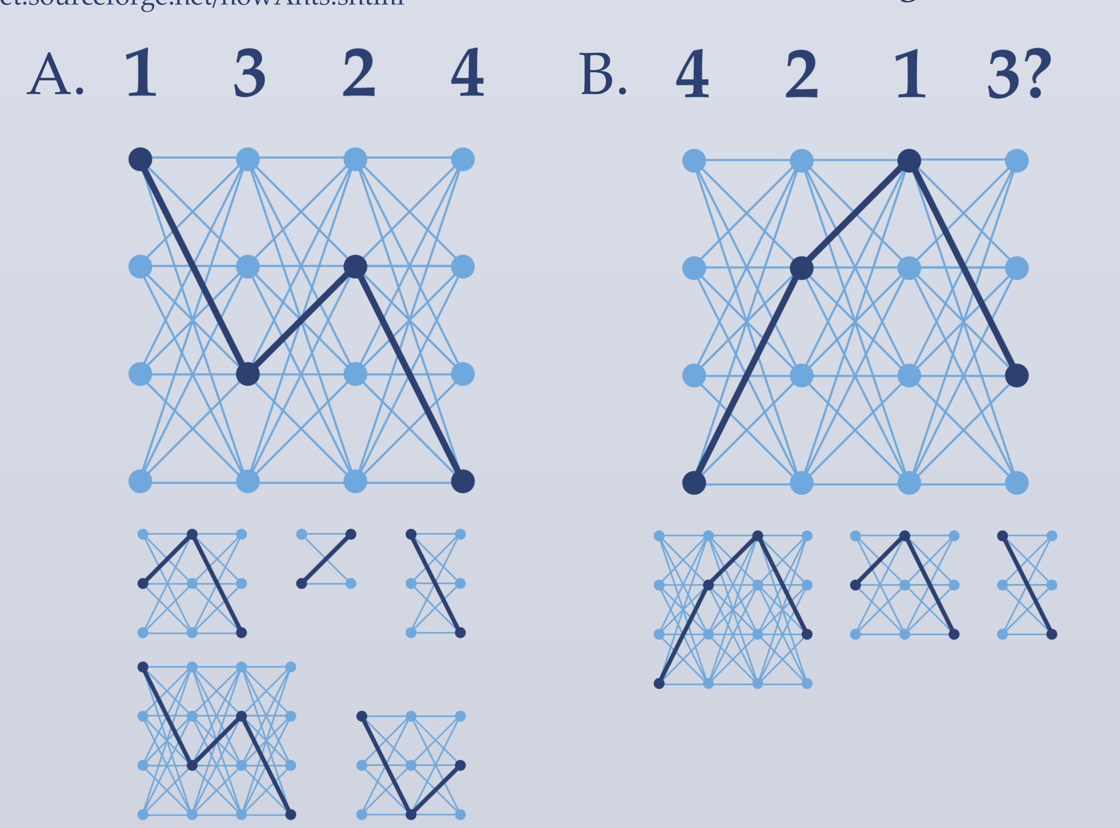

Figure 4: Path representation of permutations for pheromone association. (A) Pheromone is applied to all five unique subsequences of the permutation $(1, 3, 2, 4)$; clockwise from top left $(+1, -2)$, $(+1)$, (-2), $(-2, +1, -2)$, $(-2, +1)$. (B) All suffixes of the permutation $(4, 2, 1, 3)$ are considered when computing the ant-routing table $\mathcal{A}_{s3}$ at $s = (4, 2, 1)$; from left to right $(+2, +1, -2)$, $(+1, -2)$, (-2).

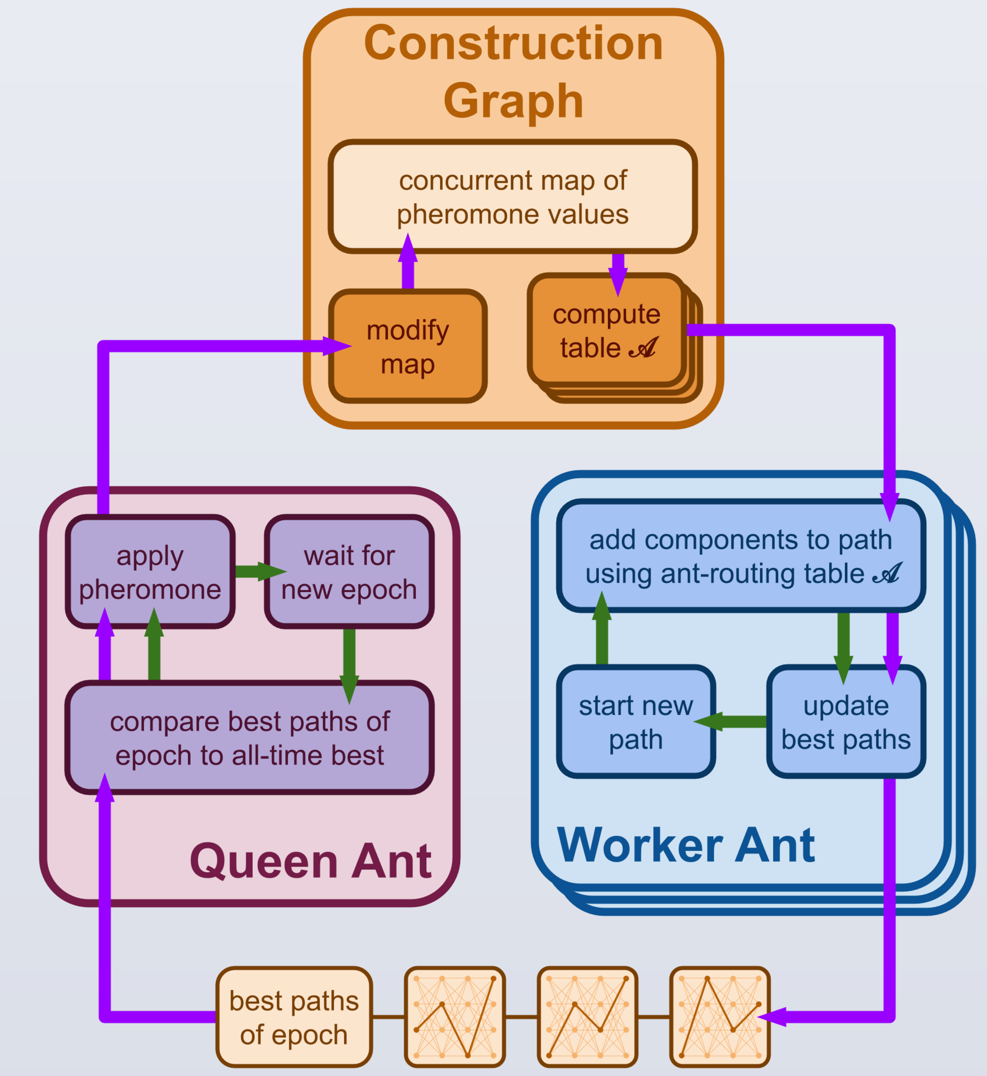

Figure 3: A high-level schematic of the parallel $\mathcal{MM}$AS framework. Purple arrows indicate the flow of information through the system; green ones indicate the flow of control.

The $\mathcal{MM}$AS framework includes two types of processes that run in parallel and indirectly communicate with each other through two shared data structures (Fig. 3).

Some number N of parallel worker ants repeatedly traverse the construction graph, constructing paths that represent possible solutions.

- Worker ants construct paths stochastically, favoring paths with high heuristic and pheromone values.
- The best (lowest-cost) paths are inserted into a shared list.
- As a new optimization, ants constructing a path the cost of which exceeds the cost of the best path found by a ***quality threshold*** ϑ abandon the path and immediately start a new one.

A single queen process periodically updates pheromone values.

- The queen operates based on ant time periods called epochs; over one epoch, each ant constructs a set number of paths.
- The queen applies pheromone to all subsequences of the best paths found (Fig. 4A).
- The amount of pheromone applied to a path's subsequences is inversely proportional to that path's cost.

The construction graph supports parallel querying of pheromone values to compute the ant-routing table $\mathcal{A}$.

- As a new optimization, the construction graph uses the longest suffix encountered when determining pheromone values (Fig. 4B).

VISUALIZATIONS

Simplified visualizations of the evolution of the pheromone graph over the course of $\mathcal{MM}$AS Costas-array searches show that the final solution found by the ants utilizes high-pheromone components of non-Costas array permutations (Fig. 5).

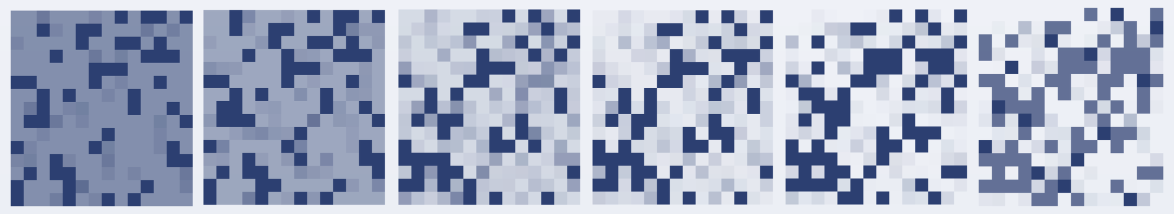

Figure 5: Visualizations of the 2D pheromone graph for an $M = 14$ $\mathcal{MM}$AS Costas-array search. From left to right, the state of the graph at time $t = 5$, $t = 15$, $t = 30$, $t = 45$, $t = 60$, $t = 75$, and $t = 90$ (in ant epochs). The color of a cell in the i^{th} row and j^{th} column represents the learned favorability of following i with j in a Costas-array candidate, with darker cells indicating higher pheromone concentration and lighter cells representing lower pheromone concentration.

RESULTS

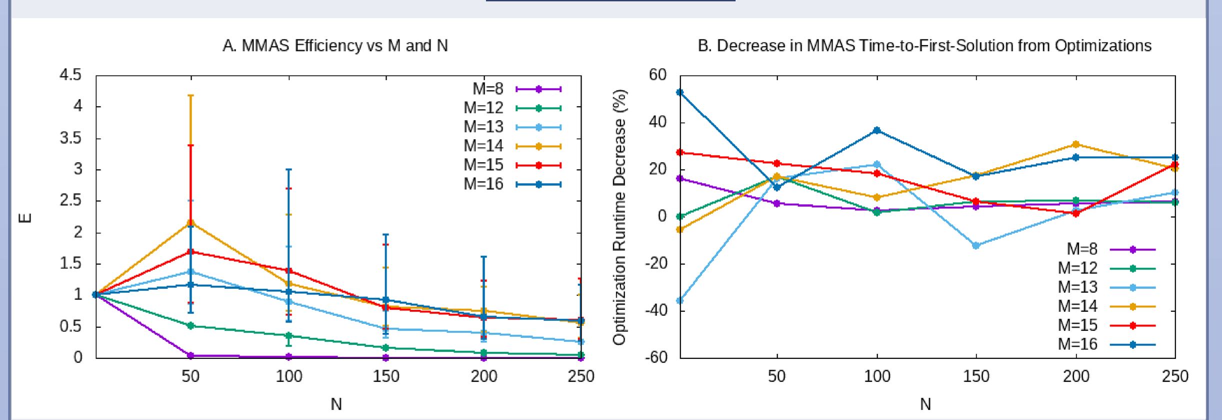

Figure 6: Graphs of $\mathcal{MM}$AS performance. (A) A graph of median E (as a ratio of times-to-first-solution) with respect to N and M ($n = 100$). Error bars represent the first and third quartiles. (B) A graph of decrease in average time-to-first-solution after new optimizations were added.

Fig. 6A indicates that, for $M \geq 14$, $E \geq 0.5$ for all values of N tested.

- For $N < 200$, E varies with M and decreases as N increases.
- For $N \geq 200$, $E \approx 0.6$ for all $M \geq 14$, implying that an efficiency of ~ 0.6 is likely to extend for greater worker counts and problem sizes.

Fig. 6B indicates that the new optimizations generally improve $\mathcal{MM}$AS time-to-first-solution.

- For $N > 1$, $\mathcal{MM}$AS with the new optimizations is consistently faster than $\mathcal{MM}$AS without (exception: $N = 150, M = 13$).
- New optimizations are most effective at larger problem sizes and processor counts; for $N = 250$ and $M \geq 14$, the new optimizations cause a $\sim 30\%$ runtime decrease.

DISCUSSION

The performance data collected indicate that the $\mathcal{MM}$AS framework developed satisfies the project goals:

- $\mathcal{MM}$AS maintains an approximately constant high efficiency across several larger processor counts and problem sizes, indicating that E is likely to remain above 0.5 even for larger M and N.
- The optimizations decrease average $\mathcal{MM}$AS time-to-first solution by a significant margin ($\sim 30\%$).

Future goals for the $\mathcal{MM}$AS program include:

- Application of distributed computing techniques to pool the computational power of multiple machines without shared memory.
- Application of $\mathcal{MM}$AS to unsolved problem instances (namely, the search for $M = 32$ Costas arrays).

WORKS CITED

Bulatov, A. A., Krokhin, A. A., & Jeavons, P. (2000). Constraint satisfaction problems and finite algebras. [Proceedings]. In *International colloquium on automata, languages, and programming* (pp. 272–282). Retrieved from https://link.springer.com/chapter/10.1007/3-540-45022-X24

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. [Proceedings]. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (cat. no.99th8406)* (Vol. 2, pp. 1470–1477). Retrieved from http://staff.washington.edu/paymana/swarm/dorigo99-cec.pdf

Drakakis, K. (2011). Open problems in Costas arrays. *arXiv preprint arXiv:1102.5727*. Retrieved from https://arxiv.org/pdf/1102.5727.pdf

Russell, S., & Norvig, P. (2009). *Artificial intelligence: A modern approach (3rd ed.)*. Upper Saddle River, NJ, USA: Prentice Hall Press.

Stützle, T., & Hoos, H. H. (2000). Max–min ant system. *Future Generation Computer Systems, 16*(8), 889–914. Retrieved from https://www.sciencedirect.com/science/article/pii/S0167739X00000431

All unattributed figures produced by student.