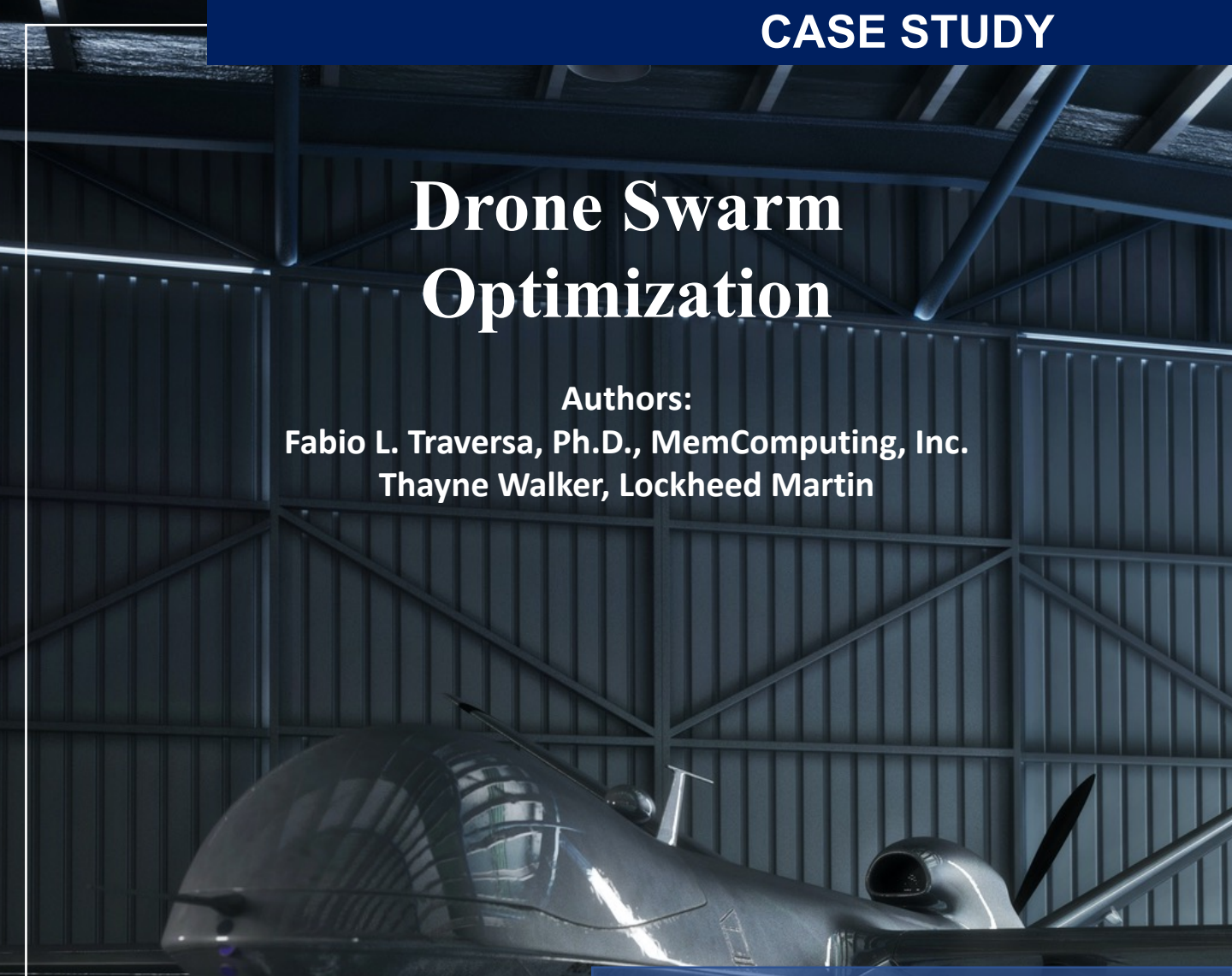
CASE STUDY

Drone Swarm Optimization

Authors:
Fabio L. Traversa, Ph.D., MemComputing, Inc.
Thayne Walker, Lockheed Martin



Learn how MemComputing solves a critical defense application for Lockheed Martin. The MEMCPU™ Platform scales to solve this problem far beyond current capabilities in tactically relevant timescales. The broad nature of this problem is relevant across numerous industries, providing a great opportunity for a variety of applications.

MemComputing, Inc.

Introduction

Artificial Intelligence (AI) is rapidly evolving thanks to advances in technology and innovative applications. From virtual assistants to self-driving cars, AI is being deployed across nearly every industry. However, there are still great computational challenges in developing advanced AI.

One well-studied problem is known as the **Multi-Agent Path Finding (MAPF) problem** [1]. The MAPF problem supports applications in warehouse automation, traffic optimization, package delivery, robotics, and autonomous vehicles. “Agents” typically refer to mobile autonomous systems like drones, robots, or other autonomous vehicles. The objective of this problem is to find paths by which many agents can safely and efficiently navigate from their respective starting positions to their respective destinations concurrently. Additionally, the agents must not block, collide, or otherwise interfere with each other, and their paths must be optimal, making this problem computationally challenging. There are tremendous benefits to solving this problem optimally or close to optimality, as it will deliver improved operational efficiencies and significant cost savings, thus driving strategic competitive advantages.

Global ecommerce giants like Amazon and Alibaba face the MAPF problem daily in their automated warehouse where robots play a critical role. Robots run 24/7 and automate their customer fulfillment performing tasks like picking, transporting, and placing goods across their massive warehouses that are many city blocks in size.

Case Study

In this case study, we show how our cloud based **MEMCPU Platform** is used to solve a complex MAPF problem for Lockheed Martin Corporation, one of the largest aerospace and defense contractors in the world. The inspiration for the use case was assuming **the agents were a swarm of drones**, each drone with its own specific mission. However, the problem applies broadly regardless of the type or purpose of the agent.

Due to the intractable nature of this problem [2], current approaches, even those used by Amazon and Alibaba, rely on **heuristics that can only provide approximate solutions**. However, approximations can lead to situations where agents must take evasive action to avoid one another or in some cases can remain stuck in deadlock. This could certainly lead to catastrophe. Here we will show how to solve this problem and problems like it using our MEMCPU Platform **on mission-scale sizes within tactically relevant timeframes.**

LOCKHEED MARTIN

Lockheed Martin is an American global security and aerospace company and is one of the largest defense contractors for the US government. Lockheed Martin is ranked 57th on the Fortune 500 list of largest industrial corporations in 2020, reporting over $65B in revenue.

For this case study, we look at the Multi-Agent Path Finding (MAPF) problem [1] and consider that the agents can be a swarm of drones. **From aerial reconnaissance to disrupting enemy communications** and even engaging enemy forces, there is a great need for battle managers to develop advanced drone swarming technologies. The coordinated planning and assignment of drones at scale could deliver significant operational advantages and greatly enhance the capabilities of the warfighter.

Problem Overview

Let us review the basics of the MAPF problem and why it is so hard to solve. Again, **the goal is to find flight plans for multiple agents in an environment without interfering with each other**, while providing the optimal path for each drone.

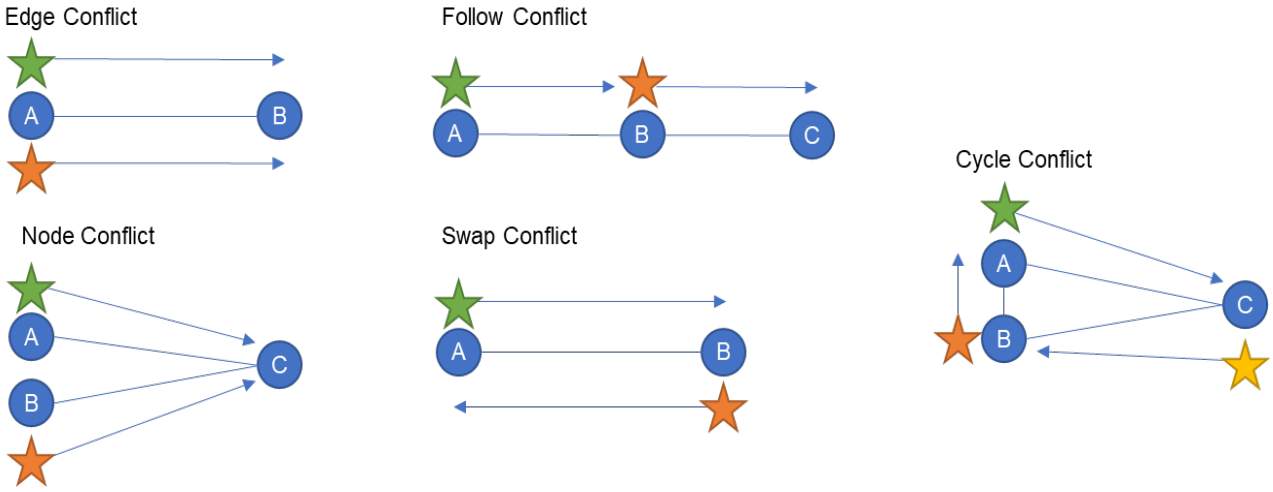
The map of travel is described by a graph. For example, a 2D map can be encoded as a 2D grid with edges connecting nearest neighbors, with four connections along the cardinal, or eight connections when including diagonal paths. Each edge might have an associated cost to traverse. Typically, this cost is the travel time but can be some other cost as defined for the mission.

In order to find a feasible solution for a group of agents, we must consider the potential for conflicts during planning. A MAPF solution is considered valid if there are no conflicts between any two single-agent plans. In this case study, we account for a number of potential conflicts:

- Edge conflicts where agents traverse the same edge during the same timestep.
- Node conflicts where agents arrive at the same node during the same timestep.
- Swap conflicts where two agents attempt to occupy each other's previous location.

Additional conflicts could easily be added to the model like:

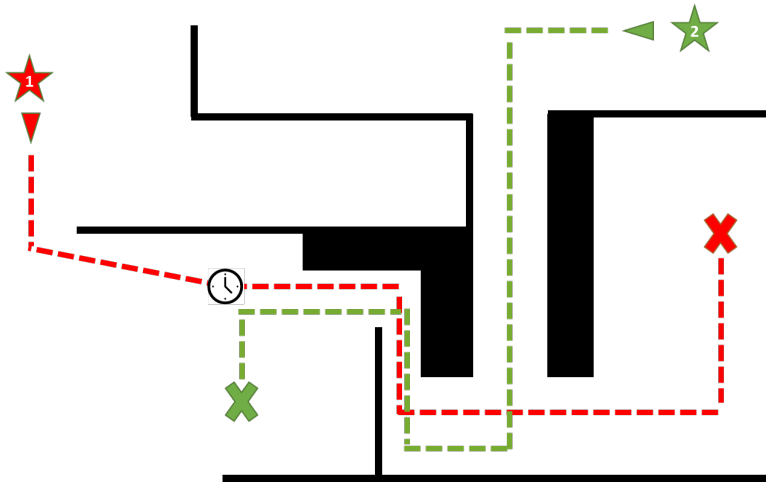
- Follow conflicts where agents occupy a previously occupied node.
- Cycle conflicts where multiple agents attempt to swap previous locations.



Multi-Agent Path Finding is a combinatorial optimization problem. That is, the complexity of avoiding collisions can grow more than polynomially as the number of agents increase linearly [2, 3]. A simple example of a MAPF problem is described in the next section.

A simple example of a MAPF Problem:

To provide an understanding of how difficult this problem is, let's consider **just two agents** on a small map. Optimal paths must be found for Agent 1 and Agent 2 to move from their starting positions to their destinations. There are many possible routes for each of them to reach their destination. Because the two must avoid conflicting with one another, the choice of route for one depends on the choice of route for the other, and moreover, it cannot be assumed that Agents move linearly on the shortest paths across the map.



For the example shown above, Agent 1 and Agent 2 start at a specified position (marked by the stars) and must reach their destination (marked by the crosses) in the shortest total time without colliding with each other or obstacles. Both paths are the shortest paths, however if the agents were to meet up where the paths intersect at the same time, the agents would collide. So, a possible choice is having Agent 1 traversing a path that includes waiting (shown by the clock icon) and accommodating the path of Agent 2, so that both may go around the narrow corner without colliding. An alternative scenario would have Agent 2 waiting for Agent 1. One would choose the scenario that minimizes an objective function like the total time or the maximum arrival time of the agents (makespan). Add one more agent and you can see how the problem complexity grows.

A very simplified estimation of the number of possible scenarios can be, for example, evaluated considering *n* agents, each of them with only one collision to be solved with each of the other agents and moreover the paths are hold fixed. In this case the potential number of scenarios grows super-exponentially like $2^{(n^2-n)/2}$ even though the number of agents *n* only grows linearly.

A full MAPF problem would also include the possibility to choose alternative paths on top of just scheduling passages, and multiple collisions among agents which adds significant complexity [2, 3]. This simple example aside, the **full MAPF problem is among the hardest optimization problems faced by industry and studied by academia**.

Case Study Continued

Each MAPF mission has an objective that drives the optimization when determining the paths associated with each agent. In the literature, the most common objective is the Sum of Costs [1, 4]. **The Sum of Costs minimizes the total time for all agents combined**, reducing the required resources to complete all paths. However, one can also minimize the makespan, i.e., minimize the longest path of all agents, minimizing the time to complete all paths [4]. While we ran studies on both the Sum of Cost and Makespan objectives, for brevity, we will just compare and contrast with the Sum of Costs in this case study.

We evaluated groups of agents ranging from 10 to 40 agents, where each agent had independent starting and target locations. As we have mentioned, these scenarios grow super-exponentially as the agents grow linearly. So, consider a scenario where each agent has 50 potential paths (a small number in real-life), we have a total of 50^n combinations of paths. For a group of 10 agents this would be about 10^{17} combinations, each having up to $2^{(n^2-n)/2} = 2^{45} \approx 10^{13}$ possible scheduling combinations, leaving 10^{30} scenarios to be considered. For 40 agents this grows to 10^{302} (there is only an estimated 10^{82} atoms in the observable universe!) scenarios making the problem completely intractable for classical technologies. It would literally take **longer than the age of the universe** to consider all these scenarios using today's fastest supercomputer. Heuristics can be used to quickly prune scenarios and help reduce the burden a bit, but unfortunately these tricks don't change the scaling of the problem, leaving it intractable for these real-world sizes [5].

Indeed, there is an urgent need to solve these problems at scale and in a timely manner using new, alternative computing architectures.

Autonomous Drone Swarming

In an autonomous swarm, each drone has its own 'mind' and flies itself using onboard AI to maintain formation, avoid collisions, and make decisions in real time. They can adapt their flying formations and re-organize autonomously to continue the mission. If each agent has knowledge of the globally optimized paths, they can use it to inform their individual-based decisions. This is possible with a MemComputing ASIC but is otherwise unthinkable for current technology (for a large number of agents).

Approach

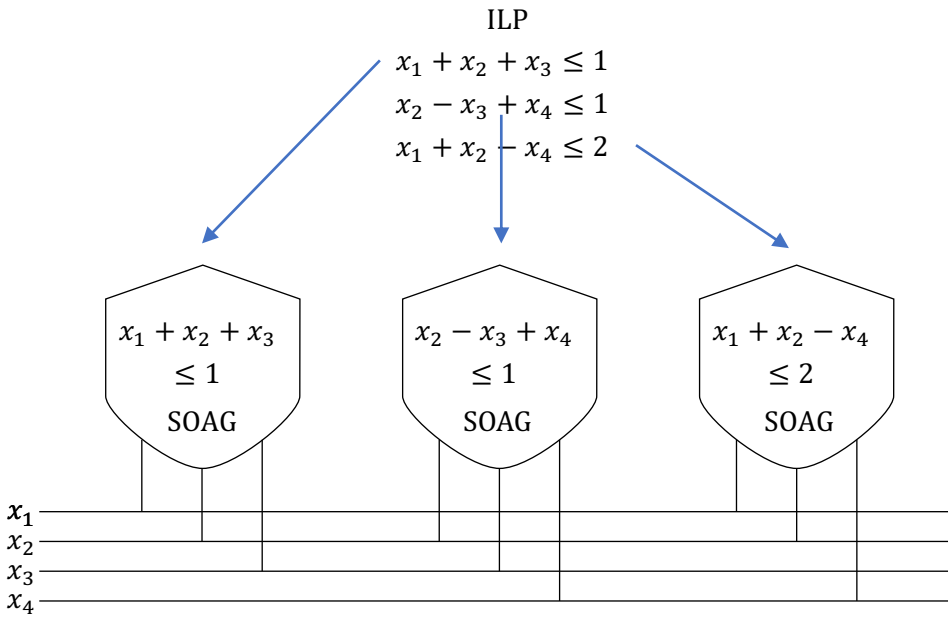
The case study was run as a collaboration between Lockheed Martin and MemComputing. **Lockheed Martin scientists are experts** in using optimized and specialized algorithms for MAPF problems. They have a long history of development and testing for internal applications as well as several academic research projects.

Lockheed Martin:
The preferred algorithm in academia and used by Lockheed Martin is the **conflict-based search algorithm (CBS**) [6]. The CBS algorithm is an optimal, dynamically-coupled, two-level A* graph search algorithm devised to solve MAPF problems defined on graphs, possibly involving multiple dependent paths. It is widely used for its efficiency in the graph search context and the possibility of providing a proof of optimality. While CBS is performant in many scenarios, scientists still cannot fully explain or predict its behavior based on problem structure alone [7], which results in very unpredictable runtimes.

- For this case study, a version of the CBS algorithm specialized for unweighted graphs, developed by the University of Southern California [8], was employed. It is considered state-of-the art [9]. Since it is a specialized algorithm, it only requires the graph in DIMACS format and the starting and ending location of the agents as input.

MemComputing, Inc.:
MemComputing used its cloud-based **MEMCPU Platform** (www.memcpu.com). It is based on the proprietary design of Self-Organizing Algebraic Gates (SOAGs) [10, 11]. Each SOAG is an electronic circuit design and the MEMCPU Platform virtualizes (or emulates) them on conventional hardware [12-15].

- For this case study we ran the MEMCPU Platform on cloud-based NVIDIA V100 GPUs. Each SOAG encodes a linear inequality for integer variables. The variables are encoded in the voltages at the terminals of the SOAGs [10, 12]. If the state of the variables satisfies the inequality, the SOAG is in equilibrium. Otherwise, current/voltage feedback mechanisms drive the change of the state of variables towards the equilibrium for the SOAG [10, 12]. Therefore, **the MEMCPU Platform can be used as a general-purpose Integer Linear Programming (ILP) solver** by encoding linear inequalities on SOAGs [10]. This results in a network of interconnected SOAGs where interconnects correspond to variables shared by different SOAGs. In this scenario the equilibrium of the entire SOAG network implies that all SOAGs are satisfied contemporarily, and therefore one can read the solution of the ILP directly encoded in the network [10]. A sketch of the embedding is reported in the figure below. To use the MEMCPU Platform to solve the MAPF problem we reformulated it in ILP.



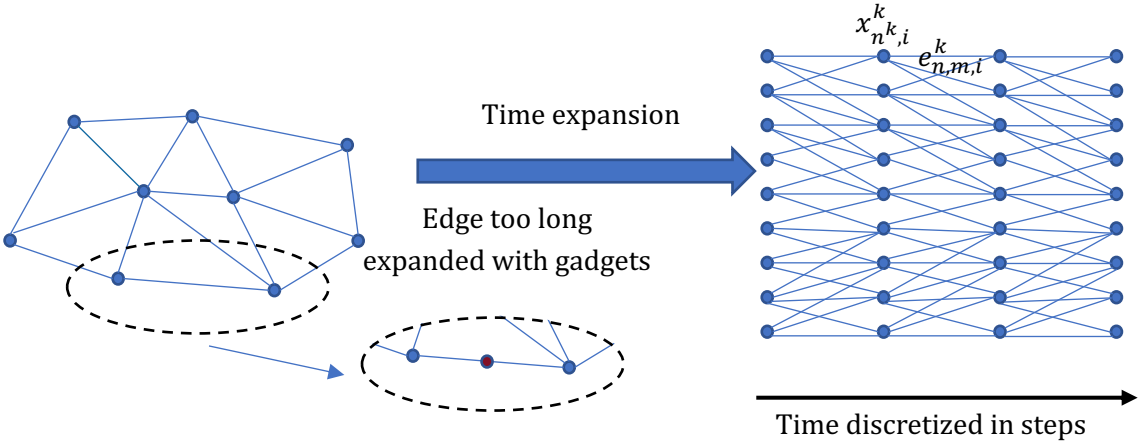
IBM CPLEX:
The graph formulation of the MAPF problem can be recast exactly in an ILP format without introducing any approximation or change to the model. Thus **the same ILP formulation** is used for both MEMCPU Platform and CPLEX. CPLEX is a **state-of-the-art commercial solver** developed by IBM [16]. CPLEX implements branch and bound on the cutting planes method that is referred to as branch and cut [17]. This algorithm is boosted by a collection of heuristics that accelerate the convergence of the branch and cut process by preprocessing the problem, finding feasible solutions, determining smart branches, and more. CPLEX is recognized as one of the best ILP solvers. For this case study, scientists from Lockheed Martin oversaw the setup and running of the ILP problems on CPLEX.

Integer Liner Programming Format

The Integer Linear Programming (ILP) formulation of the MAPF problem is not unique and it can be recast in several ways. Two main families of ILP formulations can be introduced: the time expanded graph and the graph with node transit time descriptions. Both are equivalent but present different size scaling properties in terms of graph size and number of agents. For each of them, variants are possible depending on the description of the graph (e.g., through edges and nodes, or only edges) and the definition of the constraints using linear inequalities.

For this case study we explored several variants and selected a variant of the time expanded graph with node-edge description. The selection was chosen because it provides the flexibility needed for this case study and it provides the most compact representation (linear in the product #agents × #edges × #time steps) for the benchmark we considered in this study.

ILP Format Continued



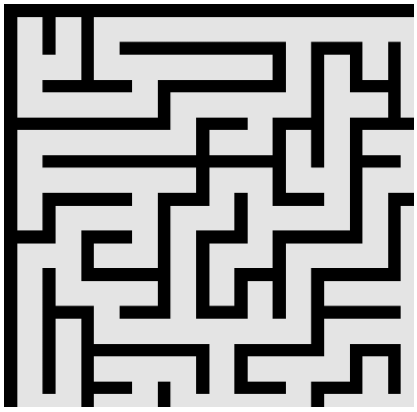
The figure above represents the time expansion method where each node of the graph is repeated the same number of time steps we want to consider for the dynamics of the agents. Each node is connected to itself, but at the next time step. That represents the possibility that an agent does not move during that time step. This node is also connected to all other nodes since the original graph shares an edge with this node. With this setup we can define the motion of the agents as well as the conflicts through the following set of linear inequalities:

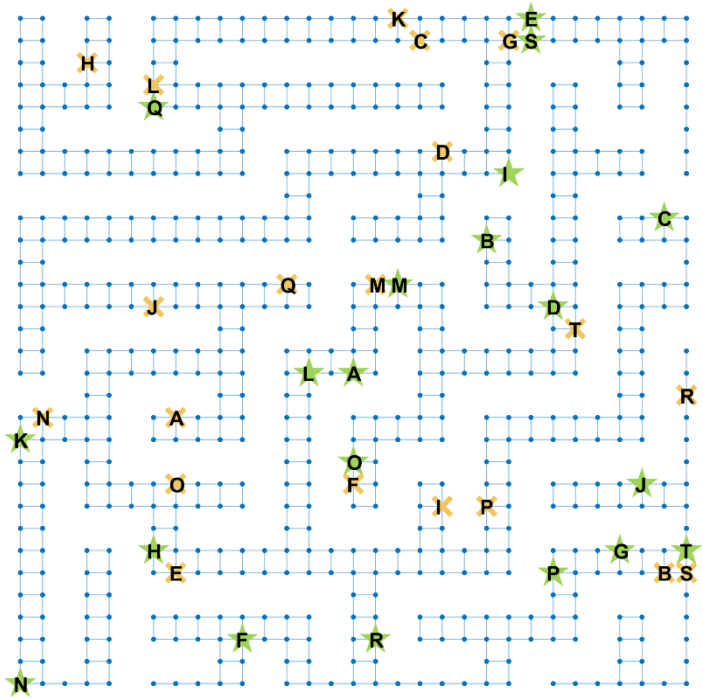
- (1) $x^k_{n=s^k,i=1} = 1,\ x^k_{n\neq s^k,i=1} = 0$ | Initialize all agents at source positions
- (2a) $\sum_n x^k_{n,i} = 1$ | Each agent visits 1 location every timestep
- (2b) $\sum_n x^k_{n+1,i} \leq \sum_n x^k_{n,i}$ | Each agent visits 1 location every timestep
- (3a) $\sum_i -x^k_{n=t^k,i} \leq -1$ | The agent must arrive to the target
- (3b) $\sum_i x^k_{n=t^k,i} = 1$ | The agent must arrive to the target
- (4) $e^k_{r^k,j,i} = 0 \quad \forall j \in \alpha^-(r^k)/r^k$ | Only self edges from target vertices
- (5) $x^k_{n\neq r^k,i_{max}} = 0$ | Ensures no agent's last stop is at a node other than the target.
- (6) $\sum_k x^k_{n,i} \leq 1$ | only one agent can be at a node at a given time i
- (7a) $\sum_k x^k_{n,i} + x^k_{n,i+1} - e^k_{n,n,i} \leq 1$ | Follow, Cycle and Swap Conflict Constraint
- (7b) $\sum_k e^k_{n,m,i} + e^k_{m,n,i} \leq 1$ | Forbid Swap only
- (8) $\sum_j e^k_{n,j,i} - x^k_{n,i} \leq 0 \quad j \in \alpha^-(n)$ | Path Definitions
- (9) $\sum_j e^k_{j,n,i} - x^k_{n,i+1} = 0 \quad j \in \alpha^+(n)$ | Path Definitions
- (8a) $\sum_{m\in\alpha^+(n)} x^k_{m,i-1} - x^k_{n,i} \geq 0 \quad j \in \alpha^-(n)$ | Path Definitions
- (9a) $x^k_{n,i-1} + x^k_{m,i} - e^k_{n,m,i} \leq 1 \quad j \in \alpha^-(n)$ | Path Definitions
- (10) $minimize\ Objective = -\sum_{i,k} x^k_{t^k,i}$ | Objective Function

This set of inequalities represents the ILP formulation that was used in this case study for both the MEMCPU Platform and CPLEX. It is also worth noting that to compact the problem somewhat, we employed a few preprocessing schemes to pre-solve part of the variables. For example, it is possible to exclude the nodes and edges that an agent would never reach within the maximum time considered.

Benchmark Description

The following map is the benchmark map for the MAPF problem being studied. The map is from the collection of benchmarks by Prof. Nathan R. Sturtevant's Moving AI Lab, a lab specialized in combinatorial optimization algorithms [18].





The map we selected is the maze reported on the right panel of the figure. On the left panel the corresponding graph represents the possible paths and corridors agents can take on the maze. The green stars on the graph represent the beginning location of each agent, while the light orange crosses represent the target locations (each letter represents the starting and target location for a single agent). Keeping in mind that numerous problem sets were run for each group size of agents, this graph represents one of the problem instances that was run using 20 agents.

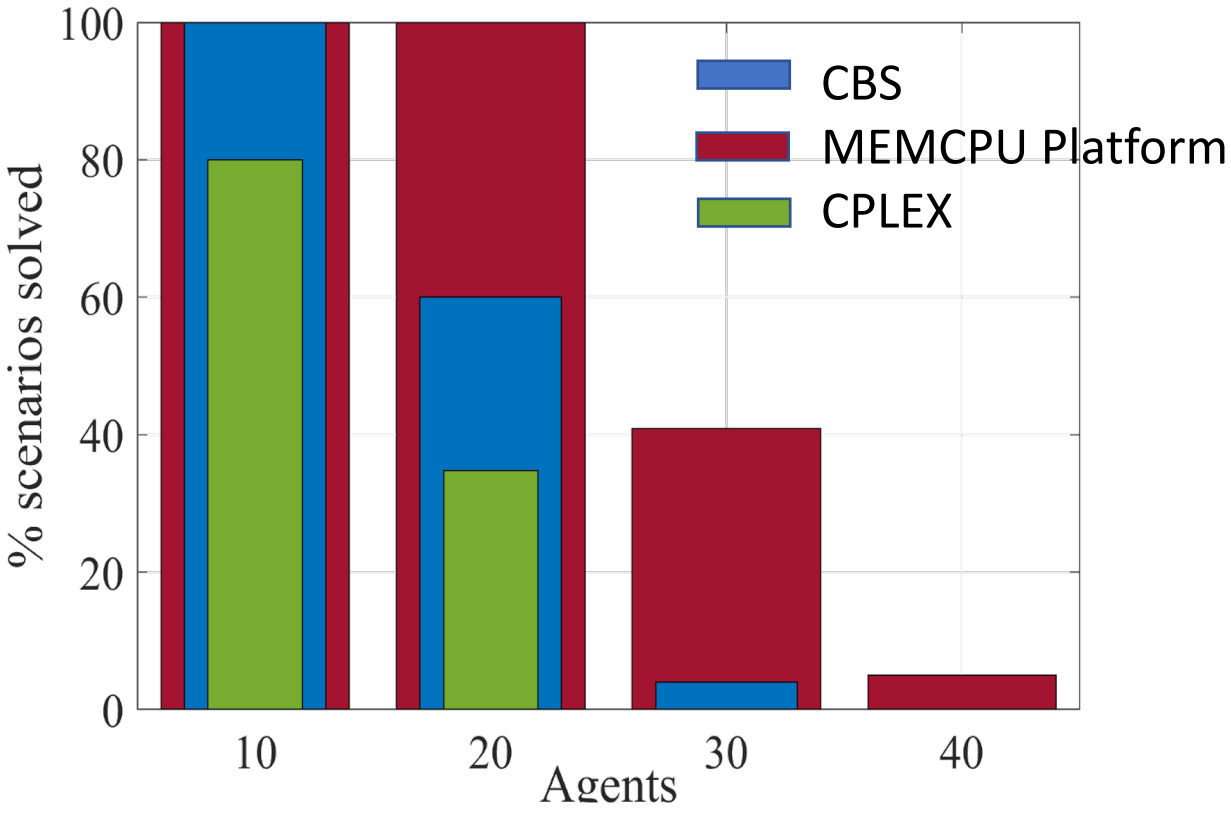
Each technology was presented with data sets representing groups of agents with **10, 20, 30 and 40 agents**. Each agent had a pre-selected starting location and desired target location that was identical for all three technologies considered. This approach demonstrates the performance of each technology as the problem scales to larger sizes.

The final item of note is that for each group size (i.e., 10, 20, 30 & 40 agents), a total of 25 different mission datasets with randomized start/goal locations were evaluated. **Each technology was given 10 minutes to solve each problem**. However, in real-life, more or less time may be afforded as determined by the application.

The maximum size was cut at 40 agents since the A*-based CBS algorithm (which was the reference for MAPF problem) topped out at 30 agents where it also showed an exponential increase in computing time. This motivation can easily be understood by looking at how dense the maze becomes with agents, causing the number of possible conflicts to grow rapidly, rendering it intractable very quickly. This decision was supported because scaling the ILP formulation to larger sizes would have required increasing the maxim number of time steps as the average wait times for the agents would grow in relation to the growing potential for conflicts.

Results

The histogram in this figure shows the results of each technology in a consistent color. The X-axis represents the number of agents, and the y-axis represents the percentage of problems solved associated with each group size. Recall that 25 different problems were run for each swarm size.

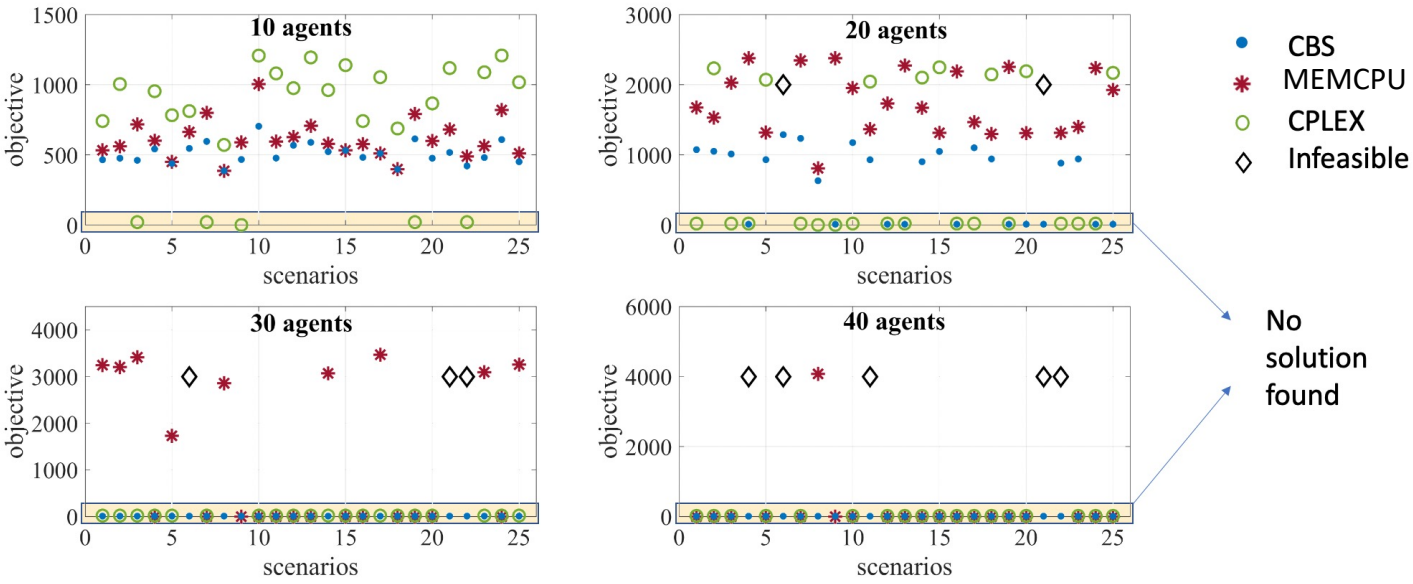


- The histogram reveals that the **MEMCPU Platform provides a solution to the MAPF problem at every instance**
- Both the A*-based CBS and CPLEX are combinatorial algorithmic approaches. As expected, CBS does better than CPLEX, as CBS is specialized for the MAPF problem while CPLEX is a general purpose solver.
- The MEMCPU Platform performs better than the CBS algorithm, thus it compares even better against CPLEX.
- Therefore, **these results clearly show** that the superior performance of the MEMCPU Platform with respect to the A*-based CBS algorithm is not related to the different formulation but is due to the fundamentally different approach to computing combinatorial optimization problems.

Results Continued

The sum of costs of the best solutions found for each scenario are plotted in more detail in the figure below. The specialized method CBS allows it to find the optimal solutions for all the 10-agent problems within the 10 minutes allotted. It is worth mentioning that the CBS algorithm applied to MAPF either returns the optimal solution or no solution at all because of its nature.

In contrast to the CBS, CPLEX and the MEMCPU Platform are general purpose ILP solvers that typically find an initial feasible solution (in the case of MAPF it means the path for all agents where the conflicts are solved but not necessarily the path with the lowest sum of costs) and then continue to improve, finding better paths, as they run. **The results from the MEMCPU Platform are uniformly better** than the results from CPLEX, evident in that the MEMCPU Platform finds a solution for all 10-agent scenarios and the solution is closer to, and sometimes matches, the global optimum provided by the CBS based solver.



For larger problems with 20 or more agents, the MEMCPU Platform continues to uniformly find solutions approaching the best value within the 10-minute run. **The advantage of the MEMCPU Platform becomes clearer as the problem scales larger**. The MEMCPU Platform is the only solver to find a feasible solution for any of the 40 agent scenarios.

Conclusion

Multi-agent path finding (MAPF) is a challenge that is rapidly expanding in importance due to the growth and promise of unmanned vehicles (including drones, autonomous cars, warehouse robots, etc.) and one that is benefiting from novel computing technologies.

The work for this case study was testing different technologies to solve the MAPF problem within a limited window of time. The main conclusion is that the **MEMCPU Platform developed by MemComputing, Inc. is competitively superior** to the specialized A*-based CBS search algorithm, developed specifically for the MAPF problem class, as well as more general-purpose solvers for Integer Linear Programming problems, such as CPLEX. As expected, CPLEX performed worse than CBS, as CBS is a specialized approach to solving the MAPF problem. These results demonstrate how the MEMCPU Platform, hence MemComputing, is well suited to efficiently solve combinatorial optimization problems like MAPF.

The acceleration of the MEMCPU Platform increased as the problems grew in scale. The MEMCPU Platform was the only solver to find solutions to all 20-agent scenarios, to most of the 30 agent scenarios, and the only one to find a solution at 40-agents. Being a general ILP solver, the MEMCPU Platform is more flexible and able to expand the scope of the MAPF problem. Using longer runtimes (beyond 10 minutes) would allow the MEMCPU Platform to continue improving its results.

Bottom line: These findings support the advantage of applying MemComputing concepts, including the MEMCPU Platform, to multi-agent AI optimization applications.

Further thoughts:
While not included in this case study, **the MemComputing solution can be accelerated even further**. For example, there are nonlinear types of implementations of the MAPF problem that we have explored that we estimate could provide an acceleration of more than 100X on the MEMCPU Platform. Ultimately the MAPF problem would be best served running on MEMCPU Platform realization on FPGA, or better yet, a MemComputing based Application Specific Integrated Circuit (ASIC). Behind the scenes, the MEMCPU Platform is the emulation of an electronic circuit, thus taking the MEMCPU Platform all the way to a specialized ASIC is on our product roadmap. A MemComputing ASIC would provide a **near-real time solution to the MAPF problem** of almost indeterminate scale.

References

[1] R. Stern, et al., Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks, Proceedings of the Twelfth International Symposium on Combinatorial Search (SoCS 2019)

[2] J. Yu, S. M. LaValle, Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs, Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013

[3] B. Nebel, On the Computational Complexity of Multi-Agent Pathfinding on Directed Graphs, Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

[4] P. Surynek, A. Felner, R. Stern, E. Boyarski, An Empirical Comparison of the Hardness of Multi-Agent Path Finding under the Makespan and the Sum of Costs Objectives, Proceedings of the Ninth International Symposium on Combinatorial Search (SoCS 2016)

[5] M. R Garey, D. S. Johnson, Computers and intractability, W. H. FREEMAN AND COMPANY New York, 1979

[6] G. Sharon, R. Stern A. Felner, N. R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence, 219, 40, 2015

[7] O. Gordon, Y. Filmus, O. Salzman, Revisiting the Complexity Analysis of Conflict-Based Search: New Computational Techniques and Improved Bounds, Proceedings of the Fourteenth International Symposium on Combinatorial Search (SoCS 2021)

[8] H. Zhang, J. Li, P. Surynek, S. Koenig, T. K. S. Kumar, Multi-Agent Path Finding with Mutex Propagation, Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

[9] O. Kaduri, E. Boyarski, R. Stern, Experimental Evaluation of Classical Multi Agent Path Finding Algorithms, Proceedings of the Fourteenth International Symposium on Combinatorial Search (SoCS 2021)

[10] F. L. Traversa, M. Di Ventra, Memcomputing integer linear programming, arXiv preprint arXiv:1808.09999, 2018

[11] F. L. Traversa, M. Di Ventra, Self-Organizable Logic Gates and Circuits and Complex Problem Solving with Self-Organizing Circuits, US patent US 9911080 B2 filed on July 12, 2015, published 2018

[12] F. L. Traversa, M. Di Ventra, Polynomial-time solution of prime factorization and NP-complete problems with digital memcomputing machines, Chaos: An Interdisciplinary Journal of Nonlinear Science, 27, 023107, 2017

[13] M. Di Ventra, F. L. Traversa, Perspective: Memcomputing: Leveraging memory and physics to compute efficiently, Journal of Applied Physics, 123, 180901, 2018

[14] F. L. Traversa and M. Di Ventra, “Universal memcomputing machines,” IEEE Trans. Neural Netw. Learn. Syst., 26, 2702, (2015)

[15] A full list of paper on peer reviewed articles on memcomputing can found at https://www.memcpu.com/publications/

[16] https://www.ibm.com/analytics/cplex-optimizer

[17] S., Alexander. Theory of linear and integer programming, John Wiley & Sons, 1998.

[18] https://www.movingai.com/