EvoMAPF: MAPF-LNS Heuristic Discovery with Evolutionary Code Agents

Lufan Yang, Tony Shen, Jiaqi Tan, Hang Ma
Simon Fraser University

Overview

Problem. MAPF-LNS is the SOTA anytime MAPF solver, and its performance hinges on the **neighborhood-selection heuristic**. Existing heuristics (RandomWalk, ADP, RWP, ADDRESS) are hand-crafted through labor-intensive trial-and-error, and learning-based methods are upper-bounded by the rule heuristics underneath.

Our proposal. We introduce **EvoMAPF** — the first framework to harness

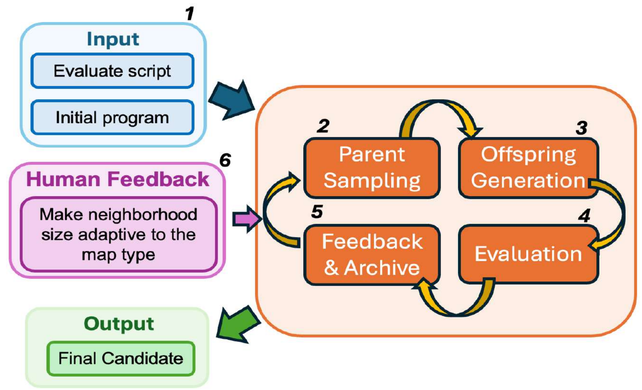
Key idea. Self-evolving agents propose, evaluate, and refine candidate heuristics as *code*. An **expert-in-the-loop** mechanism injects researcher insight mid-evolution, yielding heuristics that **outperform hand-crafted SOTA** and **generalize across diverse maps**.

Quantitative Results

Highest agent density · 300s cutoff · 5 seeds. Red bold = best, black bold = 2nd best.

Method	empty	random	wareh.	ost003d	den520d	Paris
RW	3723.5	4188.1	1035.3	4474.1	1776.3	380.5
ADP	3809.1	4236.8	**1033.2**	4863.4	1980.2	370.5
RWP	3804.5	4208.0	1118.2	3769.0	1106.1	381.6
ADDRESS	3784.1	4141.7	1098.2	4282.6	4064.1	465.4
Warehouse	3693.6	4191.7	1050.0	3369.0	1108.7	**339.4**
Den520d	**3678.4**	**4132.0**	1230.6	4868.9	990.1	371.1
Wareh_D520d	3919.5	4630.6	1033.5	7675.8	**941.9**	**362.9**
+ Adpnb	4504.9	5874.0	**1032.2**	**3006.6**	**936.3**	386.3
– Noprior	**3670.0**	**4164.0**	1092.5	**3430.2**	967.8	387.0

Evolutionary Code Optimization & EvoMAPF Pipeline



Initial Program 1

Choose certain heuristics as the initial script for evolve

```
bool LNS::generateNeighborByRandomWalk()
{
    // Seed: most delayed agent
    int a = findMostDelayedAgent();
    // 2. Expand neighborhood via random walks
    while (neighbors_set.size() < neighbor_size && count < 10)
    {
        int t = rand() % agents[a].path.size();
        randomWalk(a, agents[a].path[t].location, t, neighbors_set, ...);
        // 3. Randomly choose another seed agent
        a = random_select_from(neighbors_set);
        count++;
    }
}
```

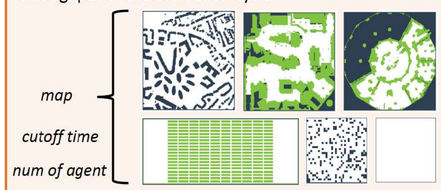
Offspring Generation 3

Prompt offers certain direction for the agent to work on

You are an expert in Multi-Agent Path Finding (MAPF) and Large Neighrborhood Search (LNS) algorithms.
Your task is to evolve a neighbor selection strategy for MAPF-LNS that maximizes solution quality.
Currently this evaluation is running on [map_names] maps.
Key insights from MAPF-LNS literature: ....
The fitness is based on final solution delay- lower is better .
The code to evolve is in the `generateNeighborByRandomWalk()` function between EVOLVE-BLOCK-START and EVOLVE-BLOCK-END markers.

Evaluation 4

Fidelity vs. Computation Cost
- exhaustive map sets and longer time limits yields the most reliable performance benchmarks
- high per-iteration cost significantly restricts the throughput of the evaluation cycle



Gen_X Evaluation Metrics

SumofDelay	ost003d	den520d	warehouse	Avg_map
Medium	1100	540	475	705
Large	8469	1168	1349	3662
Avg_size	4785	854	912	**2184**

Human-in-the-Loop 6

Adjust the task specification to bias the agent's search process toward user-defined directions of interest

The optimal neighborhood size varies across maps. Come up with an adaptive strategy to determine it.

Improve by: (1) benchmark-aligned neighbor_size adaptation using name+geometry detection (2) Stronger expansion on open maps

is_den = name contains "den520d"
is_warehouse = name contains "warehouse" OR > 55% of free cells have only 2 neighbors.

Calculate Congestion: avg_delay = total agent delays / total agents.
Determine Size (ns): Start with 16.
If **is_den**: 12 (if low delay) or 20 (if high delay).
Else if **is_warehouse**: 12 (if low delay) or 24 (if high delay).
Generel: 12 (if low delay).

Apply Limits: Clamp ns between 4 and 32.

Detailed pipeline. Only *generateNeighborBy...()* is the evolvable code block — the LNS machinery is fixed. The LLM mutates delay calc, tabu management, spatial expansion, map-adaptive logic. **Expert-in-the-loop (Step 6)** injects domain hints mid-evolution.

Sum of Delay vs. time of different heuristics