Fast and Memory-Efficient Multi-Agent Pathfinding

Ko-Hsin Cindy Wang & Adi Botea
NICTA & The Australian National University
Outline

- The multi-agent path planning problem + applications
- Related work
- Our method: FAR – Flow Annotation Replanning
- Results
Multi-Agent Path Planning

- Multiple mobile units.
- Shared environment.
- Static obstacles in the environment.
- Dynamic obstacles: other units.
- Navigate every unit to its target.

A difficult problem:
- PSPACE-hard [Hopcroft et al. 1984].
- Often, needs to be solved in real time.

Image source: http://www.supremecommander.com/
Applications

- Robotics motion planning
- Air traffic control
- Vehicle routing
- Disaster rescue
- Military operation planning
- Computer games
Related Work

- **Centralised approaches:**
  - (theoretically) optimal
  - scale up poorly in practice
  - e.g. Randomized Path Planner (RPP) [Barraquand & Latombe 1989]

- **Decentralised approaches:**
  - decompose into subproblems
  - typically faster, sub-optimal, incomplete
  - e.g. Windowed Hierarchical Cooperative A* (WHCA*) [Silver 2006], enhanced with spatial abstraction in [Sturtevant & Buro 2006];

- Subgraph abstraction & planning [Ryan 2008]
Problem Definition

- Grid maps.
- Tiles: accessible (free or occupied); blocked.
- Homogenous agents, uniform speed.
- A legal move:

  1 for a cardinal move,
  \( \sqrt{2} \) for a diagonal move.

Baldrur's Gate
Map AR0700 (320x320 tiles)

Ko-Hsin Cindy Wang & Adi Botea
The FAR Method

1. Build a flow-annotated search graph.

2. Run a complete A* search for each unit independently.

3. Execute the plan:
   - Avoid replanning;
   - Otherwise, do local plan repair.
Flow-Annotated Search Graph (1)

Step 1. Abstract the grid map into a directed graph with controlled navigation flow:

- Directed edges.
- Alternate horizontal/vertical flows to adjacent rows/columns.
- Cover entire grid with criss-crossing virtual roads.
- Initially, no diagonals.
Map Connectivity
Step 2. Additional rules ensure all adjacent nodes remain connected both ways.

- **Single-width tunnel**: bi-directional.
- **Source/sink** nodes:
  - add a diagonal incoming/outgoing edge.
  - if leading to another source/sink, make edges bi-directional instead.
Search and Execution

- **Try to avoid replanning:**
  - favour straighter paths on equal $f$-values.
  - temporal reservation: $(x,y,t)$ for $k$ steps ahead.
  - waiting.
  - traffic lights: temporal flow regulation.

- **Otherwise, when replanning has to be done:**
  - local replanning.
  - detect and break *deadlocks*. 
Deadlock Procedures

- An arbitrary size cycle of units waiting for each other to move [Coffman et al. 1971]:

- Deadlock detection launched frequently: to identify and fix deadlocks early.

- Deadlock breaking: a critical unit takes a small detour.
**Node density:**

# computed paths passing through a node.

For a unit in deadlock, the higher the density at its location, the more units are blocked.

Select a unit at the highest density node.
After selecting a critical unit, $u'$

- Let $u'$ take a step away from the deadlock, respecting the flow annotation,
- Then $u'$ replans its way back at the next time step,
- Meanwhile, units blocked by $u'$ have a chance to pass through.
Experimental Setup

- 2.8GHz Intel Core 2 Duo Mac, 2GB RAM.
- 10 largest maps from Baldur's Gate - a standard data set.
- For each map, increase $N$, the number of mobile units, by 100 at a time. Generate 10 problem instances for each $N$.
- Time limit set to 10 minutes per problem. $k = 3$.
- Compared with WHCA*(8,1), with and without diagonals [Silver 2005; Sturtevant & Buro 2006].
- Run on the Hierarchical Open Graph framework (HOG) [http://www.cs.ualberta.ca/~nathanst/hog.html](http://www.cs.ualberta.ca/~nathanst/hog.html)
AR0411SR

272x232 tiles

14098 traversable tiles
Total Distance Travelled, AR0411SR

Distance

Number of agents

WHCA* no diagonals
FAR no diagonals
WHCA* with diagonals
Future Work

• Investigate new heuristics, better waiting strategies, smarter annotations/dynamic flows.

• Analytical studies.

• Incorporate FAR into a real game, or enter RoboCup Rescue.

• Extend FAR for: planning under uncertainty; initially unknown maps; dynamic environments; moving targets.
Summary

- FAR builds a *flow-annotated search graph* inspired by two-way roads.
- Replanning is done locally, keeping the computations cheap.
- FAR solves problems more quickly and uses less memory than WHCA*.
- FAR can often solve problems with larger number of units.
- Simple approaches can be very effective in many cases.

Questions? 😊