Heuristic Search for Large Problems with Real Costs

Matthew Hatem, Ethan Burns and Wheeler Ruml

University of New Hampshire

with support from NSF (grant IIS-0812141) and DARPA (grant N10AP20029)
Motivation

- Problem: A* runs out of memory in $\sim 10$ minutes
Motivation

- Problem: A* runs out of memory in \( \sim 10 \) minutes
- Problem: IDA* fails with many duplicates and real costs
- Problem: A* runs out of memory in \( \sim 10 \) minutes
- Problem: IDA* fails with many duplicates and real costs
- Solution: A* on disk ("external memory")
Motivation

- Problem: A* runs out of memory in ~10 minutes
- Problem: IDA* fails with many duplicates and real costs
- Solution: A* on disk ("external memory")
- Problem: Previous methods assume integer costs
Problem: A* runs out of memory in ~10 minutes

Problem: IDA* fails with many duplicates and real costs

Solution: A* on disk (“external memory”)

Problem: Previous methods assume integer costs

Problem: Most previous methods use breadth-first search
Problem: A* runs out of memory in \(\sim 10\) minutes

Problem: IDA* fails with many duplicates and real costs

Solution: A* on disk (“external memory”)

Problem: Previous methods assume integer costs

Problem: Most previous methods use breadth-first search

Solution: PEDAL (this paper)

◆ Best-first search order
◆ Real costs
◆ Provably I/O efficient
◆ Exploits parallelism
■ External memory search

■ The problem with real costs

■ PEDAL

■ Results
External Memory Search
- Open List: ordered set $f(n) = g(n) + h(n)$
- Closed List: random access!
- How to put this on disk?
- Open List: ordered set $f(n) = g(n) + h(n)$
- Closed List: random access!
- How to put this on disk?
- Two ideas: Buckets and Layers
- Use a hash function to partition the space
- Duplicate nodes will be in the same bucket
- Use a hash function to partition the space
- Duplicate nodes will be in the same bucket
- Use a hash function to partition the space
- Duplicate nodes will be in the same bucket
Expand Phase

- A* Search
- Buckets
- Expand Phase
- Merge Phase
- Layers
- Summary

**PEDAL Experiments**

**Conclusion**

Only one bucket need fit in RAM to expand
Only one bucket need fit in RAM to expand
Only one bucket need fit in RAM to expand
Merge Phase

Only one bucket need fit in RAM to merge duplicates
Only one bucket need fit in RAM to merge duplicates
Only one bucket need fit in RAM to merge duplicates.
Only one bucket need fit in RAM to merge duplicates
Only one bucket need fit in RAM to merge duplicates
Depth based layers give breadth-first search, e.g., Breadth-First Heuristic Search (Zhou, Hansen 2004).

How do we perform best-first search?
- Depth based layers give breadth-first search, e.g., Breadth-First Heuristic Search (Zhou, Hansen 2004)

- How do we perform best-first search?
  - Set an $f$ limit and expand all nodes within the limit
- Divide the search space into buckets
- Duplicate detection is localized per bucket
- Duplicate detection is done in a separate phase
- Layer the search by $f$
Unit Costs

Layers grow, many nodes per layer
Problem with Real Costs

Real Costs

Many layers, too few nodes per layer
Our solution is to inflate layers
But how should we inflate layers?
Bad

Closed List  Open List

Expanded Nodes
Good

- Keep a distribution of all f-values on the frontier
- Choose a value in this distribution to guarantee I/O efficiency
- See paper for proof
Parallel External Dynamic A* Layering

- Best-first:
  - layer search on $f$

- Real Costs:
  - dynamically inflate $f$ layers for I/O efficiency

- Exploits Parallelism (see paper)

- Recursive expansions (see paper)
Experiments

Introduction

External Memory

PEDAL

Experiments

- Setup
- 15-Puzzle
- 15-Puzzle: IDA*
- 15-Puzzle: BFHS
- BFHS
- Sqrt Puzzle
- Sqrt Puzz: IDA*
- Sqrt Puzz: BFHS
- Dockyard Robots
- Doc. Rob: BFHS

Conclusion
Setup

Three domains

Dual quad-core

8GB of RAM

7 SATA disks
15-Puzzle

- Unit cost sliding tile puzzle
- Classic benchmark
- Korf 100 (Korf 1985)
- A* is unable to solve all 100 with Manhattan Distance

```
  6 2 5 14
  3 15 4 10
  7 12 11 9
  8 1 13
```
PEDAL uses disk but is faster than IDA*!
Best-first is better than breadth-first
Introduction
External Memory
Pedal
Experiments
- Setup
- 15-Puzzle
- 15-Puzzle: IDA*
- 15-Puzzle: BFHS
- BFHS
- Sqrt Puzzle
- Sqrt Puzz: IDA* CR
- Sqrt Puzz: BFHS
- Dockyard Robots
- Doc. Rob: BFHS
Conclusion
A* with proper tie breaking
BFHS must expand more nodes
Move costs square root of tile number

Simple real-valued version of well understood benchmark

Easy to reproduce
■ IDA* and BFHS did not solve any instances in time limit

■ We compare against IDA*$_{CR}$ and novel variant of BFHS
Again duplicate checking and parallelism trump disk latency
Again best-first is better than breadth-first
From Ghallab, Nau, Traverso (2004)

- All actions have real costs
- Many duplicate states
- IDA* and IDA*\(_{CR}\) fail on all instances
PEDAL is the only viable alternative
- Previous external-memory search:
  - fails on domains with real costs
  - suggests that breadth-first search is preferred

- PEDAL:
  - is best-first
  - I/O efficient for real costs
Previous external-memory search:

- fails on domains with real costs
- suggests that breadth-first search is preferred

PEDAL:

- is best-first
- I/O efficient for real costs

External memory search doesn’t have to be slow!

- faster than IDA* on standard 15-Puzzle!
Previous external-memory search:
- fails on domains with real costs
- suggests that breadth-first search is preferred

PEDAL:
- is best-first
- I/O efficient for real costs

External memory search doesn’t have to be slow!
- faster than IDA* on standard 15-Puzzle!

General purpose best-first external memory algorithm
Tell your students to apply to grad school in CS at UNH!

- friendly faculty
- funding
- individual attention
- beautiful campus
- low cost of living
- easy access to Boston, White Mountains
- strong in AI, infoviz, networking, systems, bioinformatics
Back-up Slides

PEDAL CR vs. Histogram
Recursive Expansions
BFHS vs. PEDAL Non-Recursive
PEDAL CR vs. Histogram

Corrective Reexpansions vs. Histogram

- PEDAL CR vs. Histogram
- Recursive Expansions
- BFHS vs. PEDAL
- Non-Recursive
Recursive Expansions

PEDAL

Experiments

Conclusion

Back-up Slides
- PEDAL CR vs. Histogram
- Recursive Expansions
- BFHS vs. PEDAL Non-Recursive

Matthew Hatem (UNH) Heuristic Search for Large Problems with Real Costs – 30 / 31