Research Papers

What is the primary purpose of a paper?
- To teach the reader something they didn’t know
- Survey paper -> tutorial on a topic
- Research paper -> teach the reader something new
  - Assuming basic (survey-level) knowledge of the topic

What is a secondary goal?
- To attract readers interested in the topic/problem
- To teach them in as straightforward a way as possible
  - Hence scientific writing standards

CS Paper Format

1. Introduction
2. Literature Review
   - Provide necessary background to understand paper
   - Beyond survey-level knowledge
   - Put contribution in context of previous work
3. Methodology
   - Experiment Design
   - Algorithm/System/Theory Details
4. Experimental Results
   - Present data
   - Show how data supports contribution
5. Conclusion
   - Tell the user what they learned

How to Read a Research Paper

Critical Analysis
- Identify the (claimed) contribution
- What is the problem being addressed?
- What is the solution?
- Place in context
  - Is the problem important?
  - Is the solution novel?
  - How does it relate to previous work?
- Analyze the claims
  - Is the methodology sound?
  - Does the data support the conclusion?
- Are there alternative explanations or other open issues?
- Summarize
  - What did you learn?

CS Paper Format (Introduction)

What goes in the introduction?
- A brief version of everything
- Problem Statement
- Contribution
- Approach
- Evidence
- Conclusion

Most readers will read the introduction, then decide whether to read the rest of the paper
- Some may just look at the figures, so they should tell the story too.

This is not a mystery novel – give away the story in the introduction!
CS Paper Format (Abstract)

Every paper has an abstract
- 1 or 2 paragraphs at most

Short version of introduction
- Problem statement
- Contribution
- Conclusion

Potential readers will look at the abstract to decide whether to read the introduction...

Reading Assignment #1

- Can be downloaded from the assignments page of the class web site.

Write a brief review of the paper, summarizing (1) its claimed contributions and (2) how the data presented does or does not support the conclusion.
- 1 page Maximum

Submission (Due Thursday, Feb 7th)
- On-line students: email paragraph to CS150@cs.colostate.edu
- On-campus students: print it out, bring it to class.

Local Search (General Idea)

Goal: find a termination state
- Example: SAT (find a mapping of variables to {T,F})
- Counter-example: Path planning (output is a sequence of states)

Method
- Start with an initial state
- On every step:
  - Evaluate "neighboring" states (local variations of current state)
  - Move to neighboring state with better evaluation score
- Variations on strategy
  - First improvement
  - Steepest ascent
- Variations on evaluation
  - Evaluation function is the function to be optimized
  - Evaluation function is a cheaper heuristic

Constraints
- Finite memory
- Classic form: "memory-less"
- Modifications: finite (but non-zero) memory

Adding Memory

Add just one state of memory ("best so far")

Random Restart Local Search
- Run local search from random start position
- Save result as "best so far"
- Run local search N more times
- After each, if result is better than "best so far", replace

Note: any local search strategy will do
- Generally run lots of fast searches

Adding N States

Parallel Local Search
- Run N searches in parallel
- Compares N local optima, select best
- Similar to random restart

Beam Local Search
- Initialize with N random states
- Compute N neighborhoods
- Select N best elements from combined neighborhoods
- Repeat

Ask yourself: how do these two differ?

Unlimited Memory

Tabu Search (Glover 1986)

Basic idea:
- Keep memory of visited states (tabu table)
- Do not return to tabu states

Deterministic search
- Never return to visited state

Stochastic search
- May be value to returning to visited state
- But odds of improvement are less

Issue: Memory size
Basic Tabu Search Algorithm

1. Initialize $s$ and best-so-far
2. While termination criteria not satisfied
   1. Update memory (add last state)
   2. Generate neighbor solutions
   3. Prune neighbors that are tabu (in memory)
   4. Set $s$ to best remaining neighbor
   5. Update best-so-far if necessary
3. Return best-so-far

Tabu Search Parameters

Size of memory / tabu tenure
- Fixed memory -> forgetting
- Forgetting strategies
  - Age
  - Others...

Contents of memory
- States
- State attributes
- States on path to current best solution
- States from which the next state was not chosen stochastically

Tabu Search Variants

Robust Tabu Search (Taillard 1991)
- Repeatedly choose tabu tenure randomly from an interval

Reactive Tabu Search (Batitti & Tecchiolli 1994)
- Dynamically adjust tabu tenure during search
- Store step counts with states
- Best-so-far always in tabu table
- When neighbor revisited, check interval
- Increase memory size with repetitions
- Decrease memory size when no neighbors are repeats
- Decrease memory size when all neighbors are repeats

Dynamic Local Search

Main theme: separate state quality $H(s)$ from Evaluation function $G(s)$ that guides search

Modify the evaluation function $G(s)$ during local search
$G(s)$ changes at sub-termination local optima

DLS Algorithm

Initialize state $s$
Initialize best-so-far to $s$
Initialize penalties to zero
Initialize evaluation function $G() = h() - \text{penalties}()$
While termination criteria not satisfied:
- Perform local search, starting at $s$ and using $g()$, stopping at $s'$
- If $H(s') > H(\text{best-so-far})$, best-so-far $\leftarrow s'$
- Update penalties based on $s'$
- $s' \leftarrow s$
- $G() \leftarrow h() - \text{penalties}()$
 Adapted from slides for Hoos & Stutzle’s “Stochastic Local Search”

Discussion DLS

What would a good penalty function be?