CMPT 827 - PROJECT TOPIC IDEAS

Useful resources

Book about algorithmic puzzles
https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Algorithmic%20Puzzles%20%5BLevitin%20%26%20Levitin%202011-10-14%5D.pdf

Search algorithms
http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Travaux_fichiers/Chapitres3et4-Russel.pdf

Preferences

- Implementing and testing different search algorithms for a puzzle
- Localized search
- Benefits of choosing popular puzzles - more resources

List of ideas

Idea	Comment/s
Sudoku puzzle	1st choice
Pacman puzzle	
Knight's tour problem aka A Corner-to-Corner Journey Is there a way for a chess knight to start at the lower left corner of a standard 8 × 8 chessboard, visit all the squares of the board exactly once, and end at the upper right corner? (The knight's moves are L-shaped jumps: two squares horizontally or vertically followed by one square in the perpendicular direction.)	2nd choice https://en.wikipedia.org/wiki/Knight%27s_tour https://www.geeksforgeeks.org/the-knights-tour-problem-backtracking-1/ https://core.ac.uk/download/pdf/82320262.pdf https://runestone.academy/runestone/books/published/pythonds/Graphs/KnightsTourAnalysis.html
Bloxorz Problem https://www.sciencedirect.com/science/article/pii/S187705091932160X	3rd choice
Internet search problem	

Sudoku Puzzle References:

- https://github.com/mahdavipanah/SudokuPyCSF
- https://en.wikipedia.org/wiki/Sudoku_solving_algorithms
- https://en.wikipedia.org/wiki/Min-conflicts_algorithm
- https://medium.com/@george.seif94/solving-sudoku-using-a-simple-search-algorithm-3ac44857fee8
- https://stackoverflow.com/questions/1697334/algorithm-for-solving-sudoku
- https://link.springer.com/chapter/10.1007/978-1-84800-094-0_4
- https://link.springer.com/chapter/10.1007/978-1-84800-094-0_4
- https://github.com/sraaphorst/sudoku_stochastic
  Adding few more resources
- http://lam.edu.ly/researches/uploads/Boltzmann%20Machine.pdf
- http://dl.icdst.org/pdfs/files/daabd136eb68e89427de50eec6762419.pdf

Search Algorithms if we go with sudoku:

- Hill Climbing
  - https://www.cs.rochester.edu/u/brown/242/assts/termprojs/Sudoku09.pdf
  - https://en.wikipedia.org/wiki/Hill_climbing
- BFS --- this is a CSP and it is backtracking -> naive approach
- CSP solvers
  - https://www.cs.ubc.ca/~mack/CS322/lectures/3-CSP5.pdf
- Stochastic search
  - https://arxiv.org/pdf/0805.0697.pdf
  - https://www.cs.huji.ac.il/~ai/projects/2017/csp/Suduko_3/Sudoku%20report.pdf
- Simulated annealing:
  - https://www.cs.unc.edu/~lazebnik/fall10/lec06_local_search.pdf
  - https://en.wikipedia.org/wiki/Simulated_annealing

Some specific algorithms that are used to solve sudoku

1. Crooks algorithm
2. Harmony Search algorithm
   https://link.springer.com/chapter/10.1007/978-3-540-74819-9_46
3. Tabu search algorithm
   https://www.hindawi.com/journals/cin/2015/286354/
4. Backtracking method
   https://www.geeksforgeeks.org/sudoku-backtracking-7/
5. Boltzmann machine (Stochastic)
   https://github.com/vberger/silinapse/blob/master/examples/sudoku-boltzmann-machine.rs

Proposal

Solving Sudoku puzzle using Local Search Algorithms

Sudoku is a popular puzzle game which involves placing numbers on a $n * n$ grid with no duplication. Mathematically, sudoku is a $n^2 * n^2$ combinatorial problem which is NP complete. The puzzle has n rows and n columns and requires digits from 1 to n to be filled in each region, which are the rows, columns and subgrids. Constraint to the problem in the form of pre-filled digits increases the complexity of the puzzle. The most common sudoku is of subgrids of size 3*3, with overall size of 9 * 9.

The solution to sudoku can be found out using search algorithms. In this project various search techniques are selected and the performance is compared. The category of search algorithms will be using local search algorithms. Local search will incorporate using stochastic search techniques (i.e., simulated annealing) and hill climbing search techniques. Second, we will treat the sudoku puzzle as a constraint satisfaction problem (CSP)and use backtracking. Using backtracking, because it is a brute-force approach, it will be considered our “ground truth” to compare the local search algorithms to.

A methodology section where you present the questions which you will be investigating concerning the performance of the algorithms on your problem instances, and describe the experiments you will conduct to answer those questions. What classes of instances will you be using? Make sure to use some challenging instances. If applicable, you may consider using the MAPF Benchmark instances mentioned in the individual project. Make sure to cite your sources if your instances are not hand-crafted. Some examples of questions to investigate are as follows:

- *Does there exist a class of instances on which one algorithm always outperforms another algorithm? Is this true for all instances, or is the other algorithm still sometimes a better choice?*
- *How does the performance of your algorithm(s) vary as a function of the number of agents, the percentage of obstacles in instances, the size of the instances, etc? Are there some algorithms whose performance varies less than others when certain parameters change? Is there one algorithm that is uniformly a horrible choice for this problem?*
- *If one of your algorithms sometimes returns suboptimal solutions, how close were those solutions to the optimal solutions? Did there exist a class of instances on which that algorithm was always nearly optimal?*