CS227: Assignment 4

The fourth assignment involves implementing and evaluating **planning algorithms** and engineering a **planning domain**, and writing up a report on your experiments. The assignment has three parts: (1) developing a planning algorithm, (2) developing your own planning domain, and (3) evaluating the algorithms and writing the report.

1. **Planning Algorithms:** Choose one or more of the following planning algorithms to evaluate:

   a. **FF:** Modify the FF source code (http://members.deri.at/~joergh/ff/FF-v2.3.tgz). Several recent works modify the FF algorithm in some way. Choose *one* modification to FF and implement it yourself. Compare the modification to FF. Choices include modifications described in the papers for the planners below.

      i. **YAHSP (LOBFS algorithm)**
      Vincent Vidal (2004). *A Lookahead Strategy for Heuristic Search Planning*, ICAPS 2004. http://www-rcf.usc.edu/~skoenig/icaps/icaps04/icapspapers/ICAPS04VidalV.pdf
      Implement all the algorithms described in paper.

      ii. **Identidem (Local Search w/ restarts)**
      Andrew Coles, Maria Fox, and Amanda Smith (2007). *A New Local-Search Algorithm for Forward-Chaining Planning*, ICAPS 2007.
      http://www.cis.strath.ac.uk/cis/research/publications/papers/strath_cis_publication_2151.pdf
      Implement Identidem (Algorithms 2 and 3) without lookahead actions or non-helpful actions.

      iii. **Best-First Heuristic Search**
      Rong Zhou and Eric Hansen (2004). *Breadth-First Heuristic Search*, ICAPS 2004.
      http://www2.parc.com/isl/members/rzhou/papers/breadthFirst.pdf
      Implement BFHS (Figure 3).

   b. **Graphplan:** Implement the basic Graphplan algorithm. Make sure you implement both memoization and the propagation of binary mutual exclusions. Compare the basic Graphplan procedure against Graphplan (a) without memoization; (b) without propagating mutual exclusions, i.e., the only mutually exclusive actions are the ones that directly interfere with each other (static mutexes); and (c) without both memoization and mutual exclusions propagation.

   c. **SATPLAN:** Implement the SATPLAN planning algorithm using the SAT engine you developed for Assignment 1. Convert the planning theory into a SAT formula. Simplify the formula using unit propagation and the failed literal rule. Solve the simplified formula using your SAT solver and convert the satisfying assignment back into a plan. Compare this basic procedure against a version without simplifying the formula using unit propagation and the failed literal rule.

PDDL parser: You should use PDDL 2.1 level 1 (STRIPS) as the input language to your planner. You may use the VAL code from the 3rd International Planning Competition (http://planning.cis.strath.ac.uk/VAL/VAL-3.1.tar.gz) to check to see if your plans are correct and the parser code to parse the PDDL domain and problem. FF is another option for parser code.

Problems for evaluation: Algorithms must be evaluated on planning problems from the 3rd International Planning Competition (available on the class Coursework site or http://planning.cis.strath.ac.uk/competition/CompoDomains/IPC3.tgz). Problems are available in PDDL format. Allocate at most 30 mins for solving each problem (you may want to run the harder problems longer).

2. **Planning Domain:** Develop a unique classical planning domain in PDDL 2.1 or later and create five instances (of different size). Attempt to solve all instances with three different planners, including the planner you developed in Part 1.

3. **Report:** Your report should contain descriptions of the algorithms you are evaluating in Part 1 and the optimizations you used to make the algorithms run fast. The report should contain the results of running the experiments and a discussion of your conclusions.

   In your report describe the domain that you develop in Part 2. Include a description of the actions, predicates, and objects, as well as initial states and goals in each instance. Include an analysis of the results by plotting the run times and plan lengths for each instance and each planner. The analysis should describe differences in the planners that led to the results and what aspects of the planning domain are particularly easy or hard for the planners you evaluate.

   Assignments will be graded on the description of the algorithms, the optimizations, the raw results, the developed planning domain, and the analysis of your results.

Submission: The report can be submitted electronically, in class, or directly to the TA. Submit source code electronically as a single .tgz or .zip file that unpacks into its own directory called "*your_group_name*_planner". Please include a small README file describing how to build and run your code. Clearly identify all members of the group both on the report, and in the electronic submission. Send electronic submissions to cs227-submit@lists.stanford.edu.

Assignments may be done in groups of 2-3 students. You may choose any programming language for implementation purposes, though we recommend either C or C++ for maximum efficiency. The parsers we suggest are in C/C++. We will not support the code for the parsers; the suggestions are only for your convenience.

Assignments are due by noon on **5 June**.