June 11, 2024

Robotics and Autonomous Systems

Dear Professor Gini:

We would like to thank the reviwers for their comments which have led to improvements in the clarity of the paper "GEO-SAT: A New Approach for Knowledge-Based Agent Decision Making," co-authored by Thomas C. Henderson, Amelia Lessen, Ishaan Rajan, Tessa Nishida, Kutay Eken, Xiuyi Fan, David Sacharny, Amar Mitiche and Thatcher Geary. We have considered and addressed all the issues raised by Reviewer 2 (there were no issues for Reviewer 1):

1. **What is the difference between the proposed SAT solutions finding method and the modern SAT solvers, using CDCL or local search algorithms?**
   CDCL is based on DPLL but with backjumping that is not chronological. DPLL is a complete backtracking search algorithm which as its major step assigns a value to a variable, then determines if there is a solution (i.e., it reduces the conjuncts, assigns necessary values to resulting unit clause variables, and assigns values to pure variables with only one polarity across all conjuncts). Chop-SAT can be viewed as an alternative geometric approach to this solution determination step only with no need, in general, to be embedded in a complete search. For details of the geometric method, see the answer to reviewer question 3 below.

   Change made: This information was added to the Solving SAT section.

2. **Are there any practical applications of the proposed representations and methods?**
   One current application is the use Chop-SAT as part of a Belief-Desire-Intention architecture for autonomous Unmanned Aircraft Systems agents (see "Reinforcement Learning at the Cognitive Level in a Belief, Desire, Intention UAS Agent," David Sacharny, Thomas C. Henderson, Michael Cline, and Benjamin Russon, Intelligent Autonomous Systems Conference, Singapore, pp. 431-442, June, 2021). We are conducting experiments this summer at the US Air Force Academy where such an architecture for decision making is used.

   Change made: This information has been added to the Experiments section.

3. **The contribution of the Chop-SAT claimed that it is able to determine whether a SAT solution exists, how does it work? What is the main algorithm of this decision procedure? Is it complete? Is it able to prove unsatisfiability?**
   To see how *Chop-SAT* relates to standard SAT solvers, consider DPLL. DPLL is a complete backtracking search algorithm which as its major step assigns a value to a variable, then determines if there is a solution (i.e., it reduces the conjuncts, assigns necessary values to resulting unit clause variables, and assigns values to pure variables with only one polarity across all conjuncts). *Chop-SAT* can be viewed as an alternative geometric approach to this solution determination step only with no need, in general, to be embedded in a complete search.
   Chop-SAT represents the SAT problem as a convex feasible region where if this region has a corner from the original n-D hypercube, then a solution exists; if not, then the sentence is unsatisfiable. Chop-SAT works by using linear programming (linprog in Matlab) to find extremal vertexes by projecting the feasible region onto a selected vector. Let $\mathcal{F}$ be the feasible region of a CNF sentence; then linear programming finds $x \in \mathcal{F}$ which minimizes $f^T x$ such that $Ax \leq b$ and $lb \leq x \leq ub$, where $lb = 0$ and $ub = 1$.
   For example, if $f = [1, 0, 0]^T$ then the feasible region is projected onto the x-axis (in 3-D). Of course, *Chop-SAT* when embedded in a search algorithm like DDLL is also complete, but we use it as a one-step algorithm since this usually finds a solution. Note that we can also make *Chop-SAT* complete by projecting onto all the diagonal axes of the n-D hypercube, but our original hope was that geometry would provide a polynomial time solution. *Chop-SAT* can determine that a feasible region is from an unsatisfiable sentence because of the following property: For every feasible region arising from an unsatisfiable CNF sentence, there is no point at distance greater than $\frac{\sqrt{n-1}}{2}$ from the center of the hypercube; this means the feasible region can undergo any rotation about the center of the hypercube and all points of the feasible region remain within the bounds of the n-D hypercube -– this is not the case for feasible regions containing solutions.

   Change made: This text was added to the end of the Solving SAT section.

4. **For the experiments, in its current format, it's more like a case study than an empirical study? What is the benchmark used in the experiments? Is there any detailed statistic for the benchmark? How are the solving method evaluated? Is it compared to any other solvers? What is the baseline of the experiment?**
   The experiments of greatest interest are those from Wumpus world because there

are 80 logical variables (i.e., 2^{80} models for the knowledge base!). The point is to see if the probabilistic method provides adequate variable probabilities to help make good (life preserving) decisions. The baseline of the experiment is given as the Monte Carlo method for determining these probabilities (i.e., random boards are generated that satisfy the knowledge acquired during a game and the exact variable probabilities are estimated from these board configurations). The overall statistic is that 1000 random Wumpus boards are generated, and each algorithm (human, analytic center, Chop-SAT) is compared to the ground truth approximated by the Monte Carlo method. Thus, there are 3 solvers (humans wrote programs to assign probabilities, the analytic center method provides probabilities, and Chop-SAT provides probabilities) which are then used to make decisions. Other probabilistic SAT solution methods were not used because they are all exponential complexity and either won't run on this size problem or require max entropy type assumptions which are not valid here.

Change made: This text has been added to the end of the Experiments section.

If you have any further remaining questions, please let me know.

Best regards,

Thomas C. Henderson
Professor
School of Computing