Exploring Optimization Strategies for *MasterMind*

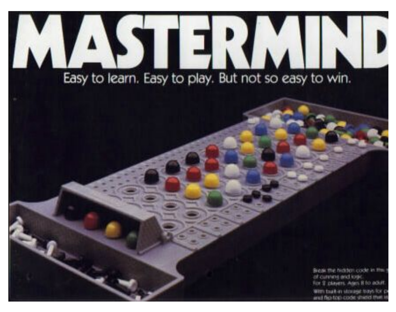
Gioia Dominedò, Amy Lee, Kendrick Lo, Reinier Maat

AM207 Stochastic Methods for Data Analysis, Inference, and Optimization

Spring 2016

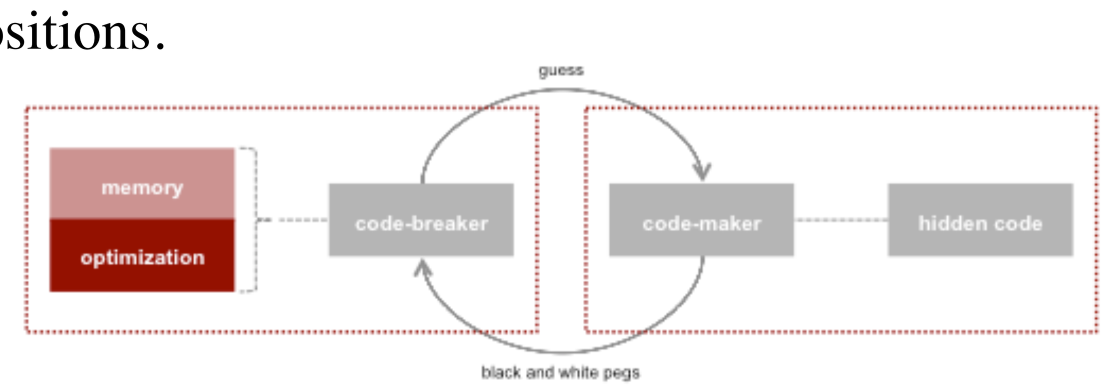
Abstract: We investigated various stochastic "code-breaking" optimization methods for solving higher-dimensional versions of the classic game, *MasterMind*. We tested our techniques on different game board variations and compared the performance of these techniques in terms of the number of guesses and computation time.

Introduction



MasterMind is a classic game that has been studied by mathematicians and computer scientists alike. It can be viewed as a dynamically constrained optimization problem in which the constraints are not known in advance but are discovered as the game progresses.

The game involves two players, a code-maker and code-breaker. The code-breaker must uncover the hidden sequence of colors (typically of length 4, with 6 possible colors) set by the code-maker through a series of guesses. After each guess, the code-maker provides feedback in the form of black and white pegs, where black pegs indicate the number of correct colors in the right positions, and white pegs denote the number of correct colors in the wrong positions.

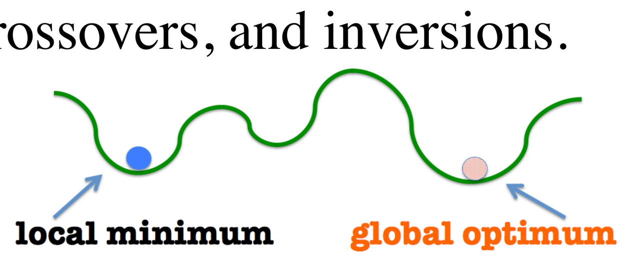


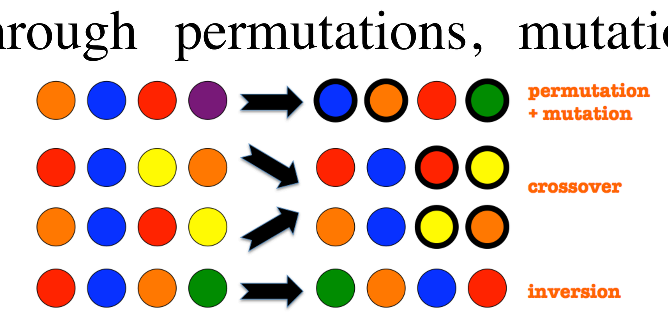
Approach

We explored approaches for solving higher-dimensional versions of the game. Five iterative "code-breaking" optimization methods and two different objective functions were investigated:

Optimization Methods:

1. *Knuth's Five-Guess Algorithm* – a well-known global optimization technique that can solve the classic game configuration in at most 5 moves.
2. *Random Sampling from Posterior* – a constrained random search algorithm involving Bayesian posterior updates of the joint distribution over all possible code sequences.
3. *Exhaustive Search with Shannon Entropy* – see function below.
4. *Simulated Annealing* – a local optimization technique adapted from the Metropolis-Hastings algorithm, used to find approximations of a global optimum within large search spaces; a temperature parameter controls the permutation and mutations of guesses to create proposals.
5. *Genetic Algorithms* – a local optimization technique that draws inspiration from natural evolution, in which generations of populations of potential guesses are formed through permutations, mutations, crossovers, and inversions.





Objective Functions:

1. *Constraint-Based Cost Function*: $\sum_{i=1}^{n} |\Delta n_w^i| + |\Delta n_w^i + \Delta n_b^i|$
   where n is the # of previous guesses and Δn_w^i and Δn_b^i are differences in white and black peg responses
2. *Shannon Entropy*: $H(\text{guess}|\text{possible codes}) = -\sum_{i=1}^{R} P(r_i) \log_b P(r_i)$
   where r_i is the i th response category and R is the # of possible responses

Experiments

We first experimented with the number of possible colors per position (e.g. 4-10) while keeping the code length constant at 4. We then varied the code length (e.g. 4-10 positions) while keeping the number of colors constant at 2. Each method was tested 20 times for each experiment, and the average number of guesses to win and the runtime were recorded.

Citations

Knuth, D.E. 1976. The Computer as a Master Mind. *Journal of Recreational Mathematics*.

Bernier, Jose Luis. Solving Mastermind using GAs and Simulated Annealing: a Case of Dynamic Constraint Optimization. *Parallel Problem Solving from Nature*: 553-563, 1996.

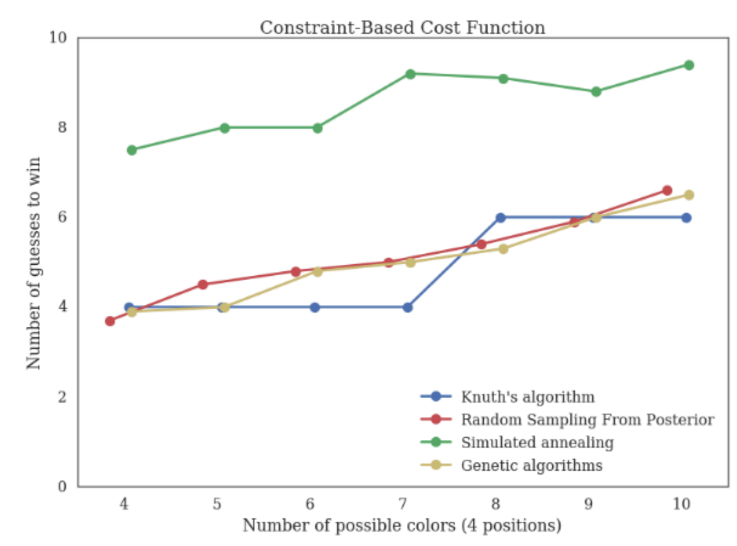
Vomlel, Jiri. Bayesian networks in mastermind. *Proceedings of the 7th Czech-Japan Seminar*. 2004.

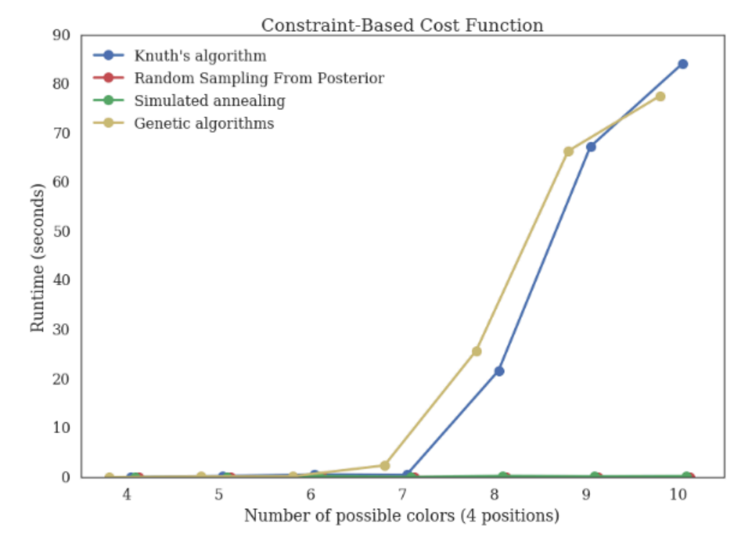
Results

Varying Color Selection

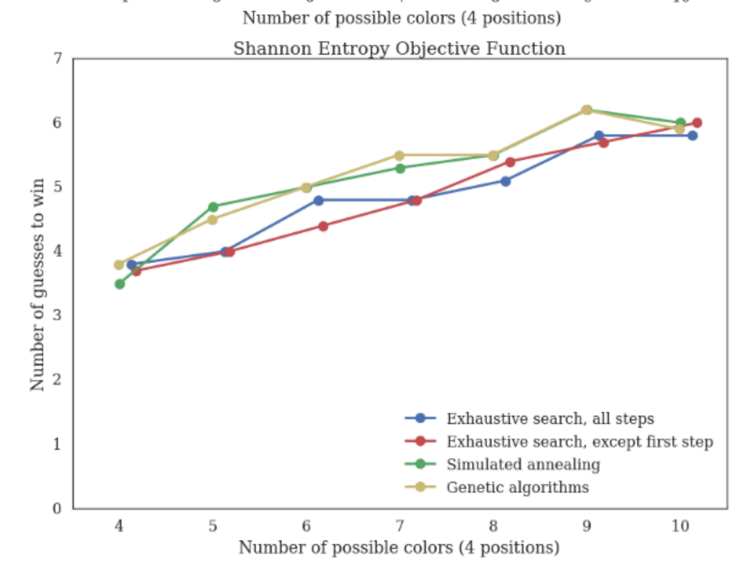
of Guesses | Runtime

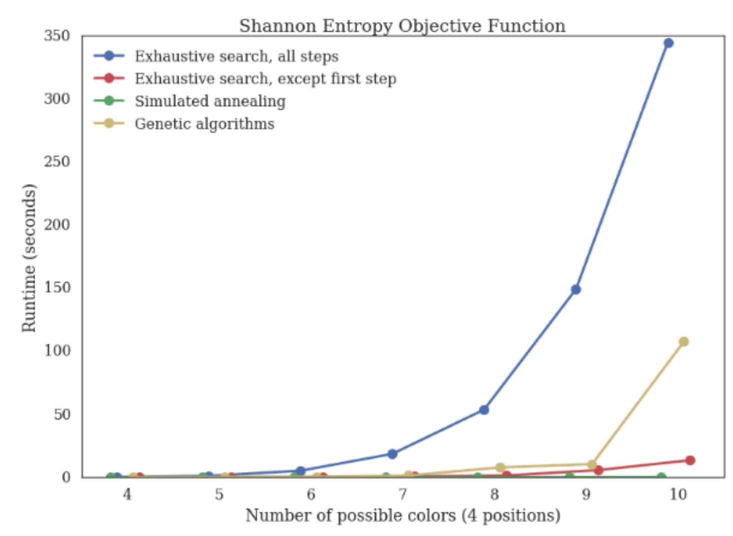
Constraint-Based





Entropy

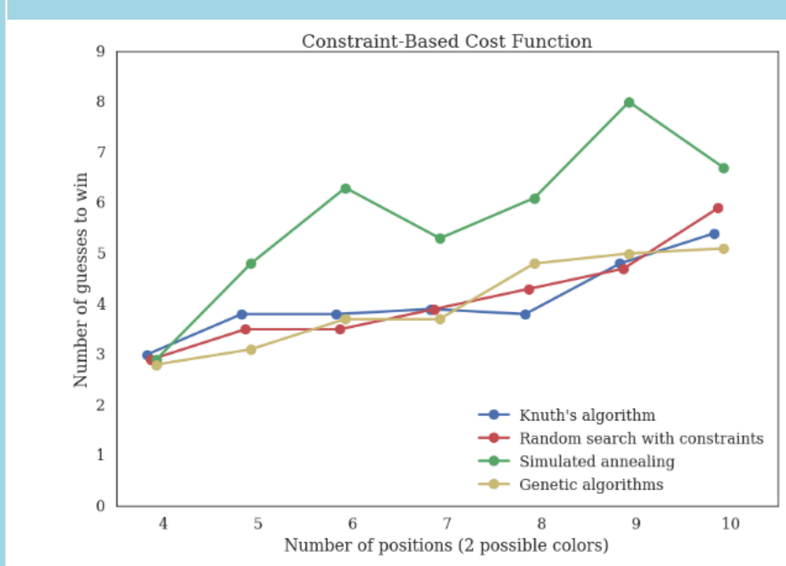


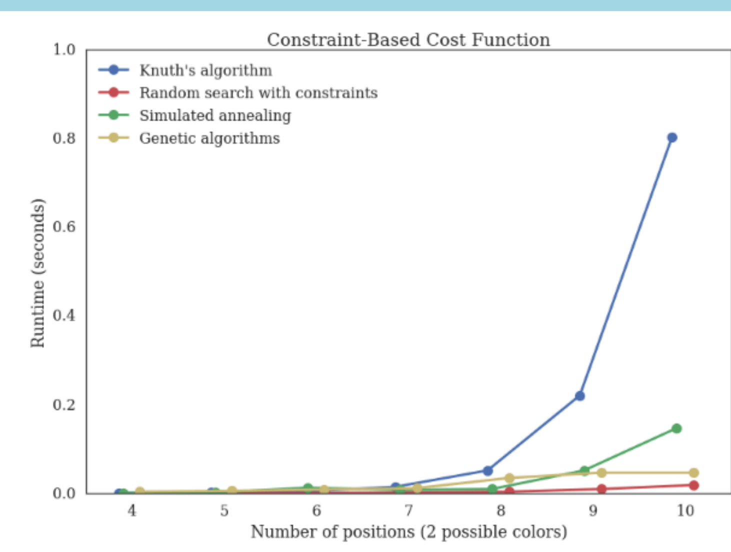


Varying Code Length

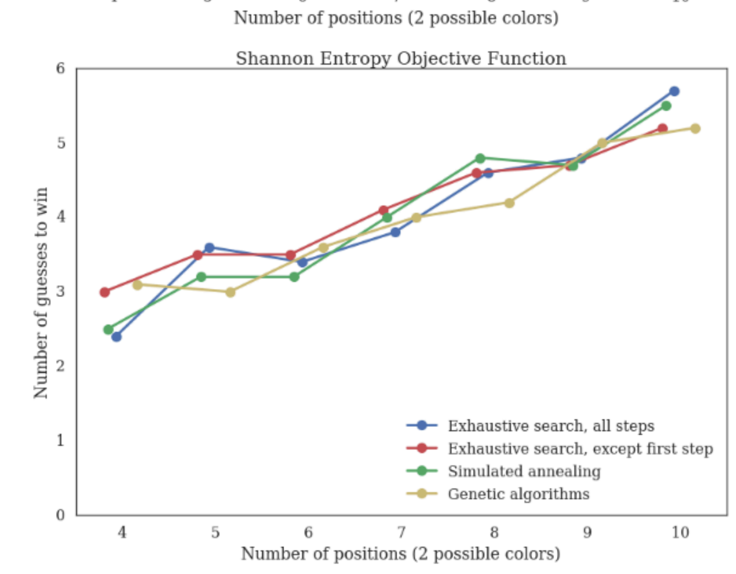
of Guesses | Runtime

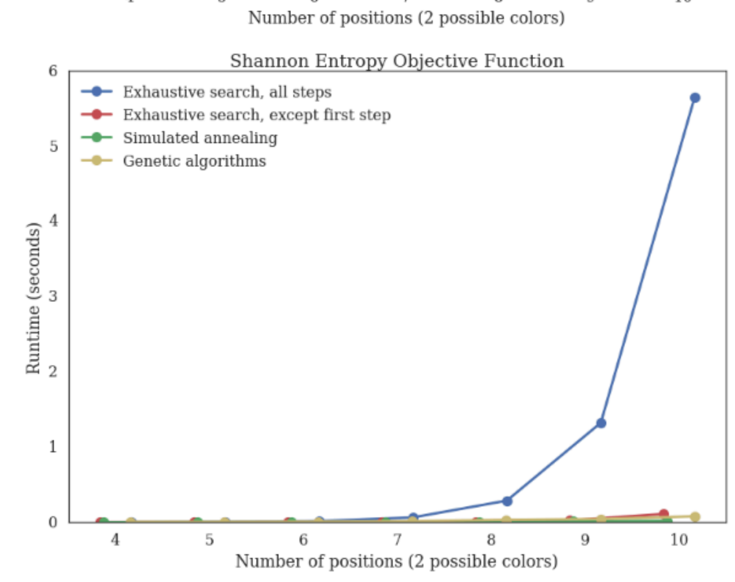
Constraint-Based





Entropy





Conclusions

Global optimization methods (Knuth's algorithm, random sampling, and exhaustive search with Shannon Entropy) were able to consistently win the game in a small number of moves, but required excessive runtimes with more complex board configurations.

Local optimization methods (Simulated Annealing and Genetic Algorithms) required more moves to win, but required less time to arrive at reasonable guesses, allowing close-to-optimal solutions to be found in a fixed period. Therefore in high-dimensional, time-limited solution, these approaches became increasingly valuable.

The best performer in terms of *balancing* number of guesses and speed (10 colors) was **simulated annealing** with the entropy objective function; the best performer that trades off these factors (10 positions) was **genetic algorithms** with the constraint-based objective function.