Crack the game “Lock and Roll” using Artificial Intelligence

Zhifei Song, Hao Luo

Introduction

Lock and Roll is a very popular ipod/iphone game nowadays. The idea of this game is to place colored dice (dice number ranges from 1 to 4) into configurations that score the highest number of points possible. There are 12 kinds of combos to score points, different combo worth different points based on how difficult to get that combo. The player could also configure a set of prime combos that not only score more points, but clear those dice from the board, freeing up space in which to continue making combos.
Our project is planning to build an agent which could play lock and roll automatically and score points as high as possible.

Literature review

Despite its huge popularity (here are some statistics from lock and roll site [1]: there are around 12,000 games played per day, and 11,400,000 games in total), It is very difficult to find relative literatures that describe formal strategies about this game, on the other side, this state of art of lock and roll is also an important motivation that drive us to develop one application to play the game smartly.
All the information we could get for lock and roll is its rule [2], Limited to the length of Proposal, we will not reproduce the rules here.

Challenges and proposed solutions

The incredibly vast feasible solution space poses a challenge for us to apply naïve search strategy in finding the optimal solution. For example, in the first round where we need to consider placing four dice into a 4x4 empty board. There are A_{16}^{4} (approximately 40,000) possibilities. And a game which scores 5000 points or above normally contains 25 rounds, the total solution space would be 40000^{25}, BFS is not suitable here because of the huge branching factor. If we use DFS, we could certainly find a solution very soon, but the score points would be very low because we only randomly placed dice on the board in each round.

Besides, we want to further argue that, lock and roll is a game that combines strategy and luck, before one round ends, we have no clue about what the 4 dice would be in the next round. That means we cannot accurately predict how good or bad a solution is. This unpredictability relies on the game rule that we roll four dice in one round and each rolling is independent. The dice we get in this round might perfectly match or totally ruin our plan in the previous rounds. If we want to plan one round ahead, we need to calculate all the joint probability for the next 4 dice and plan ahead according to that probability, which is also a great challenge.

A reasonable heuristic can greatly limit our search space and guide our search path. But for this problem we do not know what the proper heuristic should be in the first place. Rather than

predefine a heuristic, our plan is to "learn" a heuristic (utility function) of two vital factors that has good performances in average.
We will introduce two terms here:
1, potentiality (p), different grid will have different potentiality (the maximum combos could be made using that grid), we illustrate each grid's potentiality on the right.
2, combination scores(c), different combos would have different combination scores based on the rule.(Limited to the length, we will explain our algorithm more clearly in the milestone report)

5	4	4	5
4	7	7	4
4	7	7	4
5	4	4	5

We want to build our utility function as $U = \alpha * c + \beta * p$. From a set of experiments we will try to find the most suitable parameters α and β to build our utility function. We might try some more sophisticate models depending on our experiment result as well.

Success matrix

We would consider our project as a successful project if we conquered all the following procedures by the due date:

(1) Since it is a ipod/iphone game, we need re-implement it on PC based on the game rules
(2) We would implement the proposed solution to solve the game
(3) We would run the program multiple times and determine the parameters in our utility function based on the experiment result to optimize the solution.
(4) Given that a middle-level player (we have asked several of our friends to play it) could score 10,000 points on average, we hope our program would outperform the middle-level player.
(5) Finishing all of above will give us a pretty good work load, but if we still have time, we hope to modify some re-planning algorithms to make them fit into this problem for optimization. Such algorithms include D* algorithm[3]

Timeline

Feb.9th – Feb.14th: rebuild lock and roll game on PC using C++
Feb.15th-Feb.23th: implement our proposed solution, specifically, find suitable parameter α, β or find some other suitable models to tackle the problem, project milestone report
Feb.24th-March 1st: Finalize our code and add more strategies to score higher points if possible
March.1st-March.9th: write project report and prepare poster presentation

Reference

[1] http://canned-bananas.com/scoreboard/leaderboard.php
[2] http://cannedbanana.wordpress.com/lock-n-roll-rules-printable/
[3] Anthony Stentz. The focused D* Algorithm for real-time replanning. Robotics institute, Carnegie Mellon University, Pittsburgh.