visits every node, are both definitely not optimal. Exc
a Hamiltonian path is an exceptionally difficult problem
complete), but there are types of graphs that are known to
no Hamiltonian or Eulerian path.

- A *cycle* within the graph, or a path that visits the same
twice, is not desirable. This would add to the maximum po
path length as well as being unexciting for the players.

- The origin node should *connect* to every node in the t
a room that can't be reached by the players is useless.
the graph should be *spanning*, connects to every node, fro
root of the origin node. The same would apply to the exit
if the direction of every arc were reversed.

Morse, Lindsay. (2016). *On The Definition and Importance of Actual Puzzle*
cape Rooms. Retrieved from http://www.puzzlebreak.us/blog/2016/4/20/on-the-de
importance-of-actual-puzzles-in-escape-rooms

by the players. If each edge is weighted as one unit, the dif
can be expressed as the length of the *path* the players
Accordingly, possible paths should have upper and lower b
on length.

- The scenario should offer players' choices in the routes
take. So the graph should include nodes with multiple
At the same time, too many choices can be overwhelmi
players and slow the pacing of the scenario. Accordingl
out-degree, or ways to exit the node, have been limited
arbitrarily selected maximum value of 2.
- A directed graph is one in which nodes are joined by *arcs*,
can be traversed in only one direction. This will disco
players from loitering and simplify the problem of boundin
length, as well as being appropriate to the "magic portal
scenario uses.
- There should be many rooms to offer the players a se
scale. However, if players were to visit every room, it
be exceptionally difficult. An *Euler path*, a path that tra
every arc exactly once, or a *Hamiltonian path*, a path

in a manner invisible to them.
This project is devising a manner of implementing a game sce
Multiple players are navigating a "dungeon" composed of rooms linked by unidirectional portals, from an entry point exit point. The players' perceptions should be that the du is vast, that the feel urgency and a sense of momentum, and importantly that their choices are influencing the outcome. A same time, the level of difficulty must be managed so that is challenging and solvable. The scenario should be replayabl always in a different configuration.

2 Background

This problem of arrangement is typical of graph theory. By resenting the rooms as nodes of a graph and the portals be them as edges different arrangements can be compared a algorithm for generating a scenario within constraints can b plemented. The chief constraints are,

- Difficulty is roughly proportional to the number of rooms v

1 Introduction

In games or puzzles, the players' experiences are based upo
perceived difficulty, their results, and particularly their se
manipulating the outcome. Described by one puzzle designe

> Ancient Greek has this amazing word "aporia" which me
> "to be at a loss" or "to lack the resources needed" and t
> talk about it a lot ... partially, I think, because it feels
> good to get out of that feeling

From a design perspective, this can lead to contradictory
straints. Giving the player the sense that they are at a loss
allowing them to feel that they are overcoming it by their ov
sources often comes down to managing the difficulty for the

luding
(NP-
o have

e node
ossible

ree —
Thus
om the
t node

es in Es-
efinition-

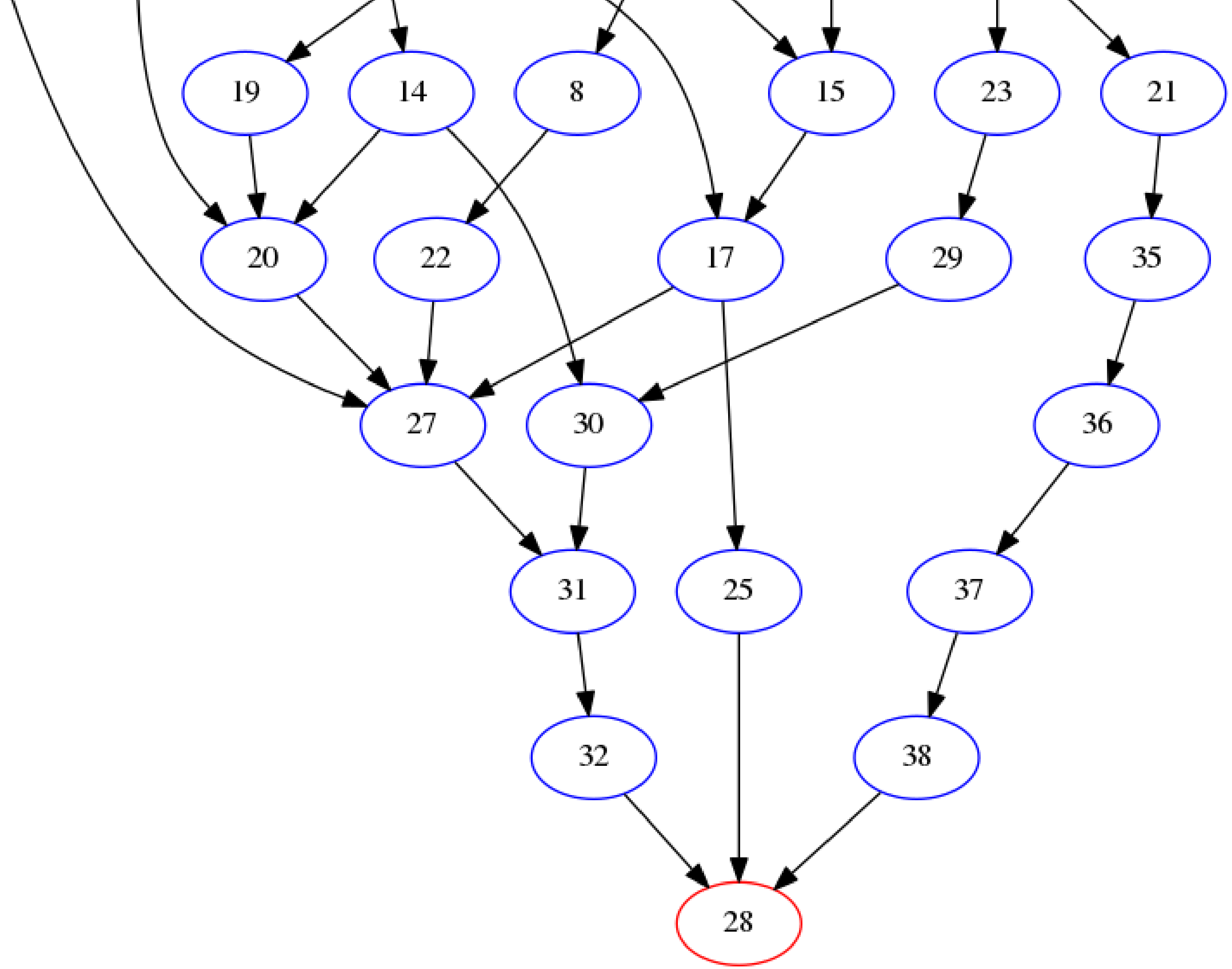
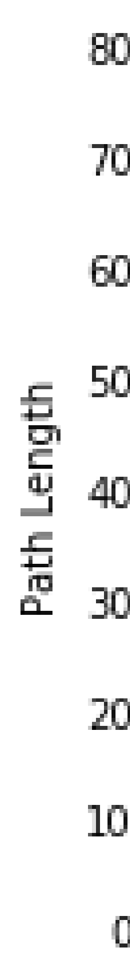


Figure 1: Diagram of graph, generated with maximum path length of 9

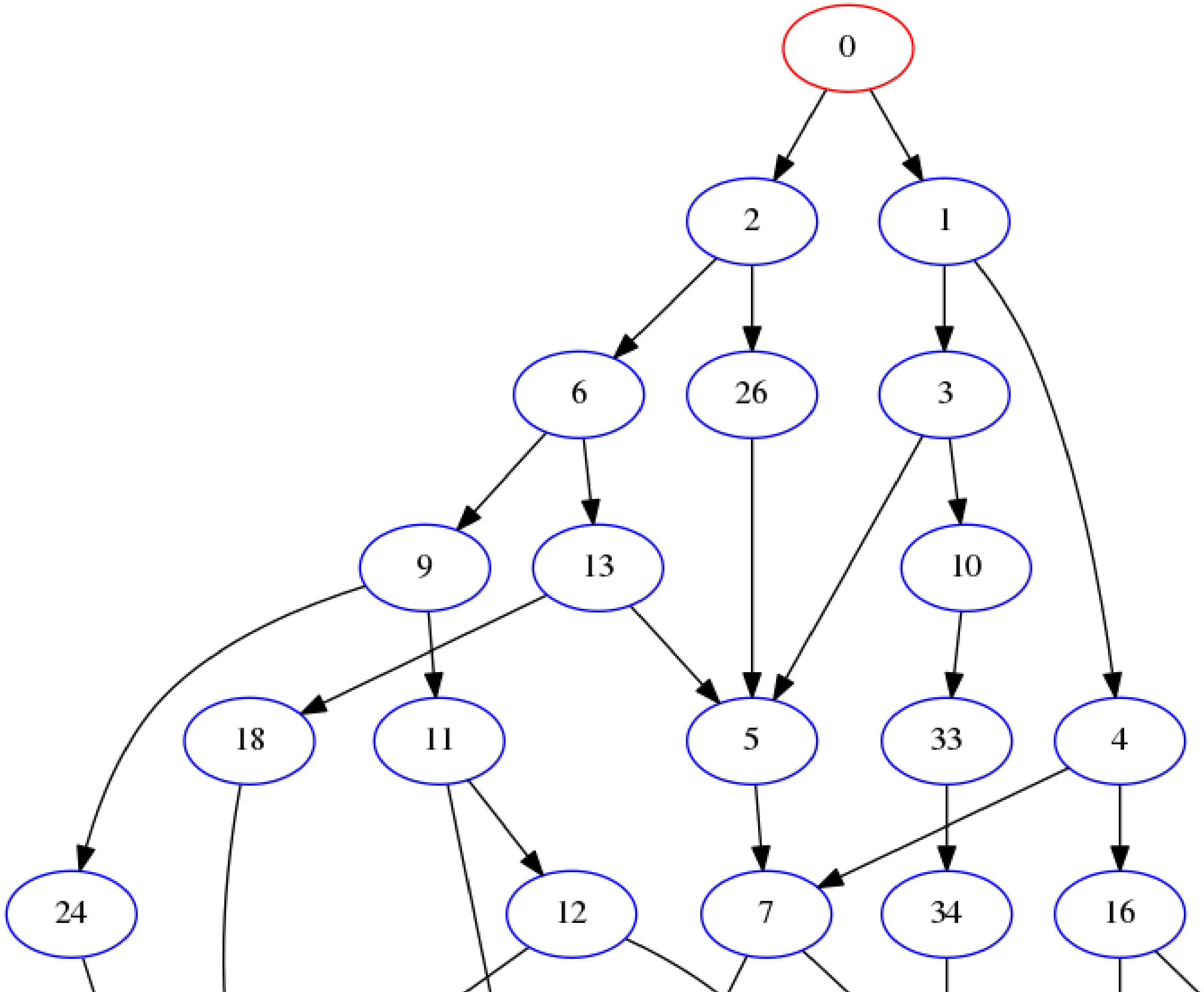
Zwick, Uri (2013). *Directed Minimum Spanning Trees.* Retrived from https://www.cs.princeton.edu/courses/archive/spring13/cos528/directed-mst-1.pdf



Figur
simul
green
the a

2. The maximum size is 2[illegible], which would occur if a perfect full binary tree of height $M - 1$ were created in the first stage of the algorithm, before a path of desired length were created.

While these outcomes are undesirable from a practical standpoint, it is not difficult to check the size of the newly created graph and discard it if the size is not within the preferred bounds.



depth (distance from the origin node) is equal to the desired minimum path length M. That node at greatest depth (distance from origin) is to be the exit node. For a directed spanning tree, the following are equivalent[1]:

- T is a directed spanning tree of G rooted at r.
- The in-degree of r in T is 0, the in-degree of every other vertex in T is 1, and T is acyclic.
- The in-degree of r in T is 0, the in-degree of every other vertex of T is 1, and there are directed paths in T from r to all other vertices.

2. Add routes until every node in the graph has a path to the exit node.

The directed spanning tree in the first stage is generated by randomly adding new nodes to the graph. Since the in-degree (for nodes other than the origin) is 1 for every node (other than the origin), and by arbitrary constraint the maximum out-degree is 2, this spanning tree is an unbalanced binary tree. Thus it follows:

1. The graph will have a minimum size of $M+1$ nodes, if it produced a simple sequence of nodes (of in-degree 1 and out-degree 1) from origin to exit.
2. The maximum size is 2^M, which would occur if a perfect full binary

Generating op

Joh

Augus

on the
ense of
er[1],

eans
they
s so

y con-
s while
wn re-
player

3 Mathematical Techniques

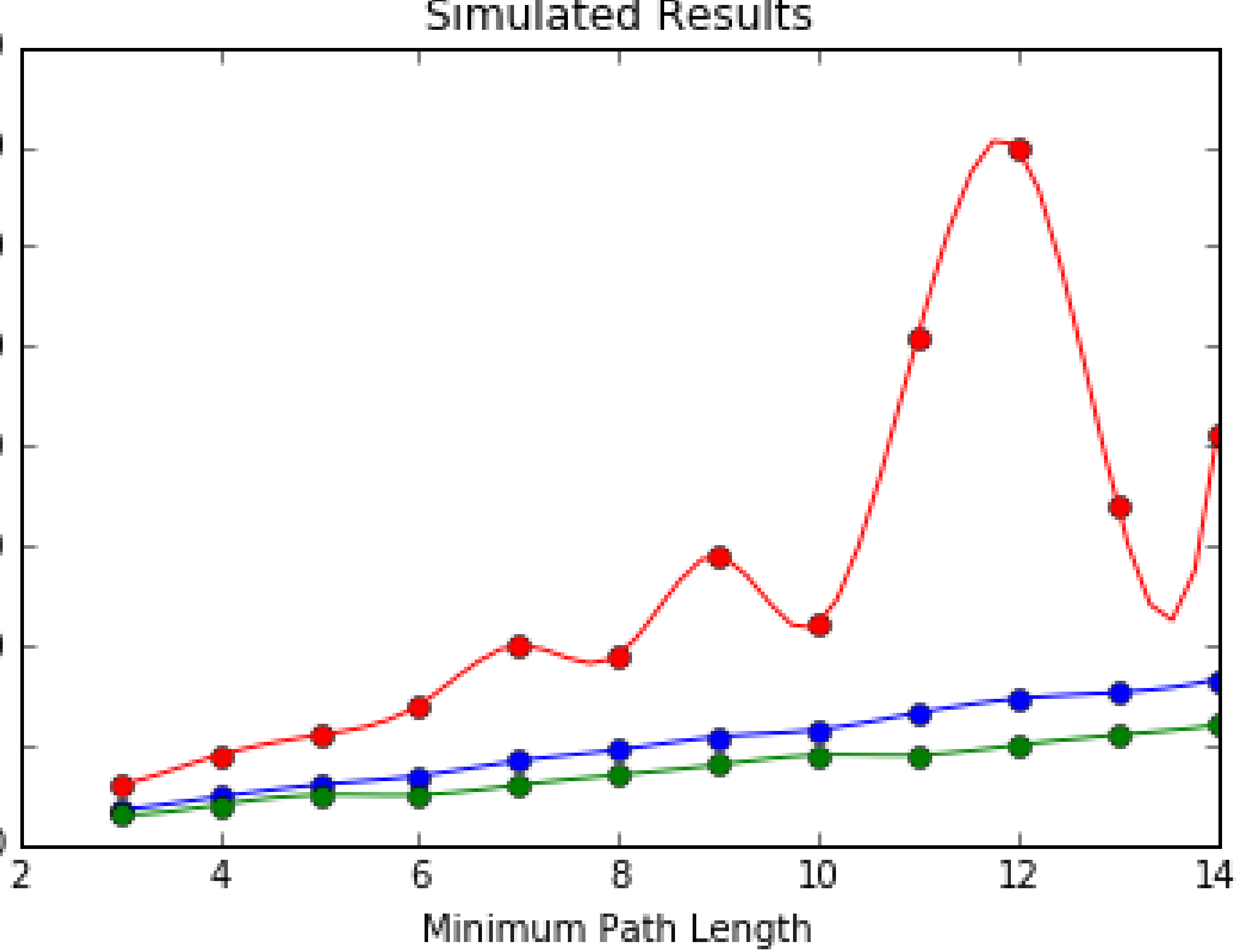
A *directed acyclic graph* is a finite directed graph with no directed cycles, and meets all of these constraints. A particularly useful property of directed acyclic graphs is that longest and shortest paths can be computed in linear time. The graph itself was generated in two stages.

1. Build a directed spanning tree rooted in the origin node, whose greatest depth (distance from the origin node) is equal to the desired minimum

4

As m
the de
accon
with
follow

1. Ma



re 2: Results of simulations, demonstrating observed path lengths of
lated players for graphs generated by different parameters. Red and
n refer to the upper and lower bounds observed, and blue refers to
verage over 1000 simulations of 10 different graphs.

F
le
ra
ol
th
gr

Do connect to the exit node.

Are of an equal or greater distance from the origin node, compared to the start node.

andomly select one node from this list to be the *finish* of a new arc. always linking nodes that are of similar distance from the origin, ths of very long length are discouraged, as we aren't zig-zagging from e bottom of the tree back to the top.

alculate the shortest paths from the origin to start nodes, and from e finish to exit nodes (in my implementation, using Djikstra's algo-hm). If adding an arc from the start node to the finish node will eate a path from the origin to the exit nodes that is of length less an M, it will violate the constraints of the graph. So:

Create a new node adjacent to the start node.

Make this the new start node.

Repeat until adding an arc from the new start node to the finish node will not violate the constraints of the graph.

ld an arc from the start node to the finish node.

neck to see if all nodes connect to the exit node.

se
th
sp
in
up
of
T
ar
un

um number of nodes is arbitrarily fixed to be 2. This is to keep the ze of the graph within manageable limits, and makes the game play ore engaging for the players as they are not overwhelmed by choices any given time.

andomly select one node from this list. Create a new node adjacent this selected node.

heck the graph for a node of the desired distance.

resulting graph is a directed spanning tree. However, including the n and exit nodes, only $M + 1$ connect to the exit node.

algorithm of the second stage repeats until every node connects to exit node.

milarly to above, make a list of all nodes that

Can accept another adjacent node.

Do not connect to the exit node.

andomly select one node from this list. This is the node that will be e *start* of a new arc.

ake a list of all nodes that

Do connect to the exit node.

ptimal dungeons

n Bush

st 19, 2016

Model

entioned above, building the graph was done in two stages. Taking
esired minimum path length M as a parameter: The first stage was
nplished according to the following algorithm. The graph is begun
an origin node, n_O. Two child nodes (n_1, n_2) are added to it. The
wing is repeated until the longest path from the origin is equal to M.

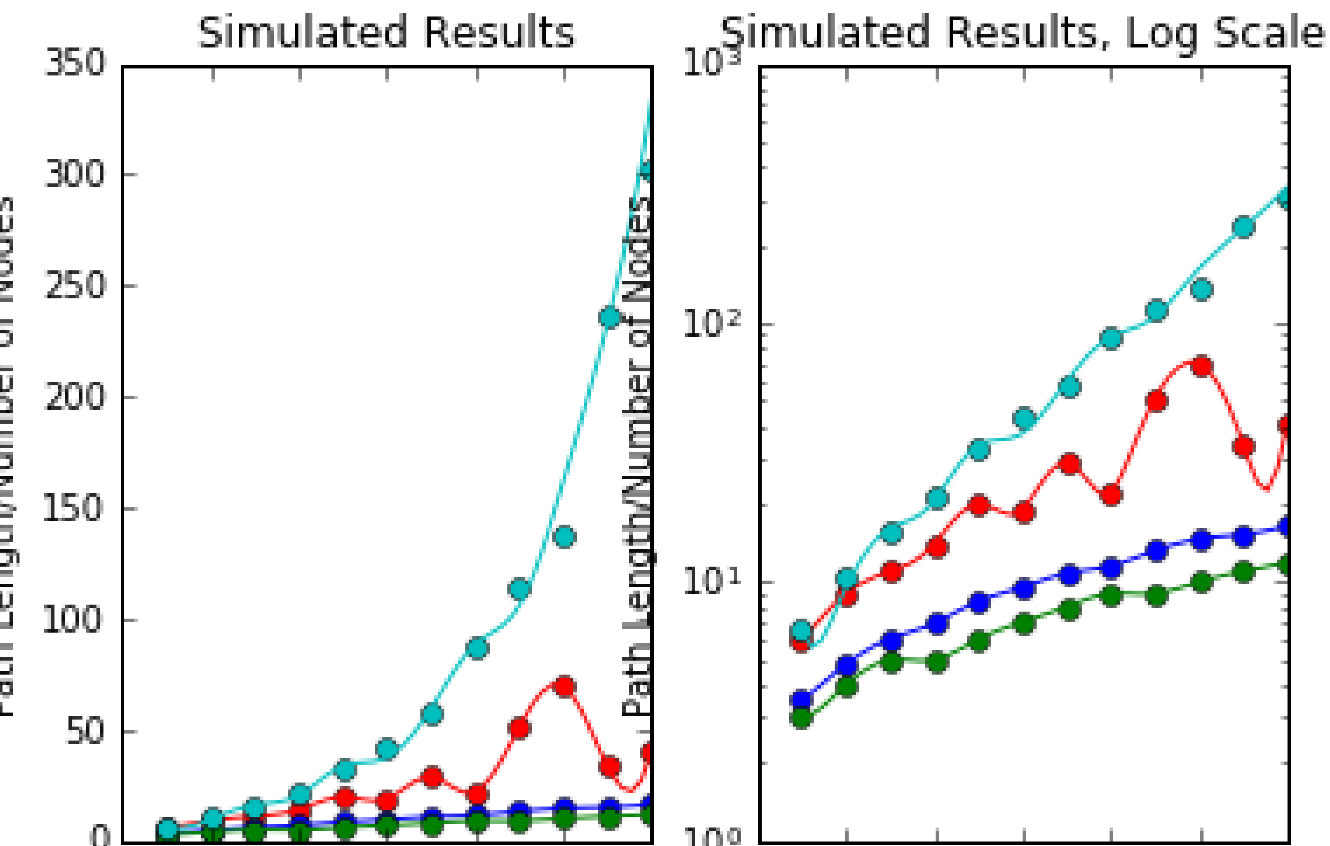
ake a list of all nodes that can accept an adjacent node. The maxi-

Si
at
ca
th
T
to
vi
n

igure 3: Results of simulations, demonstrating observed path engths of simulated players for graphs generated by different paameters. Red and green refer to the upper and lower bounds bserved, blue refers to the average path length, and cyan refers to he average number of nodes over 1000 simulations of 10 different raphs.

erved path length. Figure 3 shows the same data compared to he average number of nodes in the generated graph for the re-pective parameters. It can be seen that while the lower bound ncreases linearly with respect to the minimum path length, the pper bound increases much more rapidly, and the total number f nodes in each graph increases exponentially.

he algorithm does reliably generate graphs that fit the constraints nd are certainly practical if simulations are run and the more nsuitable graphs are discarded.



ode. The following is repeated until the simulated player reaches he exit.

. Randomly select one of the arcs that start in the currently occupied node and does not already exist in the list of visited arcs. If all arcs that start in the currently occupied node have already been visited, select one these visited arcs at random.

. For the selected arc, the start node is the current position. Replace it with the finish node.

. Check to see the new position is the exit node.

5 Solution

'igure 1 shows a graph that was generated by this algorithm for minimum path length of 9. To see how the algorithm scales, imulations were repeated for graphs of minimum path length of to 15. Ten different graphs were generated for each minimum ath length, and a player simulation was conducted on each graph 000 times. Figure 2 shows the results with observed upper and ower bounds (in red and green, respectively) and the average obimulating a player's route through a graph is much easier than creting the graph, computationally. In practice, many playthroughs an be simulated for a specific graph in order to identify those with he most desirable characteristics.

'he simulation algorithm that was implemented is very similar o a random walk. A list is made to track those arcs which were isited and the simulated player's initial position is set to the origin ode. The following is repeated until the simulated player reaches