# THIRTEEN

ICFP Contest 2017 writeup

**Organization** 1

**Writeup by advanced lightning/ConnectBot maintainer** 2

**Writeup by MetricsBot maintainer** 3

**Code metrics** 5

**Appendix** 6

## Organization

Team THIRTEEN was 7 people this year.

People were divided writing various parts of the task.

1 person wrote lightning round bot in Python. Later he converted it to Kotlin and we submitted it. It contained main heuristics that allowed it to look intelligent. This person had to leave us after the first day.

1 person continued to ever advance lightning round heuristic bot (ConnectBot), helped to improve visualizer and managed general team spirit. He wrote a writeup below.

1 person wrote shared data model and more complex bot in Kotlin (MetricsBot). He wrote a writeup, too.

1 person wrote visualizer in Java/Swing on common data model, engaged in PR and outside communications on IRC.

1 person managed time resources: collected ideas and performed general project management. Also he wrote gradient descent trainer for MetricsBot and ran it on large data on EC2 node.

2 people generated ideas, played with bots, did general QA and pair programming.

We occupied meeting room in office building, and lightning round person was with us remotely via Hangouts.

## Writeup by advanced lightning/ConnectBot maintainer

Our team's bot evolution was following:

1) First one of our members wrote a simple "Connect Bot" in Python with very basic strategy: Try to connect closest mines and when we can't connect anymore just add more rivers.

2) Most of our team uses Java/Kotlin as the first language and so our infrastructure was built using Java. Thus we quickly ported the ConnectBot from Python to Java and this is essentially what was submitted to Lightning

3) After that we worked on two different ideas: part of the team tried to improve the ConnectBot and another part started working on a MetricsBot. The basic idea behind MetricsBot was that we will generate several "metrics" to estimate "value" of each river. Then we'll calculate total "value" as a linear combination of "metrics". Then we'll run many tournaments against each other and against (hopefully improved) MetricsBot and in that way we can find optimal multipliers for different metrics

4) A lot of time was invested into inventing and implementing different metrics and implementing an optimization framework. Unfortunately it produced very unstable results in this not-so-high-dimensional space: resulted bots were highly tuned for specific enemy and list of maps.

5) Over this time ConnectBot was improved. Main improvements were: attempt to claim early "bridge rivers" i.e. rivers that are important links between different parts of map. This idea was actually stolen from one of the MetricsBot metrics. Another improvement was to add some "optimistic value" to rivers claimed at the second phase (i.e. when all mines that could be connected are connected) to select rivers that have a lot of far (i.e. "good") rivers beyond them

Generally the result of competition of the current ConnectBot vs current MetricsBot was decided by how similar the set of enemies and the list of maps used during training were to the enemies and map used at the verification. If they were close - MetricsBot was much stronger, if they were not - much weaker. Important issue was that as we only have two different bots, using 4-bot map mean we need to add a few bots of the same type and they often began to fight for the same goals and thus made life for the enemy easier. Another major issue was that ConnectBot played relatively slowly on big maps so we couldn't run optimization on big maps.

6) For the last night we tried to run a massive training on AWS to optimize MetricsBot. Unfortunately we miscalculated memory usage and half of the nodes just failed and other didn't produce good results either. Before the last night ConnectBot was on average better than "auto-tuned" MetricsBot. However over the night one of the team members has come up with a new (modified) metric and also has analyzed results and claimed that most of other metrics doesn't do anything important. Thus "hand-optimized" MetricsBot was born. It uses only two metrics and some hard-code (not "optimized") params. And still this is the bot that we are submitting with fallback to (speed-up) ConnectBot in a few cases. In our test their raw points results are close but the "hand-tuned MetricsBot" usually beats the ConnectBot. Also it seems to play better against YAGI - the only "strong" player easily available at the test server.

7) As for bonus features:

We decided that Futures are not worth bothering implementing them given their value trade-off. Splurges is an interesting feature but we had no ideas how to add it to either ConnectBot or especially MetricsBot. As for Options we started implementation in both bots but failed to get anything workable.

=================
TL;DR
Our team tried several ideas while participating this year ICFPC.
First, we implemented the bot (ConnectBot) which uses simple logical decisions.
For example, we assume "mines" are important, especially ones we can interconnect,
so this bot tries to capture shortest paths between sites. Another thing it fights for
is "bottle necks", that is narrow path segment, which could be easily captured and
won't allow the enemies to connect their distant points through bottle neck. Then our
ConnectBot just builds long path segments, starting from our claimed mines or mines clusters.

Also we created the bot, working on completely different approach. First we created a set of
(we believe) meaningful simple metrics and next we use combination of them, weighted with
pre-calculated coefficients (using gradient descent). This bot - MetricsBot - shows good
speed on big maps, but unfortunately, not as good in score (approx 10-15% worse).
Perhaps if using bigger training environment MetricsBot can be refined a lot.

## Writeup by MetricsBot maintainer

The idea was to extract set of various features from map for each edge, then find proper weighting multiplier for these features (metrics). For example, following features were implemented, from simple to more complex

1) Distance to closest mine, regardless of reachability

2) Distance to closest mine or owned subgraph connected to mine
3) Sum of connected exits (edges) on both sides of edge
4) Whether the edge is part of shortest path between some mines
5) “Bridge coefficient” - the length of detour route if edge is unreachable/taken. Bridge is in the sense of graph theory bridge, i.e. important linking edge.
6) Complex bridge coefficient: the length of detour if bridge is closed and first detour is closed.
7) the number of shortest routes coming through the edge, multiplied by bridge coefficient. This is estimation to graph flow over the edge
8) Exact score that will be achieved by us if edge is taken

Unfortunately, we did not manage to add something like “smallest cut” of a graph (but calculated bridges/flow instead). Also full flow over graph also was too complex for us to calculate.

We also did not implement calculation of metrics from opponent point of view due to lack of human time, although on maps with lots of opponents this wouldn’t give an advantage due to rapid evolution of game state. It would work only on maps with 2 opponents.

We used *weighted sum* of normalized metrics. Metrics were normalized from 0.0 to 1.0 and weights were searched by both random and gradient descent methods to maximize on given set of maps and opponents.

We did not manage to come up with the *non-linear combination* of edge metrics and Global Metrics such as game progress (time), number of opponents, size of map and general map features (such as sparseness, number of mines etc). Combination of these metrics exploded the number of tunable weights.

Also, the speed of opponent bot (ConnectBot) on large maps prevented massive tournaments and training of MetricsBot on medium and large maps (later, it was improved though).

In short, the linearity of our scoring formula prevented us from building bot with such weights that it would beat hand-written bot in one-to-one competition on larger set of maps. On three small maps, we could beat it, but I think this was due to overfitting / luck because of the discrete nature of game specially on small maps and determinism of opponent bots (our bot was also deterministic).

As contest advanced, we added more complex metrics (metrics  7 and 8), and later discovered that all previous metrics did not add value in common formula. Two metrics weights tuned by hand by one team member proved best on any combination of maps with 3+ opponents. Control training on all maps with only 2 these metrics gave similar weights. But with 2 bots only, the ConnectBot was winning more often.

What we did not have time for (implementation or testing):

1) Splitting maps/opponent numbers/sparseness of maps by categories and finding weights in each category separately
2) Adding meta-bot which would select ConnectBot for 2-bot plays and MetricsBot for others.
3) For small maps, *10 seconds of startup was enough to perform training* (search of proper weights by random search) of MetricsBot on this specific map with few bots including ConnectBot, and probably tune parameters for win. Unfortunately, although I successfully added this to init routine, some unrelated weird bug was preventing inclusion of this warm-up into final solution. Therefore, I expect us to perform poorly on smaller maps

We also didn’t implement any mini-max algorithm or similar, this could give us some boost when playing against one or even 2 opponents.

Regardless of our own suboptimal performance, we think this contest was organized poorly in terms of interactive competition (F stands for FUN). Given that it’s hard to invent interactive leaderboard for this kind of contest task, it’s a pity organizers chose to select task of this kind. It’s neither functional programming related, nor competition (during contest time), and (by its format) mostly looked like academic assignment.

## Code metrics

We wrote

- 3287 lines of Kotlin
- 1087 lines of Java
- 610 lines of C++
- 80 lines of shell script
- 208 lines of Python

On largest maps, one turn took

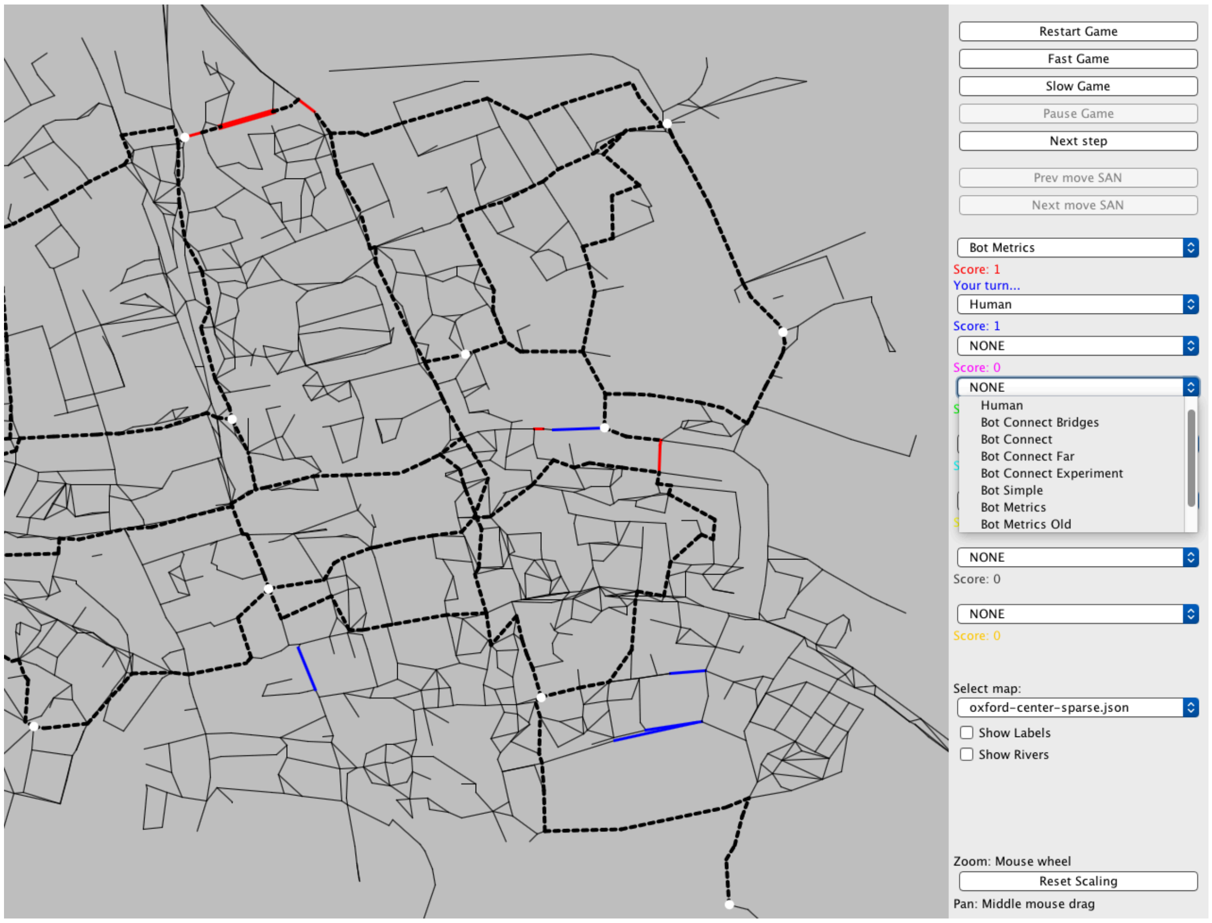
- 30 milliseconds in MetricsBot
- 150 milliseconds in ConnectBot

Git:

- Over 250 commits

# Appendix

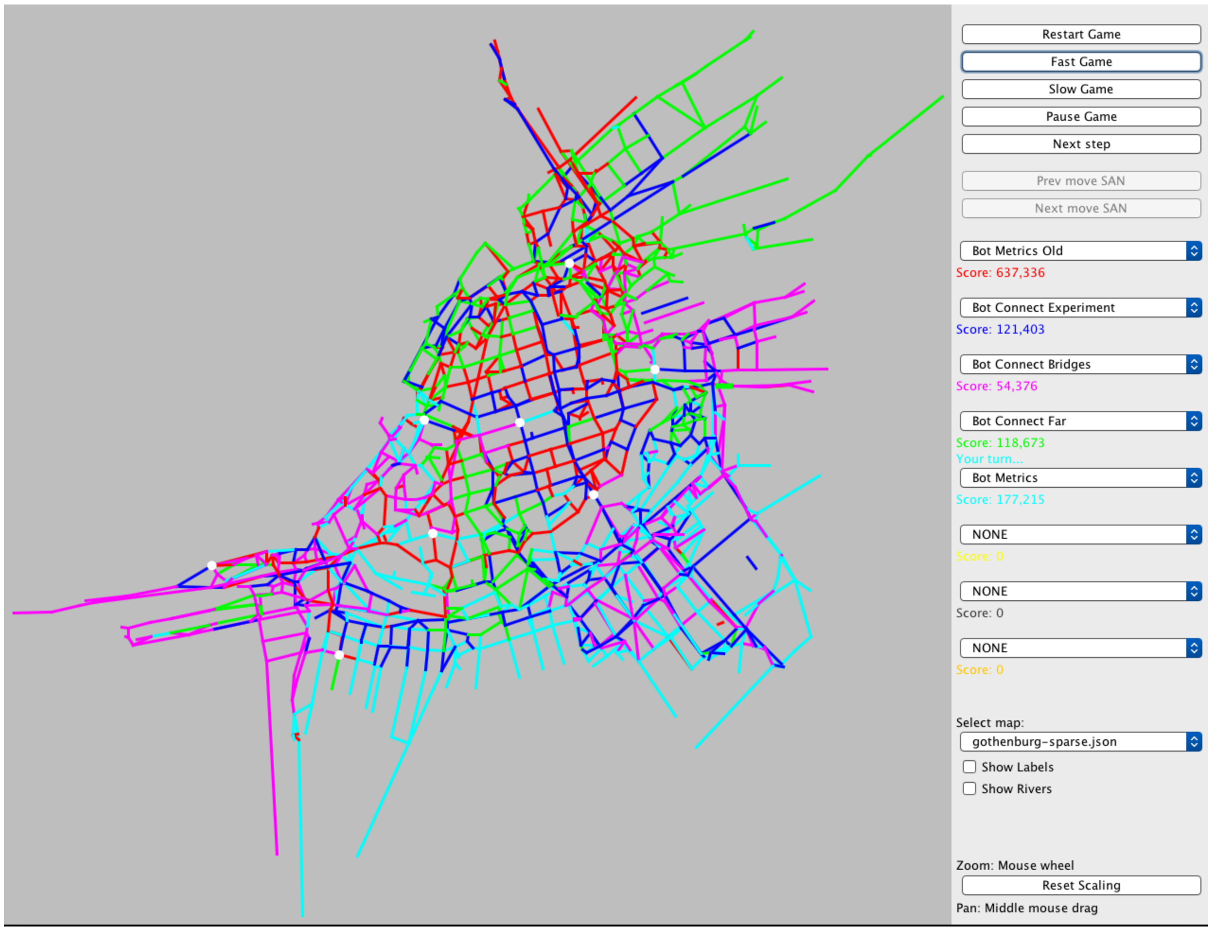
## Screenshots



Bot Metrics
Score: 1
Your turn...
Human
Score: 1
NONE
Score: 0
NONE
NONE
Score: 0
NONE
Score: 0

Select map:
oxford-center-sparse.json
☐ Show Labels
☐ Show Rivers

Zoom: Mouse wheel
Pan: Middle mouse drag



Restart Game
Fast Game
Slow Game
Pause Game
Next step
Prev move SAN
Next move SAN

Bot Metrics Old
Score: 637,336

Bot Connect Experiment
Score: 121,403

Bot Connect Bridges
Score: 54,376

Bot Connect Far
Score: 118,673
Your turn...

Bot Metrics
Score: 177,215

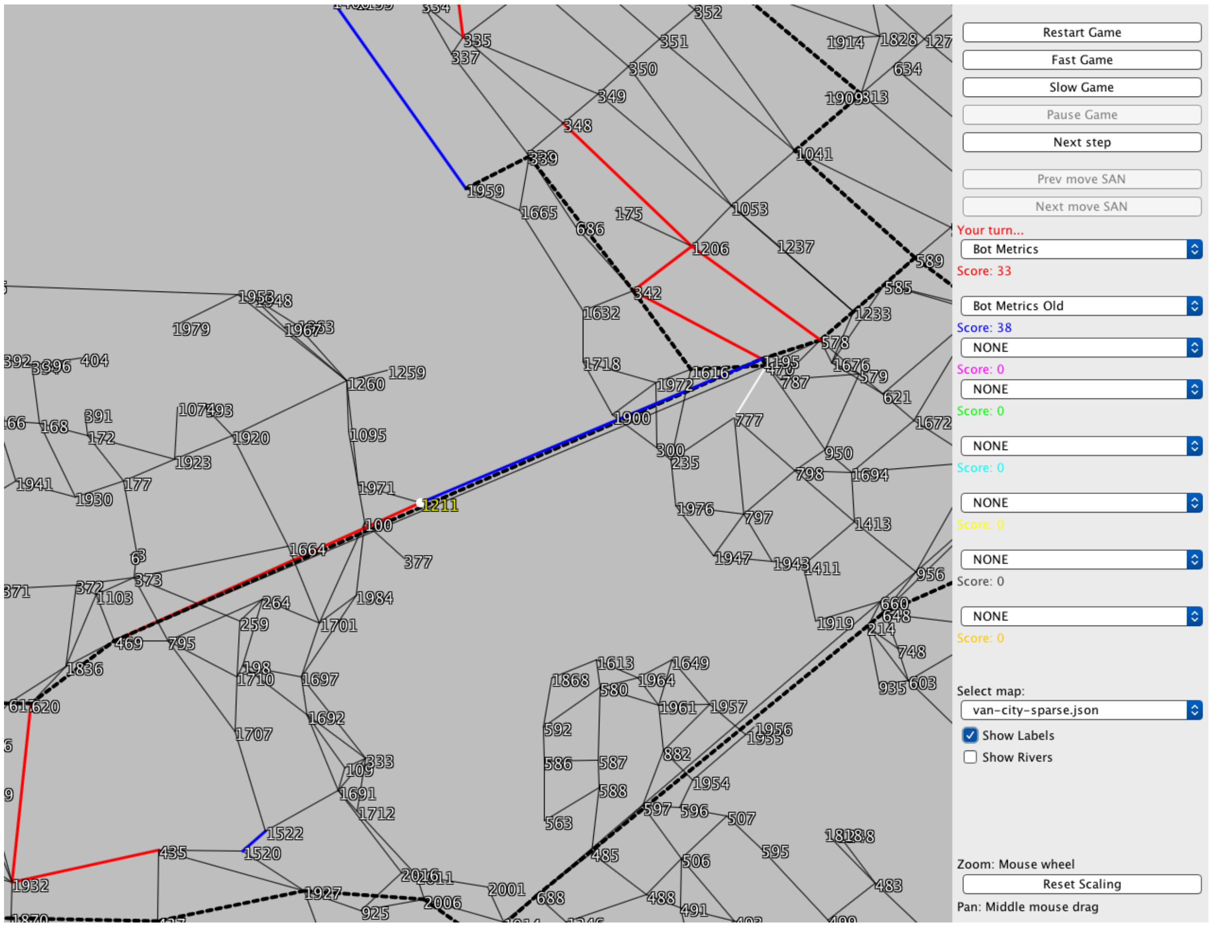
NONE
Score: 0

NONE
Score: 0

NONE
Score: 0

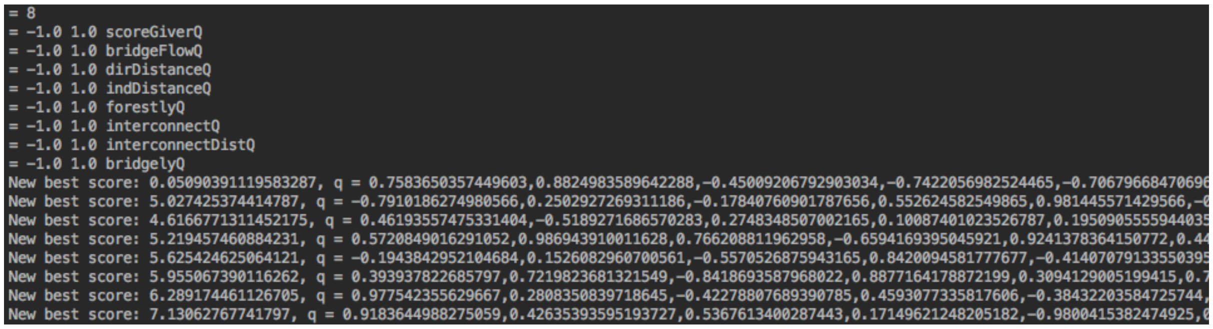
Select map:
gothenburg-sparse.json
Show Labels
Show Rivers

Zoom: Mouse wheel
Reset Scaling
Pan: Middle mouse drag



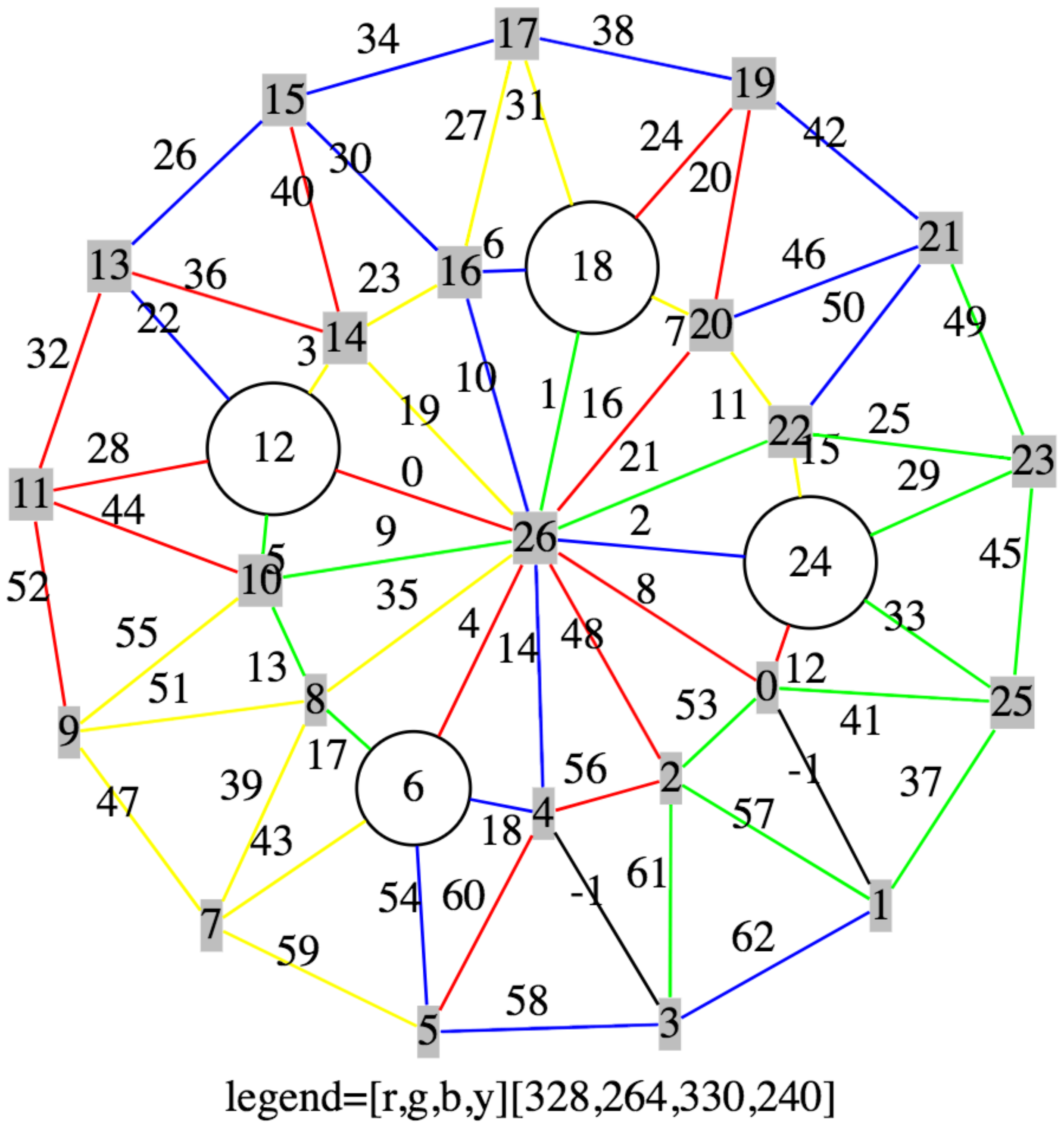
Your turn...
Bot Metrics
Score: 33
Bot Metrics Old
Score: 38
NONE
Score: 0
NONE
Score: 0
NONE
Score: 0
NONE
Score: 0
NONE
Score: 0
NONE
Score: 0
Select map:
van-city-sparse.json
[✓] Show Labels
[ ] Show Rivers

Random search of weights



```
= 8
= -1.0 1.0 scoreGiverQ
= -1.0 1.0 bridgeFlowQ
= -1.0 1.0 dirDistanceQ
= -1.0 1.0 indDistanceQ
= -1.0 1.0 forestlyQ
= -1.0 1.0 interconnectQ
= -1.0 1.0 interconnectDistQ
= -1.0 1.0 bridgelyQ
New best score: 0.05090391119583287, q = 0.7583650357449603,0.8824983589642288,-0.45009206792903034,-0.7422056982524465,-0.70679668470696
New best score: 5.027425374414787, q = -0.7910186274980566,0.2502927269311186,-0.17840760901787656,0.552624582549865,0.981445571429566,-0
New best score: 4.6166771311452175, q = 0.46193557475331404,-0.5189271686570283,0.2748348507002165,0.10087401023526787,0.1950905555944035
New best score: 5.219457460884231, q = 0.5720849016291052,0.986943910011628,0.766208811962958,-0.6594169395045921,0.9241378364150772,0.44
New best score: 5.625424625064121, q = -0.1943842952104684,0.1526082960700561,-0.5570526875943165,0.8420094581777677,-0.41407079133550395
New best score: 5.955067390116262, q = 0.393937822685797,0.7219823681321549,-0.8418693587968022,0.8877164178872199,0.3094129005199415,0.7
New best score: 6.289174461126705, q = 0.977542355629667,0.2808350839718645,-0.42278807689390785,0.4593077335817606,-0.38432203584725744,
New best score: 7.13062767741797, q = 0.9183644988275059,0.42635393595193727,0.5367613400287443,0.17149621248205182,-0.9800415382474925,0
```

Online play visualization:



Gameplay: (click link to see video)

https://youtu.be/pfIpqRmjr0k

https://youtu.be/98xiyMoikTI

10 August 2017