Artificial Intelligence for Games

Freeciv

Authors:
Łukasz Jankowski, Daniel Krzywicki

Table of Contents
Authors: .................................................................................................................. 1
Table of Contents .................................................................................................. 1
1. Introduction ..................................................................................................... 1
2. Initial ideas ....................................................................................................... 3
   Global city algorithm ......................................................................................... 3
   Population allocation by aggregating cities. ...................................................... 3
3. Description ....................................................................................................... 4
4. Experiments ..................................................................................................... 6
5. Cost estimate ................................................................................................... 8
6. Conclusions ..................................................................................................... 8
Attachments .......................................................................................................... 8

1. Introduction

This document aims to describe our work during AI for Games classes. Our goal is to improve a number of closely related issues regarding the development of an empire:

- Global optimization of city population allocation
- Improvement of assessment of potential locations of new cities
- Improvement of assessment of potential conquest of enemy cities

City population allocation is optimized separately for every city. Cities optimize their allocations successively. A city can allocate population only on tiles that hadn’t been used before by other cities. Such algorithm is of course suboptimal.
Similarly the assessment of potential locations of new cities is performed. In the assessed location a virtual city is created and it is filled with population. The total production of the city is assessed. It is done by optimizing the allocation of population exactly as in case of a real city. The virtual city can place its dwellers only on fields not taken by existing cities. Such allocation is not optimal and so the assessment of potential city locations is inaccurate.

Similarly the assessment of potential conquest of enemy cities is performed. The value of city depends on its production and it is calculated by optimizing the allocation of its population. As the city can only place its dwellers on tiles not previously occupied, the allocation is suboptimal and the assessment of city value not accurate.

We came out with several possible solutions, which will be further described in this document. Unfortunately, we didn’t succeed in implementing them in FreeCiv, due to lack of time and sufficient familiarity with the game system. However for one of the algorithms, we have done simulations that proved our idea to be correct, and also provide code which would allow someone versed enough in the game engine to implement this solution.
2. Initial ideas

Global city algorithm

One of possible upgrades to city allocation would be to treat all cities as one global city and optimize its population allocation with the same algorithm as in case of a single city. However one issue arises. A single city can place any of its dwellers in any location within its borders. A global city can not do that. Each dweller of a global city must be placed within the borders of the city he is actually from.

There arises an issue of the order at which the dwellers of a global city should be placed. If at first all dwellers of one city would be placed, than all dwellers of second city etc., then the algorithm would be exactly the same as at present. Better results would be achieved by placing dwellers in different order. Orders that definitely would be better are: random and round-robin. Since each city first places its dwellers in places that are best within its borders, with these orders it is probable that the first city to place a dweller in the disputable location, would be the city having most to gain from it. The solution would be better than currently.

1. If there exists a dweller X that has not been placed yet go to step 2, else end. (X should be chosen using random or round-robin order)
2. Place X within the borders of the city he is from using the algorithm for single city.
3. Go to 1

Population allocation by aggregating cities.

A conflict between two cities takes place when two conditions are met:

1. Cities must have a common piece of land
2. If each city was allowed to place its dwellers on any position within its borders, they would choose one or more common tiles.

The first condition is necessary. If it is met, then a potential conflict between cities exists. If condition 2 is met then an actual conflict between cities exists.

The map of potential conflicts can be presented as graph \( G_p = (V, E_p) \) with vertices being the cities and edges connecting the vertices for which condition 1 is met.

The map of actual conflicts can be presented as graph \( G_r = (V, E_r) \) with the same vertices but the set of edges \( E_r \) being the subset of \( E_p \) and containing only the edges for which condition 2 is met.

We identify the connected components of graph \( G_p \). To achieve a globally optimal allocation, cities belonging to the same connected component should be optimized together as one global city.
It is however possible that the allocations for each connected component would be in conflict with each other. It is possible because the tiles that are the effect of optimizing together the population of several cities are not the same as the tiles that are the effect of optimizing these cities separately. This means that new conflicts can appear along the edges from $E_p$ but not from $E_r$. This shows that the condition 2 is not necessary for two cities to be in conflict as the conflict can as well be indirect.

In case of such conflict between separately optimized connected components, further aggregation is necessary. Let graph $G'_p = (V', E'_p)$ be a graph with vertices being the connected components of graph $G_r$, and edges connecting the connected components that have common territory (condition 1).

Let graph $G'_r = (V', E'_r)$ be a graph with the same vertices but $E'_r$ being a subset of $E'_p$ and containing only the edges for which condition 2 is met.

Common components of graph $G'_r$ are identified. Population allocation is again performed for each common component. If necessary the aggregation is repeated until there are no more conflicts.

1. $n = 0$
2. Optimize allocation separately for every city.
3. Build graph $G^n$
4. If graph $G^n$ contains at least one edge go to point 5, else end.
5. Identify common components of graph $G^n$
6. Aggregate cities belonging to the same common component.
7. $n++$
8. Go to 2.

Optimizing population allocation through city aggregation can give globally optimal solutions in better time than the Global City Algorithm. It is possible by dividing the problem into subproblems. However in pessimistic case further aggregation will be necessary until finally one global city would be formed. In this case this algorithm would be worse than the Global City Algorithm.

3. Description

Let assume the following:

$\bar{x}_i$ - a city $i$ (some set of fields)

$s$ - a function computing a solution

$s(\bar{x}_i)$ - the solution for a city $i$ (some set of fields)
Let say that some $n$ cities have some common intersection, that is $\bigcap_{i=1}^{n} x_i \neq 0$, and moreover, the solutions for that cities also intersect, giving us a common area $\bar{y} = \bigcap_{i=1}^{n} x_i$

The current approach is to give $\bar{y}$ to whichever city comes first. It is efficient but strongly suboptimal. We can safely guess that the optimal solution would much more likely be some partition of $\bar{y}$ among the $n$ cities.

Indeed, we can show that the actual method satisfies:

$$\|Actual\| = \text{rand}_i(\|s(x_i)\| + \sum_{j \neq i}^{n}\|s(x_i - y)\|)$$

when the optimal solution would be

$$\|Optimal\| = \max_{(y_1, \ldots, y_n) \in \mathbb{Z}_n^n} \left( \sum_{i=1}^{n} \|s(x_i - y + y_i)\| \right)$$

that is, one among all possible $n$-partitions of $y$ that would maximize the total value.

Of course, such an optimal solution is computationally hard to find, as we would have to examine all the $n^{\|y\|}$ possible partitions of $\bar{y}$.

But as it often happens in practical cases, we can settle for a sufficiently good solution instead of a perfect one.

Our proposal is to iterate through all the elements of $\bar{y}$, and for each of them, assign it to the city that most needs it. To that purpose, we can define the alternate cost of $y_j \in \bar{y}$ for the $i$-th city as:
\[ c(\bar{x}_i, y_j) = \|s(\bar{x}_i)\| - \|s(\bar{x}_i - y_j)\| \]

This value represents the cost it takes for the \( i \)-th city \textbf{not} to have \( y_j \) at its disposal.

We then try to minimize the total alternate cost of all \( y_j \in \bar{y} \) over the \( n \) cities, by giving each \( y_j \) to the city with the highest value of \( c(\bar{x}_i, y_j) \).

### 4. Experiments

In an attempt to validate our idea, we wrote a appropriate simulation in Matlab. The source code is to be found in an attachment.

The situation in the simulation was the following: We assumed two cities of size 5x5, overlapping by 20%. We distributed randomly resources over the area, but increased the amount of resources in the contested zone by a factor of 2, to make conflicts to happen more often. Finally, we normalized the resources over the area. We then looked for a solution of size 15 (to make it bigger than the contested zone).

These results show the solutions values for given method. They were averaged over 5000 simulation runs:

<table>
<thead>
<tr>
<th>Method</th>
<th>solution for city1</th>
<th>solution for city2</th>
<th>overall solution (^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFS (^2) for city 1</td>
<td>0.5945</td>
<td>0.3209</td>
<td>0.9154</td>
</tr>
<tr>
<td>FCFS for city 2</td>
<td>0.3211</td>
<td>0.5947</td>
<td>0.9158</td>
</tr>
</tbody>
</table>

\(^1\) A value of 1 means that all available resources were used.
\(^2\) First Come First Served
We can clearly see that our method does not only give more predictable results, but also evenly distributes resources, and last but not least, improves the overall solution (will can clearly see the added synergy).

The improvement is not very high in this case (2.87% - 2.92%) - but if we look at it from how far it is from 1, we then see that about 30% less resources are wasted.

Our method behaves even better if the cities have an asymmetrical situation. We designed a second simulation, in which in addition to all above we also diminished by a factor of 2 the amount of resources available for the first city outside the contested zone.

These results were also averaged over 5000 simulation runs:
We can see that this time, the original results can differ greatly, depending on which city gets there first. When using our method, the distribution is again much more even, the overall improvement up to 5.7837%, and the amount of wasted resources less by up to 52%.

5. Cost estimate

The situation we are considering is not very likely to happen: A set of allied cities must intersect and so must do their respective, independent solutions. Therefore in the general case not much additional computations must be made - just checking for that condition.

If that however happen, we can assume that the number of cities is low, and the intersecting area as well - building allied cities close to each other is not very common.

We then need time proportional to the size of the area $||y||$. For each of its elements and $n$ conflicting cities, we need to compute the alternate cost. As we already have a valid solution, we just need to slightly modify it, and that can be done in a time proportional to the size of the city $||x_i||$.

Finally, we need to know the value of each solution, so a fitness function $f(s)$ must be computed.

It thus gives us $O(||y|| * n * ||x_i||) = O(f)$, which is reasonable because of the strongly bounded values of the parameters.

6. Conclusions

To sum up, be believe to have proven that our solution is correct, gives measurable benefits at an affordable cost. It can thus be considered as a possible improvement to FreeCiv’s system of city management.

Attachments

- simulate.m - A file containing the Matlab simulation code.