Implementing Last Diminisher to Assign EVE Online Corporation Tasks

Abstract

This project attempts to implement the Last Diminisher algorithm in order to divide typical tasks among cooperating players in the MMO video game, “EVE Online”. The tasks are considered indivisible in nature, and the algorithm attempts to divide these tasks among the players, given their preferences for how difficult they view a task to be. The algorithm is written in such a way that it can take input for any list of indivisible tasks and any number of individuals who have submitted their preferences; in this way, it is not specific to EVE Online.

Project Inspiration

I have been an avid EVE player for several years. Like many MMO players, the social aspect of the game keeps me interested; achieving a common goal with friends is satisfying. To that end, I started a “corporation” (a team of players whose interests, goals, and personalities align) within the game. This corporation provides a common space for pilots (the players) to interact, but equally importantly, it provides several benefits to pilots that they could not otherwise obtain independently. These can be monetary, strategic, or simply the ability to open other areas of the game that would remain too difficult to conquer alone. However, these benefits must be generated somehow; tasks (chores) can span the various facets of EVE and
require teamwork to achieve optimum efficiency. Some pilots may be proficient at some tasks, while others may simply prefer different tasks. This sort of problem clearly lends itself to implementation of a fair-division algorithm.

I chose the Last Diminisher algorithm because it seemed reasonable to implement, while being able to assign tasks in discrete steps, rather than a moving-knife algorithm. The other strong consideration was for Fink's Lone Chooser, because of its unique advantage of being able to add more players after the division had already begun. However, in practice, the list of tasks changes at least as rapidly as the list of pilots available to complete them. I decided that a preference list for each character, updated whenever and however they choose, would best serve the needs of the corporation. Then, the algorithm could be run anytime a new list of tasks is set, and players can be assigned tasks reflecting their most recent preferences.

**Implementation**

As previously mentioned, the fair-division algorithm I chose to implement was the Last Diminisher. It's implementation went relatively smoothly, with assistance from a dynamic programming solution to the knapsack problem for deciding the task lists, as was suggested by someone else with a similar project in class. The algorithm reads in a comma separated value file with names of pilots listed in the first row, followed by a row for each task and each pilot's positive integer rating of how difficult it will be to complete that task. Pilots may specify any positive integer for any difficulty, and are not limited to a “pool” of difficulty points in any way. This allows any pilot to effectively refuse a particular task, by assigning it an extremely large number, given that some other pilot doesn't feel as strongly about it. The algorithm also randomizes pilot order.
Last Diminisher works by having the first pilot create a task list that they view as proportional. Because the tasks are discrete, there must be a level of error allowable, up to the value of one task. This list is then passed to the next pilot, who has the choice to either agree that it is proportional, and pass it along, or believe that it is smaller than a proportional share, and add tasks to it until it becomes proportional. The last pilot to add tasks to the task list in a given round, receives that list as his or her list of tasks. After the Last Diminisher goes through enough rounds to reach two players, a simple divide and choose (again, random order) selects the remaining tasks. It is also important to note that the Last Diminisher algorithm is not envy-free; any pilot who is assigned a list earlier may envy a pilot who receives a later list.

Analysis

Data was obtained by requesting preference lists from a select group of pilots within the corporation, on an example set of tasks. Tasks were divided by the algorithm in somewhat puzzling ways, for several reasons. First, it became evident that pilots were not able to judge their own relative valuation of a medium to large number of tasks efficiently. This could be mitigated in time, as pilots receive tasks they don't like, because they could alter their preferences for the next time tasks are run and task lists assigned. Secondly, the algorithm does not account for synergies among tasks. For example, it might make sense for one person to haul manufactured products to the trading station, sell those products there, buy new materials for the next round of manufacturing, and haul them back to the home starbase, instead of having those tasks split up. This is a large hurdle that could possibly be overcome with a different format of preferences that uses some super-set of tasks, but then the first problem of valuing tasks
relatively becomes even more daunting. Thirdly, the number of tasks given for the number of pilots available leads to some strange results; some pilots may not have any tasks, because of their order in selection coupled with their valuations of tasks. There is no way currently to track this, and have such pilots “make up” their work the next time the algorithm is run on the next set of tasks. Finally, the algorithm cannot take into account the possibility of some people not being able to perform some tasks. For example, newer members are not typically trusted with high-value cargo, in fear that they may run off with it. This would leave them incapable of completing some tasks, but the algorithm does not see that.

Results

Results from the algorithm are straightforward. Running it on the webpage shows the tasks assigned to each pilot; particularly interesting are the amounts of proportionality that each pilot thinks he or she received. No pilot (of 12 considered) received more than 8.75% of the whole task list, in their own view; this is a good example of the algorithm holding proportionality.

Conclusions

The algorithm has its shortfalls, documented in detail in the Analysis section; namely, there seems to be a tradeoff between ease of defining preferences for the pilots and the ability for the task lists to efficiently represent all combinations of tasks, as well as the inability for the algorithm to take into account “lucky” results for some pilots in the next iteration. Envy in general can be a problem, even if the aforementioned issues were minimized; the nature of the algorithm does not guarantee envy-freeness. Given more time, it would be interesting to
investigate and quantify the amount of envy each pilot may have, and compare that with a subjective measure of that pilot's estimation of another pilot's task list. This could also be used to subjectively judge if a pilot's valuation of the tasks were accurate. Ultimately, the corporation feels as though the algorithm could be implemented in regular use, if synergy issues were solved without added complexity to the pilots. This may or may not be possible with the Last Diminisher algorithm, but the results so far have shown potential.