G53DIA Assignment 2 Report

Joel Cox psyjc5

Introduction

The agents used in this task use the same architecture as those in assignment 1 (report included), with one errata: The architecture is reactive with state, not deliberative as previously stated.

The task is similar to that of assignment 1, except that there are now multiple agents being used to achieve the same task. The goal is still to maximise score, however score is now total waste collected divided by the number of agents used.

Multi-Agent Problem

The expected score for multiple agents is around 70-80% of the single agent runs. This is due to co-ordination overhead, as well as saturation of the environment with agents, which all try to accomplish the same task, lowering available tasks per agent.

A good explanation for a drop in score is thus:

If a tanker can reliably complete 8 of 10 available tasks, it can be seen as being near-optimal. However, if a second tanker joins in to help, they will each only be able to average 5 of the 10 available tasks, since each tanker will take away available tasks from the other. Whilst the overall completed tasks will be higher (higher waste disposed), the average per tanker will not be as high as the original 8 of 10 tanker.

Task Scheduling and Co-Ordination

An issue with the base tanker from assignment 1 is that it beelines for any available task. When grouped together, all the tankers aimed for the same task, but only one would be able to claim it.

Base Approach

As mentioned, the agents had no way of knowing if another agent was heading for a specific task, and this was inefficient.

The basic approach to fix this was to implement a simple boolean on tasks, to identify if a tanker was heading to it. This resulted in the agents being very effective, operating at a very similar efficiency to a single agent environment. This meant that any modifications to the task scheduling approach would provide meaningful information about how they affected the score.

Contract Net Protocol

Contract Net Protocol (CNP) is a widely used, simple task scheduling protocol. The implementation of it for the multi agent system has a large time issue due to the sequential processing of the agents, increasing co-ordination overhead. The overhead cost using a traditional protocol implementation would waste up to 3 time steps for every task found – signal, response, and confirmation (the expediting step being a meaningful step) – which is too slow. Instead, the initial planned implementation would use an additional manager class to handle all task assignments, effectively reducing the overhead cost.

Implementation

Whilst implementing a version of contract net protocol it became apparent that the current tanker architecture was unsuitable for using contract net protocol.

Unsuited Architecture

The way the Efficient Tanker and its derivatives operate is reactive. They cannot plot a route from A to B, because they will only ever head towards locations they don’t have to plan around; locations within fuel range. This means that the either an overhaul of the tanker architecture would be needed, to enable tankers heading to tasks that are far away, or the CNP manager would only assign tasks to tankers within their fuel range; this would not be different in practice from operating without CNP.

Not Required

A major benefit of using CNP in some systems is the ability to request the best agent for a task. However, all tankers are identical in ability, albeit slightly different in behaviour. This means that it will effectively just select the tanker which is either closer or has a higher waste capacity. The tankers themselves already filter distance using biases, and also have considerations for waste capacity. In addition to this, testing from assignment 1 showed that it seems to be better to choose lower distances over using the remaining tanks efficiently.

CNP: Conclusion

Whilst I would have liked to investigate a CNP system for this assignment, it is unfortunately unsuited to the tanker architecture used. If the tankers were capable of plotting routes between fuel pumps in order to get from A to B, or if the tankers had distinct differences in ability, such as reduced waste capacity in exchange for increased fuel efficiency, or vice versa, than utilising CNP would be explored. However, CNP currently offers no distinct benefits, and requires overhauling of the tankers. Additionally, a simpler approach has worked very well for the tankers thus far, and so the current system of “claiming” tasks will be continued, instead of CNP.

Task Scheduling: Conclusion

The result of investigating CNP leads me to believe that perhaps the simple Base Approach could be one of the most optimal. When paired with Lonely Tankers (see below), the tankers manage to split up across the environment with good efficiency. This leads me to believe that a complex task scheduling system would not benefit the simple tankers, and rather modifying the tankers themselves would provide better results.

Implicit Task Management

A facet of the simple “claiming” task acquisition system, when paired with the tankers biases for task selection, is the idea that the tankers will implicitly manage the tasks between themselves. It is not a sophisticated system, but by continually choosing tasks further away from each other, as the Lonely Tanker does, the tankers effectively take up part of the environment for themselves.

This implicit task management was not planned, nor really considered part of the system. I just thought it was an interesting perspective worth highlighting.

Dynamic Fleet Scaling

Dynamic Fleet Scaling (DFS) is based on the theory that the potential waste available to deliver is restricted by the amount of Fuel Pumps present in the environment. This idea stems from the fact that each tanker is restricted by fuel limits, and so cannot stray too far from a “safe” area around fuel pumps. This limit is avoided, and multiple fuel pumps are exploited by adding an additional Tanker per fuel pump found in the environment.

Tests from the single agent assignment indicated that a single fuel pump environment would result in a score of approximately 70k-90k; this is the target score which was aimed for, although a reduction was expected due to the cost of running multiple tankers.

Changes to the Simulator:

In order for DFS to work within the given simulator, non-substantial code (1 line) must be changed. This allows adding extra tankers mid-simulation, and was discussed and approved. A guide to modify the code is provided alongside the source code.

Potential Issues

There is an issue when introducing additional tankers, due to the way the average score (and thus overall score) is calculated. If a single tanker obtains 1000 score in 100 steps, then a second tanker introduced after 100 steps would have to effectively obtain 2000 score in the next 100 steps, to make up for the lost time. This catch-up can be performed over the tanker’s entire lifetime, so it is not that impactful for the second tanker, or even the third. However, when the program reaches higher numbers, the potential catch-up distance becomes a lot greater:

Assuming that the first 3 tankers are all contributing equally, and averaging 1500 score in 1500 steps; When a fourth tanker is introduced at 1500 steps, it now has to effectively average 6000 (1500 base + 4500 extra) score in 1500 steps. With 10 000 maximum steps, that leaves an actual time frame of 8500 steps to make up that last score. This means that the actual demand (assuming a continued expected base average of 1 score per step) is an additional 4500 score over 8500 steps, which works out to an additional 0.529 score per step. The impact of this is that the fourth tanker, if introduced at 1500 steps, has to work 1.53 times as effectively as the previous three tankers. If step 1500 were to be the upper limit for the last tanker to be introduced, the working cost works out as such:

Tanker Number	Required extra score	Multiplier
4	4500	1.53
5	6000	1.71
6	7500	1.88
7	9000	2.06
8	10500	2.24
9	12000	2.41
10	13500	2.59

From the above table it is apparent that the multiplier for additional tankers becomes frighteningly large. Fortunately, due to the low rate in which tasks appear in the environment, not much score is obtained before 1500 steps. After ~1500 steps, tanker effectiveness ramps up considerably due to task density increasing. This is why no additional tankers are added after 1500 steps. A normal tanker is expected to achieve around 1000 score in the first 1500 steps, but then is expected to average around 10 000 score per 1000 steps after 1500 steps. This means that the catch-up multiplier is easily reached. If the cut off point were earlier than 1500, less tankers may be added, and an environment may not be fully exploited, and if it were later than 1500, the required effectiveness of tankers would increase too much to be feasible.

After testing a hard step limit, a limit of 750 was found to be better than 1500. The score differences were not too large, so it is likely that the tankers were simply wasting the extra steps.

Next, I decided to try testing limits of scores. By sheer luck, the first limit I tested, 800, turned out to result in near exact results to a step limit of 750 (differing in average by only 20 across 30 test runs), which was previously being used. This gave a decent benchmark for further tests. These further tests showed that there was quite a margin within which this value could sit. All values between 750 and 1500 proved to yield identical results for the test set, with values either side of this range being worse. I decided to stick with 1500 as a limit, as it – rather conveniently – matches the maths above.

Results

The DFS approach used a single starting scout tanker, and an additional tanker for each additional fuel pump found. This means that a single fuel pump should only have due a single tanker – the scout. However, as the assignment is about multiple agents, in the rare case of a single pump environment, an additional tanker will be added, giving a minimum of 2 tankers. Testing environments did not include a guaranteed single pump environment.

Early tests of the DFS approach with Leash Tankers showed poor results; limiting the Tankers to a certain fuel pump stopped them from exploiting particularly rich areas of the environment, or ‘bouncing’ between two pumps with high task density.

A second version of DFS used Lonely Tankers instead, which proved to be very effective. An average score of 85k-105k was observed, with total waste delivered hitting over 1 000 000. This was surprising, as it indicated certain seeds supported using 10 tankers efficiently, without losing to much to the cost of excess tankers. For comparison, basic tests with the Base Approach using 4+ tankers had proved very inefficient, averaging around 60k, peaking at 80k. The high score range also exceeded the expected target range, showing that the cost of running multiple tankers is not actually that high if used sparingly. However, the co-ordination used for DFS was primitive, just using the Basic Approach, so there was minimal overhead costs.

The lowest scores observed were about 55k-65k, which shows there are some seeds where this approach is not very efficient. Some of these environments were those that the single agent assignment had also struggled with, but others were ones where it had succeeded, indicating that there was occasionally a real cost.

Also observed was the pattern that average score seemed to correlate slightly to tanker amount. Whilst previously thought that lower fleet sizes would put up better results, DFS showed results to

the contrary, with the highest scores, 100k+, being between 4-10 tankers. The environments that supported 10 tankers would reliably show scores of 95k+.

In conclusion, DFS was considered a success, putting up scores that were impressive, albeit less than the Base Approach. Normally, total waste would not be heavily considered, but the fact that it can show total waste amounts of over 1 million, as well as putting up better scores with up to 10 tankers than the Base Approach could with more than 3 tankers, means it is a serious consideration for a final approach.

NOTE: Unfortunately the processing time for DFS was annoyingly slow, so testing was difficult. Further testing might have yielded stronger results, or revealed that the approach was actually unreliable.

Simple Multi Agent Approach

Also tested was simply using multiple agents with no dynamic components. For example, using 3 Lonely Tankers, or 2 Efficient Tankers and a Scout. Further Detail about each tanker type can be found at the end of the report.

Testing Combinations.

Initial tested used only Efficient Tankers, the base tanker type from assignment 1. These tankers actually yielded very high scores – averaging 110 000 score over 30 runs. However, it became apparent these agents would get stuck trying exploit a small section of the environment available to them. To improve this, several options were considered

Scouts

One attempt was to have a tanker dedicated to exploring the environment for the early part of the simulation. Due to the low task density at the beginning of the simulation (detailed earlier), these tankers should not miss out on much efficiency.

Testing showed that pairing 2 Efficient Tankers with a single Scout Tanker was better than just using Efficient Tanker, giving a 2-5k rise in average score.

Lonely Tankers

Another approach was to implement a bias to encourage the tankers to go to different parts of the environment, to better exploit it. This also proved quite effective, although performed similarly to the prior test of 2 Efficient and 1 Scout.

Results

Overall, running a set 3 tankers did provide a slightly higher average than return by the DFS approach, reaching 112 000 at best.

Tests with only 2 tankers were also performed, and these performed better, as expected due to a lower impact of the multi agent problem. However, using only 2 tankers is not very interesting, and not good enough of an improvement to use instead of more interesting methods.

Conclusion

Final Approach

In the end, whilst using a simple 3 tanker approach showed better overall results, they were not too much of an improvement over the DFS approach. Since DFS is much more interesting, and in concept much more efficient too, I decided to use DFS as my final approach. In the end the limits I used resulted in 2 agents being used more frequently than I would have liked, however, this was only about 30% of seeds tested, which leaves a large share of the seeds using 3 or more agents. The good side of some seeds using 2 agents, is that it shows the advantage of being able to switch between 2 and 10 agents, and still return similar scores.

Task Scheduling

Originally I had planned to use a more sophisticated task scheduling protocol. However, simple agents can perform well with simple task scheduling, and so I left that area unexplored.

Appendix: Agent types

Listed below are the Tanker agent types used for this assignment. A small part of the descriptions below are repeated from the report, for easier referencing. Other small details are included, which did not fit in other parts of the report. All Tankers use a “claim” based task management system.0

Efficient Tanker

The Efficient Tanker agent is an adaptation of the single agent from assignment 1. It uses a reactive with state “decision” methodology to choose which nearby task is best. All the decisions made are based on hard coded biases.

Scout Tanker

The Scout Tanker agent is very similar to the Efficient Tanker. It starts differently, exploring for a longer period of time than the Efficient Tanker, before reverting to the same behaviour. The explore time is limited, as after a certain point it is likely to have explored as far as possible, and would be better off completing tasks.

Certain environments would allow for limitless exploration, but even then, the other agents may have a difficult time exploiting the explored area efficiently, and the Scout Tanker needs to justify the penalty for additional agents, so it switches to completing tasks.

Lonely Tanker

The Lonely Tanker (and Lonely Scout Tanker) operate near identically to their base counterparts, with the exception of the getBestTask function. These tankers use a weighted bias – loneliness – to try to prioritise tasks that are further away from other tankers.

Interestingly, whilst the Efficient Tanker and Scout Tanker result in different scores when used exclusively, the Lonely Tanker and Lonely Scout Tanker give the same results, despite the Lonely Scout not performing tasks for the first 1500 steps. This is likely due to the loneliness bias encouraging the tankers to explore further than Efficient Tanks would.

Lonely Scout Recruit Tanker

Based upon the Lonely Scout Tanker, the Recruit Tanker adds an extra Lonely Tanker to the Fleet every time it finds a new Fuel Pump before 1500 steps, up to a maximum of 10 tankers. After 1500 steps, it resumes regular task execution.

Leash Tanker

The Leash tanker simply sticks with the same designated fuel pump. Designed with the Dynamic Fleet Scaling in mind, it performed worse than a basic Lonely Tanker and was replaced as such.