COMP2310 Assignment1

GLIDING IN SPACE

Student Name：

Student ID：

Submission Date:2020-9-28

Content

[1 Problem description 3](#_Toc52222112)

[1.1 Main idea 3](#_Toc52222113)

[1.2 message passing structure 3](#_Toc52222114)

[1.3 Recharging 4](#_Toc52222115)

[2 Stage B/C 4](#_Toc52222116)

[2.1 Choose message 4](#_Toc52222117)

[2.2 Set destination 5](#_Toc52222118)

[2.3 Experiment result 8](#_Toc52222119)

[3 Stage D 9](#_Toc52222120)

[3.1 Consensus algorithm 9](#_Toc52222121)

[3.2 Experiment result 10](#_Toc52222122)

[4 Improving directions 11](#_Toc52222123)

[5 reference 11](#_Toc52222124)

# 1 Problem description

## 1.1 Main idea

The main idea of the program is to make sure as many vehicles alive as possible. And for each vehicle, it needs 4 operations:

* Find globes around locally
* Send messages
* Receive messages
* Set Destination and Throttle.

First is Find globes around locally. We can use function Energy\_Globes\_Around which can detect the globes around this vehicle. when the vehicle finds globes, which means this message is the latest, so the message can send to others. And it makes sure this message is not out of date. But if the vehicle did not find globes, it needs to get messages from other vehicles or use the message last time received. In these messages, we can have globes movement information. According to this information we can set destination and throttle.

## 1.2 message passing structure

In the message, we need to decide what message we need to pass. Smaller message size can make message passing more efficient.

The message structure showed below as:

* comm\_time timestamp when finding the globes or send the message
* globe\_num how many globes we found
* globe\_get globes array we get
* my\_energy how much energy is left.

Each vehicle has two messages, latest\_message, and message\_get. Depending on the different circumstances they can have a different meaning, the message wants to send, the message received, or the latest message.

## 1.3 Recharging

According to the description, the vehicles will recharge when they approach the gobbles. However, as the paper mentions, vehicles may not approach the destination when multiple vehicles are going to the same destination. What we need to do is distribute vehicles to different destinations and they can get recharged in the destination.

# 2 Stage B/C

## 2.1 Choose message

From the description above, we know that vehicles may detect messages by themselves or get a message from other vehicles, which means we need to choose which message we should follow. They can be classified as 4 cases.

* Case 1: have detected globes by themselves and did not receive messages from others.
* Case 2: have detected globes by themselves and received messages from others.
* Case 3: have not detected globes by themselves and received messages from others.
* Case 4: have not detected globes by themselves and did not receive messages from others.

Case 1

Case 2

For case 1 and case 2, the black vehicle detected the globe by himself which means this message is the latest and this globe is closest to the vehicle (vehicle can detect this globe). So it is unnecessary to hold other’s message, what we need to do is set destination with this detected message and send to others.

Case 3

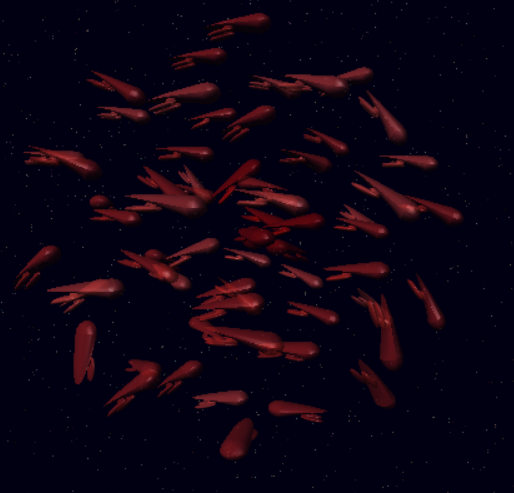
Case 4

In case 3, the black vehicle did not detect globes, but it receives messages from other vehicles, in this case, we need to compare this timestamp with the last message and restore the latest one. After storage, send the latest message to others. Also set destination with this latest message. In case 4, the black vehicle neither detects the globes nor received a message, which means there are no other vehicles nor globes around him, it has to rely on itself’s latest message. and set the destination with 0.5 throttles. because there is no vehicle around him, It is unnecessary to send a message to others.

In general, with every message update, the vehicle needs to send it to others, and whether they received it depends on themselves. This mechanism making sure every vehicle’s message is the latest and message passing efficiently.

## 2.2 Set destination

When setting a destination, we need to consider the globe’s location, vehicle’s movement, current charge. When this part is designed not good enough, it is easy to cause “Herd Effect”, which vehicles are stuck together and no one can reach the destination.

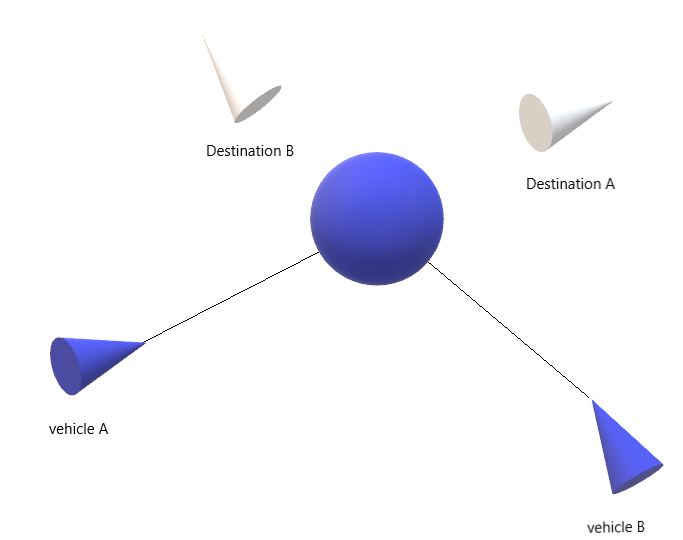


Herd Effect

In stage c, we need to find which globe is the closest to the vehicle. It is easy to accomplish with a loop. And distance can calculate with

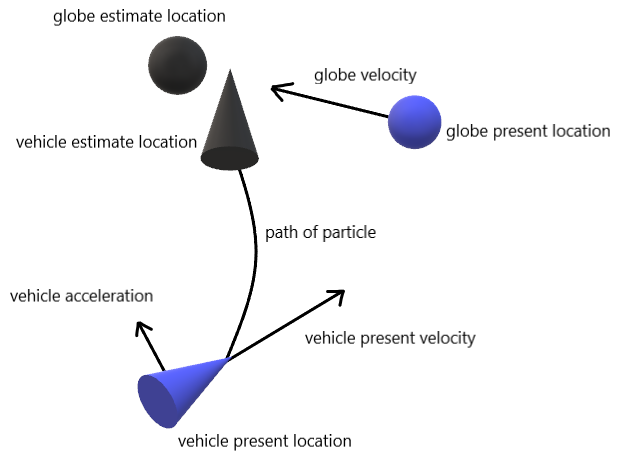
.

After finding the closest globe, there may be many vehicles that need to reach that globe, which would cause the “Herd Effect”. Here we make 2 strategies. First, set each vehicle to a different destination around the globe. Here we have the globe’s location and vehicle’ s location . So we set the destination to .



Set destination

When adding a unit vector with the vehicle and globe direction, can cast vehicles to different destination. After times of reliable experiments, we found 0.025 is more stable. Second, before setting the destination, the vehicle needs to ensure it is on a low charge level, and the current charge is lower than other vehicles who are waiting to recharge, which is realized by comparing the current charge and my\_charge in the message structure. To make the vehicle realize it is in a low charge level. At first, I try to use the globe’s present location, velocity, vehicle’s location, and velocity to estimate the vehicle’s charge level. but I did not find a way to solve the Quadratic equation with Ada. So I set the charge threshold to 0.4. this has a good performance in most conditions.



Estimation model

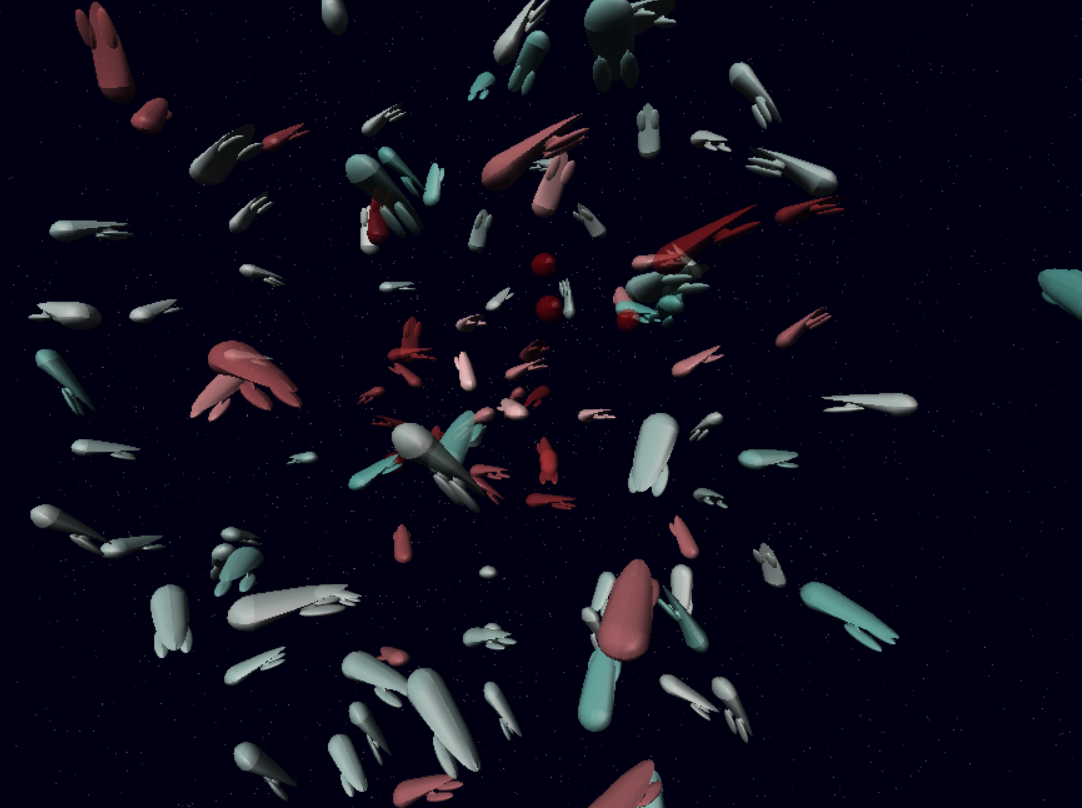
Also setting a large throttle is helpful to relieve the “herd effect”. Because when there is a large throttle, generally vehicles have a higher velocity.

However, when vehicles are in a large number, still exist “Herd Effect”. In this case, the vehicle’s who received messages and current charge greater than 0.3 will move to zero\_vector\_3D with 0.5 throttles.

When setting throttle, I have tried to change throttle with the change of the distance to the destination. Throttle setting as below.

When the vehicle is far from the destination, we set a larger throttle. However, it just performed well in some particular situations.

In stage c, there is a trick. When a globe vanishes may result in a vehicle's group stuck. When the globe vanished there is a group of vehicles rushes to this globe, they can not recharge and passing messages in their small group. There are 2 ways to break this dilemma. First, another vehicle passed by and transfer the message. second is set destination to zero vector.



Group stuck ( with the blue circled )

Before setting the destination to zero vector, it needs to analyze whether vehicles are stuck. There are 2 ways. The first one is to analyze the vehicle’s distance to the destination. The second is to analyze the vehicle’s velocity. When vehicles are stuck together, they can never reach the destination. So when the vehicle is close to the destination or velocity is close to zero. At the same time, there are no globes around, this vehicle is stuck.

## 2.3 Experiment result

Stage B:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Init vehicle number | duration | Test time | Average result | Survival rate |
| 16 | 2 min | 20 | 15.7 | 98.1 |
| 32 | 2 min | 20 | 31.6 | 98.7% |
| 64 | 2 min | 20 | 63.5 | 99.2% |
| 100 | 2 min | 20 | 98.5 | 98.5% |

During the experiment, 64 vehicles seem very stable for one globe in the space. However, more vehicles may cause a “Herd Effect”, and fewer vehicles mean they lost the globe. In this case, 100 vehicles test how is the model solves “Herd Effect”. And 16/32 vehicles test how is the message passing in the model if message passing is inefficient, the vehicles may die soon.

As the chart above, this model performs well in stage B, survival rates are above 98%.

Stage C:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Init vehicle number | duration | Test time | Average result | Survival rate |
| 32 | 2 min | 10 | 22.4 | 70% |
| 64 | 2 min | 10 | 56.5 | 88.2% |
| 128 | 2 min | 10 | 108.5 | 84.76% |

For multiple globes, 64 vehicles performed best. There enough vehicles passing messages, and it has a suitable structure. When there are fewer vehicles, The vehicle's distribution is too dispersed. It is hard to find another globe when the previous globe vanished. So it is easy for a small number of vehicles to fall into group stuck. When there are more vehicles, the structure is loosely organized, when the globe vanish the vehicles around him may not have enough energy to get to other globes.

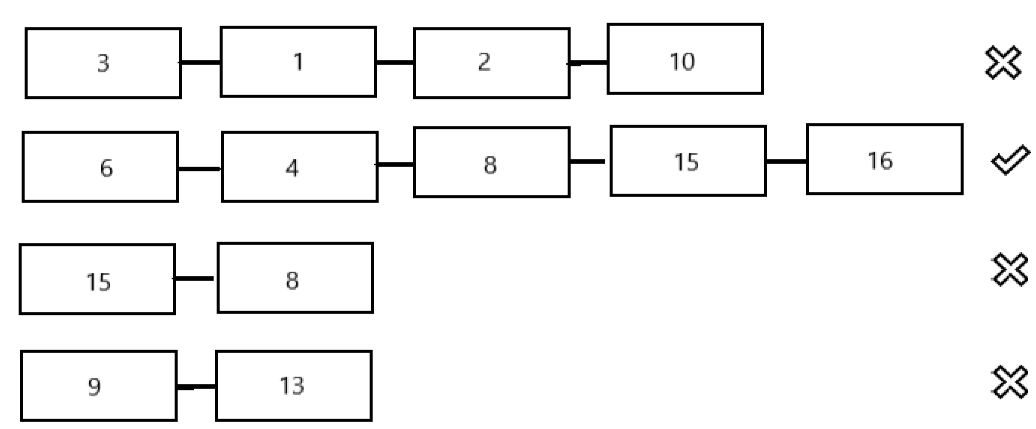
# 3 Stage D

## 3.1 Consensus algorithm

For stage D, what we need to do is let vehicles communicate who is going to destroy himself and who will survive. Based on the message structure above, we add

* surviver\_num data structural: positive
* surviver data structural: an array of positive
* update\_time data structural: Real\_time.time

To make this work, every vehicle should guarantee the same array of the survivor. Each vehicle has rights to add himself to this array. Here we use the idea of blockchain, which is the consensus algorithm. At first, each vehicle holds an array with his number in the first place. When communication, the vehicle always chooses the longer array and when arrays are the same length, always choose the one update earlier. Who can finish the array earlier who will decide which vehicles survived.



Finally, every vehicle holds the same array. When the vehicle-No is not in the array, it means this vehicle is not a survivor. It will spend the energy until vanishing.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Init vehicle number | duration | Test time | Target number | Average result | Survival rate |
| 32 | 4 min | 10 | 26 | 20.6 | 79.2% |
| 64 | 4 min | 10 | 42 | 37.5 | 89.2% |
| 128 | 4 min | 10 | 100 | 87.4 | 87.4% |

## 3.2 Experiment result

Multiple globes:

Like the chart above, this model has the best performance when the initiation number is 64. Comparing to the single globe, when the globe vanishes, some vehicles who should survive die. Also, in some particular situations, vehicles should die may trap the vehicle should survive. When the vehicle number is small, it is hard to keep the vehicle number in a stable situation.

Single globe

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Init vehicle number | duration | Test time | Target number | Average result | Survival rate |
| 32 | 4 min | 10 | 26 | 25.6 | 98.4% |
| 64 | 4 min | 10 | 42 | 39.2 | 93.3% |
| 128 | 4 min | 10 | 100 | 93.9 | 93.9% |

The best performance in the single globe is 32 vehicles. Because the small size group can pass messages more efficiently, which will reduce the useful vehicle’s death rate.

In summary, when vehicles in the survive array die, this model can not add another vehicle to the array. And there may have a way to reduce the trap as above mentioned.

# 4 Summary

This model performed well in the experimental conditions. The model can solve “Herd Effect” and the group’s stuck as the above mentions. At the same time, some reusable functions are designed as functions. It improves maintainability. And the model is designed dependable. However, there is no denying that there is room for improvement and some defects in this model.

# 5 Improving directions

* Dynamically programmed the surviving array. When some vehicles die accidentally another vehicle can be added into the array.
* Estimate the destination more precisely, using the model in 2.2 may have an improvement.
* Estimate when to recharge more precisely, as the introduction above, using a constant threshold is not a good idea.
* Setting a dynamic throttle is also important, just like I tried before. But I believe there are more efficient ways to set throttle.
* Solving “Herd Effect” more efficiently. When the vehicle number is greater than 150, the present method shows a bad performance. So there is room for improvement.

# 6 reference

* An introduction to Consensus Algorithm

<https://www.bilibili.com/video/av78588312?from=search&seid=1453987515383608625>

* Consensus Algorithm <https://www.cnblogs.com/linguoguo/p/10178196.html>
* Ada <https://www.ada.gov/>