Lehigh University
Lehigh Preserve

Theses and Dissertations

2015

Using Planning Landmarks to Control Camera Movement in DOTA 2 Games

Jundong Yao
Lehigh University

Follow this and additional works at: http://preserve.lehigh.edu/etd

Part of the Computer Sciences Commons

Recommended Citation
Yao, Jundong, "Using Planning Landmarks to Control Camera Movement in DOTA 2 Games" (2015). *Theses and Dissertations*. 2895.
http://preserve.lehigh.edu/etd/2895

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

المنارة للاستشارات
www.manaraa.com

Using Planning Landmarks to Control Camera Movement in DOTA 2 Games

By

Jundong Yao

A Thesis

Presented to the Graduate and Research Committee

Of Lehigh University

In Candidacy for the Degree of

Master of Science

In

Computer Science

Lehigh University

May 2015

www.manaraa.com

©2015
Jundong Yao
All Rights Reserved

للاستشارات المنارة

www.manaraa.com

SIGNATURE SHEET

The thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science

____

Date

____

Thesis Advisor

____

Chairperson of Department

المنارة للاستشارات

www.manaraa.com

Table of Contents

Abstract 3

1. Introduction 4

2: Background 10

2.1 Planning 10

2.2: Planning Landmarks 12

2.3: The Dota 2 Game 14

3. Computing Planning Landmarks 18

3.1 Algorithm Description 18

3.1.1 Weight Calculation Algorithm 18

3.1.2 Similarity Algorithm 21

3.2 Example in Transportation Domain 22

3.2.1 Weight Algorithm in Transportation Domain 26

3.2.2 Similarity Algorithm in Transportation Domain 28

3.3 Example in Dota 2 29

3.3.1 Weight Algorithm in the Dota 2 Domain 30

3.3.2 Similarity Algorithm in the Dota 2 Domain 31

4. Implementation 33

4.1: New Transportation Implementation 33

4.1.1 Parsing Traces 35

4.1.2 Similarity Algorithm Implementation 38

4.1.3 Weight Algorithm Implementation 40

4.2: Dota 2 Implementation 41

4.2.1 Preparing the Input Traces 43

4.2.2 Finding Landmarks Using Weight and Similarity Algorithm 45

4.2.3 Managing the Camera in the Dota 2 Game 47

5: Experiment 49

5.1: Experimental Setup 49

5.1.1 Transportation Domain 50

5.1.2 Dota 2 Domain 50

5.1.3 Amazon Mechanical Turk 52

المنارة للاستشارات
www.manaraa.com

5.2: Experimental Results 54
5.2.1 Transportation Domain Results 55
5.2.2 Dota 2 Domain Results 61
5.2.3 Amazon Turk Results 65
5.3: Discussion 69
6. Final Remarks 72
6.1: Conclusions 72
6.2: Future Work 74
Bibliography 75
Vita 76

المنارة للاستشارات
www.manaraa.com

Abstract

This thesis introduces a new method for automatically finding important events from video game replays. We use these events to control the movement of the camera in the Dota 2 game. This method is based on the idea of finding landmarks, which are events that always takes place in a game. Our method automatically highlights important events in the Dota 2 game. The method combines two algorithms: one assigning weights to events and another one computing similarities between events. We discuss the motivation and implementation of these two algorithms and we test them in a transportation domain and in the Dota 2 game. Several experiments are performed in both domains and the results show that the method extracts landmarks effectively. Based on the experiments on Amazon mechanical turk, camera control based on landmarks shows potential benefits.

المنارة للاستشارات

www.manaraa.com

1. Introduction

In this thesis we investigate the question of automatically discovering important events from videogames feeds. Learning to find important events in videogames is useful in understanding the game process and help towards attaining automatic generation of game narratives. It can also be used in other applications such as video surveillance. We use the Dota 2 game in our study. Dota 2, which is short for Defense of the Ancients 2, is a very popular online multiplayer video in which two teams of players are tasked with protecting their own ancient, a unique building in the game, while attempting to destroy the opponent's ancient. Each player controls one character in the game and obtains experience and gold from the death of characters or non-player controlled units called creeps in the opponent's team. The objective of the game is to destroy the opponent's ancient.

Important events are difficult to distinguish from non-important ones because there is no fixed scenario in Dota 2. Important events will show in every replay of the Dota 2 game but they are not subject to time, location or the strength of each team. Defining if an event is important or not is also a central concern in the rapidly changing runtime situation. But it is very hard to identify.

المنارة للاستشارات
www.manaraa.com

Figure 1.1 A Dota 2 screen shot of a team fight happens near one team's ancients

Figure 1.1 shows an informative to the game because it happens when one team is attacking their enemies' base (i.e., where the ancient is located), which is a game-winning situation. This is an important event in the game and it can be identified as a milestone or landmark.

Writing a computer program to detect these important events in Dota 2 game is challenging, but it can be very important because the learning results can help people understand this game better, or predict the winner in a particular round or contribute towards ongoing efforts on automated narrative generation of videogame competitions.

Our hypothesis is that important events are events that will happen every time in the game. No matter when or where this event happens, they are milestones in the game. The team fight event is an eligible event that will happen in all games. Fights can show the current situation of both teams and also provide information about the whole game. The event "the ancient is destroyed" is the most important event in the game every game

المنارة للاستشارات
www.manaraa.com

will go through this event. The four pictures (Figures 1.2-1.5) below show events in Dota 2. Three of them are important events, one of them is not important event.

Figure 1.2: An important event happens near a location called radiant's height. In this event, there is a team fight in the area where the game camera is now focusing on.

Figure 1.3: An important event happened near a location called radiant's middle lane, close to the base. This event is important because if the attacking team wins during this team fight, they continue to destroy the inner tower and win the game.

المنارة للاستشارات
www.manaraa.com

Figure 1.4: An important event happened near the end of the game. This event is important because an important structure the dire’s ancient is being attacked. The radiant will win the game after this ancient is destroyed.

Figure 1.5: a screenshot of a normal scenario in dota 2. The viewer cannot tell which team will win or what the trend in this game is. This is an unimportant event

To automatically identify important events in Dota 2, we use the notion of landmarks to find important events in the game. A landmark is a particular situation or state that must occur when some final goals are achieved. We make two observations about landmarks. Firstly, a landmark is a state or a sequence of consecutive states that

المنارة للاستشارات
www.manaraa.com

always occur when achieving the final goals, while starting from a particular state. Secondly, a landmark is a special situation, which given multiple possible game trajectories to achieve the final state, it will always appear in each of those trajectories. No matter how the game trajectories vary, the landmark will always be visited every time.

We proposed an algorithm that can automatically find the landmarks in the Dota 2 game. We are given multiple game replays, which are recognized as traces of the game. Each replay consists of a series of consecutive states or trajectories, from the beginning of the game until one ancient is destroyed. The state is defined as a captured moment, or a screen shot with all heroes and units' properties in a particular time. We used similarity metrics to discover the similarities between two separate traces and then we calculated the distance start from one state in the first trace to all the other states in other traces. Distance can be shortened when considering points in different traces that are near to one another. Then we calculate the weighted value of each states in this trace, the highest landmark will appear when a particular state has the lowest similarity value. The lower the similarity value one state has, the higher landmark value it possesses. Also, an ordinary state has the highest value and lowest landmark.

There is another algorithm proposed by Julie Porteous, Laura Sebastia, and Jorg Hoffman, 2014, will be able to find landmarks in their domain (e.g. the blocks world domain). Also Amy Mcgovern and Andrew G. Barto, 2001 they used an algorithm to find landmarks by calculating the trajectory using reinforcement learning. Neither of these two algorithms can solve the problem of finding landmarks in the Dota 2 game because they require an over-simplified way to define the actions in the game (i.e., as

المنارة للاستشارات
www.manaraa.com

preconditions and effects). In a game like Dota 2 player's actions have a complex definition and it will be either not possible to use the preconditions and effects to define the actions or even if it is possible it would be too cumbersome.

The algorithm we propose is a good match to solve the problem we have in the Dota 2 game. In the Dota 2 game, each state of the game cannot be predictable based on the previous one (this is called non-determinism; after taking an action in a state, there are multiple possible states that might occur next). Our algorithm finding the landmarks, no matter how complex the input trace is, will always generate results.

المنارة للاستشارات
www.manaraa.com

2: Background

2.1 Planning

A plan is a sequence of actions that starting from a particular state, and ends at a prescribed goal state. The plan may consists of several actions may to realize the setting goal. Planning is a branch of artificial intelligence that is concerned with how to generate a plan. Plans are to be generated by automated robots or artificial intelligent agents, rather than using human's conventional way to analyze problems and optimize solutions. Given the planning operators, the planning process starts from the initial state, applying an action each time that transforms the state until finally reaching the final state. Multiple sequences of the actions (i.e., plans) may be transform same start state until the end state, analogous to starting from an initial point, and converge to a same final point by using multiple trails. Planning can tackle more logical complicated and more time consuming domains, and give possible solutions by using a variety of planning algorithms. With the help of computers, planning can be much more efficient and straightforward to generate solutions than manually generating the plans.

A state is a collection of primitive atoms representing the conditions that are true in the world. A state can reflect the scenario of a particular situation, like a screenshot in the game. A state is a description of the objects in the domain. For example, a package is in location A, a player in the game possesses the experience of 100 at time 12 o'clock. So the planning trace consists of a sequence of many states. In order to achieve the goal, the planning must system must search a path between the initial state and the final state. We denote a state as S, and the initial state is s_{init}, final state is s_{final}. The planning is using the actions to realize the ordered set of the states$(s_{init}, s_1, s_2, s_3, \cdots, s_{final})$.

المنارة للاستشارات
www.manaraa.com

Operator is a state converter that can change one state into another state. The operator consists of four major elements: operator head, precondition, delete list and add list. An operator will match with the state that is same with its precondition, delete the primitive atoms in its precondition, and add new atoms to generate a new state. Based on this procedure, the newly generated state will be applied into another operator to change to a third state, so on and so forth. When an operator is applied in an state, it is called an action. For example, $\{s_{init}|a_1, a_2, a_3\}$ is a state that has 3 primitive atoms. For the operator $o = (h, pre, del, add)$, in which $pre = s_{init}$, $del = \{d_1|a_3\}$, $add = \{add_1|a_4, a_5\}$, the newly created state $s' = \{s_1|a_1, a_4, a_5\}$. So in general, when the operator is modifying one state to another, it can be written as $s' = Modify(s, a) = (s\backslash del) \cup add$. Also, as the goal and the initial state is given, we are able to have the relation between states and operators, which is $s_{final} = Modify\ (Modify(Modify(\cdots Modify(s_{init}, o_1) \cdots), o_{n-1}), o_n)$, where o_n is the last operator convert the second last state into the final goal state.

The example of transportation domain is a good way to illustrate the planning problem we are discussing. In transportation domain, a package is in location A, and it needs to be delivered to location B in the same city. We have a truck in location A that can carry the package to travel within this city, so the truck is able to get this package delivered by the action called drive truck. In order to make the package onto the truck, we have an action called load, and on the other hand, we also have an action called unload to release the package from the truck. This is quite a straightforward and easy to understand domain that can even be figured out by hand. But it includes all the elements we mentioned above. In this domain, we have all the states listed in Table 2.1:

المنارة للاستشارات
www.manaraa.com

Initial state	Interim state 1	Interim state 2	Goal (Final state)
Package in Loc-A	Package on Truck	Package on Truck	Package in Loc-B
Truck in Loc-A	Truck in Loc-A	Truck in Loc-B	Truck in Loc-B

Table 2.1: A trace of states from the initial state to the final state

Also we have a list of operators in this domain: Load package, Unload package, and Drive truck. For the first operator convert initial state into interim state 1, $precondition = \{s_{init}|package\ in\ locA \cup truck\ in\ locA\}$, $del = \{del_1|package\ in\ locA\}$, $add = \{add_1|package\ on\ truck\}$. After doing the modifying job by using this operator (Load package from location 1 to the truck), it satisfy the effect that is showing in interim state 1.

2.2: Planning Landmarks

There are several methods for finding landmarks in a particular domain. Julie Porteous, Laura Sebastia, and Jorg Hoffmann, 2014 proposed a way to extract landmarks in the blocks world domain. The candidate landmarks are extracted by their so called relaxed planning graph (RPG). In this building process, they ignore all the delete item lists to relax the planning task. The GRAPHPLAN's planning graph will then been constructed and chain forward from the initial state to a graph level where all the goals are achieved. The GRAPHPLAN-style planner IPP is a kind of planner that they used in the paper. After the relaxed planning graph is built, they went back through the the RPG and then extract the landmark-generation tree.

المنارة للاستشارات
www.manaraa.com

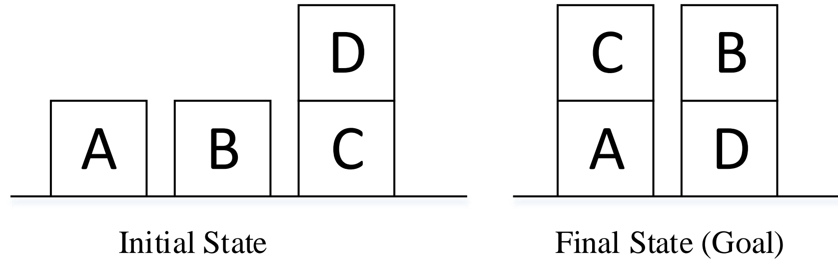

Figure 2.1: An example of a task in blocks world domain. (Julie Porteous, Laura Sebastia, and Jorg Hoffmann, 2014)

For the process of extracting the potential landmarks, all the top level goals are added into the landmark generation tree, and then these goals are solved by relaxed planning graph based on their previous level. For each goal in a level, all the actions to achieve the goal is grouped, and the intersection of all the pre-conditions are calculated. Then moving down to a lower level to do the above procedure again. Table 2.2 shows a relaxed planning graph in blocks world domain, to achieve the goal: C is on A, and B is on D, from the initial settings: A, B, and C are on table, D is stacking on C.

L0	A1	L1	A2	L2	A3	L3
on-table A on-table B on-table C on D C clear A clear B clear C arm-empty	pick-up A pick-up B unstack D C	holding A holding B holding D clear C	stack B A stack B D stack B C put-down B … pick-up C …	on B A on B D on B C … holding C …	stack C A stack C B stack C D	on C A on C B on C D

Table 2.2: A relaxed planning graph for the blocks world domain showing in Figure 2.1. (Julie Porteous, Laura Sebastia, and Jorg Hoffmann, 2014)

The goals are on(C A) and on(B D), and the corresponding landmark generation tree is $N = \{on(C\ A), on(B\ D)\}$, the goal on(C A) is in level 3 and goal on(B D) is in level 2. The goal on(C A) can only be achieved by the operator: stack C on A. So C is

المنارة للاستشارات
www.manaraa.com

holding on hand and A's surface is clear are its pre-conditions. Then the atom C is holding will be put down to level 2. Then the procedure proceeds until it reaches to the level 0. By applying a verifying and ordering procedure, landmarks are extracted. For example, the state: C's surface is clear is a landmark that no matter how the procedure varies, this state must show up.

For the work from Qiang Yang, Kangheng Wu, and Yunfei Jiang, 2005, they make assumptions that the states are unknown and their goal is to reconstruct the literals in transportation domain. Based on their research, the planning structure with minimal logical action model will be created. Their algorithm can be summarized in three steps: (1) initialize plans and variables, (2) build action and plan constraints, (3) build a weighted MAXSAT, which is a fast algorithm for discovering association rules.

2.3: The Dota 2 Game

Dota 2 is an online multiplayer video game whose ideas and setting are mostly inherited from DotA, a Warcraft III: Frozen Throne based battle arena map. It is developed by Valve Corporation and its chief game designer and developer is IceFrog, who once served as a major role in redesigning and upgrading the DotA game. This game can be ran on Microsoft Windows, OS X and Linux platforms, and STEAM is its game community which is also developed by Valve Corporation.

The origin of Dota 2 is DotA, which is short for "Defence of the Ancients". It ran on the game platform "Warcraft III: Reign of Chaos" and customize some of its specific configurations. DotA is firstly created by "Eul". The DotA game comes to be well known after Blizzard published a subsequent version "The Frozen Throne". The developer

المنارة للاستشارات
www.manaraa.com

"Guinsoo" developed the "all-star" map, and another developer "IceFrog" started adjusting and optimizing the DotA after 2005. DotA it is very popular among young people. It is one of the most famous battle games in the world level electronic video game competition. As its 3D engine was out of date and lost its customers, its subsequent follower Dota 2, was developed by Valve Corporation. "IceFrog" is now an employee in Valve.

The huge square shape map is the unique map used in Dota 2 game. Its terrain contains slopes, heights, jungles, and also rivers. It is majorly divided by diagonal northwest to southeast river with two competing teams known as Radiant and Dire team on each side. The Radiant team is located at the southwest corner of the map, and the Dire's team is located on the other side. Each team has their unique ancient as well as some surrounded buildings and towers, which protect the ancient from attacks by the other opponent team. The densest building area is located in both the southwest corner and northeast corner. They are placed on the heights of each side so they are made more difficult to attack if someone wants to occupy the heights. There are three lanes omitting from the height, two lanes are placing alongside the map, which are horizontal and vertical respectively, and also one center lane connecting two bases. Figure 2.1 shows a screen shot of the Dota 2 game map with three lanes and two bases on each corner of the map. On each side of each lane, there will have two defensive towers protecting its lane by attacking the heroes and creeps that do not belong to their team. The inner tower or ancients are attackable only if the outer tower is destroyed. When the inner tower is destroyed the game ends.

المنارة للاستشارات
www.manaraa.com

Figure 2.1: The radiant's base is located at the bottom left corner of the map. The dire's base is located at the top right corner of the map. There are three lanes: top, middle and bottom connecting two bases.

The Dota 2 game allows at most 10 players play in the game. These 10 players are divided into two teams fit into the Radiant and Dire's team respectively. Each controls the same hero in the game until the game ends. Their birth place when the game start is the same ax their respawn location (when the hero dies it reappears in this location): behind their ancients and on another heights.

Each hero is born with zero experience and limited gold, they can earn the gold by killing enemies' creeps and enemy heroes. Also there are some neutral creeps in the jungle that can also provide gold and experience when killed. The heroes can also get extra gold when enemies' defensive tower is fallen or other buildings are destroyed. Meanwhile, heroes can use the gold they earned to purchase items in the store, to improve their damage value or do positive effect to win the game. The better weapon or

المنارة للاستشارات
www.manaraa.com

equipment they have, the more possibility they may have to win the game. Experience can be used to acquire more powerful skills such as the hero is able to learn new skills and upgrade their equipped skills when they level up. Heroes can also learn ultimate skill, which only be available every six levels and will do great damage or assistant comparing to normal skills.

The game ends when either of the ancients is fallen. One side will try to approach to the heights that belongs to the enemies, while on the other hand, the enemies will try their best protecting their ancients. Team fights happen at this time when both teams' heroes gathered together and use their physical attack or magical ability to put damage on their enemies. So the team fight will always happen in the game, and many team fights are the turning points in the game that can greatly affect the result of the game.

المنارة للاستشارات
www.manaraa.com

3. Computing Planning Landmarks

3.1 Algorithm Description

In this thesis, we proposed a new general algorithm to find the landmarks. As we discussed in chapter 2.2, the other two methods proposed from other papers cannot solve the problem we have in Dota 2 because either it is too difficult or even not possible to define operators. In contrast, our algorithm can efficiently find landmarks in Dota 2 without requiring that operators are given.

Our algorithm requires that the states should all start from a state and end in one state, which means even though we may have many input traces, they should be starting from same origin point and same final point. No matter how we take the trace to achieve the goal, and no matter how the traces are twisted or intersected, there should definitely have these two landmarks. One is the initial state and the other is the end state.

The algorithm is consist of two parts, calculating the weighted value and calculating the similarity between two states from two traces.

3.1.1 Weight Calculation Algorithm

We used an algorithm to calculate the weighted value across all the states among all the traces we are given as input. The main idea in this algorithm is trying to find a difference in value that reflects the relative importance of particular states. The weighted value of potential landmarks should be different from the ordinary states in our traces. This algorithm maintains current state $s_{current} \in Trace_1$, the runner state $s_{runner} \in Trace_1$, and the seeker state $s_{seeker} \in Trace_i$, where $Trace_1$ is the base trace and $Trace_i$ is another trace that we want to visit (i≠1). The intersection point $s_{intersect} =$

المنارة للاستشارات
www.manaraa.com

$\{s|s \in Trace_1 \cap s \in Trace_i\}$ are the states that has the same set of primitive atoms between two traces. And also s_{seeker} and s_{runner} are the instance of the intersection states set. We call the connection between two intersection states the tunnel $T_{s_{seeker},\ s_{runner}}$. It links the two traces up and the neighboring states must use this tunnel to visit another trace. The method to choose the runner state, the seeker state, and the tunnel between these two states will be discussed in Section 3.1.2 (the similarity algorithm).

The weight of a state will be computed starting from this state, and finding a path through the other traces by using the tunnels bridging the traces. The distance between two states $s_{1i} \in Trace_1$ and $s_{2j} \in Trace_2$ is high if they are not the states next to the tunnel. On the other hand, the distance will be small if two states are close to the tunnel. The algorithm will start from the current state, and move the state runner forward. Meanwhile, the state runner will find the shortest way to get access to other traces by trying to match the state seeker in other traces. The state seeker will also run through the trace where it belongs to, until it reaches the final state in that trace. If the state seeker can be matched to the state runner, it means the tunnel can be created between these two traces. The distance between these two states (s_{seeker} and s_{runner}) is set as 1. On the other hand, if the state seeker reaches the final state of that trace, this means the state runner cannot find a similar state in the other trace, and the distance between these two states are set as infinity. The algorithm may not choose this seeker and runner pair as the tunnel because the cost to visit the other trace is really huge. If the tunnel is found in which the distance of this tunnel is a small, the algorithm will choose it, and the neighboring states of the current state will also get access to the other trace by using this

المنارة للاستشارات
www.manaraa.com

tunnel. If the current state is located at the place where tunnels appear on both sides near it, the algorithm will choose a shortest one that can decrease the distance.

$$Distance_{s_{runner},s_{seeker}} = Similarity_{s_{runner},s_{seeker}} = \begin{cases} 0 & (tunnel\ exists) \\ 1 & (tunnel\ exists) \\ \infty & (tunnel\ doesn'texist) \end{cases}$$

For the distance from one state to the other state in another trace in general,

$$Distance_{s_{ip},s_{jq}} = Distance_{s_{ip},s_{irunner}} + Distance_{s_{irunner},s_{seeker}} + Distance_{s_{jseeker},s_{jq}}$$

Where $s_{ip}, s_{irunner} = \{s|s \in Trace_i, 1 < p, runner < n\}$, n is the number of states in $Trace_i$. $s_{jq}, s_{jseeker} = \{s|s \in Trace_j, 1 < q, seeker < m\}$, m is the number of states in $Trace_j$.

The distance will be gathered and added every time from the current state to all the other states on all the other traces. The total weight of the current state is calculated by dividing the sum of distances between the states it visited.

$$Weighted\ Value_{s_{ip}} = \frac{\sum_{j=2}^{k} \sum_{q=1}^{m} Distance_{s_{ip},s_{jq}}}{m \cdot (k-1)}$$

Where k is the number of traces we have in our input set, m is the number of states in $Trace_j$.

By using this algorithm, the ordinary states will have larger value on the final calculated weighted value (and hence, will not be considered landmarks). The reason is quite straightforward. The ordinary states are more distant to the state runner, which is added to the distance to other trace. Meanwhile, the special state, which sits exactly on

المنارة للاستشارات
www.manaraa.com

one side of the tunnel, the state runner itself, will have the lowest distance. The difference of the distance value is a good measurement of the importance of the landmarks we want to analyze; the lower the distance or weight, the more likely the state is a landmark.

3.1.2 Similarity Algorithm

The similarity algorithm is a method we proposed to determine whether two states from two different traces are close to one another. This algorithm can solve the problem in HTN planning that given several input traces, the method that simply counting the occurrence of states will leads to a failure. Inspired by the work of Amy McGovern and Andrew G. Barto, the landmark may also exist in a high dense region, that many trajectories staying really close to each other within an area, but not intersect with anyone else. In this case, no landmark will be extracted by using counting occurrence of the states since there is actually no intersections in the domain. The phenomenon that we can discover in this case is that the dense region will still be recognized as a landmark by the similarities in this region. If we have such an algorithm that can filter out this similarity, we are then safe to say some states which are staying close enough but not intersected will also be considered as a kind of landmark.

The actual similarity algorithm is slightly different between domains (i.e., it is domain dependent), but the key idea in similarity is comparing the degree of how two states with their primitive atoms are shared. For example, for $s_i = \{s | s \in Trace_p\}$ and $s_j = \{s | s \in Trace_q\}$, each of them has a list of primitive atoms that may have slightly different with each other.

المنارة للاستشارات
www.manaraa.com

s_i	s_j
$Atom_1$	$Atom_1$
$Atom_2$	$Atom_3$
$Atom_3$	$Atom_4$
$Atom_4$	$Atom_5$

Table 3.1: List of the primitive atoms in two states.

So the similarity may consider two states similar when few atoms are different. The more differences these two states may have, the higher similarity value we may have. This is related to the distance calculation in Chapter 3.1.1 in which the similarity value is used as the distance between the state seeker and state runner. Also the function of calculating the similarity value is not linear, which means we are using different stages to determine the similarity. If two states are sharing exactly the same atoms and we cannot tell any difference besides which trace it belongs to, the distance is 0. And also if there is only minor difference between two states, we set the similarity to 1 or 2 to differentiate them from equality case, but still having very close distance with each other. Finally if the two states have a lot more differences, we similarity is larger. We assign infinity between two states to if they share no atoms in common.

$$Similarity_{s_i,s_j} = \begin{cases} 0 & (exactly\ same) \\ 1 & (1\ difference) \\ 2 & (2\ difference) \\ \ldots & \\ \infty & (no\ commonality) \end{cases}$$

3.2 Example in Transportation Domain

The transportation domain is a good example in HTN planning. It is focusing on transporting a package from one location to another. Major objects concerned is the

المنارة للاستشارات
www.manaraa.com

package, truck, airplane, so on and so forth. The package carrier transports the package between locations, but are limited to the transport area so another transportation methods should be continued. It is a classical and typical model in planning. It has the primitive atoms like “package is in location A”, it has the states that can be converted by operators or actions, it has a start state and a final goal, and it has multiple choices that can achieve the goal.

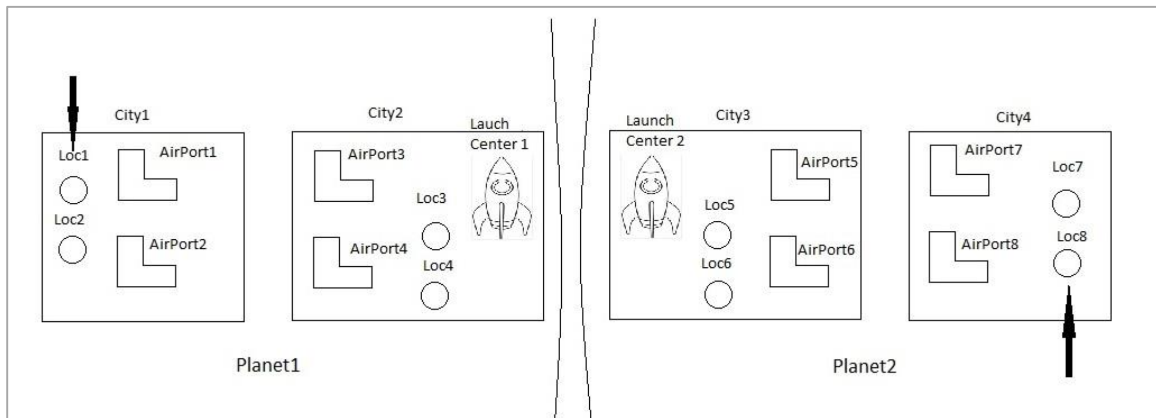

Figure 3.1: A description of new transportation domain.

We now introduce a more complex version of the transportation domain that introduces more elements than the example we mention in Chapter 2.1. In this new transportation domain, packages still need to be delivered to another place, but this time these two locations are not necessarily in the same city, not even not in the same planet. For example, consider the following scenario: the package is initially locate in location 1, city 1, planet 1, while the destination location 8 is in city 4, planet 2. In our scenarios, we have two planets, four cities, eight airports, two launch centers, and eight ordinary locations. Each city has two airports and two ordinary locations. For city 2 and city 3, each of them has a launching center that has a rocket waiting on it.

www.manaraa.com

In order to express the initial state formally, we write the primitive atom in the format like this: $Atom_i = (Adverbial_i \ \ Object_i \ \ Object_{container})$. The atom format is used to indicate that the $Object_i$ is on $Object_{container}$ in the name of $Adverbial_i$. And then we can have a detailed description of the initial state we have.

City	(ON-PLANET CITY1 PLANET1)	(ON-PLANET CITY2 PLANET1)	(ON-PLANET CITY3 PLANET2)	(ON-PLANET CITY4 PLANET2)
Airport	(IN-CITY AP1 CITY1)	(IN-CITY AP2 CITY1)	(IN-CITY AP3 CITY2)	(IN-CITY AP4 CITY2)
	(IN-CITY AP5 CITY3)	(IN-CITY AP6 CITY3)	(IN-CITY AP7 CITY4)	(IN-CITY AP8 CITY4)
Location	(IN-CITY LOC1 CITY1)	(IN-CITY LOC2 CITY1)	(IN-CITY LOC3 CITY2)	(IN-CITY LOC4 CITY2)
	(IN-CITY LOC5 CITY3)	(IN-CITY LOC6 CITY3)	(IN-CITY LOC7 CITY4)	(IN-CITY LOC8 CITY4)
Launch Center	(IN-CITY LC1 CITY2)	(IN-CITY LC2 CITY3)		
Airplane	(AIRPLANE-AT PLANE1 AP1)	(AIRPLANE-AT PLANE2 AP5)		
Truck	(TRUCK-AT T1 LOC1)	(TRUCK-AT T2 AP3)	(TRUCK-AT T3 LC2)	(TRUCK-AT T4 AP7)
Rocket	(ROCKET-AT ROCKET1 LC1)			
Package	(OBJ-AT PACK1 LOC1)			

Table 3.2: A detailed list of atoms in the initial state

Based on the result run by JSHOP, a hierarchical task network (HTN) based planning software, a way to achieve the goal “send the package to location LOC8” can have one of the possible solution in the form below:

Plan # 1

(!load-truck pack1 t1 loc1) 1.0 (!drive-truck t1 loc1 ap1) 1.0 (!unload-truck pack1 t1 ap1) 1.0 (!load-airplane pack1 plane1 ap1) 1.0 (!fly-airplane plane1 ap1 ap3) 1.0 (!unload-airplane pack1 plane1 ap3) 1.0 (!load-truck pack1 t2 ap3) 1.0 (!drive-truck t2 ap2 lc1) 1.0 (!unload-truck pack1 t2 lc1) 1.0 (!load-rocket pack1 rocket1 lc1) 1.0

المنارة للاستشارات
www.manaraa.com

(!fly-rocket rocket1 lc1 lc2) 1.0 (!unload-rocket pack1 rocket1 lc2) 1.0 (!load-truck pack1 t3 lc2) 1.0 (!drive-truck t3 lc2 ap5) 1.0 (!unload-truck pack1 t3 ap5) 1.0 (!load-airplane pack1 plane2 ap5) 1.0 (!fly-airplane plane2 ap5 ap7) 1.0 (!unload-airplane pack1 plane2 ap7) 1.0 (!load-truck pack1 t4 ap7) 1.0 (!drive-truck t4 ap7 loc8) 1.0 (!unload-truck pack1 t4 loc8) 1.0)

So the hierarchical structure in this transportation domain can be described as:

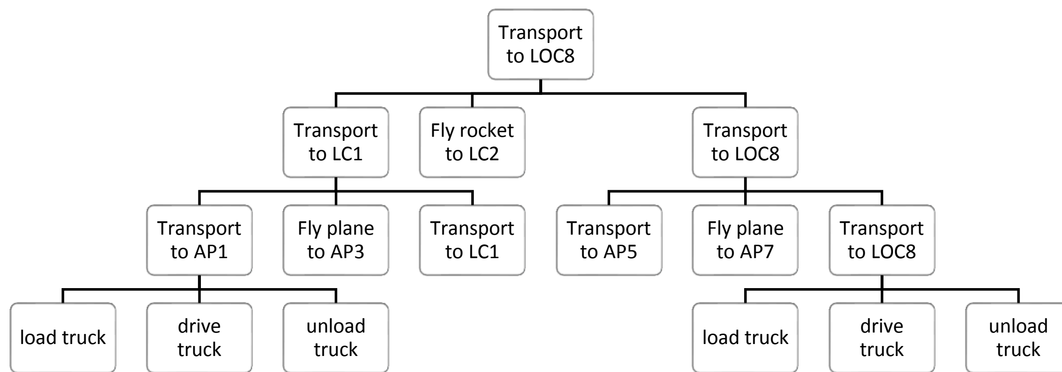

Figure 3.2: A hierarchical structure in transportation domain.

We may have multiple possible solutions on this domain, if we want to transfer the package to LOC8. Since the package can be picked up by the truck at LOC1, and go through LOC2 and then arrive at AP1, or directly arrive at AP1 without any other stops, there are two possible solutions in the sub goal “transferring the package to the airport AP1”. In addition, in city CITY2, when the package is picked up at airport AP3 (suppose the airplane PLANE1 arrive at airport AP3), the truck TRUCK2 can choose to visit LOC3 or LOC4, or both of the locations in city CITY2, then arrive at the launch center LC1, or directly arrive at LC1.

www.manaraa.com

By observing the transportation domain hierarchical structure, we can discover that the package should be arrive at the launch center LC1 and LC2 every time, and also the transportation between two planets cannot be avoided because the original place and the destination are on different planets. Independent of how we choose the routes within the city, and how many traces we may have when the package is transporting from LOC1 to AP1, or even to AP2, the launch center is the place that it will visit every time. So this is a landmark. We claim that our similarity algorithm will discover this landmark even when the package is arriving at different airports but in the same city.

We are going to discuss about the weight calculation algorithm and similarity algorithm in the transportation domain.

3.2.1 Weight Algorithm in Transportation Domain

The major steps in using the weight calculation algorithm is based on the pseudo code 3.1 shown below:

procedure Weight_Algorithm(Set of traces T)

$Trace_{base}$ = T_1

S_{curr} = the first state S_{11} from $Trace_{base}$

while (S_{curr} < the last state in $Trace_{base}$)

Distance = 0

if ($S_{curr} == S_{runner}$)

S_{runner} = Similarity($Trace_{base}$, S_{curr})

S_{seeker} = Similarity($Trace_i$, $Trace_{base}$)

end if

while (S_j < the last state in $Trace_i$)

المنارة للاستشارات
www.manaraa.com

Distance = Distance + $|S_{runner} - S_{curr}|$ + $|S_{seeker} - S_j|$

$S_j = S_{j+1}$

end while

$Weight_{curr}$ = Distance / number of states visited

$State_{curr} = State_{curr+1}$

end while

end procedure

Pseudo code 3.1: Weight calculation algorithm major procedures

First, we need to choose the first trace as our base trace. There is no preference on the method of choosing this base trace, because the landmarks are supposed to exist in every trace, no matter how we generate the trace as our base trace, the end result will not change. Then we are using this base trace and another companion trace to calculate the weight value on the base trace. We need to do this procedure for every other traces because we may have lots of input traces, the accuracy can be improved as more trances are given. Then we need to use our state runner and state seeker to find the tunnel between two traces. The runner state is taken from the base trace, and the seeker state is from the companion trace: $S_{runner} \in Trace_{base}, S_{seeker} \in Trace_{companion}$. The distance will be calculated by summing up the distance from the current state to the state runner, from state runner to state seeker, and then the distance from state seeker to all the other states in the companion trace.

After we collect all the distances from a particular state to all the other states in other traces, we divide the sum value by the number of the states we visited. This averaged value is a final measurement of the importance of this state whether we can

www.manaraa.com

determine it is a landmark in our transportation domain or not. Since we have series of the states in our base trace, the values will vary a lot because the distance each state to the tunnel is different. Ideally the states which has the minimum weight value is considered as the landmark in our domain.

3.2.2 Similarity Algorithm in Transportation Domain

Although we are able to find the states as the landmark by matching states, this is only working when two states are exactly the same. In our transportation domain, the airplane PLANE1 can fly to both AP3 and AP4 with the same probabilities. If we have ten traces but nine of them are going to airport AP3, only one out of ten is going to airport AP4, then AP4 will not be considered as a valid airport that has the potential to be considered as the place where landmark exists. This is also the reason why we cannot use the simple counter to count the occurrence of the places the package has visited.

The similarity algorithm we discussed in Section 3.1.2 can solve this problem and can also work with weight calculation algorithm to get the landmark we want. When AP1 is being considered, the similarity algorithm will consider AP2, which is also in CITY1 and it is also a place that can transport the package to CITY2. AP3 and AP4 are considered as a same airport because they are both in CITY2.

Using the similarity algorithm in the transportation domain results in a number of matched primitive atoms we have in each state. The reason why AP1 and AP3 are not considered as similar airport is because AP1 is an airport in CITY1, and AP2 is in CITY2. The algorithm searches for the AP1 and AP3 and finds that AP1 and AP3 are from different cities based on the atoms: (AIRPORT AP1) (AIRPORT AP3) (IN-CITY

المنارة للاستشارات
www.manaraa.com

AP1 CITY1) (IN-CITY AP3 CITY2) (CITY CITY1) (CITY CITY2). The algorithm will also search for the item CITY1 and CITY2, and realize it as the same items based on the atoms: (CITY CITY1) (CITY CITY2) (IN-PLANET CITY1 PLANET1) (IN-PLANET CITY2 PLANET1). So the similarity between AP1 and AP3 is smaller than the similarity between AP1 and AP2, because AP1 and AP2 are in the same city, which in the algorithm are considered to have higher relationship.

3.3 Example in Dota 2

Dota 2 is an online battle video game that allows at most 10 players to play the game. As we have introduced some basic concepts about Dota 2 in Section 2.3, we discover that the domain of the Dota 2 game also have the characteristics of planning domain in our research. Even though the gaming process can be totally different between each game played by different players, we still can find many features are shared accross all of the games in Dota 2. For example, the game will start at the scene that all the five players who are grouped as radiant team will be born in the radiant's base, and the other five players are born in the dire's base. Then the game will end when the ancient of either team is destroyed. If the ancient building is destroyed by the enemies, all the players and creeps are frozen and players cannot make any control on their heroes. Also the ancient is attackable only if the outer defensive towers are collapsed (i.e., destroyed by the enemies), the team fight are mostly happened near the end of the game. No matter how many times the games are played, these features are fixed in the game. This feature applies to the planning model characteristics so we can use the Dota 2 domain in finding important events in the game, or in other words, finding the landmark in the Dota 2 game.

المنارة للاستشارات
www.manaraa.com

The transportation domain and the Dota 2 domain are different. In our new transportation domain, each state in an input trace is connected by the operators, which are given from a set of countable operators. Each next state is predictable in our domain because there is an add list and delete list in each operator. By applying the adding list and the deleting list onto the pre-condition state, an effect state can be generated. Although we can have hundreds of valid input traces in our transportation domain generated by JSHOP, the number of valid inputs are not as many as the input we can collect from the Dota 2 game. The transportation domain is a classical model that all of its possibilities can be created by permutations. Basically, all the possible solutions can be ideally populated by JSHOP, which means the complexity of the solutions are dependent on the restrictions we set. We can only have no more than 10 valid input traces when we downgrade the transportation domain, which means we make the package delivered to a location in the same city, and there are only 2 locations that the truck can visit. So the transportation domain is a simple and predictable domain. While in the Dota 2 domain, we are not able to collect all the operators because every next state is non-predictable. The next state in Dota 2 are majorly depend on the players' decision, or the effects given by other players. The location where the heroes choose to stand, the time when the ability is used on himself or others, and the lane they choose to approach to the enemies' ancient, can have countless progress and results just like the real world. So we cannot use exactly the same algorithm in the Dota 2 domain.

3.3.1 Weight Algorithm in the Dota 2 Domain

We still employ the weight calculation algorithm in the Dota 2 domain but we make some changes according to the game feature. As we know from the section 2.3 and

المنارة للاستشارات
www.manaraa.com

previous paragraph, Dota 2 does not have operators between states, the next state is non-predictable from previous state that we may have all the state traces different from each other, which means the number of traces we can employ is unlimited. Another issue we need to consider is the length of the trace. A regular Dota 2 game will last for at least 30 minutes, and each tick will capture a state in the game. The tick period is as short as 0.06 second, so we can have at least 30000 states in a single input trace. So the number of states in a trace in the Dota 2 domain is much more than we have in the new transportation domain. In our algorithm, we need to import multiple input traces based on our domain, and this results to the problem of calculating the sum of distance with hundreds of thousands of states. This is both time and space consuming.

As we have discussed in section 3.1.1, we pick up one trace as our base trace. The trace in the Dota 2 domain is the file of one replay file. Then the current state in the base trace will find a tunnel, which will be calculated by our similarity algorithm, to visit other states in other traces. Then the value of all the distance from the current state to other states will be summed up and will be divided by the states it visited. This is the weighted value of the current state. Then the algorithm will move the current state to next state, do the previous procedure again and calculate the new weight value for the current state. The pseudo code is no much difference with the one in section 3.2.1

3.3.2 Similarity Algorithm in the Dota 2 Domain

The similarity algorithm in Dota 2 domain is different from the one in the transportation domain. First, in transportation domain, we make an exact match between two states from two different traces, while in the Dota 2 domain, we cannot have operators which contains add list and delete list, so there is no exact match to be made.

المنارة للاستشارات
www.manaraa.com

Second, even if we only take heros' health, mana, location x and y, and time as five variables in our domain, there are no same states with one or two variables differences. So we use a range similarity here to determine two states whether they are similar or not.

We use heroes' health, mana, location x and y in map, and time, these 5 variables in our domain to simplify our landmark searching complexity. These 5 variables are considered to be the key factors to determine a team fight, which is a very important event in the game. The team fight is supposed to happen in all the traces and the location, time when team fights happen can give useful information during the game. If at least two heroes are gathering together from each team in a very small range of location, and with very quick health and mana consuming or curing in a short time, we will say this is the location to have a team fight. If two states from two different traces both of them have this scenario, we put them as sharing a similarity feature.

The quick health and mana consummation can be defined as, starting from above 80 present of mana and health value, the value drop or resume to below 10 percent during the range which at least two heroes from each team are standing closely. For the range of the location we consider as a close state, we set the percentage by experience at first, and then automatically adjust by the algorithm to increase the percentage in stage until we find the team fight. If there is no scenario in the trace that meets our algorithm after the algorithm enlarges five times, we consider the trace an invalid trace and we will delete this trace from our trace set.

المنارة للاستشارات
www.manaraa.com

4. Implementation

4.1: New Transportation Implementation

We discuss several steps needed before discussing the code (see the figure below). First, our algorithm needs enough valid input traces. We use the planner JSHOP to obtain the traces in transportation domain. SHOP, is short for Simple Hierarchical Ordered Planner, was developed by Dana Nau, Yue Cao, Amnon Lotem, and Hector Munoz-Avila, 1999. JSHOP is the Java version of SHOP. JSHOP plans the order of the tasks in hierarchical structure to achieve the final goal. By given the domain description and problem description which contain operators declaration, axioms, initial state restrictions and final goals, JSHOP is able to generate all the possible solutions for achieving problems such as "transporting the package from location 1 to location 8 between different cities, different planet".

Second, the input traces must be in a particular format as a text output, so we need to parse these traces and translate them into another format that can be parsed by our weight and similarity algorithm module.

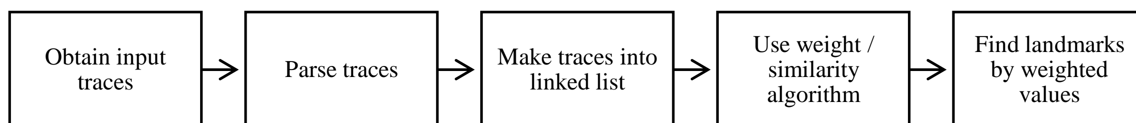

Figure 4.1: Workflow diagram for finding the landmarks in transportation domain.

By assigning tasks to this procedure, we modularize our code into three parts: the data generation module, the trace parsing module, and the landmark finder module. The data generation module is responsible for constructing our new transportation domain and

المنارة للاستشارات
www.manaraa.com

set the initial state and the final goal to our problem. The output of this module is all solutions in the required format. Here is an example:

Plan # 1

((!load-truck pack1 t1 loc1) 1.0 (!drive-truck t1 loc1 ap1) 1.0 (!unload-truck pack1 t1 ap1) 1.0 (!load-airplane pack1 plane1 ap1) 1.0 (!fly-airplane plane1 ap1 ap3) 1.0 (!unload-airplane pack1 plane1 ap3) 1.0 (!load-truck pack1 t2 ap3) 1.0 (!drive-truck t2 ap3 lc1) 1.0 (!unload-truck pack1 t2 lc1) 1.0 (!load-rocket pack1 rocket1 lc1) 1.0 (!fly-rocket rocket1 lc1 lc2) 1.0 (!unload-rocket pack1 rocket1 lc2) 1.0 (!load-truck pack1 t3 lc2) 1.0 (!drive-truck t3 lc2 ap5) 1.0 (!unload-truck pack1 t3 ap5) 1.0 (!load-airplane pack1 plane3 ap5) 1.0 (!fly-airplane plane3 ap5 ap7) 1.0 (!unload-airplane pack1 plane3 ap7) 1.0 (!load-truck pack1 t4 ap7) 1.0 (!drive-truck t4 ap7 loc8) 1.0 (!unload-truck pack1 t4 loc8))

All these traces are generated by the output file in JSHOP folder, and then we need to parse them by using the trace parsing module. The trace contained in the output file is a sequence of operators that can be applied to achieve the final goal, so we need to translate them into a sequence of states linked by these operators. In the trace parsing module, we wrote a C++ program to automatically read strings from the output file containing all the operator sequences, and extract information to generate the actual linked states. The states are linked as linked lists and such lists are the actual input traces that will be analyzed in the next module. In our landmark finder module, which is the subsequent module of parsing procedure, we use our weight algorithm and similarity algorithm to calculate the weight value of each state in the base trace. As explained in the

المنارة للاستشارات
www.manaraa.com

previous chapter, the landmarks are those that have the lowest value among all the states in one trace. The lower the weight value one state has, the higher chances it is a landmark. The modularization and procedure can be found in figure 4.2. Finally we get an array of weight values that compute the importance of each landmarks in each trace, and examine whether they are real landmarks in our transportation domain.

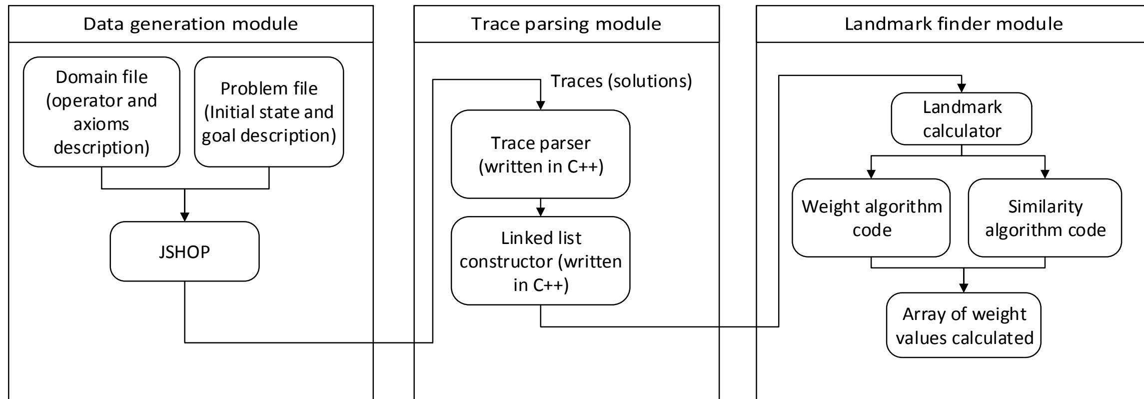

Figure 4.2: Modularization of finding landmarks in transportation domain

We will discuss the trace parsing module and the landmark finder module in detail, while for the first module, we can get output from JSHOP directly so we do not discuss it and refer to (Nau et al., 1999).

4.1.1 Parsing Traces

Given an output file containing a series of solutions, we need to firstly do the stream input to read characters from this file and then try to translate the sequence of operators into a sequence of states. The output of this module should be a set of traces, which contains sorted states in it. We use an array to store all the traces. In each cell of this array, there is a bi- directional linked list. Each state, except for the initial state and final state, has pointers referring to its previous state and next state. For each operator

المنارة للاستشارات
www.manaraa.com

instance, it will have pointers pointing to the pre-condition state and effect state respectively. The data structure is shown in figure 4.3.

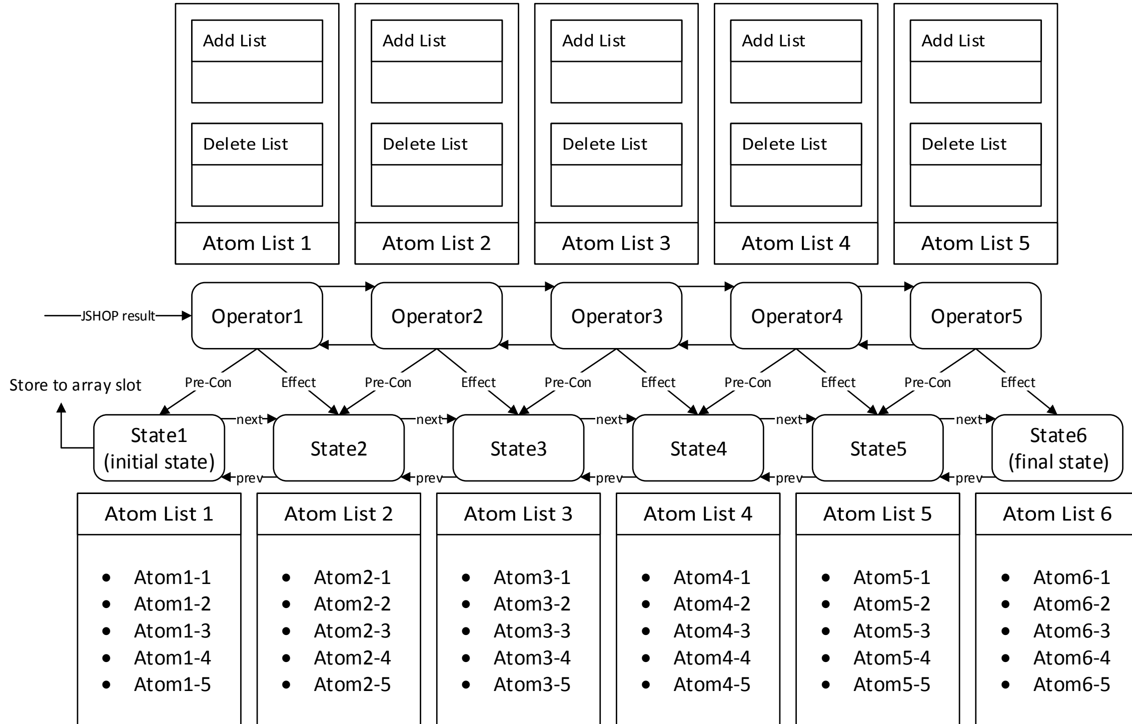

Figure 4.3: the data structure of the states and operators.

For parsing the output file from JSHOP, we write a program in C++. First, all the operators are known to us, so we create a list of operator instances that stores all the possible operators that could be used in the parsing procedure. Second, we load the initial state by reading its atom list. Third, we load the first operator from the output file in the first module. Note that each operator in the output file is shown in the format: (!operator_name object from to), so the code will search for the operator's name in the operator list. The matched operator will return a list of add-list and delete-list that will be implied on the initial state, some atoms will be added, and some atoms from the initial state atom list will be deleted. This will generate the next state in that trace, equipped

www.manaraa.com

with a new atom list that is different from the initial state. Then the procedure goes on, loading the next operator and getting the add- and delete-list, and creating a third state based on the second state. Also, the previous state has a next-state pointer, while the next state will also have a previous-state pointer. For the linked list of operator, they have additional pointers pointing to the previous state and the next state, which are the pre-condition and effect respectively.

The constructing procedure can be described in the following pseudo code 4.1. The SeachOperator function is a matching procedure that will fetch the add- and delete-list from the matched operator.

procedure Construct_States ($Operator\ Sequence, S_{init}$)

$S_{curr} = S_{init}$

$O_{curr} = O_1$

while ($O_{curr} <= O_{final}$)

(AddList, DeleteList) = SearchOperator(O_{curr})

$S_{curr+1} = (S_{curr} \cup AddList) \setminus DeleteList$

$(Pointer_{prev} \in S_{curr+1}) = S_{curr}$

$(Pointer_{next} \in S_{curr}) = S_{curr+1}$

$S_{curr} = S_{curr+1}$

$O_{curr} = O_{curr+1}$

end while

end procedure

Pseudo code 4.1: State constructing procedure in transportation domain

المنارة للاستشارات
www.manaraa.com

4.1.2 Similarity Algorithm Implementation

As we have discussed in Section 3.2.2, even if the package arrives at a different airport when these two airports are located in the same city, we consider these two airports as similar airports, but the distance from one trace to the other may be more than one unit because they are not the same. The method we use here is to look at the difference between two states from two traces first, then we focus on the different atoms to examine whether one atom from each trace can be classified as a similar atom. If we have two traces segments as shown below:

State 2	State 3	State 4	State 5
(on-truck pack1 t1) (truck-at t1 loc1) (truck-at t2 ap3) (truck-at t3 ap4) (plane-at plane1 ap1) (plane-at plane2 ap5) (rocket-at rocket1 lc1) (in-city ap1 city1) (in-city ap2 city1)…	(on-truck pack1 t1) (truck-at t1 loc2) (truck-at t2 ap3) (truck-at t3 ap4) (plane-at plane1 ap1) (plane-at plane2 ap5) (rocket-at rocket1 lc1) (in-city ap1 city1) (in-city ap2 city1)…	(on-truck pack1 t1) (truck-at t1 ap1) (truck-at t2 ap3) (truck-at t3 ap4) (plane-at plane1 ap1) (plane-at plane2 ap5) (rocket-at rocket1 lc1) (in-city ap1 city1) (in-city ap2 city1)…	(obj-at pack1 ap1) (truck-at t1 ap1) (truck-at t2 ap3) (truck-at t3 ap4) (plane-at plane1 ap1) (plane-at plane2 ap5) (rocket-at rocket1 lc1) (in-city ap1 city1) (in-city ap2 city1)…

Table 4.4: A segment from an input trace. Truck 1 goes to airport 1.

State 2	State 3	State 4	State 5
(on-truck pack1 t1) (truck-at t1 loc1) (truck-at t2 ap3) (truck-at t3 ap4) (plane-at plane1 ap1) (plane-at plane2 ap5) (rocket-at rocket1 lc1) (in-city ap1 city1) (in-city ap2 city1)…	(on-truck pack1 t1) (truck-at t1 ap2) (truck-at t2 ap3) (truck-at t3 ap4) (plane-at plane1 ap1) (plane-at plane2 ap5) (rocket-at rocket1 lc1) (in-city ap1 city1) (in-city ap2 city1)…	(obj-at pack1 ap2) (truck-at t1 ap2) (truck-at t2 ap3) (truck-at t3 ap4) (plane-at plane1 ap1) (plane-at plane2 ap5) (rocket-at rocket1 lc1) (in-city ap1 city1) (in-city ap2 city1)…	(obj-at pack1 ap2) (truck-at t1 ap2) (truck-at t2 ap3) (truck-at t3 ap4) (plane-at plane1 ap2) (plane-at plane2 ap5) (rocket-at rocket1 lc1) (in-city ap1 city1) (in-city ap2 city1)…

Table 4.5: A segment from another input trace. Truck 1 goes to airport 2.

We can observe from two tables that the truck 1 goes to airport 1 while in another table the truck 1 goes to airport 2. By using our similarity algorithm, when we are looking for matching state of state 4 in trace 1 (the first table), we cannot find any matching state

المنارة للاستشارات
www.manaraa.com

in trace 2 (the second table). But when we take a deeper look at two traces, we find that state 4 in trace 1 and state 3 in trace 2 they have one atom sharing the same atom name, but one of the parameters is different. Those atoms are: (truck-at t1 ap1) from the first trace and (truck-at t1 ap2) from the second trace. Then the similarity algorithm explores into these two atoms by looking at the similarity between ap1 and ap2. The algorithm firstly observes that ap1 and ap2 are both airports by finding the same atom these two states both have: (AIRPORT AP1), (AIRPORT AP2). Then the algorithm takes checks the triple (atom_name obj_name obj_container), that is, (in-city ap1 city1), and (in-city ap2 city1). The similarity algorithm finds that their distance should be 1 because there is only one difference between state 4 in trace 1 and state 3 in trace 2, and ap1 and ap2 are of the same type and occur in the same triple.

procedure Similarity ($S_{runner}, Trace_i$)

$S_{seeker} = S_2 \in Trace_i$

while ($S_{seeker} < S_{final} \in Trace_i$)

distance = MAX_INT

if ($Atom_{seeker} = Atom_{runner}$) // means exactly match

distance = 0

else if ($Atom_{seeker} \cap Atom_{runner} = n\ atoms\ difference$)

if (all atom names can be matched && n <= threshold)

if (each matched atom pairs has same $Obj_{container}$)

distance = 1 * n

end if

end if

end if

$S_{seeker} = S_{seeker+1}$

المنارة للاستشارات

www.manaraa.com

end while

return distance, S_{seeker}

end procedure

If the distance we calculated from the similarity algorithm is larger than our distance threshold, the returned distance should be put as INT_MAX, because the distance is larger than our threshold and hence it cannot give us an accurate estimate of distinguish value that landmarks may have. In our new transportation domain, we set the threshold to be 2 so that any distance calculated by the similarity algorithm will be marked as a positive infinite integer sending it back to the weight algorithm.

4.1.3 Weight Algorithm Implementation

The weight algorithm is not much different than the algorithm we proposed in Section 3.2.1. We first pick a trace as our base trace and set the second state as the current state. We do not need consider the initial state, because the initial state is always a landmark, as was explained before. We have the same initial state across all traces, so we can just start from the second state. The next step is to run the state runner and state seeker to find the tunnel between two traces. We use our similarity algorithm at this time to compute the distance between state runner and state seeker. If the distance is the positive infinite, the state seeker and runner cannot make a tunnel for the two traces. Then the state seeker keeps running until a similarity is found. If there is no exact matching or similar matching, the state runner will move to the next state in the base trace, and then the state seeker will traverse the other trace, searching for a matching tunnel. For each current state, it needs to compare the distance from the previous tunnel and next tunnel, and it will pick a tunnel that is shortest to visit other traces.

المنارة للاستشارات
www.manaraa.com

4.2: Dota 2 Implementation

The Dota 2 domain is a very different domain from the transportation domain. It does not have operators, pre-conditions, and effects. Actions in Dota 2 can be non-deterministic (more than one event can follow the action). But in Dota 2 replay files, the whole sequence of the game is recorded. Starting from the beginning of the game, when heroes are picked, and until the very end of the game, when the ancient of one team is destroyed, each state is stored in the replay file in every game tick. By using an open source program called Skadi and Tarrasque, which are written in Python, we are able to parse the Dota 2 replay files into another readable format that contains series of the states. We choose to use JSON as our output format because it has an informative and clear to view structure. We use those JSON files to calculate the weight values by running our Python code. The landmark timestamp and location are recorded and used in our Dota 2 runtime scripts, to allow the camera moving to a particular place and focus on a hero to have a continuous view of the whole event.

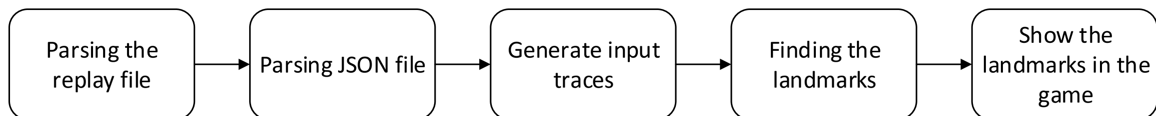

Figure 4.4: A working sequence of finding the landmarks in Dota 2 domain

We divide the whole project into four modules. They are the module of parsing Dota 2 replay files, the module of translating the JSON files into input traces, the module of finding the landmarks in each trace, and the module of locating the runtime event based on the landmarks we found. The place where we download the replay files is called

المنارة للاستشارات
www.manaraa.com

DOTABANK (www.dotabank.com). This website allows the users to upload their game replays into their database, and let others download and share the replay files with no copyright issues. The number of replay files in dotabank is huge, which gives us plenty of resources to analyze.

We use the Tarrasque project, a library developed in Python. Terrasque, which is based on the Skadi project, is a tool to allow the easy and straightforward analysis of Dota 2 replay files. The Skadi project is a kernel that parse the Dota 2 replays, but it is not as convenient as Tarrasque. So the Tarrasque project is an upper level API that can let users have easier access to the parsed data in the memory. We use Tarrasque to extract heroes' health, mana, locations, and time information in each state, and make them into a structured JSON file as the output.

For the translation module, we translate the data stored in JSON files into the arrays needed for the input traces. Each input trace contains an array of states. This enables us to get access to these states and the information included in each state. The translation module is written in Python. For the landmark searching module, we use our similarity and weight algorithm to analyze all input traces and calculate the weight values in each state. We use Python to program in this module. We record the landmarks and get it prepared for the next module, the game runtime module. A very straightforward way to check whether the landmarks we find is really a landmark in the Dota 2 game is replaying the game again and move the camera to the place where the landmarks happen. There is a built-in command in the Dota 2 game that can allow setting the style and the location of the camera, focusing on a particular hero, and view the events just as the real scenario.

المنارة للاستشارات
www.manaraa.com

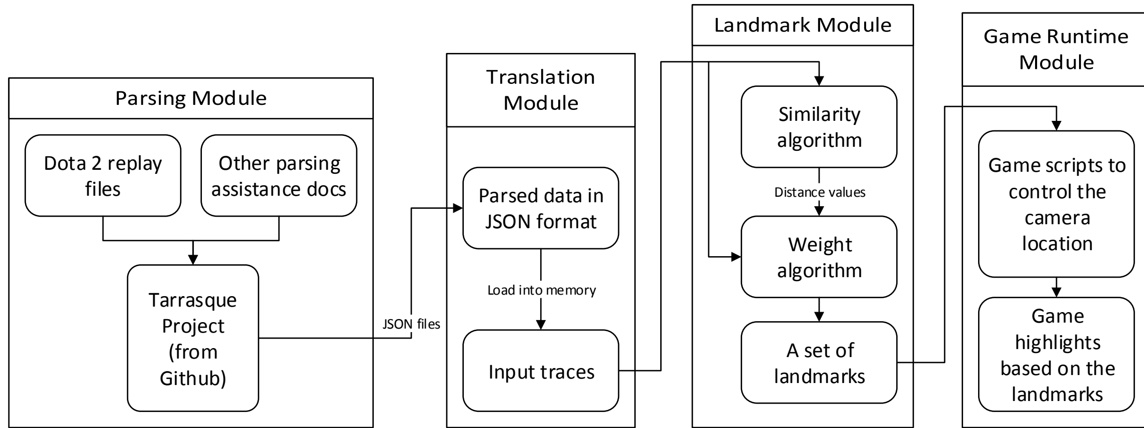

Figure 4.5: Procedure and component description classified by modules

4.2.1 Preparing the Input Traces

We download ten replay files from www.dotabank.com, a website that players can upload their game replays and share with each other. The replay file is encoded into a format and the information inside is highly compressed. We use Tarrasque project on Github to parse the replay file and give the output in a form that is easy to use in the next steps. Restricted by the version update of the Dota 2 game, we cannot use a higher compressed version replay files that were published after July 2014, so the Tarrasque project can only parse the replay files prior to this time, which means we have ten replay files, each of them has a file size larger than 100MB.

We coded in Python our implementation. First, we need to import the Tarrasque library to get access to all the parsing functions. Then we load the replay file directory to import the target file we want to parse. We can easily parse the replay files by using the function: tarrasque.StreamBinding.from_file(target_file_directory), to load all the data into computer's memory.

المنارة للاستشارات
www.manaraa.com

The first step in our code is searching for the player names in the game. We need to know who plays in the replay file and which team (radiant or dire) does he belongs to. We need to take care of the name tag on each player. Tarrasque will throw an exception on the characters that are not in UTF-8. It means that some replay files cannot be correctly parsed because the user ID given by the players can be accepted by the game platform, but cannot be accepted in the Tarrasque. The replay files should be carefully picked on dotabank.com. This problem can be solved by viewing the detailed information on the web page of that replay file, and we can get rid of such character failure beforehand without downloading and loading them in our code.

The second step is reading hero names and matching them to the players that is playing in this game. We create a file in JSON format that records all the localized hero names. These names are official names in the Dota 2 game. These 107 hero names are easier to use than looking up the players' ID. After we load this file, we put the localized hero names onto the player IDs. By using this block of code we are able to view each player's hero and we make these heroes into an ordered array, that we know the index of each hero in our output JSON file.

The last step in parsing replay files is iterating over each tick to extract from the game trace the health, mana, location x, location y, time information in that tick. The tick is adjustable, but in our code we just use Dota 2's default tick, which is approximately 0.06 second in each interval. In each tick, we firstly get the official name of the hero from the array we created in our previous step, then we use this name and the tick to locate the detailed memory location, to get the health, mana, location x, and location y in that tick. We formulate this data into a formatted form in JSON and dumping them into an output

للاستشارات المنارة
www.manaraa.com

JSON file. The actual parsed file is usually more than one hundred MBs, because it is no longer highly compressed. On the other hand, the data in the parsed file is easier to view. Here is the format:


```
{
        "max_mana": 299.0364990234375,
        "mana": 299.0364990234375,
        "player": 1,
        "health": 492,
        "y": -6784.0,
        "x": -6912.0,
        "max_health": 492
},
…
```


The JSON file we parsed needs to be translated into input traces. Based on the JSON library in Python, we load the JSON file and make all the ticks in the file into an array. The array we created in the memory is the input trace that is used in the finding landmarks module. We have several reasons for creating the JSON file as a middle point between the replay file and input traces. First, it is easier to examine whether we have a correct parsed output, since we cannot understand the raw data in replay files. Second, the JSON file we created can be used in both Linux and Windows operating systems, while we can only use Linux to run the Tarrasque Parsing tool.

4.2.2 Finding Landmarks Using Weight and Similarity Algorithm

For the similarity algorithm, we need to define what a team fight is. The team fight is an event in which at least two heroes from each side is attaching their enemies in

المنارة للاستشارات
www.manaraa.com

the same region. The region size can vary, but should be a small region compared to the total size of the game map. So we set up a threshold that can restrict the region where the team fight happens. This threshold is based on experience, starting from 2% to 5% of the map size. There is no need to make our threshold larger because the threshold larger than 5% is a large rectangle in the game, which can even cover two defensive towers; this means some cases that are actually not team fights will be recognized as team fights, so it would lead to misclassification. We increase the threshold by 1% every step. Also, we use the health, mana information to determine whether the event is really a team fight because when the team fight happens, heroes have dramatic health and mana fluctuation or even died (health equals to zero).

In calculating the similarity values, we generate all possible combinations of heroes on the map, and search for the maximum location x, location y, and minimum location x, location y in each state. If the range satisfy the threshold we set, this is a similar state in that trace.

$$Potential\ Landmark = \left\{ S \left| \begin{array}{l} \left(MAX(Location_{x_i}) - MIN\left(Location_{x_j}\right) < Map\ Size_x * Threshold_x\right), \\ \left(MAX(Location_{y_i}) - MIN\left(Location_{y_j}\right) < Map\ Size_y * Threshold_y\right) \end{array} \right. \right\}$$

The similarity states are marked as tunnels between two traces, and the distance in this tunnel depends on the state location in that trace. If two states are similar states and both of them are located at the end of the trace, the difference of the location percentage is very small, so the tunnel cost (the similarity value) is small. And if one state is at the beginning of the game in one trace, but the other is at the end, the similarity value of

المنارة للاستشارات
www.manaraa.com

these two traces are very big. Then the weight algorithm's state runner and state seeker are less likely to choose this tunnel because it has a higher value of visiting another trace.

We use the general weight calculation algorithm as we have discussed in Section 3.3.1. We choose a base trace at first, and then set up the current state, state runner on that trace. The state runner will run to the tunnel state and calculate the distance from the current state to all the other states on the other trace. The tunnel is created by the state runner and the state seeker; the runner's and the seeker's distance is determined by the returned value from similarity algorithm. In the Dota 2 game, even though a team fight usually takes only a few seconds, it still occupy many ticks. For example, if the team fight takes ten seconds, and the tick period is 0.06 second per tick, we will have more than 150 continuous states in a team fight. So in the Dota 2 domain, a landmark is a period of states that satisfies our requirement as opposed to a single state.

4.2.3 Managing the Camera in the Dota 2 Game

The best way to examine the correctness of the landmarks is by putting them back into the replay files and view in Dota 2's own replay mode. In order to see whether the landmarks really work, we need to be able to move to a particular time in the replay, and move the game camera to the location where a team fight is supposed to occur. Also, the camera's focus is needs to be put on a particular hero so that we can move the camera according to the hero's location change during the team fight. We use Dota 2 console commands to control the time and location of the camera in replay mode.

In Dota 2 replay mode, we have four camera mode: directed camera, free camera, player perspective, and hero chase. For the directed camera mode, the camera will

المنارة للاستشارات
www.manaraa.com

automatically focus on the area filtered by Dota 2 itself. This may including team fights, first blood (i.e., the first time here is combat and one of the heroes loses health points), the defensive tower destroyed, attacking Roshan (i.e., the most powerful neutral creep in the game), etc. Some of these events do not happen in all the replays, so we cannot use this camera mode to check the landmarks. The hero chase mode is a good view point but it has its flaws. In this mode, people can also view the manipulation by the players, including the mouse hits, ability used, which can be distracting. But they are not our major concern.

We use a combination of the console commands to choose the replay file, run the game, move to a particular time according to landmarks, and move the camera to the location where the landmarks happen. We notice that not all heroes participate in the team fight every time, so we look back to the parsed file, and select a hero who participated in this team fight, and is alive for the duration of the team fight. By making these restrictions, we wrote several console commands in Dota 2's configuration file (see Table 4.6.

dota_spectator_mode 3	Change the camera mode to "hero chage"
dota_camera_lock 1	Lock the camera
demo_gototick 123456	Go to the 123456th tick in this replay
dota_spectator_hero_index 3	Pick hero in index 3 and focus on him
Dota_camera_getpos	Get current position(x, y, z) in the game

Table 4.6: A list of Dota 2 console commands that we used in representing the landmarks.

المنارة للاستشارات
www.manaraa.com

5: Experiment

In this chapter, we report on our experiment for finding the landmarks in the transportation domain, the Dota 2 domain, and the experiment taken on the Amazon mechanical turk to collect the feedbacks from Dota 2 expert players.

In the new transportation domain and the Dota 2 domain, we implement the code and report on results from our programs. We provide figures and tables to show the result calculated by our weight and similarity algorithms as we discussed in Chapter 3 and Chapter 4.

For the Amazon mechanical turk section, we by using a series of videos. They show when and where the landmarks occur. We upload ten pairs of the videos onto the Amazon mechanical turk, and receive feedbacks from turkers (i.e., online mechanical turk workers) who are experts in the Dota 2 game. Then we analyze the data given by these users and draw our conclusions. Our hypothesis is that the weight and similarity algorithm is a good implementation in finding the landmarks in the transportation domain, and it is also very useful in finding the important event in the Dota 2 game.

5.1: Experimental Setup

In this section we introduce our experimental setup for the transportation domain experiment, the Dota 2 domain experiment, and the Amazon mechanical turk setup. In order to discover the landmarks in these domains, we wrote different code to import the input traces from their data sources, and we realize our weight and similarity algorithm in two different ways.

المنارة للاستشارات
www.manaraa.com

The software, program and code mentioned in this section is compiled and run on the computer which is equipped with Intel Core i7-4770 CPU, 3.40GHz processor, 16.0 Gigabytes installed memory(RAM), and the operating system is a 64-bit Windows 7 professional.

5.1.1 Transportation Domain

The data sources in our new transportation domain are from the JSHOP program, which is developed by Dana Nau, Yue Cao, Amonon Lotem, and Hector Munoz-Avila, 1999. The input files are called "domain.shp" and "problem.shp". In the domain file we wrote a complete list of operators with their own adding list and deleting list. We also add some other rules and axioms in this file. In the problem file we wrote the initial state and we described our final goal.

We use C++ to code the parsing and finding landmarks modules. The integrated development environment is Microsoft Visual Studio 2012. The input files are from the results given by JSHOP, and the output forms are list of float values that can show the landmarks which have lower values. These final output data is stored in a text file.

5.1.2 Dota 2 Domain

In order to generate the replay files that are played by other players, we use www.dotabank.com to download 10 Dota 2 replay files. The suffix of these replay files are "*.dem" and all these files are parsed by the tool called Tarrasque. Tarrasque is downloaded from Github.

The Tarrasque project requires to be running on a Linux operating system, so we set up the environment by using the virtual box version 4.3.2. We installed the Linux

المنارة للاستشارات
www.manaraa.com

operating system: a 64-bit Ubuntu 12.04 LTS version, and we set up a virtual Python compiling and running environment on Ubuntu. After the Tarrasque project is successfully installed, we import 10 replay files and run our Python code. The Tarrasque project is actually a library so we do not need to worry about the code inside this project.

After we have all the parsed 10 JSON files, we run our Python code, which contains the similarity and weight algorithms, to explore landmarks. The results are also stored in a text file that contains a list of landmarks in that replay. In order to run the python code on the Windows 7 operating system, we use the Python 2.7.9 for Windows and we make some modifications on our code to satisfy the input/output requirement on Windows operating system.

The Dota 2 game is already installed in our computer, and we write a Dota 2 console command list in an automatic execute file. We save this file under "Dota2\Steam\SteamApps\common\dota 2 beta\dota\cfg", and we rename this file to "exec.cfg". Before Dota 2 is opened, we need to add "-console" in the launch options of Dota 2 properties in STEAM platform. When the Dota 2 game is opened, we need to firstly open the command line window and use the command: playdemo *.dem to play the replay file and "exec exec.cfg" to apply the camera settings and load the time and locations where our landmarks are.

We use a video recording software "Lukool Recorder" to make videos that is around ten to twenty seconds. The videos we take are corresponding to the landmarks we discovered in previous steps.

المنارة للاستشارات
www.manaraa.com

5.1.3 Amazon Mechanical Turk

In order to ask people in Amazon mechanical turk to answer the questions in our experiment, we passed the web-training certification from The National Institute of Health (NIH) Office of Extramural Research and obtain approval from IRB.

We prepare 10 pairs of videos corresponding to 10 input traces we choose in our experiment. For each pair, one shows the landmark that has lower calculated weight value based on the weight and similarity algorithm, and the other shows a higher weight value calculated in the same circumstance. The landmark which has lower value should be more informative than the other one. Subjects will have no control over the game replay itself, and each video will be unlabeled so as to reduce bias. Subjects will not know how the play scenes were selected nor will they know that the selection was based on what we consider strong/weak versions of our algorithm.

The person who take this experiment will be asked for several questions to be examined whether s/he has a prior experience in playing the Dota 2 game, and whether s/he is an expert player. We will divide participants in 3 groups: (1) the expert in playing Dota 2, (2) those who have played Dota 2 before but do not have much experience in this game, and (3) not played any Dota 2 before. The participants will not know which group they have been classified to. Each participant will be shown a 5-minute video showing the fundamental concepts of Dota 2. Then each participant will be shown between 10 pairs of videos as described before. Each video lasts around 10 to 20 seconds.

After each pair of videos are shown, each participant will be asked two questions that evaluate both objective and subjective criteria in relation to the videos. The

المنارة للاستشارات
www.manaraa.com

participants will be asked which of the two videos the participants find (1) more informative about the events occurring in the game and (2) more entertaining to watch.

Participants (called turkers) will be compensated monetarily via the Amazon Mechanical Turk platform. We will collect all results and run a student t-test to determine if there is any significance in difference between preference on the pairs of videos (for both, the "informative" and the "entertaining" categories).

As with watching videos, there is a risk of stress, fatigue, or hyperventilation which could be the onset of seizures. If you experience any of the above symptoms, please stop the study immediately.

IF YOU SUFFER FROM EPILEPSY, PHOTOSENSITIVITY OR GENERALLY FEEL SICK WHEN WATCHING FAST-MOVING VIDEOS, YOU MAY NOT PARTICIPATE IN THE STUDY.

The benefits to participation are:

While participation in this study will not directly benefit you, the knowledge gained from this study will enable us to evaluate the effectiveness of algorithms for automated camera control.

- **Compensation**

  You will be given compensation according to the market value in Amazon Mechanical Turk.

- **Confidentiality**

  Amazon Mechanical Turk handles your confidentiality.

- **Contacts and Questions**

  If you have any questions or comments contact: hem4@lehigh.edu

  Questions or Concerns:

  If you have any questions or concerns regarding this study and would like to talk to someone other than the researcher(s), you are encouraged to contact Naomi Coll, Lehigh University's Manager of Research Integrity, at nac314@lehigh.edu or 610-758-3021. All reports or correspondence will be kept confidential.

- **Statement of Consent**

  I have read the above information. I have had the opportunity to ask questions and have my questions answered. I consent to participate in the study.

Agree with the above terms: ☑ OK

(need to check box to continue)

Figure 5.1: Segment of the consent form of our experiment on Amazon mechanical turk.

Pre-test Questionnaire

The following questions are used to test whether you are an expert of Dota 2

This is **not** part of the experiment. Please fill out these questions honestly.

1. Have you every played any "MOBA" (or DotA 2) games before?
   ○YES ○NO
2. How many games you have played in Dota 2? How many battle points you have?
   Number of games you played:
   Battle points you earned:
3. Is the Base Tower attackable if the front tower is still existing?
   ○YES ○NO ○Don't know
4. Farming is most important for the following type of hero:
   ○Jungler ○Tank ○Carry ○Support ○Don't know
5. Would you say that you 'feed' your team or 'feed' the enemy team?
   ○'feed' your team ○'feed' the enemy team ○Don't Know
6. Which item could be purchased in the secret shop? (Multi Select)
   ☐Morbid Mask
   ☐Demon Edge
   ☐Blink Dagger
   ☐Ring of Health
   ☐Don't Know

المنارة للاستشارات
www.manaraa.com

Figure 5.2: The pre-test questionnaire that is used to classify people whether he is an expert in play the Dota 2 game.

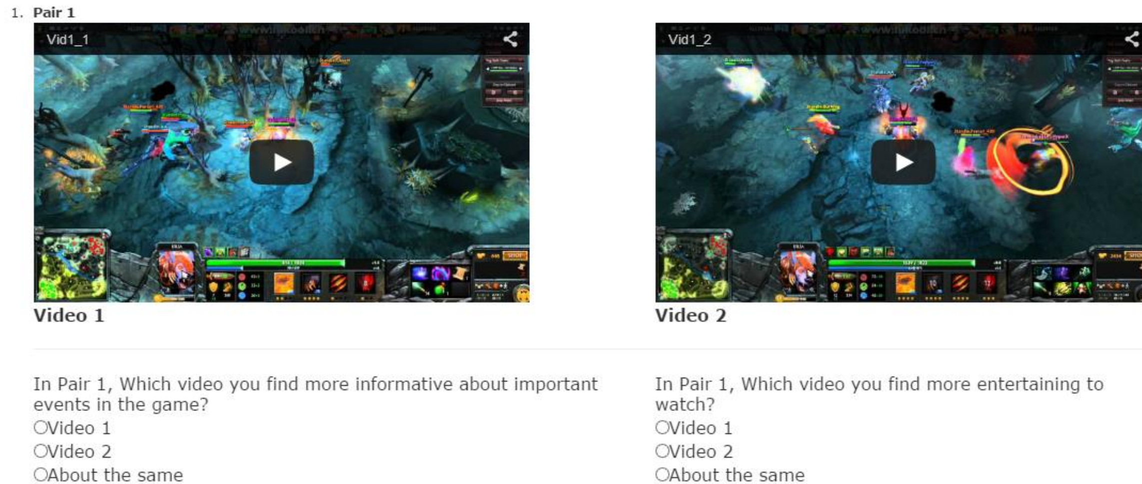

Figure 5.3: Two questions are asked after the participants finished watching a pair of videos.

5.2: Experimental Results

In this section, we discuss the experimental results on the new transportation domain, the Dota 2 domain, and our Amazon mechanical turk survey results. We analyze these results and assess our implementations in both transportation and Dota 2 domain. Based on the results we analyzed, we found that our method, which is using the similarity and weight algorithms to find landmarks, or important states in both domains, can effectively identify important events for the most part. We also discuss some difficulties in our experiments. We examine these difficulties and suggest how to solve them in future works.

المنارة للاستشارات
www.manaraa.com

5.2.1 Transportation Domain Results

The final output format in transportation domain is a list of weight values in each trace. These values can reflect the importance of a state whether one state is important enough to be considered as a landmark. Based on our algorithms, the lower (smaller) value in the final output list, the more important event; this state is a landmark in our domain. In our transportation domain experiment, we imported 40 traces and in the end we have a list of 40 weighted values. The landmarks, as we have discussed in Section 2.2, are a kind of states that every trace will visit, independent of how the traces differ from one another. When the code outputs all the lists of weighted values, we can look into the weight data and expect a distinguished value difference of these landmarks from the other normal states.

The resulting weight values are shown in Table 5.1. We show four digits after the decimal. We show them into two columns to have a better appearance. We also show the actual input trace in Table 5.2 (the operator trace, not the state trace), which is generated by the JSHOP program.

State 1 (Initial state)	12.2270	State 12	12.2270
State 2	12.2270	State 13	12.2270
State 3	12.6381	State 14	12.2770
State 4	12.6381	State 15	12.6396
State 5	12.6381	State 16	12.6396
State 6	12.6381	State 17	12.6396
State 7	13.0035	State 18	13.0042
State 8	13.0035	State 19	13.0042
State 9	12.2270	State 20	12.6396
State 10	12.2270	State 21	12.2270
State 11	12.2270	State22(final state)	12.2270

Table 5.1: The final output weight value list of the 1st trace calculated by our algorithms

المنارة للاستشارات

www.manaraa.com

State 1	load-truck pack1 t1 loc1	State 12	unload-rocket pack1 rocket1 lc2
State 2	drive-truck t1 loc1 ap1	State 13	load-truck pack1 t3 lc2
State 3	unload-truck pack1 t1 ap1	State 14	drive-truck t3 lc2 ap5
State 4	load-airplane pack1 plane1 ap1	State 15	unload-truck pack1 t3 ap5
State 5	fly-airplane plane1 ap1 ap3	State 16	load-airplane pack1 plane3 ap5
State 6	unload-airplane pack1 plane1 ap3	State 17	fly-airplane plane3 ap5 ap7
State 7	load-truck pack1 t2 ap3	State 18	unload-airplane pack1 plane3 ap7
State 8	drive-truck t2 ap3 lc1	State 19	load-truck pack1 t4 ap7
State 9	unload-truck pack1 t2 lc1	State 20	drive-truck t4 ap7 loc8
State 10	load-rocket pack1 rocket1 lc1	State 21	unload-truck pack1 t4 loc8
State 11	fly-rocket rocket1 lc1 lc2	State 22	

Table 5.2: The states associated with their operators in the 1st trace

From the result in Table 5.2, we find that the weight value 12.2270 is the minimum value in the first state trace. We use the first state in Table 5.2 as an example to analyze the weight value. In the first state of the 1st trace, we can say this state is a landmark because all traces will start from this state. This satisfy the definition of landmark, which is a state that is visited in every trace. The minimum value is expected because of the following two reasons: (1) in the similarity algorithm, we need to find a tunnel between the state runner and state seeker, which is the shortest distance from the current state to the other trace. As the similarity algorithm is calculating the distance between the first state in the first trace and other states from other traces, both the first states from two traces can be perfectly matched as they are the same. As a result, the similarity algorithm will consider the distance between every first state is 0. (2) in the weight algorithm, starting from the current state (the first state from the first trace), the tunnel found by the similarity algorithm is applying a distance of 0 to visit other states in other traces. So the sum value is 10542, and the number of visited states in this case is 862. The result is 12.2770 and this value is a landmark.

المنارة للاستشارات
www.manaraa.com

For the values 12.6381, 13.0035, 12.6396, and 13.0042, they can still be considered as landmarks for the following two reasons. We use state 3 in Table 5.1 as an example. (1) state 3 is generated by applying the adding list and deleting list of the operator: *drive-truck t1 loc1 ap1*. In this state, the truck 1 goes to airport 1, while it has another option, which is driving the truck to airport 2. So in the similarity algorithm, when we are using the first trace, in which the truck 1 goes to airport 1, and we are trying to find a similar state in other traces, we have two situations: (i) the truck 1 still choose airport 1 in the other trace, and (ii) the truck 1 chooses airport 2 to deliver the package. For the first situation, the two states from two traces can be perfectly matched, but for the second situation, two states are recognized as similar states, and the distance should be added by 1 because airport 1 and airport 2 are different airports but they are in the same city. (2) in the weight algorithm, the shortest distance from state 3 to other states is the tunnel between state 3 and similar states from other traces, so the sum of the distance will be greater than 10542. Even though the final value is greater than 12.2770, we still consider these states are important state because they are only having the similarity difference from 0 to 1, which means the state runner is actually the current state itself. They are still landmarks, but not as important as the ones which has the minimum value.

The first trace is a special trace because it includes all the important states in our domain. Tables 5.3 and 5.4 show another final result from the 8th trace in our input trace set.

State 1 (Initial stete)	12.2810	State 16	12.2810
State 2	12.2810	State 17	12.2810
State 3	13.0281	State 18	12.2810
State 4	12.6885	State 19	13.0298
State 5	12.6885	State 20	13.4693

المنارة للاستشارات
www.manaraa.com

State 6	12.8946	State 21	12.7131
State 7	12.8946	State 22	12.7131
State 8	12.8946	State 23	12.7131
State 9	13.6979	State 24	12.9215
State 10	12.8932	State 25	12.9215
State 11	13.7004	State 26	13.6892
State 12	13.7286	State 27	12.5741
State 13	12.2810	State 28	13.4603
State 14	12.2810	State 29	12.2810
State 15	12.2810	State30(final state)	12.2810

Table 5.3: The final output weight value list of the 8th trace calculated by our algorithms

State 1	load-truck pack1 t1 loc1	State 16	unload-rocket pack1 rocket1 lc2
State 2	drive-truck t1 loc1 loc2	State 17	load-truck pack1 t3 lc2
State 3	drive-truck t1 loc2 ap1	State 18	drive-truck t3 lc2 loc5
State 4	unload-truck pack1 t1 ap1	State 19	drive-truck t3 loc5 loc6
State 5	load-airplane pack1 plane1 ap1	State 20	drive-truck t3 loc6 ap5
State 6	fly-airplane plane1 ap1 ap4	State 21	unload-truck pack1 t3 ap5
State 7	unload-airplane pack1 plane1 ap4	State 22	load-airplane pack1 plane3 ap5
State 8	drive-truck t1 ap3 ap4	State 23	fly-airplane plane3 ap5 ap8
State 9	load-truck pack1 t2 ap4	State 24	unload-airplane pack1 plane3 ap8
State 10	drive-truck t2 ap4 loc3	State 25	drive-truck t4 ap7 ap8
State 11	drive-truck t2 loc3 loc4	State 26	load-truck pack1 t4 ap8
State 12	drive-truck t2 loc4 lc1	State 27	drive-truck t4 ap8 loc7
State 13	unload-truck pack1 t2 lc1	State 28	drive-truck t4 loc7 loc8
State 14	load-rocket pack1 rocket1 lc1	State 29	unload-truck pack1 t4 loc8
State 15	fly-rocket rocket1 lc1 lc2	State 30	

Table 5.4: The states associated with their operators in the 8th trace

From the Table 5.3, the minimum value is 12.2810. This matches our domain definition, which the package is only allowed to use rockets to be transported to another planet. The difference between the first trace and this eighth trace is that, in the 8th trace, additional states are added. The additional states mean several states occur in the 8th trace, but not in the other traces. For example, for the state: truck 1 is in location 2, truck 2 is in location 3 and location 4, etc. only occurs in the 8th trace. For these states that only occur in one trace, they are definitely not landmarks because they are single occurrence events. In the similarity algorithm, the additional state cannot find its matching or similar

المنارة للاستشارات
www.manaraa.com

states in other traces, so it has to visit the nearest tunnel to visit other traces. This adds up to the total distance from these additional states, and the averaged values must be higher than those states which are standing on the state runner. We can get a better understanding of the landmarks and those ordinary states by analyzing Figure 5.4 and Figure 5.5.

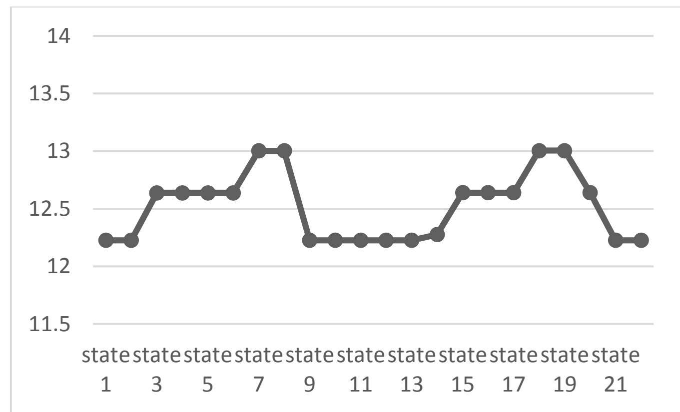

Figure 5.4: The weight value distribution in the 1st trace.

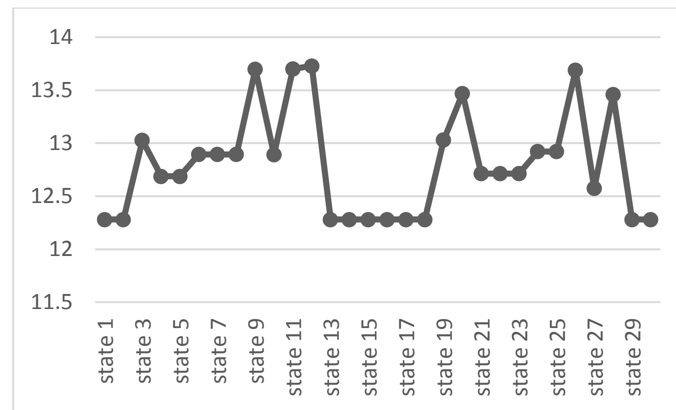

Figure 5.5: The weight value distribution in the 8th trace.

المنارة للاستشارات
www.manaraa.com

Figure 5.4 and Figure 5.5 show that the landmark has a low value. Independent of the final weight value we choose from the input trace set, the first few states and last few states always have the lowest weight values (and hence are considered landmarks because every trace starts from the same state and ends in the same state. Also, we noticed that in the middle part of each trace, there exists a series of landmarks: when (1) the package arrives at the launch center, (2) the package is loaded into the rocket, (3) the rocket is arrived at the planet 2, and (4) the package is located in launch center 2. So our algorithms can accurately find the highest landmarks (the ones that has the minimum values) by overlapping all the results together. The highest landmarks also match our previous expectations, which is that taking the rocket is the only way to deliver the package to the destination, if the destination is on another planet.

From Figure 5.5, we are also able to observe a distinguishing difference between ordinary states (the points which has highest values) and the points whose values are in the middle level in the figure. Compared to Figure 5.4, we see that the middle level points are having the same height with the higher value set of points in Figure 5.4. Knowing that the 1st trace has included all the important states and no other additional states, we can conclude that in the 8th trace (shown in Figure 5.4), all the states having a middle level values are also landmarks. These landmarks are not as important as the ones that are having the minimum value because the similarity distance is applied, the distance is more than 0.

المنارة للاستشارات
www.manaraa.com

5.2.2 Dota 2 Domain Results

In the Dota 2 domain, we use our weight and similarity algorithms to find important events in the Dota 2 game. The output format for each input trace is a list of time and location when the team fight happens. We use these lists of time and location information to trace back the events in the Dota 2 game replay mode. So the real replay mode can check if the landmarks the system finds correspond to important events in the game.

We have ten lists of locations and the time information. We organized them in Table 5.5. The lower row shows a lower landmark period, and the higher row shows a higher landmark period. The hero index is the index we will use in controlling the Dota 2 camera in the game. The "from" tick and "to" tick is the start and end time when team fights take place.

Trace #1			
	Hero Index	From (tick)	To (tick)
Lower	3	1184.33239746	1185.39880371
Higher	3	2560.29980469	2563.31005859
Trace #2			
	Hero Index	From (tick)	To (tick)
Lower	5	2034.3248291	2277.50317383
Higher	0	3469.5637207	3473.64428711
Trace #3			
	Hero Index	From (tick)	To (tick)
Lower	1	3289.58398438	3292.59423828
Higher	1	3948.49511719	3949.36474609
Trace #4			
	Hero Index	From (tick)	To (tick)
Lower	2	1576.96984863	1589.56677246
Higher	7	2231.47973633	2233.15209961
Trace #5			
	Hero Index	From (tick)	To (tick)
Lower	0	3254.49780273	3257.44116211
Higher	6	3272.02416992	3277.77709961
Trace #6			

المنارة للاستشارات
www.manaraa.com

	Hero Index	From (tick)	To (tick)
Lower	4	3065.72143555	3067.1262207
Higher	4	3404.64257812	3406.04736328
Trace #7			
	Hero Index	From (tick)	To (tick)
Lower	2	911.399047852	912.26550293
Higher	1	2377.61083984	2390.58837891
Trace #8			
	Hero Index	From (tick)	To (tick)
Lower	0	2152.51074219	2165.02001953
Higher	0	2475.81201172	2501.70019531
Trace #9			
	Hero Index	From (tick)	To (tick)
Lower	2	938.759033203	939.825439453
Higher	2	1134.17797852	1135.57763672
Trace #10			
	Hero Index	From (tick)	To (tick)
Lower	4	2223.95410156	2225.92749023
Higher	4	2668.1003418	2706.49780273

Table 5.5: The output result from Dota 2 domain

We then use the start time and end time from Table 5.5, put them into the script for controlling the camera in Dota 2 replay mode. We took 10 pairs of videos. Each pair of video records two periods in the game, including the lower landmark and the higher landmark respectively. In Dota 2 replay mode, the time in selecting heroes and pre-preparation is not counted into the time we have in Table 5.5. So we make some modifications on the replay mode. We skip to the beginning of the battle starts rather than beginning from the file is read. We compare the videos we record with the landmarks we found by checking whether the event happened at our setting time and location can show the importance in the whole game.

المنارة للاستشارات
www.manaraa.com

Figure 5.6: A team fight extracted by our algorithm. This team fight has a lower landmark.

Figure 5.6 and Figure 5.7 show team fights that were found by our similarity and weight algorithm, and also control the camera in a good view point. The time where the team fight happens may have some bias so we picked a proper time in the game to start recording the videos. The start time calculated by our algorithms can still help us find when and where this team fight happens while a 5% bias is occurring in real-time.

المنارة للاستشارات
www.manaraa.com

Figure 5.7: A team fight extracted by our algorithm. This team fight has a higher landmark.

The mini map is a useful tool in Dota 2 which can help players and reviewers have a straightforward view on the whole game at that time. The mini map is located in the left-bottom side of the game screen. We can obtain heroes' locations from the mini map, and we can predict the trends of a game by checking how many defensive towers are destroyed by the enemies. When we look at Figure 5.6 and Figure 5.7, from their mini maps, we know that for Figure 5.6, the team fight is happening in the upper lane of the middle map and many heroes are participating in this team fight. For Figure 5.7, we see that the team fight is in dire's base, very close to dire's ancients. After viewing both videos, we can predict that on Figure 5.7, the team fight which has a higher landmark and the radiant's team will win this game. In Figure 5.6, we cannot tell which team will win because this video does not provide enough information. This means the pairs of videos

المنارة للاستشارات
www.manaraa.com

we take can match the importance of the landmarks we extracted from our weight and similarity algorithms. For the other 9 pairs of videos, no matter which team (radiant or dire) wins the game, the team fight will be more informative when it happens near, or inside, their opponents' base (ancients). We are more likely to predict who will win from those team fights which have higher level landmarks (i.e., lower weighted values in our algorithm).

5.2.3 Amazon Turk Results

We posted our experiment on Amazon mechanical turk website. The experiment name was "Dota 2 MOBA Camera View Experiment". The experiment we completed in a very short time; it was completed within 8 hours. We have 90 valid records submitted from turkers. A turker took, on average, 12 minutes 59 seconds to finish our experiment, which falls within our allocated time of 30 minutes.

We divide 90 participants into 3 groups: the expert group, the intermediate group, and the novice group. For the expert group, a turker must answer at most one qualification questions incorrectly. For the intermediate group, a turker must answer exactly 2 questions correctly. And for the novice group, there are two cases, one is a turker answers at most one question correctly, the other is his or her answers is not making any sense. Figure 5.8 shows the resulting grouping.

المنارة للاستشارات
www.manaraa.com

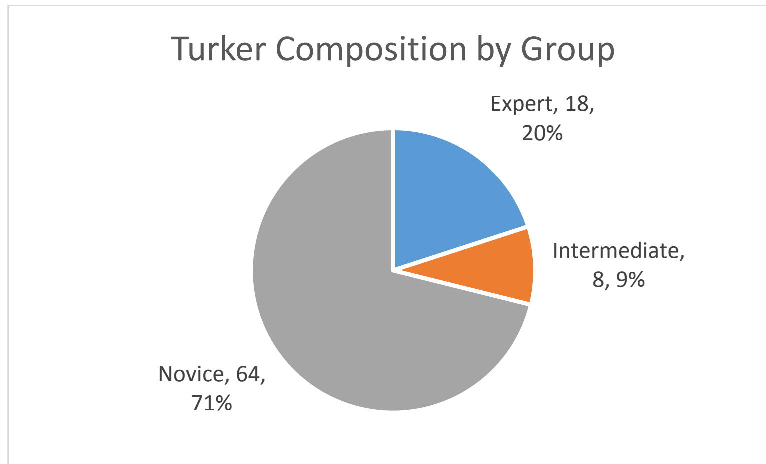

Figure 5.8: The composition of participants in each group

We will focus on the expert group and the intermediate group because only these two groups of participants have the needed qualifications for our analysis. Only the person who answers at least two questions correctly can be recognized as an experienced player in the Dota 2 game. Their test answers are analyzed in Table 5.6.

Pair 1	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	7	6	5	3	5	10
Expert+Interm	9	9	7	3	8	14
Pair 2	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	8	6	4	3	7	8
Expert+Interm	12	8	5	5	10	10
Pair 3	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	11	2	5	5	4	9
Expert+Interm	15	3	7	6	8	11
Pair 4	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	14	4	0	5	3	10
Expert+Interm	17	6	2	7	6	12
Pair 5	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	15	0	3	9	4	5
Expert+Interm	19	2	4	12	5	8

المنارة للاستشارات
www.manaraa.com

Pair 6	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	13	2	3	3	4	11
Expert+Interm	17	4	4	6	4	15
Pair 7	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	7	8	3	5	4	9
Expert+Interm	10	11	4	5	8	12
Pair 8	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	4	12	2	1	9	8
Expert+Interm	7	14	4	4	9	12
Pair 9	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	8	7	3	3	4	11
Expert+Interm	10	9	6	3	5	17
Pair 10	Question 1 (more informative)			Question 2 (more entertaining)		
	Video 1	Video 2	Same	Video 1	Video 2	Same
Expert	7	6	5	2	4	12
Expert+Interm	7	8	10	2	5	18

Table 5.6: Amazon mechanical turk experiment result. (Expert and expert+ intermediate)

Pair 3, Pair 5, Pair 6, Pair 7, and Pair 9 are classified as the 'matching' results among the 10 pairs. In these pairs, the videos which are having higher landmarks (this means it is more likely to happen in the games and more important than the other one) are having in common that one team is going to win the game. Compared to the other video in the pair, the other one cannot provide enough information that make it a meaningful team fight because it is not clear who will win.

It is more common that the participants will choose higher landmark video when the situation is very clear. For example, participants will more likely choose the correct video (the one with the higher landmark) when one lane's towers and barriers are all cleared off, which means this team will be able to attack enemies' base which has a high winning possibility. Commonly, people are more willing to decide that one side will win

المنارة للاستشارات
www.manaraa.com

by comparing the situations of both team. If the attacking team wins the team fight when they are attacking their enemies' base, a majority of the participants will choose the correct answer. In Pair 3, Pair 5, Pair 6, the percentage of choosing the desired video is higher than the other two (the other video, or about the same). In Pair 7, even though the dire's team will win, video 2's people is only a little more than video 1, but it can still indicate that the dire's team will win in the game.

Participants will not choose a higher landmark video when the situation is not clear enough. In Pair 9, even though the radiant's team is trying to attack dire's base, the result does not tell much difference between two videos; video 1's percentage is marginally higher than video 2. This is because dire's tower and barriers are not destroyed totally in one lane, which means which team will win is still unknown in the higher landmark video.

Pair 2, Pair 4, Pair 8 are classified as the 'not matching' results among 10 pairs. All these videos have distinguished difference between video 1 and video 2. This means that the results are conclusive. The quality of the game, which might be the key point in this 'not match' class, is the reason for this result. The quality of the game consists of many elements. For example, in Pair 4, one hero is not killed in the team fight when facing with four enemies, although the HP of that hero is really low but nevertheless survived. Then with the assistance with his/her teammates, they won the team fight in the end. This team fight is a good one to say it is very informative, and entertaining, but it it is not a landmark. Also in Pair 8, the team fight efficiency is very high, which means the radiant's team ended up the team fight by killing all the enemies very quickly without much difficulty. But it is not the video that has higher landmark. Participants' decisions

المنارة للاستشارات
www.manaraa.com

are made by the ‘more informative’ and ‘more entertaining’ elements, so whether a team fight is considered exiting made a difference here.

Also, all these three videos pairs are not indicative of which team will win in the end. The participants cannot observe any lane’s towers or barriers been destroyed.

Pair 1, Pair 10 are classified as the ‘no much difference’ among the 10 pairs. For Pair 1, ignoring the mini map in the left bottom corner of the screen, the participants may have difficulty in assessing the importance of the game situation by only looking at the team fight. The mini map is a very useful tool to have a view of the buildings and location of each heroes in the game. We did not notify the participants to consider the mini map and this may have led to it not been considered in the first video pairs.

For Pair 10, there is no much difference in the selection of the two videos. Even though it is very clear that video 2, which has higher landmark may be the one that is mostly chosen, the result shows the answer is mostly selected as ‘about the same’. Some participants did not finish the experiment because they left the question of last pair videos blank. The data collected then is not as accurate as expected.

5.3: Discussion

Based on the analysis of the result we get in our new transportation domain, we found that the true landmarks can be identified by our similarity and weight algorithm designed for the transportation domain. The result shows that the most important landmarks have the minimum value among all the states in one trace as we had predicted. These minimum values vary in different traces, but the region where the minimum values show up overlaps in all the traces. The highest, or the most important landmarks are

المنارة للاستشارات
www.manaraa.com

highlighted as the ones that have the minimum values. We also found that the lower-score landmarks (i.e., not true landmarks) can be found by the values with greater weights and they tend to be uniformity in a small range of values. These lower-score landmarks have smaller values than the states that are not considered as landmarks. From the experimental results in Section 5.2.1, we noticed that the difference between normal states and lower level landmarks is not large. In our similarity and weight algorithms, the difference should be more obvious because the normal states need to add additional distance to visit the tunnel bridged between two traces. We may have a more distinguished difference between normal states and the lower landmark states, if we add more locations to visit. Because in our scenarios, we only have two locations in each city, which means the truck will have only 2 choices to go from one location to one airport: (1) to directly go to the airport, and (2) to visit another location and then going to the airport. If we have 3 locations in each city, for a single activity: drive the truck, we can have 5 possibilities. One is directly going to the airport, two are visiting only one location, and two are visiting two other locations. So if the number of locations increases, for those normal states, they will require more distance to visit the nearest tunnel to reach to the other trace, and this will increase the final weighted values of normal states which we can have a more distinguished comparison between normal states and lower landmarks states.

For the experiment on Amazon mechanical turk, we discovered that the participants made decisions very clearly based on the situations that the videos show in that pair. For the "matching" class, the choices of "which video is more informative" question is quite clear. Participants are able to tell which team will win the game if the higher landmark video happens in one of the team's ancients. This means the attacking

المنارة للاستشارات
www.manaraa.com

team has destroyed all the defensive towers in at least one lane and will try to attack their enemies' ancients. Also, for the "not matching" class, the participants cannot tell which team will win the game when both videos in one pair do not show a clear situation in the game. From the experimental results, we noticed that participants will mostly vote to the video which has more exiting events in the team fight itself, rather than considering the importance of the team fight to the whole game.

We found that the mini map provides a good view on the whole situation of the game during the time the video is taken. Participants can have a very straightforward view of which lane of defense is cleared, which team is approaching their enemies' ancients, and where is the team fight happening. We also noticed that among the pairs after pair 2, the participants are making very clear choices among the videos. In contrast for the first two videos, the choice distribution is quite even, which means people are not making a very clear decision on which video to choose. We believe that the reason for this is because we did not mention the participants to consider the mini map, and suggested that viewing the mini map as an important element to decide which video is more informative and which video is more entertaining to watch.

For the pairs that their results are not matching our expectations, we speculate that participants may be more likely to choose the higher landmark video if they are given a little longer video. For example, if we extend the video for pair 2, 4, and 8 from three to five minutes, participants might be more likely to choose the ones that match our expectations. Also the results accuracy will be enhanced if we can have more valid Dota 2 experts in our experiment.

المنارة للاستشارات
www.manaraa.com

6. Final Remarks

6.1: Conclusions

In this thesis we designed a method to find important events, which is also called landmarks in a new transportation domain and the Dota 2 game domain. This method is realized by our similarity algorithm and our weight algorithm. We present the general calculating procedure of these two algorithms and implement them into both domains we have in our thesis. For the similarity algorithm, it is capable of searching for matching or similar states from two different input traces. The similarity extent is returned as a distance value based on how similar these two states are. Tunnels are bridged between two traces if the similarity value satisfies our requirements. For the weight algorithm, a value of an averaged distance from one state to all the other states on other traces is calculated, and become a measurement of how important this state is in the whole trace of states. The method we use in this thesis can present an automatic and straightforward implementation in helping people to find important events in a particular domain. And no matter how the domain changes, the similarity and weight algorithms are the general idea in realizing searching for important events, also called landmarks, in different domains.

The experiments are taken to test the realization of the weight and the similarity algorithms in both new transportation domain and the Dota 2 domain. For the new transportation domain, these two algorithms work well and the landmarks are correctly found. The highest landmarks, which are remarked as the states that have minimum weighted values, and the landmarks that every trace will visit. The lower landmark states are also found based on the greater values compared to the highest landmarks. The ordinary states have highest calculated values and it shows the difference with the higher

المنارة للاستشارات
www.manaraa.com

or lower landmarks clearly. We also find some flaws in our transportation domain because we discover the difference between ordinary states and landmarks is not so distinguished as we expect, and we propose to increase the number of locations to give more trace vary possibilities in our domain.

Our algorithms can successfully find important events, also called landmarks in the Dota 2 domain. The algorithms can also locate the time and the locations where team fights happen. We put the landmarks back into the Dota 2 replay mode, direct the camera to the time and place based on the results calculated by our algorithms. Although the camera has some bias in the start time and end time of the team fight which the landmark data refers to, it still can match the team fight correctly.

The results from Amazon mechanical turk show that most turkers can correctly choose a video which has a higher landmark (more important) when the situation of the game is quite clear. Turkers have no difficulty to choose a video in which one team is attacking their enemy team's ancients, which means the attacking team will win the game. They cannot make uniformed answer when both videos in a pair do not have a clear view of the team fight that happens near the ancients. The mini map at the corner of the screen is also a very good tool to view the situations during the video period. So the participants may forgot to view the mini map and cannot make correct decisions. The problems in this experiment can be concluded in the following aspects: (1) the duration of the video which has higher landmarks but does not show the situation as one team is having the team fight near the ancients will be extended to have a longer length, so the experiment participants can have a better understanding of the game situation and make correct decisions. (2) the mini map should be mentioned before the experiment

المنارة للاستشارات
www.manaraa.com

participants begin to take the experiment. This can let them have a better view point on the game process. (3) the number of valid Dota 2 game expert player should be enlarged. Based on the qualified participants we have in our Amazon mechanical turk experiment, the lack of valid feedbacks give us a fluctuation on the analysis of our conclusion.

6.2: Future Work

The Amazon turk experiment should be improved. This is a good implementation in checking the landmarks we find by our algorithms can be accepted and understood by most of the Dota 2 game expert players. First, we need to prepare a more simplified but efficient pre-test questionnaire, to check whether the experiment participant can be qualified into our expert or intermediate group. We can also enlarge the number of people which can participant in our experiment. Secondly, we are going to have a new combination of the pairs of videos in our experiment. One is the video in pair 1 has a lower landmark, and the other has higher landmarks. The video which has higher landmarks is taken place in one team's base, very close to the ancients. Another pair of the videos is the video in pair 2 has a lower landmark, and the other has higher landmarks. But both of these videos are not taken place near the ancients, which is used to compare to pair 1. In pair 3, we still use the pairs of video in pair 2, but the duration of the video which has higher landmarks may be extended to be more than 1 minute. This can allow us make a comparison to pair 2. Our expectation of participants' choices is in pair 1, most people choose the video which has higher landmarks. For pair 2, the percentage of choosing video 1, video 2, and same option may be equal. And for pair 3, most of people will choose the video which has higher landmarks again.

المنارة للاستشارات
www.manaraa.com

Bibliography

McGovern, Amy, and Andrew G. Barto. "Automatic discovery of subgoals in reinforcement learning using diverse density." *Computer Science Department Faculty Publication Series* (2001): 8.

Yang, Qiang, Kangheng Wu, and Yunfei Jiang. "Learning Actions Models from Plan Examples with Incomplete Knowledge." *ICAPS*. 2005.

Elkawkagy, Mohamed, Bernd Schattenberg, and Susanne Biundo. "Landmarks in Hierarchical Planning." *ECAI*. 2010.

Porteous, Julie, Laura Sebastia, and Jörg Hoffmann. "On the extraction, ordering, and usage of landmarks in planning." *Sixth European Conference on Planning*. 2014.

Nau, Dana, et al. "SHOP: Simple hierarchical ordered planner." *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc., 1999.

Hogg, Chad, Héctor Munoz-Avila, and Ugur Kuter. "HTN-MAKER: Learning HTNs with Minimal Additional Knowledge Engineering Required." *AAAI*. 2008.

Hogg, Chad, Ugur Kuter, and Héctor Muñoz-Avila. "Learning Hierarchical Task Networks for Nondeterministic Planning Domains." *IJCAI*. 2009.

المنارة للاستشارات
www.manaraa.com

Vita

Jundong Yao was born in Hangzhou, Zhejiang Province, China. After completing his schoolwork at Hangzhou Xuejun High School in Hangzhou in 2009, Jundong entered Huazhong University of Science and Technology in Wuhan, China. He received a Bachelor of Engineering degree with a major in Automation from Huazhong University of Science and Technology in June 2013. During the following two years, he entered the Graduate School of Lehigh University in Bethlehem, PA.

المنارة للاستشارات
www.manaraa.com