Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 834616, 6 pages
doi:10.1155/2008/834616

Research Article

Strategic Team AI Path Plans: Probabilistic Pathfinding

Tng C. H. John,[1] Edmond C. Prakash,[2] and Narendra S. Chaudhari[1]

[1] *School of Computer Engineering, Nanyang Technological University, Singapore 639798*
[2] *Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M1 5GD, UK*

Correspondence should be addressed to Edmond C. Prakash, e.prakash@mmu.ac.uk

Received 29 September 2007; Accepted 13 December 2007

Recommended by Kok Wai Wong

This paper proposes a novel method to generate strategic team AI pathfinding plans for computer games and simulations using probabilistic pathfinding. This method is inspired by genetic algorithms (Russell and Norvig, 2002), in that, a fitness function is used to test the quality of the path plans. The method generates high-quality path plans by eliminating the low-quality ones. The path plans are generated by probabilistic pathfinding, and the elimination is done by a fitness test of the path plans. This path plan generation method has the ability to generate variation or different high-quality paths, which is desired for games to increase replay values. This work is an extension of our earlier work on team AI: probabilistic pathfinding (John et al., 2006). We explore ways to combine probabilistic pathfinding and genetic algorithm to create a new method to generate strategic team AI pathfinding plans.

1. INTRODUCTION

A popular game with heavy team AI is "Full Spectrum Warrior" (FSW), a console game on Xbox. This game is a downsized version of "Full Spectrum Command" on the PC platform. The game "Full Spectrum Command" is actually a simulation of the real-world behavior of the US Army. This game was originally used in the military for leadership training as well as decision making training. The game includes real-world army movements such as "bounding" and ducking. The main feature of the game is its team AI. The team AI of the game is actually derived from the real world and simulates how a real person will behave. The purpose of this game is to command two teams of soldiers to accomplish a mission.

Even though it is a great game, one area it can improve on is the team AI plan. Players may feel that opponents always appear at the same places after playing repeatedly, as most team AI plans use A* algorithm or look up tables as their main pathfinding techniques. These techniques always produce the same path if the source and destination locations are the same. On the other hand, Probabilistic is able to produce variations to the path even if the source and destination locations are the same.

Replay value can easily be added to the game by creating variations to the opponent team plans. When the opponents have different plans, they move differently, thus the players cannot always predict the opponents' locations. This work uses probabilistic pathfinding algorithm to obtain variations of team AI path plans.

Section 2 talks about related work, existing problem, and a scenario that strategic team AI path planning can be applied. Section 3 gives an introduction about team AI and probabilistic pathfinding. It explains how team AI is created in [1] by combining different systems. Section 4 describes the main method of strategic team AI path plan generation. Section 5 suggests a method that can be used to optimize this work. Section 6 shows some interesting strategies generated by this team AI path plan generation method. The final section, Section 7, comprises conclusion, other applications, and future suggestions for this work.

2. RELATED WORK

In this paper, team AI pathfinding and team AI plans are the same. Team AI pathfinding refers to the different paths taken by teammates to reach a desired place. An example is a team of enemies enter a room from different doors to trap and capture the player. Team AI pathfinding plans are popular in computer games. With good team plan, the game difficulty level increases, making the game more challenging and helps to showcase intelligent behavior of the game.

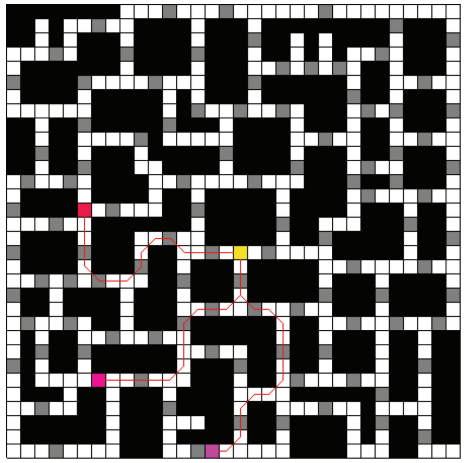

Figure 1: Situation without team AI.

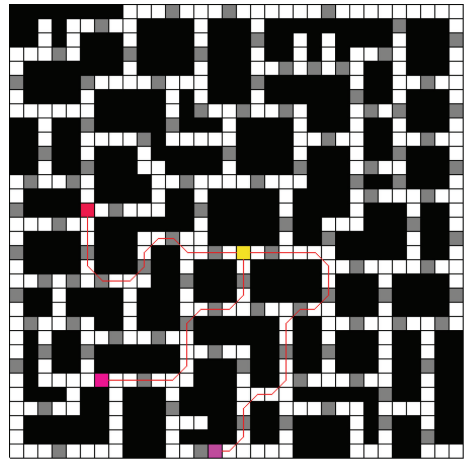

Figure 2: Situation with team AI.

In this work, we use and extend our earlier work on probabilistic A* pathfinding algorithm [1]. Further readings on A* pathfinding can be found in [2–4]. Bourg and Seeman [4] provide other data representation of the A* pathfinding. A very useful article by Pinter [5] discusses methods to modify a raw path generated from a path search to form a more convincing traveling path.

An interesting work by Kamphuis et al. [6] attempts to simulate tactical pathfinding in urban environments for a small group of characters for games and simulations. The characters use tactical information such as road maps and special locations for pathfinding. This work uses common A* data structure and a list of gateway points. The tactical pathfinding [6] algorithm is able to run in real time with the help of a preprocessing step. For this work to run in real time, no preprocessing is needed. With optimization, it can even run more efficiently.

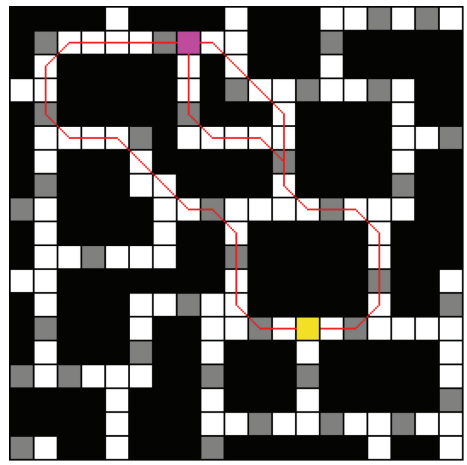

Figure 3: Different paths generated by probabilistic pathfinding.

3. PREVIOUS WORK

This section is an introduction to team AI: Probabilistic pathfinding. The detail implementation and algorithm can be found in [1]. Team AI can be shown to exist if teammates coordinate to trap or capture an opposite team. Figure 1 shows a situation when team AI is not enabled. The enemy teammates find the shortest path to capture the player.

Figure 2 shows a situation with team AI enabled. The enemy teammates surround, trap, and capture the player.

Probabilistic is a modified version of A* algorithm with an addition ability to generate different paths controlled by a probability variable. The variable controls how different the paths differ from the shortest path. The paths may not be the shortest, but they are one of the shorter paths. In general, if the variable is 0, then probabilistic pathfinding will behave exactly the same as a usual A* pathfinding. If the variable is set to 1, it will always produce a different path that is not the shortest. For more details refer to [1]. Figure 3 below shows an example of an enemy character following different paths to pursuit the player character.

Notice that there are light grey squares in Figures 1, 2, and 3. They are actually the gateways of the map. Gateway is narrow path in the environment that opens up to a bigger path before and after the gateway. That is, a gateway is a narrow link between two spaces. Figure 4 illustrates a gateway.

A blackboard messaging system is developed in [1] to facilitate communications between characters of a team. In short, blackboard is a place for characters to "write" useful information and let other teammates read it. After every teammate read, the message is deleted by the message writer. Figure 5 shows the concept of a blackboard.

Putting together probabilistic pathfinding, gateways information, and message system, we achieved what is shown in Figure 2, the complete working team AI in [1].

4. PATH PLAN GENERATION

In this section, we illustrate the method used to generate high quality team AI path plans. The main idea is to test the team AI path plans with a fitness function. The fitness will

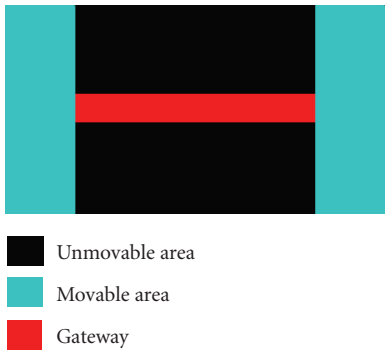

Figure 4: Illustration of a gateway.

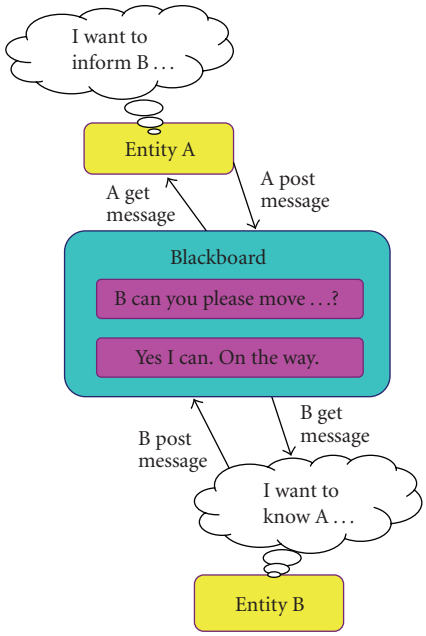

Figure 5: Different paths generated by probabilistic pathfinding.

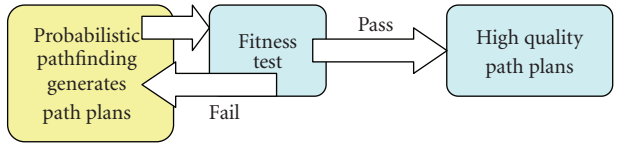

Figure 6: Path plan generation flow chart.

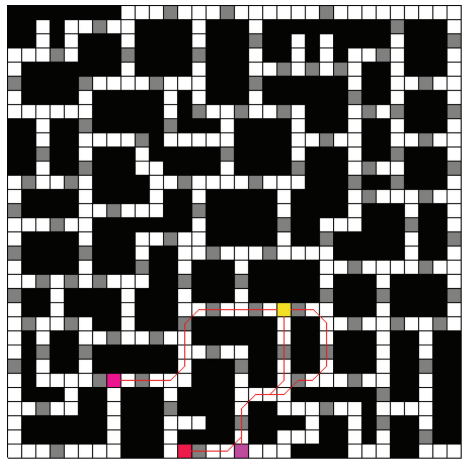

Figure 7: Path with overlapping regions.

determine whether the team plan is good enough. If the quality is bad, a new team AI path plan search will be conducted. Figure 6 shows the flow chart for path plan generation.

This idea is mainly inspired by genetic algorithms [7], where a fitness function is used to test the quality of the genes combination. The genes combination is formed based on its parents, some manipulation, and some randomness. The better the quality of the combination of genes is, the higher chance it will survive. The bad quality gene combinations get eliminated. Team AI path plan works the same way. Different path plans are generated by probabilistic pathfinding. Treat each path plan as a combination of genes. A fitness function is used to test path plan. If it is not good enough, it will be eliminated. A new search will be conducted. The cycle repeats until a satisfactory quality path plan is obtained.

A fitness test can be a simple function that calculates the distance between two characters. For example in a game, it is not acceptable for two teammates to get too near to each other or their paths overlap. An alternative could be for the team to explore a bigger area of the map to gain resources and familiarize with the terrain. It all depends on the game play. Therefore, the fitness test for such a path plan will fail if two teammates get too close to each other. Fitness test can also be constraints of the team path plan.

Figure 7 shows a team path plans that are not acceptable because of overlap paths.

The path of the bottom-right enemy character follows the shortest path to the player. With team AI enabled, that bottom two characters have to trap the player through different entrances. However, due to overlap paths constraint, the path plan is discarded. A new path plan is conducted and shown in Figure 8. By comparing the plan shown in Figure 7 and Figure 8, the team path plan in Figure 8 is better than that of Figure 7 according to overlap constrains.

The following are three ways to test the team paths with the fitness function. They are illustrated below.

4.1. *Iterative test*

The fitness test is conducted after the whole team found its path. If the fitness test fails, a new path plan search will be conducted. However, the old path plan is saved. This is to prevent the system from doing too many searches and slow down the game. The user can specify a number of maximum searches to perform. If the maximum number of searches is reached, the path plan with the highest fitness will be selected. The user may also choose to terminate the

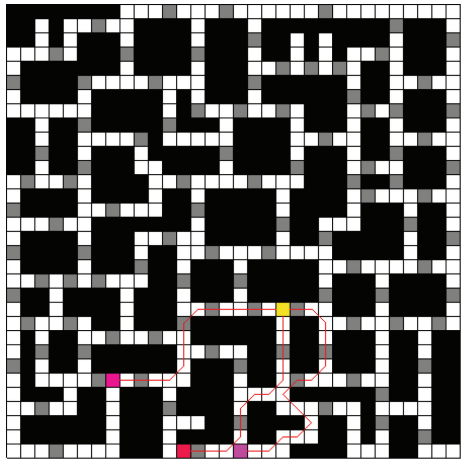

Figure 8: Path without overlap.

```
Path QualityPathPlanForWholeTeam() {
    Path pathOfWholeTeam;
    do {
        pathOfWholeTeam = null;
        While(TeamPathPlanNotComplete()) {
            pathOfWholeTeam + =
                ProbabilisticPathForOne();
            }
    } while(FitnessTest(pathOfWholeTeam)==fail);
  Return pathOfWholeTeam;
}
```

Algorithm 1

search once a path plan passed the first fitness test. This method is efficient if the fitness test seldom fails on path plans. This method is the easiest and fastest to implement. No modification is needed for probabilistic pathfinding generation algorithm. No modification is needed for team path planning. The exact algorithm is already shown in Figure 6. Algorithm 1 shows the pseudo code algorithm.

4.2. Step test

Algorithm 2 shows the pseudo code of the step test.

The fitness test is conducted at every segment of the path. This is the extreme opposite end of iterative test. On selection of the next node (using probability pathfinding search), if it does not pass the fitness test, another node will be chosen instead. This method is good if iterative test always fails and the number of characters is small. As opposed to iterative test, modification is needed for probabilistic pathfinding generation algorithm. The fitness test function has to be included into the probabilistic pathfinding algorithm.

4.3. Progressive test

The test will be conducted after each character found its path. This is a middle solution between the iterative test

```
Path QualityPathPlanForWholeTeam () {
    Path pathOfWholeTeamSoFar = null;
    While (TeamPathPlanNotComplete ()) {
            pathOfWholeTeamSoFar + =
    ProbabilisticPathForOne (pathOfWholeTeamSoFar);
      }
      Return pathOfWholeTeamSoFar;
}
Path ProbabilisticPathForOne (Path
pathOfWholeTeamSoFar) {
      Path currentMemberPath = null;
      Path temp = null;
      do {
            currentMemberPath += selectANextNode ();
            do {
                  temp = pathOfWholeTeamSoFar +
currentMemberPath.selectADifferentNode ();
            } while (FitnessTest(temp) == fail);
    } while
(currentMemberPath.SearchNotComplete ())
    return currentMemberPath;
}
```

Algorithm 2

```
Path QualityPathPlanForWholeTeam () {
    Path pathOfWholeTeamSoFar = null;
    Path temp = null;
    Path memberPath = null;
    do {
        do {
            memberPath = ProbabilisticPathForOne ();
          temp = pathOfWholeTeamSoFar + memberPath;
          } while (FitnessTest(temp) == fail)
          pathOfWholeTeamSoFar += memberPath;
      } while (TeamPathPlanNotComplete ());
      Return pathOfWholeTeamSoFar;
}
```

Algorithm 3

and the step test Table 1. It is based on each character. After each character has found its path, the fitness test will be performed. If the test fails, the character will choose another path. This is the best method if the iterative test and step test always fail. Modification needs to be made to team path planning. Fitness test is conducted per character (see Algorithm 3).

5. OPTIMIZATION

Fitness testing should be cheap if it does not involve calculation of huge set of constraints and variables. The load of this system comes from A* pathfinding. This means that all optimization techniques applicable to A* pathfinding are here. A common optimization technique for A* pathfinding

Table 1: Summary fitness function test.

Methods	Advantage	Best for cases
Iterative test	Easiest and fastest to implement	Many high quality solutions Fitness test mostly passes
Progressive test	A general solution that can solve most cases	Average solution In the middle of iterative test and step test
Step test	Guaranteed to have a solution if it exists	Few high quality solutions Fitness test mostly fails

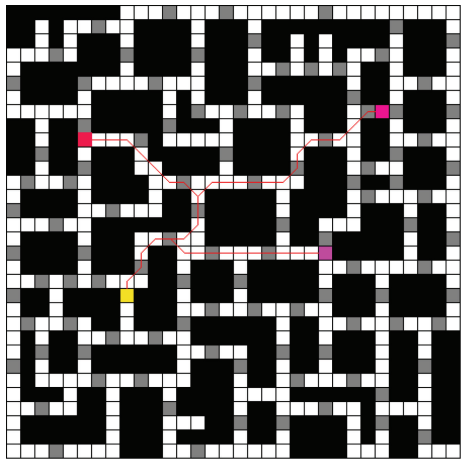

Figure 9: Combine force strategy.

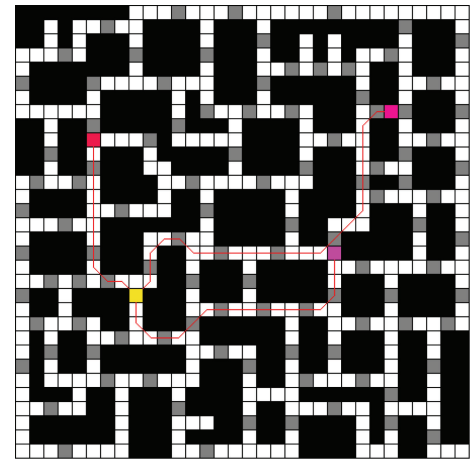

Figure 10: Trap strategy.

is search by parts so that the A* search execution is spread out over many frames.

6. ANALYSIS OF STRATEGIES

One useful feature of this team AI path plan generation method is to generate various useful strategies. These strategies can be applied to first person shooting team games as AI opponents or enemies. With such interesting strategies, the replay value of the game increases. The difficulty level increases and it becomes more challenging for player to defeat the enemies. This section analyzes interesting strategies generated.

Figure 9 shows the combined force strategy. In this particular strategy, it is the reverse. Joining teammates together synergized the power of the enemy team and increased the chance of success for eliminating the player. From Figure 9, the enemies join forces along the way and attack the player together in a single path. The fitness function to such a plan is to test the path before the destination (the player position) and ensure that before the destination all three teammates must be together. This is the first constraint. This fitness function will eliminate plans that do not combine forces before they encounter the player. The second constraint controls can be how early the teammates must combine their forces before they encounter. The earlier they meet the higher chance of success in their mission. For this example, the second constrained fitness function is not tested. As long they are able to meet before encountering the player, it is a good strategy.

Figure 10 shows a trap strategy. It is the reverse strategy of Figure 9. In this case, the team may have higher chance of success for killing the player. The trap strategy aims to trap the player at different directions. As far as possible, this means that the enemy teammates should not have overlap paths. So there are two constrained fitness functions for this example. The first constraint means, as far as possible, that the teammates must not encounter the player in the same direction. In general, use the pigeonhole theorem [8]. Number of teammates in same direction cannot be greater than teammates divided by number of directions. This is to ensure equal distribution of teammates in each direction. The second constraint is to avoid crossing paths of the teammates. This is to create higher chance of search space if the player escapes somewhere else.

Figure 11 shows a similar strategy as in Figure 10. However, a third fitness constraint is applied. The third constraint is the minimum distance away from the path of teammates. This is the *explore and trap strategy* . No doubt, the main objective of the teammates is to trap and capture the player in different directions and paths. In addition, the enemy teammates have a secondary objective to explore a wider area of the map. This will facilitate their future plans, actions, or operations. From Figures 10 and 11, with an additional fitness function, the path generated is different. A secondary objective can be included with additional constraints.

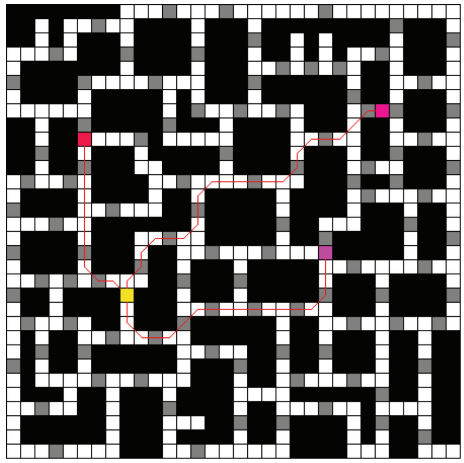

Figure 11: Explore and trap strategy.

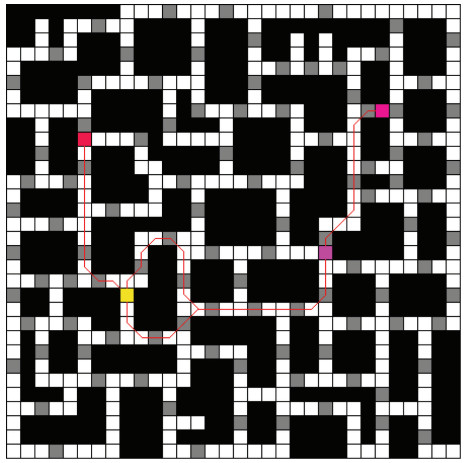

Figure 12: Combine and split trap strategy.

Figure 12 shows the most interesting strategy. One of the teammates follows the path of another teammate. To make analysis easier, the enemy teammate right of the player is known as $E1$ and the enemy teammate on top of $E1$ will be $E2$. In this situation, the map can be full of land mines. For every grid path that a character moves, it needs to remove all the landmines and make sure it is safe to travel before it can proceed to the next grid. $E1$ is ahead of $E2$ to the player. $E2$ tries to follow the path of $E1$ because in that case, $E2$ does not need to waste effort removing all landmines. This is the first constrained fitness function, which is try to overlap paths if they are along the way.

This is also a trap strategy because all teammates trap the player from different directions. This should be the second constraints. That is, all teammates should try to trap the player in different directions.

Using these two constrained fitness functions, a very nice strategy is generated from the team AI path plan method. The teammates know what they are doing. They work together and save effort for removing landmines. When they are near to the player, they split their ways to trap the player.

7. CONCLUSION

This paper proposes a new method inspired by genetic algorithm to generate interesting and high quality path plans for team AI. Probabilistic pathfinding is used for path search and blackboard architecture is used for communication between teammates. The path generation algorithm runs in real time without any preprocessing. Only the standard graph data and a list of gateway points are needed for the A* search. Controllable randomness allows the path generation to be tuned easily. The dynamic generation of path plans adds replay value to games. In addition, interesting path plans generated from this method are able to showcase the AI intelligences of the game which is a good selling point for games.

This strategic path plan generation method can apply to other applications that involve path searching. It is best for applications that have many solutions. A good example is traffic control system or GPS system. Such systems can plan the path of vehicles to avoid congestion. Congestion condition is used as constraints for fitness test to fail. For example, a congestion condition can be that the number of vehicle traveling along a road must be less than a maximum number. Another good application is goal planning with many different ways of achieving the goals. This path plan generation method can generate good quality path plans to achieve the goals. A real example is a mission-based game where there are many ways to solve a mission. Choosing a good plan to solve the mission can bring out the intelligence of game characters.

The path plan generated using this method is by trial and error. Generate a path, test it, and discard it if it is not good enough. A future step to go from here is to generate path plans by functions or heuristic. An example is to add a fitness function as a heuristic function to the probability pathfinding algorithm. With the fitness heuristics function, the path plans generated will always be good. This will prevent all the wasteful discards of low quality path plans.

REFERENCES

[1] T. C. H. John, E. C. Prakash, and N. S. Chaudhari, "Team AI: probabilistic pathfinding," in *Proceedings of the International Conference on Game Research and Development*, vol. 223 of *ACM International Conference Proceeding*, pp. 191–198, Perth, Australia, December 2006.

[2] S. Rabin, *AI Game Programming Wisdom 2*, Charles River Media, Hingham, Mass, USA, 2004.

[3] M. Buckland, *Programming Game AI by Example*, Wordware, Plano, Tex, USA, 2005.

[4] D. M. Bourg and G. Seeman, *AI for Game Developers*, O'Reilly, Sebastopol, Calif, USA, 2004.

[5] M. Pinter, "Gamasutra," http://www.gamasutra.com/features/20010314/pinter_01.htm.

[6] A. Kamphuis, M. Rook, and M. H. Overmars, "Tactical path finding in urban environments," 2005, http://www.cs.uu.nl/centers/give/movie/index.php.

[7] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ, USA, 2002.

[8] D. B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2000.