Deterministic Rendezvous Problem

COMP 5703 Seminar Project
Adam Bennett

Abstract

The purpose of this paper is to describe the Deterministic Rendezvous Problem and to explore the basic ideas of the algorithm presented in 2014 by Ta-Shma and Zwick which solves it.

Introduction

The Deterministic Rendezvous Problem is a variant of the more general *Rendezvous Problem*. The Rendezvous Problem is a problem in game theory where two robots are in a graph and must find one another. The robots are considered to have found one another if they both occupy the same node of the graph at the same time. [5]

One of the original examples of the Rendezvous Problem is the Astronaut Problem. In it, two astronauts land in different places on a celestial body which is much larger than the range of the equipment they use to detect each other. The astronauts need some kind of strategy to find each other. The Rendezvous Problem has applications which include search and rescue, communications, and networks and operating systems. [2]

The Deterministic Rendezvous Problem adds the restriction that the instructions given to the two robots must be deterministic; they cannot include any instructions which are executed based on probabilities.

More formally, in the Deterministic Rendezvous Problem, a pair of robots are placed in an unknown, finite, connected, undirected graph. Each robot knows:

- *T*, the number of time steps since it has been activated
- *d*, the degree of the node it is at
- *L*, the value of a distinct label that it was assigned

The robots must be provided with a set of deterministic instructions that will allow them to find each other. [5]

Many of the parameters of the problem are set by the *adversary*, or user:

- The size and layout of the graph
- The initial positions of the two robots
- When each robot is activated
- The value of the unique label assigned to each robot

[5]

A variety of algorithms exist which solve the Deterministic Rendezvous Problem.

Dessmark et. al [2006] presented an algorithm that solves the problem in time proportional to $O(n^5\sqrt{\tau l}+n^{10}l)$, where:

- n is the size of the graph
- l is the length of the shorter of the two labels
- τ is the difference in activation times of the two robots

[3]

In 2008, Kowalski and Malinowski presented an algorithm that solves the problem in time proportional to $O(n^{15}+l^3)$. This is a significant improvement because its runtime is no longer dependant on τ, which is set by the adversary and can thus be arbitrarily large. This solution has one major drawback, though. It makes use of *backtracking*, where the robots must keep track of each edge that they have traversed. This is a drawback because it places assumptions on the structure of the graph (namely, how it is labeled), and the robots' sensory and memory capabilities. [4]

Recently, Ta-Shma and Zwick [2014] presented an algorithm that solves the Deterministic Rendezvous Problem in time proportional to $O(n^5 l)$. This solution is notable in that it doesn't rely on τ and it doesn't make use of backtracking. Instead, it works by using *Universal Traversal Sequences*, which will be explored next. [5]

Universal Traversal Sequences

A *Traversal Sequence* is a series of instructions that define a traversal of every node in a particular *d*-regular graph from a given starting node. Each node in the sequence specifies which neighbour of the current node to visit next. [1] These steps are relative to the current node, not absolute. For example, if the current node is v_j, and v_j has *d* neighbors, then the traversal sequence will specify the next node to visit, v_{j+1}, as the *i*th neighbor of v_j, where $1 \leq i \leq d$.

A Universal Traversal Sequence (UTS) is a traversal sequence for a particular number of nodes, *n*, which covers all nodes of any *d*-regular, *n*-vertex graph; it also doesn't matter which node is chosen as the starting node. Aleliunas et al [1979] presented a proof which states that for any value of *n*, a UTS exists that covers all graphs of size *n* and has size proportional to $O(n^5)$. [1]

The graph provided by the adversary is not necessarily *d*-regular, as is assumed by a UTS. In such a case, a UTS is used where *d* corresponds to the largest degree of any node in the graph provided. The robots can simply remain idle if a UTS instructs them to traverse an edge which doesn't exist; this is essentially equivalent to adding self loops to any node with less than *d* edges until the node has *d* edges. [5]

The remainder of this paper assumes that for any *d*-regular, *n*-vertex graph, a UTS is known for those values of *d* and *n*.

Ta-Shma and Zwick's Solution

The basic idea of Ta-Shma and Zwick's solution is that if one of the robots completes a complete traversal of the graph while the other robot remains idle, or rests, then the two robots are guaranteed to meet. Since the size of the graph is unknown, the robots run UTSs for increasing values of *n*, while periodically resting. Whether a robot rests before or after a traversal depends upon the value of its label.

For example, one of the robots could run the sequence:

$$U_1 0^{u_1} U_2 0^{u_2} U_4 0^{u_4} U_8 0^{u_8} \ldots U_{2^i} 0^{u_{2^i}} \ldots$$

while the other robot runs the sequence:

$$0^{u_1} U_1 0^{u_2} U_2 0^{u_4} U_4 0^{u_8} U_8 \ldots 0^{u_{2^i}} U_{2^i} \ldots$$

where U_i is a UTS for a graph of size *i*, u_i is the number of steps in that UTS, and 0^k represents *k* steps where the robot rests. Eventually, the UTS for the size of the actual graph will be run by one robot while the other rests for the number of steps in that traversal. This only works if the two robots are activated at the same time, however.

To cover the case where the robots are activated at different times, the sequence to run will include rest periods of length u_i after each step (either in the traversal or the rest period). For example, one of the robots would run the sequence:

$$\sigma_1 0^{u_1-1} \sigma_2 0^{u_1-1} \sigma_3 0^{u_1-1} \ldots \sigma_{u_1} 0^{u_1-1} 0^{2u_1^2}$$
$$\pi_1 0^{u_2-1} \pi_2 0^{u_2-1} \pi_3 0^{u_2-1} \ldots \pi_{u_2} 0^{u_2} 0^{2u_2^2} \ldots$$

where $\sigma = U_1$ and $\pi = U_2$.

In order to formally present the sequence that each robot will run, some additional notation is needed.

Let:

- $\sigma^b =$
  0 if $b = 0$
  σ if $b = 1$
- $\bar{L} = 1 - L$
- $\sigma^{m_1 \ldots m_k} = \sigma_1^m \ldots \sigma_k^m$
- $D_k(\sigma_1 \ldots \sigma_m) = \sigma_1 0^k \ldots \sigma_m 0^k$

For the purpose of this report, assume that one robot is assigned $L = 0$, while the other robot is assigned $L = 1$. In this case, the sequence run by each robot is:

$$D_{u_{1-1}}((U_1 U_1)^{L\bar{L}}) D_{u_{2-1}}((U_2 U_2)^{L\bar{L}}) \ldots D_{u_{2^k}-1}((U_{2^k} U_{2^k})^{L\bar{L}}) \ldots$$

[5]

The sub-sequence

$$D_{u_{i-1}}((U_i U_i)^{L\bar{L}})$$

is known as a *block*, and can be rewritten as

$$D_{u_{i-1}}((U_i U_i)^{L} (U_i U_i)^{\bar{L}})$$

If $\sigma = U_i$ and $m = u_i$, the block can be further simplified to:

$$(\sigma_1 0^{m-1} \ldots \sigma_m 0^{m-1} \sigma_1 0^{m-1} \ldots \sigma_m 0^{m-1})^L$$
$$(\sigma_1 0^{m-1} \ldots \sigma_m 0^{m-1} \sigma_1 0^{m-1} \ldots \sigma_m 0^{m-1})^{\bar{L}}$$

Each of the above lines of the block are known as *chunks*. One chunk consists of a UTS (with interleaved idle periods), while the other chunk is an idle period of length $2m^2$.

If the robot's label is 0, then each block that it runs is equal to:

$$0^{2m^2}\sigma_1 0^{m-1}\ldots\sigma_m 0^{m-1}\sigma_1 0^{m-1}\ldots\sigma_m 0^{m-1}$$

If the label is 1, then a block is equal to

$$\sigma_1 0^{m-1}\ldots\sigma_m 0^{m-1}\sigma_1 0^{m-1}\ldots\sigma_m 0^{m-1}0^{2m^2}$$

Correctness Proof

Let:

- b_i = the number of steps in the block $D_{u_{i-1}}((U_i U_i)^{L\bar{L}})$
- w_i = the number of steps in $D_{u_{i-1}}((U_i)^{L})$, or a quarter of the number of steps in a block

A chunk of the block n has $2u_n^2$ steps in it, so $w_i = u_n^2$

From Aleliunas et al [1979], it is known that $u_n = O(n^c)$, so $w_n = O(n^{2c})$ [1]

Therefore, $b_n = 4w_n = O(n^{2c})$

Assume that $4u_n \leq u_{2n}$, for every $n = 2^i$. This assumption can be made because $4u_n = O(n^c)$ and $u_{2n} = u_n^2$ for all $n = 2^i$. Squaring both sides gives $16w_n \leq w_{2n}$. Multiplying both sides by four gives $16b_n \leq b_{2n}$. This results in the important property that for all $j \geq 1$,

$$\sum_{i=0}^{j} b_{2i} < \frac{1}{15} b_{2j}$$

which implies that if one robot is executing block i when the other robot is activated, then the former robot will be less than a quarter of the way through the i+1th block when the latter begins executing block i+1.

We now have everything we need to prove the correctness of the algorithm. Let K be the index (index in this case means the value of i in $D_{u_{i-1}}((U_i U_i)^{L\bar{L}})$) of the first robot to be activated when the second robot to be activated starts running block n, where n is the size of the graph. There are two cases: the case where $u_K \geq b_n$ and the case where $u_K < b_n$.

Case 1: $u_K \geq b_n$

In this case, the first robot rests between each step for at least as long as it takes the second robot to run a block, so the robots must meet (as shown in Figure 1). [5]

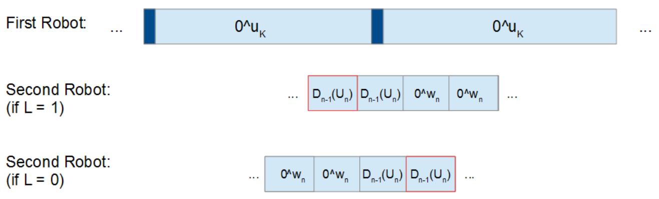

Figure 1: The second robot runs a complete traversal sequence while the first robot is resting

Case 2: $u_K < b_n$

$u_K = O(K^c)$ and $b_n = O(n^{2c})$, so $K < O(n^2)$

The second robot finishes block K and begins block $2K$ after $O(K^{2c}) = O(n^{4c})$ steps. The first robot must still be in the first quarter of block $2K$ at this time. If the first robot's label is 0, then the second robot will finish a complete traversal before the first robot finishes resting. Otherwise, the first robot will complete the entirety of its second traversal while the second robot is resting. These cases are shown in Figure 2. [5]

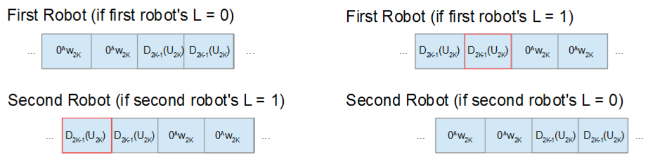
First Robot (if first robot's L = 0)
0^w_2K
0^w_2K
D_2K-1(U_2K)
D_2K-1(U_2K)
First Robot (if first robot's L = 1)
D_2K-1(U_2K)
D_2K-1(U_2K)
0^w_2K
0^w_2K
Second Robot (if second robot's L = 1)
D_2K-1(U_2K)
D_2K-1(U_2K)
0^w_2K
0^w_2K
Second Robot (if second robot's L = 0)
0^w_2K
0^w_2K
D_2K-1(U_2K)
D_2K-1(U_2K)

Figure 2: One of the robots will complete an entire traversal while the other rests

In both cases, one robot completes a full traversal sequence while the other robot rests, therefore the robots must find one another.

Complexity Analysis

As in the correctness proof, two cases will be considered.

Case 1: $u_K \geq b_n$

It was shown above that the robots will meet by the time the second robot completes block n. Block n is the $\ln(n)^{th}$ block, and each block has length $b_n = O(n^{2c})$, so the robots will meet $\ln(n)O(n^{2c}) \leq O(n^{4c})$ steps after the second robot is activated. [5]

Case 2: $u_K < b_n$

Recall from earlier:

- $u_K = O(K^c)$ and $b_n = O(n^{2c})$, so $K < O(n^2)$
- The second robot finishes block K and begins block $2K$ after $O(K^{2c}) = O(n^{4c})$ steps

It was shown that the robots will meet in the first half of block $2K$, which has length b_{2K}, so the robots will meet after $O(n^{4c}) + b_{2K}$ steps.

$b_{2K} = O(K^{2c}) = O(n^{4c})$

$O(n^{4c}) + b_{2K} = 2O(n^{4c}) = O(n^{4c})$

Therefore the robots will meet $O(n^{4c})$ steps after the second robot is activated. [5]

Conclusion

It has been proven that the robots will meet, and that the meeting will take place $O(n^{4c})$ steps after the second robot is activated. Ta-Shma and Zwick also go on to show how the robots can be made to meet only $O(n^c)$ steps after the activation of the second robot and how to deal with arbitrary labels. This is beyond the scope of this report, which is intended to show how Ta-Shma and Zwick's solution is one which runs in polynomial time not dependant on τ (which can be arbitrarily large) and which does not use backtracking (which makes assumptions on how the graph is labelled and the sensory/memory capabilities of the robots). [5]

References

(1) R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. 1979. Random walks, universal traversal sequences, and the complexity of maze problems. In FOCS. 218-223

(2) Alpern, Steve, Shmuel Gal and MyiLibrary, The Theory of Search Games and Rendezvous (Kluwer Academic Publishers, 2003) vol 55

(3) A. Dessmark, P. Fraingnaud, D. Kowalski, and A. Pelc. 2006. Deterministic rendezvous in graphs. *Algorithmica* 46, 1 (2006), 69-96

(4) D. R. Kowalski and A. Malinowski. 2008. How to meet in anonymous network. *Theoretical Computer Science* 399, 1-2 (2008), 141-156

(5) Amnon Ta-Shma and Uri Zwick. 2014. Deterministic rendezvous, treasure hunts, and strongly universal traversal sequences, universal exploration sequences. ACM Trans. Algor. 10, 3, Article 12 (April 2014), 15 pages.