Algorithms for the Online Time Dependant Freeze-Tag Problem

Dan Hill

Cameron University
mail@danhill.us

Abstract

The abstract lives here.

I. Introduction

The Freeze-Tag Problem (FTP) [1] is a problem in the field of swarm robotics in which a strategy to awaken a swarm in the minimum makespan must be found. In the original problem as proposed in [1], robots have two states, sleeping and awake. Sleeping robots are awakened when touched by an awake robot. At the beginning of the problem we are given a set of sleeping robots and a single awake robot known as the *source robot.* The source robot then awakens sleeping robots which then help awaken the other robots.

In this paper we examine a variation of FTP [2] in which each robot has a release time associated with it. Awake robots do not become aware of a sleeping robot's existence until it's release time has been met. We call this variation the Online Time Dependant Freeze-Tag Problem (OTDFTP).

Related Work. Hammer et al. [2] found the Offline Time Dependant Freeze Tag Problem has a lower bound of $7/3 - \epsilon$, for any $\epsilon > 0$.

In [3], Sztainberg et al. have proven a lower bound for the FTP using the nearest neighbor heuristic in which robots do not claim targets is at least $4 - \epsilon$ times optimal for any $\epsilon > 0$[3, Theorem]. The authors go on to find the strategy also has a tight approximation bound of $\Theta(\sqrt{\log n})$ for points on a plane and $\Theta((\log n)^{1-\frac{1}{d}})$ for points in d dimensions.

Preliminaries. Let $R = \{r_0, r_1, r_2, \ldots, r_n\} \subset M$ be the set of n robots in some continuous metric space M. M is a d-dimensional Euclidean space with distances measured according to an L_p metric.

Unless otherwise stated, the robot r_0 is the *source robot* and is the only robot that is not in sleep mode at the beginning of the problem. Each robot has 3 modes. In awake mode, a robot is fully functional and free to move. In sleep mode a robot is completely inactive and unable to be activated even if touched by an awake robot. Once the release time of a robot has been reached it enters *wait mode* and awake robots can now sense it's position. In wait mode a robot is available for activation.

In order to avoid robots from traveling in a single pack, awake robots can claim a target and no other robot will target a claimed robot. [3]

the OTDFTP can also be seen as a sequence of requests where $\sigma = (r, v)_1, (r, v)_2, \ldots (r, v)_m$ is a set of locations of robots ordered by their release time and the makespan is the amount of time it takes for all requests in *sigma* to be served.

A solution to the OTDFTP is considered *rational* if:

1. Each robot with no target will immediately choose an unclaimed robot in wait mode and begin moving toward it.

2. A robot does not move other than to advance on it's target. If a robot has no target, it will not move.

This definition is similar to the one given in [1] with slight modification for release times.

Summary of Results. We find that the competitive ratio of the Nearest Neighbor heuristic is 5/2.

II. Nearest Neighbor Algorithm

The Nearest Neighbor Algorithm is a *rational* wake-up strategy for the OTDFTP. Each robot chooses the closest unclaimed robot in wait mode. Once a robot has chosen a target it will not choose another target until it has reached it's current target.

Algorithm 1 Returns the nearest unclaimed sleeping robot.

Precondition: A is the set of sleeping robots and t be the robot's current target.

```
function NEAREST NEIGHBOR(A, t)
    if t ≠ NULL then
        return t
    if A.size = 0 then
        return NULL
    m ← NULL
    for robot in A do
        if ¬robot.claimed then
            if m = NULL then
                m ← robot
            else
                a ← dist(self, robot)
                b ← dist(self, m)
                if a < b then
                    m ← robot
    return m
```

Theorem 1. *For any $\epsilon > 0$, there exists an instance of the OTDFTP for which the Nearest Neighbor Algorithm results in a makespan less than $\frac{5}{2} - \epsilon$ times optimal.*

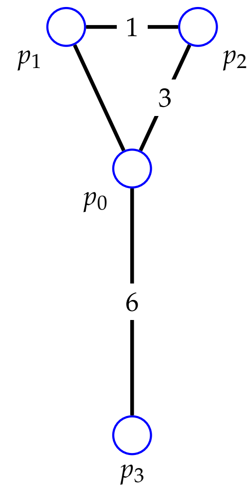
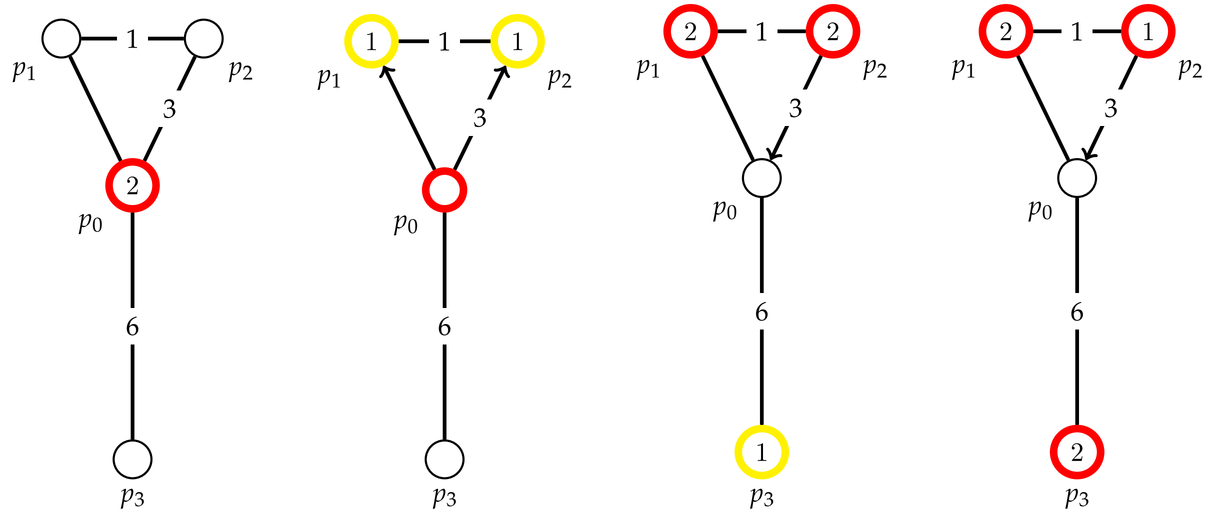


Figure 1

Proof. Let $G = V, E$ be the graph in Figure 1. The source robot r_0 activates at vertex v_0. At time $t = 0$ robot r_1 enters the sleeping state and is immediately awakened by r_0. At $t = 3$ r_2 enters sleeping state at v_1 and r_3 enters sleeping state at v_2. r_0 claims r_2 and begins moving down edge (v_0, v_1). r_1 claims r_3 and begins moving down edge (v_0, v_2). r_0 and r_1 arrive and awaken their respective targets at $t = 6$. At the same time r_4 enters sleeping state at v_3 and is claimed by r_0. r_0 travels down (v_1, v_0) to v_0 then travels down (v_0, v_3). r_0 arrives at v_0 at $t = 15$.

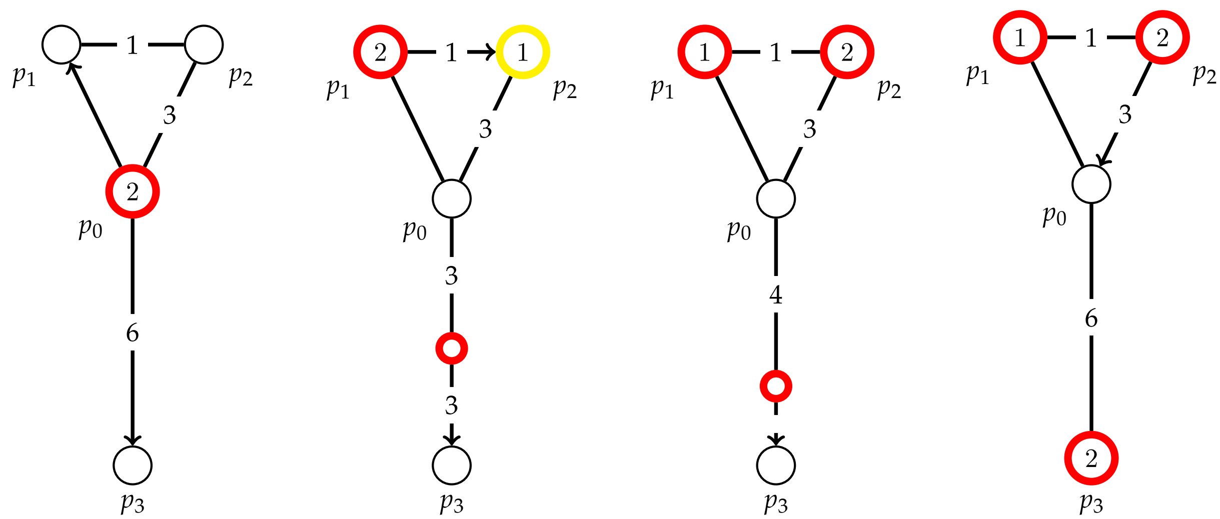
In the optimum solution, the source robot r_0 still starts at v_0 and awakens r_1 at $t = 0$. r_0 immediately moves down (v_0, v_3) and r_1 moves down (v_0, v_1). At $t = 3$, r_1 arrives at v_1 as r_2 and r_3 are entering sleeping state. r_1 immediately claims and awakens r_2 then claims r_2 and begins moving down (v_1, v_2). r_1 arrives at v_2 and awakens r_3 at $t = 4$. At $t = 6$ r_0 arrives at v_3 as r_4 is entering sleep mode and immediately awakens it.

This gives us a makespan of 15 for the Nearest Neighbor algorithm and a makespan of 6 for the optimal solution to Figure 1. This gives us a competitive ratio of $\frac{5}{2}$. $\square$



(a) $t = 0$ **(b)** b **(c)** $t = 6$ **(d)** $t = 15$

Figure 2: *Illustration of the competitive ratio proof for the Nearest Neighbor algorithm.*



(a) $t = 0$ **(b)** $t = 3$ **(c)** $t = 4$ **(d)** $t = 6$

Figure 3: *Illustration of the optimum solution for the problem in Theorem 1.*

I. Empirical Analysis

I.1 Experiment Setup

The experiment is based on a Python 2.7.9 simulation of swarms of varying sizes where the robots are running an implementation of the Nearest Neighbor Algorithm. The simulations were performed on a 64-bit PC running Ubuntu Linux.

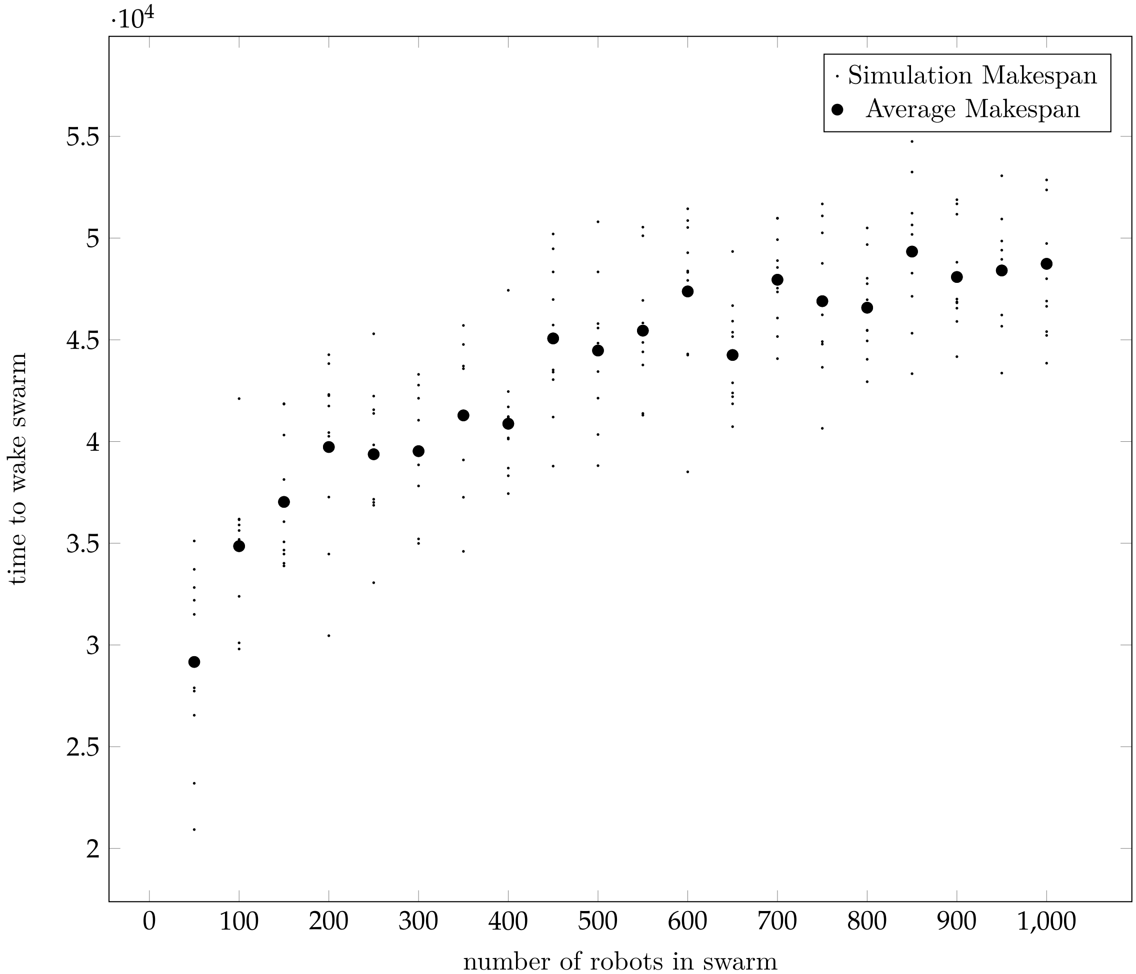
Dataset. The simulations are run with 210 different datasets consisting of varying numbers of robots randomly placed on a 10000x10000 2D Euclidean plane. The release time of each robot is a random number between 0 and 5000 time units.

Performance Measure The performance of the algorithm on the datasets is based solely on the makespan.

II. Experiment Results

III. Conclusion

IV. Future Work



References

[1] Esther M. Arkin, Michael a. Bender, Sandor P. Fekete, Joseph S B Mitchell, and Martin Skutella. The freeze-tag problem: How to wake up a swarm of robots. *Algorithmica (New York)*, 46(2):193–221, October 2006.

[2] Mikael Hammar, Bengt J. Nilsson, and Mia Persson. The Online Freeze-Tag Problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3887 LNCS, pages 569–579. 2006.

[3] Marcelo O. Sztainberg, Esther M. Arkin, Michael A. Bender, and Joseph S. B. Mitchell. Analysis of Heuristics for the Freeze-Tag Problem. In *Algorithm Theory — {SWAT} 2002*, pages 270–279. 2002.