Optimising Robotic Swarm Movement

Manish Kumar Bera
Mentor: Dr. Indranil Saha

June 21, 2017

1 Introduction

Swarm robotics is a field of multi-robotics in which large number of robots are coordinated in a centralized[1] or distributed[3] way. It is supposed that a desired collective behavior emerges from the interactions between the robots and interactions of robots with the environment. With recent technological developments, producing a large number of inexpensive robots that are equipped with sophisticated sensing, computation and communication tools has become a reality[5]. Swarm robots can be used to form various spatial arrangements for any required purpose. Robots need to move simultaneosly, to go from one position to another, to collectively form a new configuration. Describing complex spatial specifications for swarms is a non-trivial task.

2 Objective

Describing complex spatial specifications for swarms is a non-trivial task. Our goal is to optimise the cost of the collective motion of the robots from initial configuration to final configuration.

3 Methodology

We consider a grid A of size $n \times n$. We will refer the collction of all the robots in the grid as *swarm*. We will have R number of robots, such that a robot will occupy a cell of the grid. No two robots can occupy the same cell of the grid, and a robot will occupy only a single cell of the grid. We will refer to a *move* as the movement of a robot. We define a *timestep* as the time period after which a robot can execute a *move*. A robot can execute a *move* on each *timestep*. Each move will be selected from a set of *Motion Primitives*.

Let L be the total number of *timesteps* which the *swarm* takes to go from initial configuration to final configuration. Let x_t^i be a variable that describes the position of ith robot after t time steps. We define *swarm position at timestep* t, X_t as follows:

$$X_t = (x_t^1, x_t^2, x_t^3, ..., x_t^R)$$

i.e., X_t is an ordered collection of tuples, with the ith tuple describing the position of the ith robot after t *timesteps*. Thus, X_t describes the position of the *swarm* in the grid after t *timesteps*. So, we can say that $X_0 = (x_0^1, x_0^2, x_0^3, ..., x_0^R)$ represents the initial configuration of the *swarm*, and $X_L = (x_L^1, x_L^2, x_L^3, ..., x_L^R)$ represents the final configuration of the *swarm*.

The movement of the *swarm* will obey a set of rules. The progress from one *swarm position* to another will always follow certain conditions. Our aim is to develop a set of constraints, whose solution will give an optimal for the *swarm*. We will frame these constraints using a set of boolean variables, which will be encorporated into a formula ϕ. We define a *solution* S:

$$S = [X_0, X_1, X_2, ..., X_L]$$

where S is a sequence of *swarm positions* which occur in the chronological order of *timesteps*. The symbol S represents the sequence in which the *swarm* movement can occur from initial to final configuration. A *solution* S is said to be valid iff:

$$\begin{aligned} S &\models \phi \qquad (1)\\ \Leftrightarrow [X_0, X_1, X_2, ..., X_L] &\models \phi \qquad (2)\end{aligned}$$

The problem will thus be reduced from an optimal path finding problem to a problem of finding satisfiability(SAT)[2]. We will use a SAT solver (like the MiniSAT[4]) to compute the solution to the above model.

4 Work done till now

4.1 boolean variables

I define $X(t,k,x,y)$ as a boolean variable s.t. $X(t,k,x,y)$ is *TRUE* iff at time step t, the kth robot is at position (x,y).

4.2 initial and final states

I define (x_0^k, y_0^k) to be the initial position of the kth robot. Similarly, (x_L^K, y_L^k) is the final position of the kth robot.
The constraints for initial and final positions will be:

$$\begin{aligned}\forall t \forall k, X(0,k,x,y) &= TRUE;\ if\ x = x_0^k\ and\ y = y_0^k\\ &= FALSE;\ otherwise\end{aligned} \qquad (3)$$

$$\begin{aligned}\forall t \forall k, X(L,k,x,y) &= TRUE;\ if\ x = x_L^k\ and\ y = y_L^k\\ &= FALSE;\ otherwise\end{aligned} \qquad (4)$$

4.3 motion primitives

I assumed that the robots will have five simple motion primitives. each robot can:
1. Move up by one cell
2. Move down by one cell
3. Move right by one cell
4. Move left by one cell
5. Stay in that cell
Each robot has to perform one of the above mentioned motion primitives at each time step. Whether the robots are allowed to perform a particular primitive at a give timestep also depends on other constraints.

The constraints for motion primitives will be:

$$\forall t \forall k,\ X(t{+}1,k,x,y) \Rightarrow X(t,k,x,y) \vee X(t,k,x{-}1,y) \vee X(t,k,x,y{-}1) \vee X(t,k,x{+}1,y) \vee X(t,k,x,y{+}1)$$

4.4 obstacle avoidance

Let the number of obstacles be G. Let (x_{obs}^g, y_{obs}^g) be the position of the gth obstacle where $g \in \{0, 1, ..., G-1\}$. Now each robot has to avoid each obstacle at every timestep.
The constraints for obstacle avoidance will be:

$$\forall t \forall k,\ X(t,k,x,y) = FALSE;\ if\ \exists g \in \{0, ..., G-1\}\ s.t.\ x = x_{obs}^g\ and\ y = y_{obs}^g$$

4.5 collision avoidance

Each robot must avoid getting hit by another robot. There will be two types of collision.

4.5.1 same space collision avoidance

This refers to the collision when two robots try to occupy the same cell of the grid. The constraints for this type of collision avoidance will be:
$\forall t \in \{0, 1, ..., L\};\ \forall k_1, k_2 \in \{0, 1, ..., R-1\}\ s.t.\ k_1 \neq k_2;\ \forall x, y \in \{0, 1, ..., n-1\};$

$$\neg(X(t, k_1, x, y) \wedge X(t, k_2, x, y))$$

4.6 headon collision avoidance

This refers to the collision type where two robots move into each other. The constraints for this type of collision avoidance will be:
$\forall t \in \{0, 1, ..., L\};\ \forall k_1, k_2 \in \{0, 1, ..., R-1\}\ s.t.\ k_1 \neq k_2;\ \forall x, y \in \{0, 1, ..., n-1\};$

$$\neg((X(t, k_1, x, y) \wedge X(t, k_1, x+1, y)) \wedge (X(t, k_2, x+1, y) \wedge X(t, k_2, x, y)))\wedge$$

$$\neg((X(t, k_1, x, y) \wedge X(t, k_1, x, y+1)) \wedge (X(t, k_2, x, y+1) \wedge X(t, k_2, x, y)))$$

4.7 mark1: The first prototype

I used the MiniSAT, a general purpose SAT solver for solving SAT problem instances.

4.8 mark2

4.9 mark3

4.10 mark3.1

4.11 mark3.2

References

[1] Iman Haghighi, Sadra Sadraddini, and Calin Belta. *Robotic Swarm Control from Spatio-Temporal Specifications.* 2016 IEEE 55th Conference on Decision and Control (CDC), ARIA Resort & Casino, December 12-14, 2016, Las Vegas, USA.

[2] Wikipedia *Boolean satisfiability problem.*
`https://en.wikipedia.org/wiki/Boolean_satisfiability_problem`

[3] Caroline Perry. *A self-organizing thousand-robot swarm.* August 14, 2014.
`https://www.seas.harvard.edu/news/2014/08/self-organizing-thousand-robot-swarm`

[4] *The MiniSAT Page.* Niklas En, Niklas Srensson
`http://minisat.se/Main.html`

[5] Self Organizing Systems Research Group. *The Kilobot Project.*
`https://www.eecs.harvard.edu/ssr/projects/progSA/kilobot.html`