Multiagent Coordination in Roombas: From a Neural Network Perspective

Jimmy Lin
Barry Feigenbaum

Prof. Risto Miikkulainen

Department of Computer Science
The University of Texas At Austin

December 3, 2014

Table of Contents

Section 1

Introduction to Roomba

The Roomba Environment

The Roomba Environment: Cont.

- Objects:
  - Agent: dynamic dirt collectors (Black Disks)
  - Crumb: static dirt to clean (Blue Dots)
  - Wall: static boundary of the world
  - Chair/Desk: static decorations as salient/hidden obstacles
- Goal: figure out the best dirt collection efficiency.
- Objectives: within a given period,
  - collect as many crumbs as possible
  - make as few collisions as possible

Section 2

Problem Formulation

A Ideal Case

- The most ideal case is to set up a system with
  - Centrailization: full control over all agents
  - Global View: full observations over all crumbs
- In this system, all distances between crumbs can be pre-computed and the crumb collection can be formulated as an integer programming problem.
- Little improvement can be achieved by experiential learning in this case.

A Realisitc Case

- Relax the "Centralization" and "Global View" constraints, but allow limited communication between agents. These yield a system with
  - Autonomy: all agents should decide on its own
  - Local View: only local observation of crumbs are available for agents
  - Limited communication: a limited amount of information can be shared between agents
- Target at learning some high-level multiagent behaviors by which efficiency of the crumb collection is maximized.

Challenges and Difficulties

- Expected Multi-agent Behaviors:
  - Work Balance.
  - Collision Avoidance.
  - Competition Avoidance.
- Efficient Coding for Sensors:
  - Sensation: what information does each individual agent perceive in its local view?
  - Communication: what information to share between agents?
- Social Learning Policies that facilitate the learning process:
  - Egalitarian Social Learning (ESL).
  - Opportunistic Cooperative Learning (OCL).

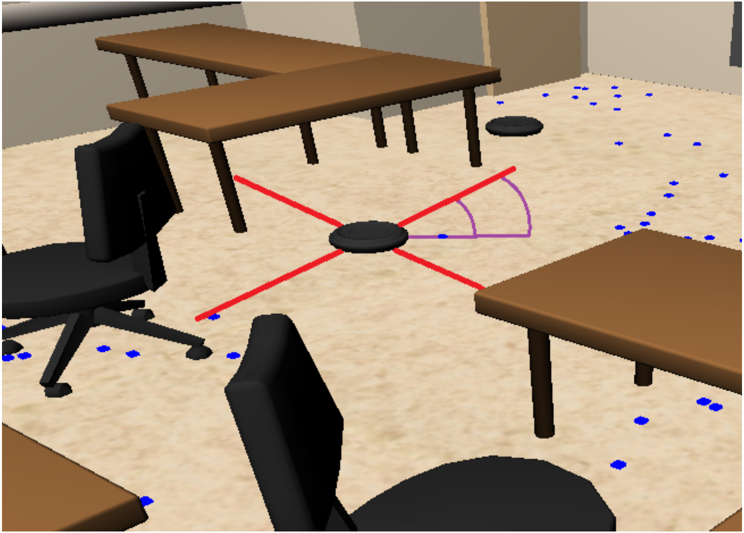
Section 3

Implementation

Approaches

- Learning mechanisms:
  - Neroevolution (Real-time NEAT)
  - Q-learning
- Techniques to support cooperation:
  - Communication between agents
  - Social learning
  - Centralized control / coordination
- Compare against hardcoded behavior as a baseline:
  - Random agents
  - Purely greedy search

Roomba Sensors



Training Plan

We are incrementally training agents to solve more difficult problems:

1. Simple greedy search
2. Simple cooperative / coordinated search
3. Greedy search with collision avoidance
4. Cooperative search with collision avoidance
5. Social learning and other advanced multi-agent techniques

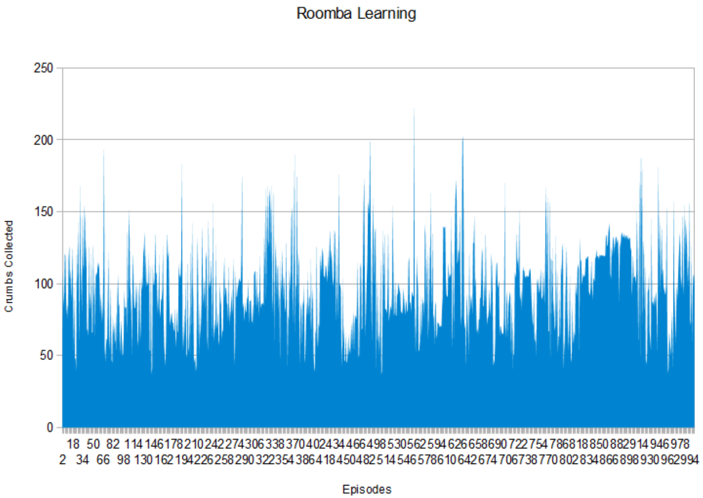
Section 4

Experimentation

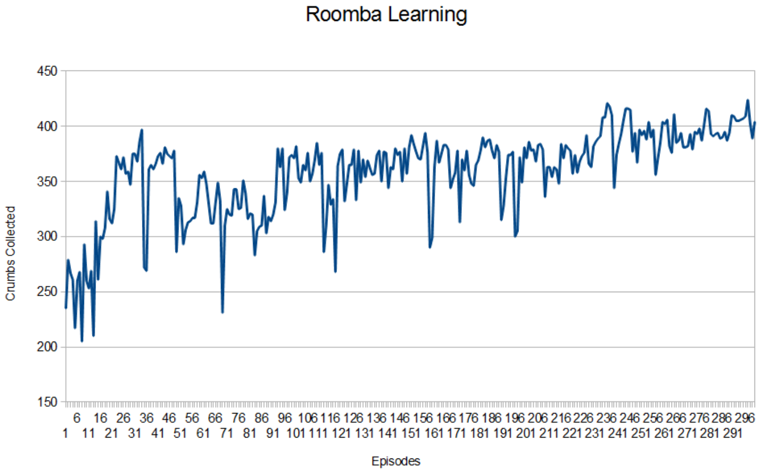
Progress Thus Far

- Initial results are unexciting - agents don't learn
- Roomba module needs work just to be used as a baseline
- Most of our effort has been spent getting the Roomba mod on track

Unmodified Roomba Environment



Modified Enviroment



Conclusion / Future Work

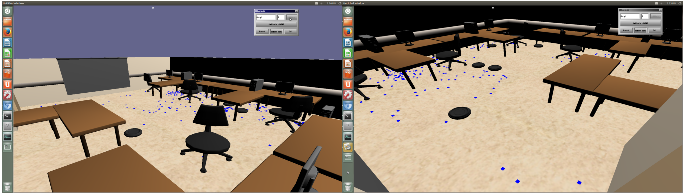
- Most of our effort has been spent getting the Roomba mod on track
- A major issue is learning obstacle avoidance
- We're experimenting with different sensor configurations
- Exploring learning algorithms other than rtNEAT
- And of course, there are exciting ideas like social learning, emergent communication / language, stigmergy, etc.

Section 5

Discussion

Discussion Time

- Any questions or suggestions?



- Or email us at
  - jimmylin at utexas dot edu
  - baf at cs dot utexas dot com

Acknowledgement

Thanks.