Genetic Programming of Autonomous Agents

Bradley University 2011

By: Scott O'Dell

Advisor: Dr. Joel Schipper

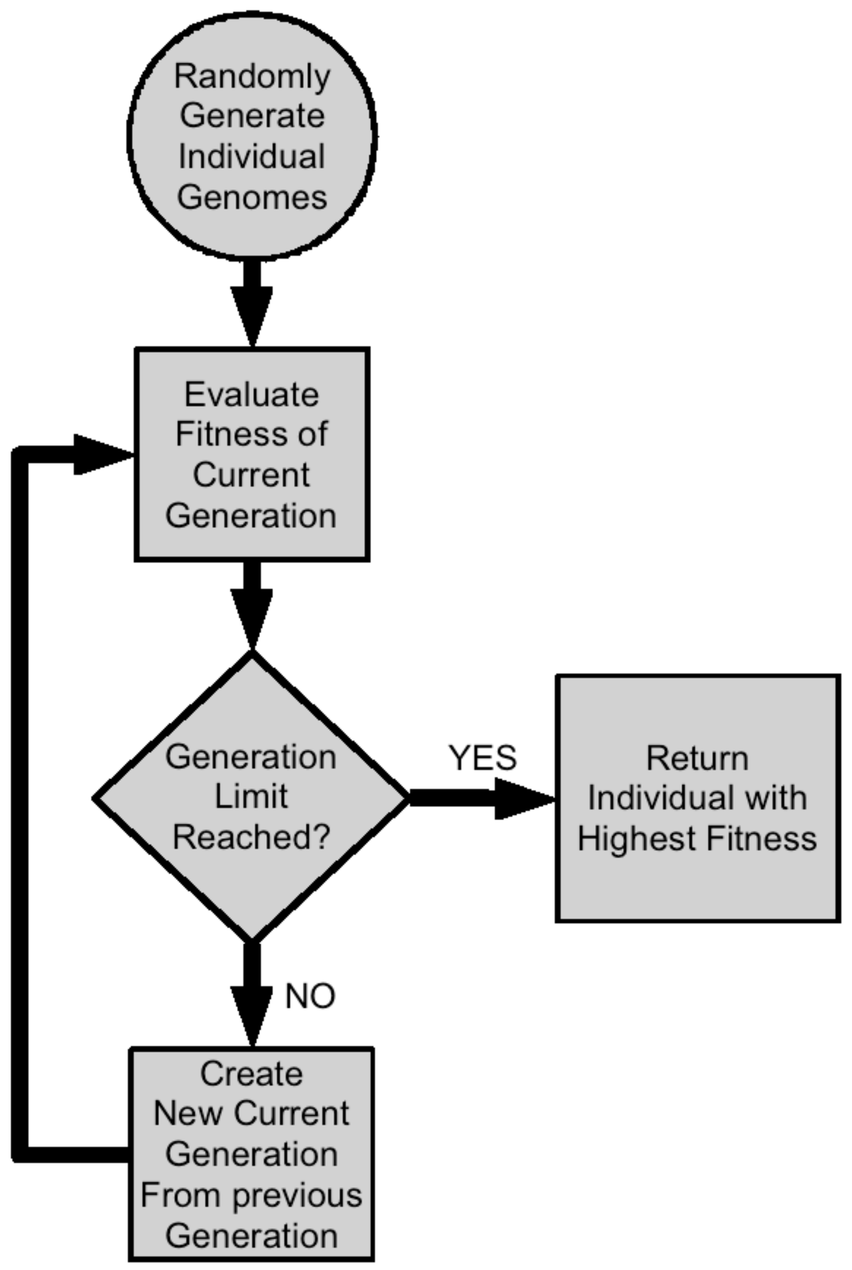
Abstract

Research in genetic programming (GP) for the control of autonomous agents often employs grid-based simulations to evolve prototypical solutions. However, grid-based simulations produce solutions that are often impractical on physical robotic platforms. Our research evolves perimeter maintenance control programs to illustrate the limitations of grid-based approaches and explore the advantages of using a continuous simulator when evolving programs for physical autonomous agents.

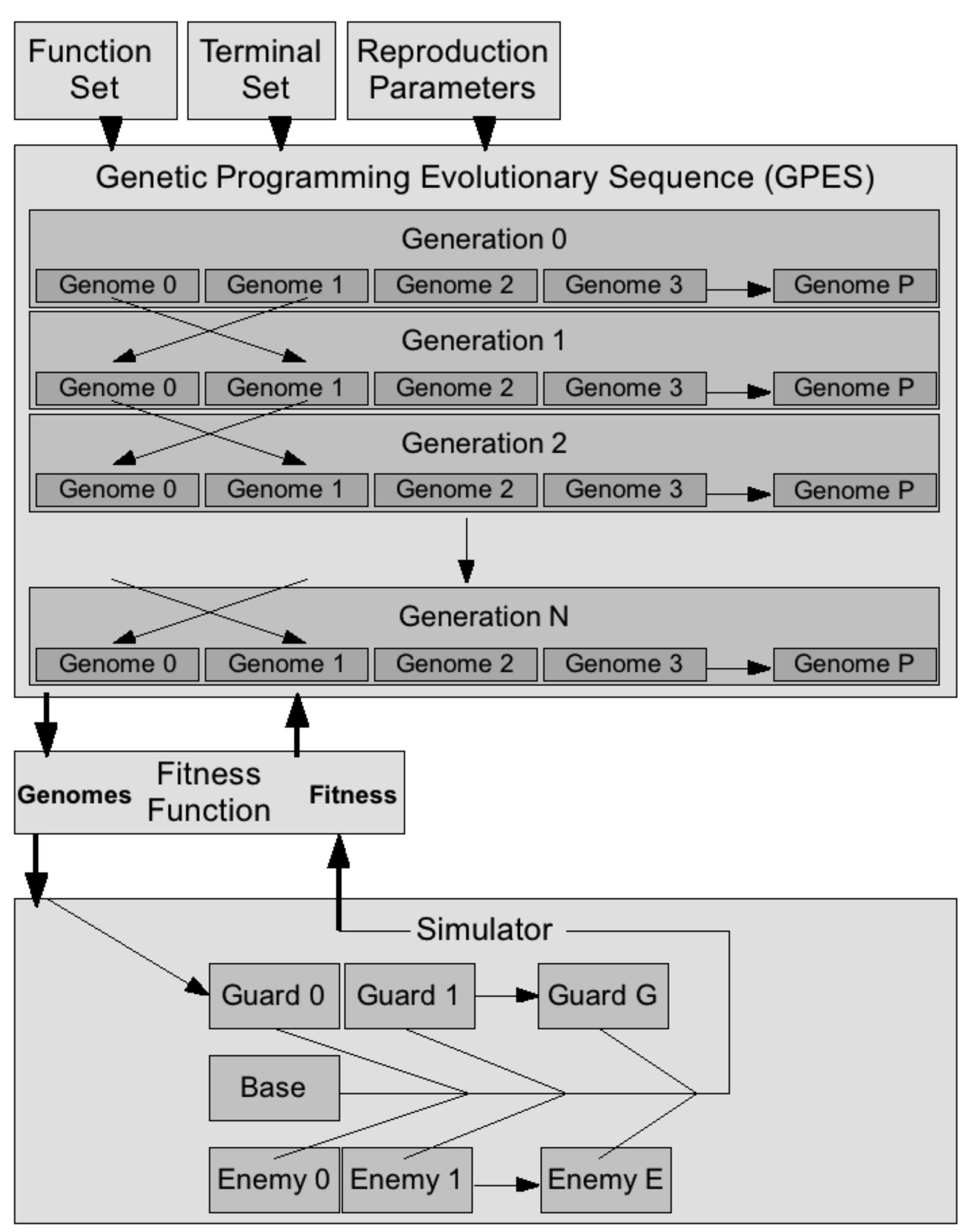
Perimeter maintenance is a task where a group of autonomous "guard" agents are deployed around a "base" to detect "enemy" agents before they reach the base. For this research, GP software and simulators were developed to approximate the movements of physical agents. To illustrate the limitations of evolving autonomous agent control programs using a grid-based simulator, early experiments evolved perimeter maintenance agents in the grid domain. Later experiments replaced the grid-based simulator with a continuous simulator to demonstrate its advantages when generating control programs for physical autonomous agents. The experiments performed explore multiple fitness functions for evolving homogenous and heterogeneous teams of guards as well as the co-evolution of guards and enemies. The approach has produced autonomous agent control programs that display intelligent perimeter maintenance behavior in a continuous simulator by patrolling a circular area around the base.

Genetic Programming Framework

The software framework and simulator is written in the Ruby programming language. Ruby's object-oriented nature contributed to the rapid development of reusable software components.



Process of Genetic Programming



Architecture of GP Framework

Results of Evolutionary Sequences

Evolution of Guards

Each guard is controlled using the same program. Enemies start on the edge of the simulation and moves directly toward the base.

Co-evolution of Guards & Enemies

Enemies are evolved with the same primitive set as the guards with the addition of terminals that provide the position of the closest guard. Evolving the enemies simultaneously with the guards produces less exploitable solutions from the guards.

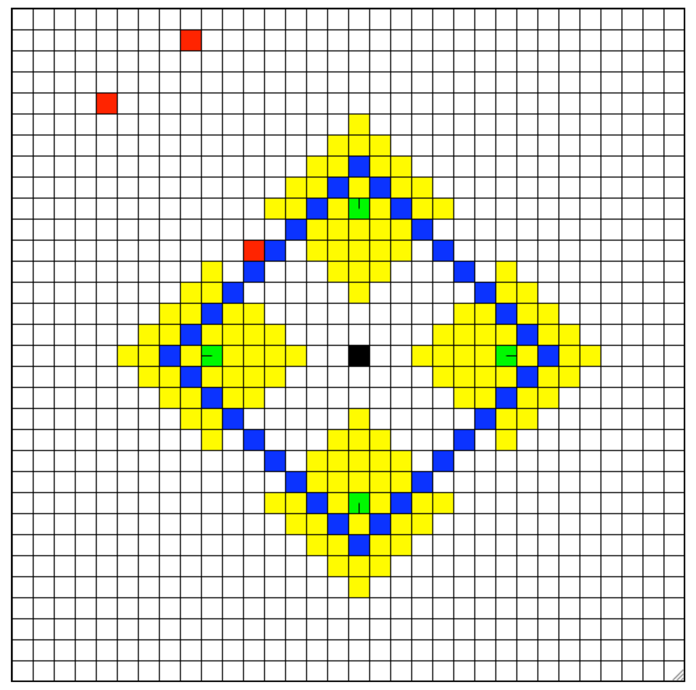
Grid-Based Simulations

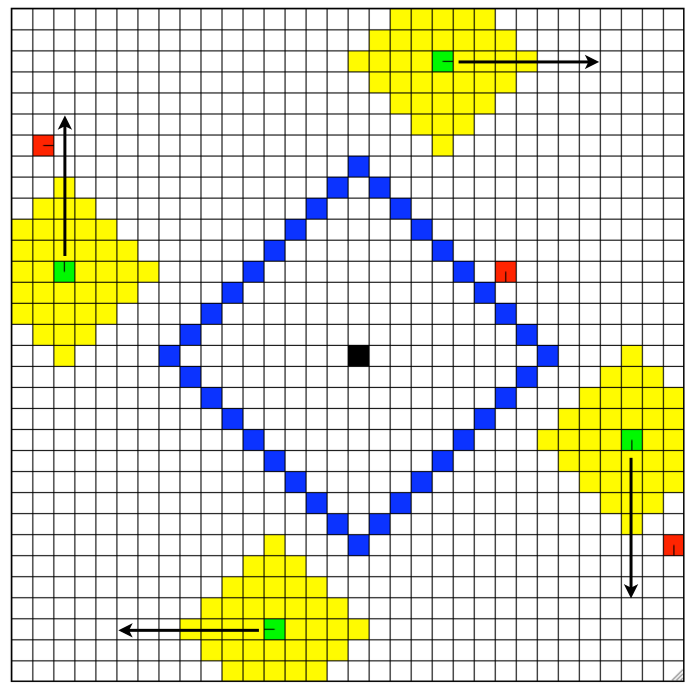
Function Set

Function	Arity	Pseudo-code
prog	2	(a) then return(b)
>	4	if (a > b) then (c) else (d)
+, -, *, /, %	2	[standard integer arithmetic]

Terminal Set

Terminal	Effect
base	returns Manhattan distance from guard to base
forward	moves agent forward, returns "base"
left	turns agent left 90 degrees, returns "base"
right	turns agent right 90 degrees, returns "base"
0-6	constant integers





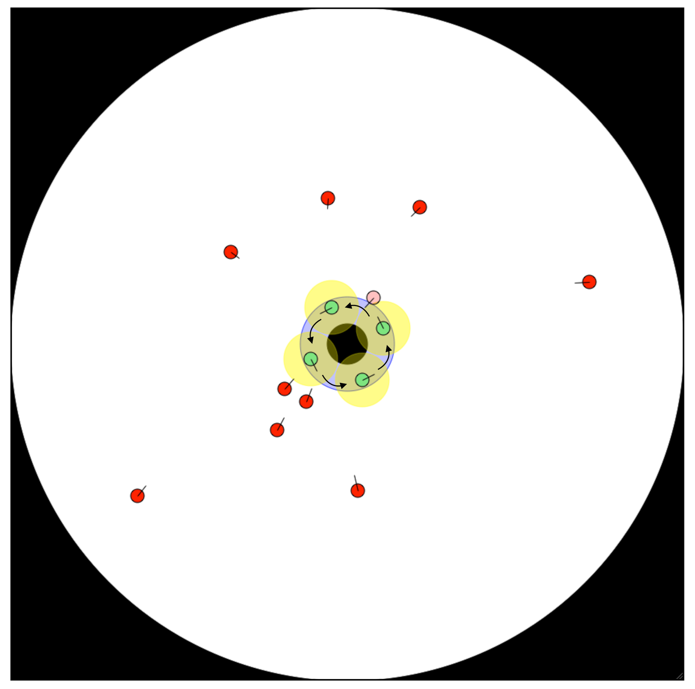
Continuous Simulations

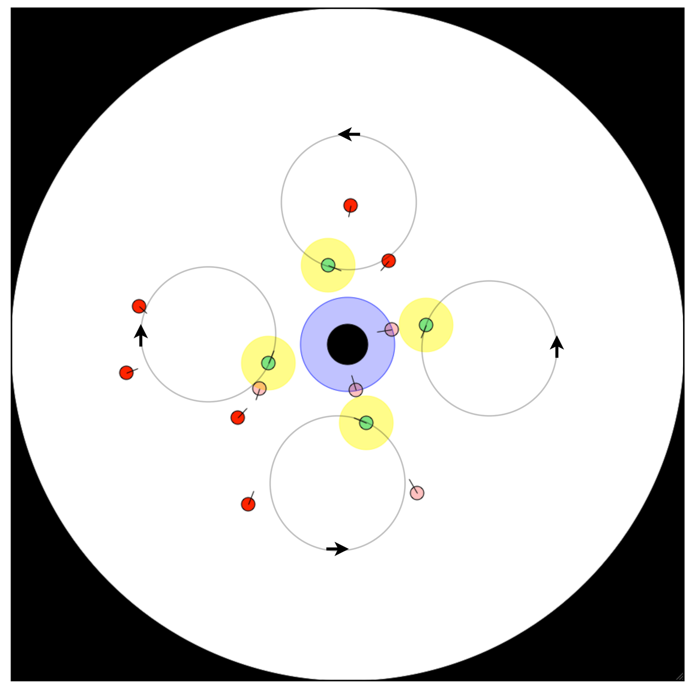
Function Set

Function	Arity	Pseudo-code
prog	2	(a) then return (b)
> (Magnitude)	4	if (a.mag > b.mag) then (c) else (d)
> (Angle)	4	if (a.ang > b.ang) then (c) else (d)
X =	1	variable X = (a)
+, -. *	2	[standard vector arithmetic]

Terminal Set

Terminal	Effect
base	returns vector from guard to base
direction	returns unit vector representing guard's heading
X	return (variable X)
Vector	[static vector]





Noisy Simulations

To test if genetic programming can deal with uncertainty in the environment, Gaussian noise was added to the simulations. Any terminal that is based on sensor data has noise with a constant variance, and each movement is subject to noise with a variance that is equal to 1/10th the magnitude of the ideal movement.