Introduction to Particle Swarm Optimization and its Application to Mobile Swarm Robotics

Outline

- Introduction, Thanks, Discussion
- The Story
    - First researched in undergraduate studies for swarm robotics work
        - 1st sem.
            - Began investigating robot collaboration solidifying the design of one robot
            - Job: seek light
            - Q-learning reinforcement technique
            - Starting with something simple so to remain scalable to more complicated targets and environments
        - 2nd sem., added localization (gps), communication (rf)
    - Realized the problem: moving the swarm intelligently
    - Wouldn’t it be great if there was an intelligent way to choreograph the robots’ movements without too much computational overhead or complexity, yet remain robust to significant changes in the environment?
    - Found PSO, researched and implemented, but not tested due to time constraints and platform issues
    - Revisited in graduate studies as project in optimal control theory class
- PSO Introduction [Kennedy]
    - Authors
        - James Kennedy (Social Psychologist, Bureau of Labor Statistics, Washington, DC)
        - Russell Eberhart (Electrical Engineer, Purdue School of Engineering and Technology, Indianapolis, IN)
    - Original paper in 1995, *Proceedings of IEEE International Conference on Neural Networks*.
    - Originally created to simulate social behavior models, inspired by “graceful but unpredictable choreography of a bird flock”
    - Component Methodologies
        - Artificial Life: bird flocking, bee swarming, fish schooling, group of seemingly intelligent units performing one task.
        - Evolutionary computation: genetic algorithms and evolutionary programming
        - Conceptually, this is what gives the algorithm its power. After all, science and technology inspired by nature, benefits from the millions of years of

        development that has already happened.
    - Growth and Applications
        - First published application: tuning artificial neural networks
        - Algorithm is general and can be applied to multiple areas
        - Riccardo Poli, Department of Computer Science, University of Essex, Colchester, UK
            - 1100 papers in ieee xplore, 650 cited papers, 26 top journal paper topics
            - Optimal Control
            - Investment Decision Making
            - Graphics and Visualization
- PSO Algorithm
    - There are many variation and hybridizations of the algorithm. Here we examine the original presented in the paper.
    - Definitions
        - f: $\mathbb{R}^n \rightarrow \mathbb{R}$, maps n-dim to 1
            - Fitness / cost function to be minimized
            - Takes a candidate solution, and assigns a fitness / cost
            - The gradient of f is not known
        - S, number of particles each with position $\mathbf{x}_i \in \mathbb{R}^n$ and velocity $\mathbf{v}_i \in \mathbb{R}^n$
        - $\mathbf{p}_i$ best known position for particle i
        - $\mathbf{g}$ best known position for the swarm
        - Tuning factors
    - Initialize
        - Uniformly distribute the particles' positions: $\mathbf{x}_i \sim U(\mathbf{b}_{lo}, \mathbf{b}_{up})$
        - Uniformly distribute the particles' velocities: $\mathbf{v}_i \sim U(-|(\mathbf{b}_{up} - \mathbf{b}_{lo})|, (\mathbf{b}_{up} - \mathbf{b}_{lo}))$
        - Over every particle:
            - Set each particle's best with it's starting position: $\mathbf{p}_i = \mathbf{x}_i$
            - If the particle's best is better than the global, update the global: $\mathbf{g} = \mathbf{p}_i$
    - Iterate
        - Over every particle:
            - Generate uniformly distributed random values
            - Assign a new velocity based on factors of
                - Previous velocity
                - The distance to the local best
                - The distance to the global best
            - Update the position based on the new velocity
            - If the new position is better than the local, update it
            - If the new position is better than the global, update it
    - So already you might be able to see some carry over into mobile robotics.
- Benchmark Behavior
    - Functions:

        - Rastrigin Function: non-convex, 2D (could be more / less), "sagging egg-crate bed cushion", minimum at (0, 0), many discrete minimums and maximums
        - Rosenbrock Function (1960, computer journal): non-convex, 2D, "banana valley", minimum at (1, 1), easy to find valley, [Rosenbrock]
    - Benchmark Performance:
        - Marco Nobile (M.S. Evolutionary Computing and Complex Systems. Milano, Italy)
        - http://vimeo.com/17407010
- Qt Demo #1
    - Cost function, minimize: $f(x,y) = \text{sqrt}((t_x - x)^2 + (t_y - y)^2)$
    - 4 Hz or fps or iterations per s
    - "Slow Approach"
        - $\omega = 0$ (previous velo)
        - $\phi_i = .025$ (distance to local)
        - $\phi_g = .025$ (distance to global)
    - Without investigating the algorithm further, we already have the foundation for a path finding or room mapping algorithm.
- Application: Path Finding / Room Mapping
    - Rafaelle Grandi, Computer Engineer, University of Bologna, Industrial robotics and swarms.
    - Tuning not known, cost function is 3D, position (x,y) and time robot is in a position
    - The video is left as an exercise for the student
    - What if the goal moves?
- Five principles of swarm intelligence
    - Millonas, (1994). Swarms, phase transitions, and collective intelligence
        - Proximity: "population should be able to carry out simple space and time calculations"
        - Quality: "population should be able to respond to quality factors of the environment", the principle that the population must be able to sense the appropriate stimuli in the environment
        - Diverse Response: "population should not commit its activities along excessively narrow channels", react in various ways to stimulus
        - Stability: "population should not change its mode of behavior every time the environment changes"
        - Adaptability: "population must be able to change behavior mode when it's worth the computational price." if there is a modest change in the environment, the population should be able to adapt to it
    - 4 and 5 are complementary, a trade-off must be balanced
    - Our optimizer can suitably operate as a swarm intelligence.
    - What if our goal moves?
- Qt Demo #1 Revisit

    - Move goal within the swarm
    - Move goal outside the swarm, Uh oh!
    - Wait, wait. How'd that path finding example work then?
    - Consider the 3D gradient.
- Tuning Values [Pedersen]
    - Magnus Pedersen, HVass Laboratories, 2010, determined "good" choices for tuning values for a number of scenarios
    - Used a meta-optimization technique, PSO to find PSO values
    - General tunings
- Qt Demo #2
    - Get back to the question: What if the goal moves?
    - "Slow Approach"... Bad
    - "Good"
        - ω = .3925 (previous velo)
        - ϕ_i = 2.5586 (distance to local)
        - ϕ_g = 1.3358 (distance to global)
    - What if the optimizer over-tunes?
- Qt Demo #3
    - "Oscillations"
        - ω = 1 (previous velo)
        - ϕ_i = 1 (distance to local)
        - ϕ_g = 1 (distance to global)
    - Unbounded, oscillates forever
    - "Oscillations Modified"
        - ω = .99 (previous velo)
        - ϕ_i = 1 (distance to local)
        - ϕ_g = 1 (distance to global)
- Application: Original, seek robustly in changing environment
    - Moving intelligently? Yes, the PSO follows the five principles of warm intelligence.
    - Is there considerable computational overhead and complexit? No, PSO iterations are simple, nonlinear calculations.
    - Is the swarm robust to significant changes in the environment? Yes, even with intentional oscillations, the PSO collapses to a solution.

References

Grandi, R. (2009). "Robot Swarm driven by Particle Swarm Optimization algorithm". https://www.youtube.com/watch?v=RLIA1EKfSys.

Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". *Proceedings of IEEE International Conference on Neural Networks*. 1942–1948.

Millonas, M. M. (1994). Swarms, phase transitions, and collective intelligence. In C. G. Langton, Ed., *Artificial Life III*. Addison Wesley, Reading, MA.

Nobile, Marco. (2010). “Particle Swarm Optimization”. http://vimeo.com/17407010.

Pedersen, M.E.H. (2010). "Good parameters for particle swarm optimization". *Technical Report HL1001*. Hvass Laboratories.

Poli, R. (2008). "Analysis of the publications on the applications of particle swarm optimisation". *Journal of Artificial Evolution and Applications*: 1–10.

Rosenbrock, H. H. (1960). “An Automated Method for finding the Greatest or Least Value of a Function”. *The Computer Journal*: 175–184.