International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Memory Based Learning Augmented with Simulated Annealing for a Two Joint Arm

Patrick McDowell[1], Kuo-pao Yang[2]
[1,2]Department of Computer Science, Southeastern Louisiana University, Hammond, LA, USA

Abstract**—** *This paper discusses a simple, biologically plausible, methodology for approaching control problems, mostly focused on joint control and coordination. The method relies on a modified simulated annealing routine, coupled with a nearest neighbor-based memory system. The algorithms were tested in a python-based simulation and are presented in pseudo-code, along with the motivation for the work and the criteria that drove the solution.*

Keywords**—** *Annealing, coordination, learning, state-space exploration.*

I. INTRODUCTION

This paper describes a simple algorithm whose purpose is to enable a two jointed arm to learn to reach positions of interest. At the lowest level, the algorithm employs a simulated annealing-based search in order to find the correct joint angles for the end effector of the arm to reach specified positions. Built into the system is a state memory so that the arm can reduce exploration time when trying to reach new positions. The annealing system and state memory work in concert so that the arm can quickly move to known locations, and figure out how to reach new locations, based on their proximity to known locations.

II. BACKGROUND AND MOTIVATION

Robotic limbs are common in both mobile robots and in assembly robots in manufacturing plants. To find the position of the end effector of the arm, forward kinematics is used [1]. To determine the joint angles needed to put the end effector in some desired position, inverse kinematics is used [2]. For limbs with 2 or 3 degrees of freedom, there are algebraic/trigonometric solutions that are fairly straight forward. Fig. 1 provides a diagram showing the various angles and limb lengths required for a 2 degree of freedom solution. Note that there are two possible solutions to the problem, as shown in the figure.

The following section briefly details the algebra of a 2-degree of freedom (DOF) solution. The location of P_1 will be assumed to be (0,0), and the lengths of the segments, and the angles between the segments are known. Using these values, the locations of the ends of the segments, P_2 and P_3 can be calculated using forward kinematics as is shown below.

$$P_{2x} = L_1\cos\theta_1$$
$$P_{2y} = L_1\sin\theta_1$$

$$P_{3x} = L_1\cos\theta_1 + L_1\cos(\theta_1 + \theta_2)$$
$$P_{3y} = L_1\sin\theta_1 + L_1\sin(\theta_1 + \theta_2)$$

The following definitions and equations describe the locations of the relevant parts of the 2 DOF arm in an inverse kinematics situation. This situation is very important because it tells what θ_1 and θ_2 need to be for a desired end effector's position (P_3).

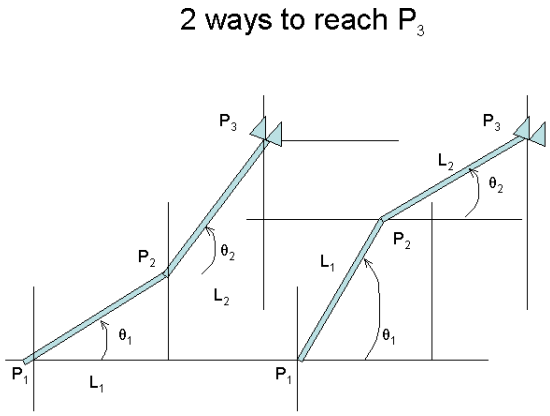

Fig. 1. This figure shows that for the arm to reach point P_3 there are two different solutions.

Two important formulas that will be used to algebraically solve these equations will be the Law of Sines and Law of Cosines. They are stated below:

Law of Sins:

$$\frac{a}{\sin(A)} = \frac{b}{\sin(B)} = \frac{c}{\sin(C)} \quad \text{[see Fig. 2]}$$

$$D = \tan^{-1}\left(\frac{P_{3y} - P_{1y}}{P_{3x} - P_{1x}}\right)$$

$$c = \sqrt{(P_{3x} - P_{1x})^2 + (P_{3y} - P_{1y})^2}$$

Law of Cosines:

$$a^2 = b^2 + c^2 - 2bc\cos A$$

rearranging we get:

$$A = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2bc}\right)$$

$$\frac{a}{\sin(A)} = \frac{c}{\sin(C)}$$

$$C = \sin^{-1}\left(\frac{\sin(A)c}{a}\right)$$

Patrick McDowell, and Kuo-pao Yang, "Memory Based Learning Augmented with Simulated Annealing for a Two Joint Arm," *International Research Journal of Advanced Engineering and Science*, Volume 5, Issue 4, pp. 244-247, 2020.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Total Angle = D + A

Or

Total Angle = D - A

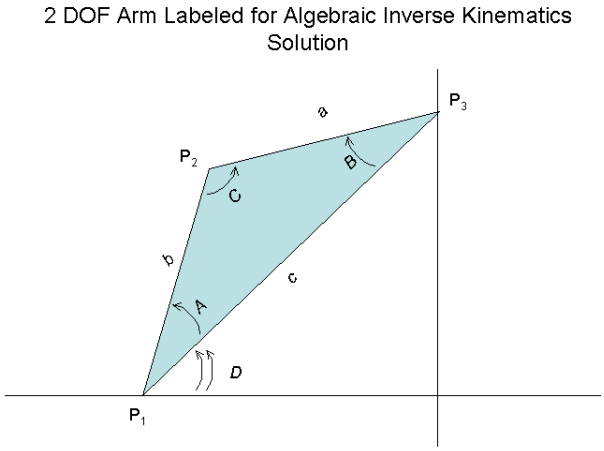

Fig. 2. This figure shows a triangle labeled with the angles and side lengths needed for the 2DOF inverse kinematic solution. Side b and c represent the robot arm segment lengths.

The closed form solutions described above are well proven, however, part of the goal of this work is to find a simple learning solution to the problem. As part of this goal, the criteria for the solution should embody some or most of the following criteria:

- The solution should be simple, robust, and effective.
- The general framework of the solution should be extendable to other problems.
- The nature of the solution should be something that is biologically plausible. That is, are the steps taken to solve the problem something that an animal like a cat would do?

The criteria above outline a desire to find a biologically plausible approach to solving joint control, joint coordination, and body movement, problems. It is readily apparent that all of these problems have been studied from many different angles, including direct solutions using forward and reverse kinematics, neural networks [3], neural oscillators, etc. They have also been approached using various learning systems, paired with genetic algorithms [4], simulators and the like. Each of these approaches has merit. What differentiates the approach described here can be summed up by the following:

"When an animal is learning to position its limbs, balance, and walk, is it using its memory, senses of touch, etc., or is its brain whirring away solving what is the equivalent to sets of equations and the like?"

In all fairness, it is likely that the answer is yes to both sides of the question, or that the questions above are really just one question, posed two different ways. That being said, this work concentrates on the first side of the question.

III. Approach

The approach to the problem was separated into two components, exploration and memory driven action. The exploration is based on simulated annealing.

Simulated Annealing [5] is a problem solving technique which has its roots in metallurgy. When a blacksmith is fabricating an implement he works the metal, letting it cool slowly while he fashions it to the desired shape. The process of slow cooling allows the molecules to locate themselves in the lowest energy configuration, while the working of the metal helps jostle the high energy molecules into the low energy positions, ultimately producing a strong, non-brittle product.

Simulated annealing attempts to model this process. Initially the solution is a random one and the system starting temperature is set to a high value. Once an energy value for the random solution is calculated, the system is perturbed (Analogous to the blacksmith working the metal). An energy value for the perturbed system is calculated. Acceptance of the new solution is controlled by a function that is system temperature dependent. If the system temperature is high, less optimal (higher energy) solutions have a chance of being accepted in the hope that it will lead to an overall lower system energy in the future. If the energy value of the perturbed system is lower than that of the previous solution it will most likely be accepted and the perturbation process will repeat. If a solution is not accepted, the system is returned to its previous state, and the perturbation process is repeated. While this process continues, the system temperature is slowly lowered, making the probability of a higher energy solution less and less likely. When the system temperature reaches the end temperature, the annealing process is complete.

A. *Using Simulated Annealing to Find Joint Angles*

When the simulated 2 jointed leg is trying to reach a position it has not reached before, the annealing/exploration process begins. At its most basic level, it is described by the pseudo code below:

```
if (annealing) {
    d0 = getDist(foot, target);
    myJoint = selectRandomly(hip, knee);
    perturbDirection = getRandomDirection();
    moveJoint(myJoint, perturbDirection);
    d1 = getDist(foot, target);
    if (d1 < d0) {
        keepNewPosition();
        if (d1 is within tolerance to target) {
            annealing = False;
        }
    }
    else {
        restoreOld();
    }
}
```

This basic system will find the joint angles that the leg needs to move its "foot" to the target position. Note: in this version of annealing, only better solutions are accepted; i.e. the probability of accepting a worse solution is zero. In this implementation, the system is written in python, and displayed using the basic vector drawing of the pygame library. Mouse positions selected by mouse clicks provide the target positions.

Patrick McDowell, and Kuo-pao Yang, "Memory Based Learning Augmented with Simulated Annealing for a Two Joint Arm," *International Research Journal of Advanced Engineering and Science*, Volume 5, Issue 4, pp. 244-247, 2020.

The arm/leg will reliably find all positions that are within reach. A timeout has been added to shut the process off if the target supplied is out of reach.

While the target is reliably reached, the movement to the target is not smooth, unless the algorithm is altered somewhat. The modification that is needed, is that once a "good move" is found, that is, once a move is found that brings the end effector closer to the target, that action is continued until either it is not good anymore, or the target is reached. By doing this, the movement is made to be much smoother. This change is reflected in the pseudo code below:

```
if (annealing) {
    d0 = getDist(foot, target);
    if (goodMove) {
        myJoint = goodJointToMove;
        perturbDirection = goodDirection;
    }
    else {
        myJoint = selectRandomly(hip, knee);
        perturbDirection = getRandomDirection();
    }
    moveJoint(myJoint, perturbDirection);

    d1 = getDist(foot, target);
    if (d1 < d0) {
        keepNewPosition();

        if (d1 is within tolerance to target) {
            annealing = False;
        }
        else {
            goodMove = True;
            goodJointToMove = myJoint;
            goodDirection = perturbDirection;
        }
    }
    else {
        restoreOld();
        goodMove = False;
    }
}
```

B. *Using Memory to Return to Old Positions and Find New Ones Quicker*

Each time a targeted position is found, its location and the joint positions needed to reach that location are kept in a memory. If the target needs to be returned to, its joint positions can be retrieved, and the arm can be transitioned from its current position to the retrieved target by moving the hip and knee joints to the retrieved values. This simple system allows for very quick movements between known target positions.

To make finding new positions quicker, a nearest neighbor lookup is used. The previously unseen target location is compared against all known positions. If there is a known position closer to the new target than the current position of the end effector, the arm is moved towards the known position. If at any point, along the way, the new target position is closer than the old position, the system switches to the annealing system to complete the process. To sum up the process, known positions are used to find new positions more quickly.

As can be expected, the longer the process of moving to positions goes on, in general, the more quickly each new position can be found. This is especially true if the memory contains an evenly dispersed set of target positions. The pseudo code below outlines the process:

```
# First we need to see if there is a stored position that is closer
# than the current position of the end effector.
tx, ty = getTargetLocation();
closeOne = getNearstNeighbor(tx, ty, mySpots)
if (closeOne is closer than myArm.endEffector) {
    traverse2Nearest = True;
    nearX, nearY = closeOne.getPosition();
    nearHipAngle,nearKneeAngle = closeOne.getAngles();
}
else {
    annealing = True;
}
```

In the part of the code that controls the movement of the joints this pseudo code transitions the leg/arm towards the saved position, until either it gets there, or it gets closer to the target position than the saved position. Either way, when these conditions are detected the annealing system is then enabled to complete the process.

```
# If there is a position that is closer, we use it to get to the new
# position.
if (traverse2Nearest) {
    # Get current distance of foot from target.
    dist2TargetMyFoot = dist(myLeg.fx, myLeg.fy, tx, ty)

    # Get current distance of foot from target.
    dist2TargetNearSpot = dist(nearX, nearY, tx, ty)

    if (dist2TargetMyFoot > dist2TargetNearSpot) {
        if (myLeg.hipAng < nearHip) {
            myLeg.moveHip(1)
        }
        else if (myLeg.hipAng > nearHip) {
            myLeg.moveHip(-1)
        }
        else if (myLeg.hipAng == nearHip) {
            doNothing();
        }

        if (myLeg.kneeAng < nearKnee) {
            myLeg.moveKnee(1)
        }
        else if (myLeg.kneeAng > nearKnee) {
            myLeg.moveKnee(-1);
        }
        else if (myLeg.kneeAng == nearKnee) {
            doNothing();
        }

        if ((myLeg.hipAng == nearHip) and
            (myLeg.kneeAng == nearKnee)) {
            traverse2Nearest = False;
            # Anneal until the leg gets close to the target.
```


Patrick McDowell, and Kuo-pao Yang, "Memory Based Learning Augmented with Simulated Annealing for a Two Joint Arm," *International Research Journal of Advanced Engineering and Science*, Volume 5, Issue 4, pp. 244-247, 2020.

```
                annealing = True;
            }
        }
        else {
            traverse2Nearest = False
            # Got real close to target on the way to the nearest spot.
            # Anneal until the leg gets close to the target.
            annealing = True;
        }
    }
```

IV. Summary and Conclusions

This paper discusses a simple memory/annealing based methodology of finding the correct joint angles for a two jointed arm to reach target positions. The annealing like exploration is not a true annealing system in that it only accepts perturbations that bring the system closer to a solution, versus a true annealing system that would accept some adverse solutions using a temperature based probability system. The systems effectiveness is enhanced by repeating helpful perturbations until they are found to not be helpful.

The annealing exploration finds solutions that are then stored in a memory that is used to increase the overall efficiency of the system. When a new target position for the arm to reach is supplied, a nearest neighbor recall of the previously found positions is used get a jump on the exploration. By doing this, the system uses known positions to reduce the exploration time needed to reach unknown positions.

Control of the jointed arm is a testbed for this exploration/learning methodology. It is a first step on the way to finding a simple, biologically plausible, method of control that is aimed at joint control, balance, and body coordination problems that occur in simulations and robotics.

References

[1] V. Gupta, R. Chittawadigi, and S. Saha, "RoboAnalyzer: Robot Visualization Software for Robot Technicians," *Proceedings of the Advances in Robotics (AIR '17)*, Article No. 26, 2017.

[2] K. Erleben, S. Andrews, "Inverse Kinematics Problems with Exact Hessian Matrices," *Proceedings of the Tenth International Conference on Motion in Games (MIG '17)*, Article No. 14, 2017.

[3] Simon Haykin, "Neural Networks: A Comprehensive Foundation, Macmillan," pp. 18-26, 1994.

[4] (2013) Genetic Algorithm. In: Dubitzky W., Wolkenhauer O., Cho KH., Yokota H. (eds) *Encyclopedia of Systems Biology*. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-9863-7_100560

[5] van Laarhoven P.J.M., Aarts E.H.L. (1987) Simulated annealing. In: *Simulated Annealing: Theory and Applications*. Mathematics and Its Applications, vol 37. Springer, Dordrecht. https://doi.org/10.1007/978-94-015-7744-1_2

Patrick McDowell, and Kuo-pao Yang, "Memory Based Learning Augmented with Simulated Annealing for a Two Joint Arm," *International Research Journal of Advanced Engineering and Science*, Volume 5, Issue 4, pp. 244-247, 2020.