Harmony Search: Optimization of penalty kick Agent

Maitry Trivedi

Kevin Vora

***Abstract*—A band of musicians playing a trio or quartet, takes years to master which note to play next. Based on this ideology emerges an evolutionary strategy that aims to find an optimal solution. The main idea is to improvise based on imitating musicians. In order to find the optimal solution to a complex quality function, various algorithms like random search and genetic algorithm perform well but there is something unique about the Harmony Search Algorithm(HSA). This report summarizes use of Harmony search algorithm to train a penalty kick agent such that it learns to kick the ball at optimal velocity and angle.**

***Index Terms*—Harmony search, Optimization**

I. Introduction

In a band of musicians, a musician faces dilemma of choosing the note to play next. This selection determines quality of harmony. In order to find a solution to this dilemma, musician can use their knowledge and skill to pick a note that sounds good in context of song or from previous experience. Usually the notes played recently synced alright, so usually they are good choices. One idea an artist keeps in mind is its audience. So it's important to play a note that excites audience by adjusting pitch away from expected note. This results into new and probably better harmony.

In the same way harmony search algorithm has been tailored. There are various parameters that keep track of recent harmony memory (population), selection of new harmony (new solution), modification of existing harmony and few other criteria in order to reach an optimal solution. Harmony search generates "harmonies" of inputs which it then evaluates for quality, and iterates this process until it finds the best one possible.

In this report, we explain how Genetic Algorithm (GA) and Harmony search algorithm (HSA) differ in section II. This section is followed by section III algrithm. Here the entire optimization algorithm is explained. In algorithm section, intuition is translated into mathematical notation and iterating over the core of the algorithm, exploration and exploitation is done to find an optimal solution. Since we know how this algorithm works, we can test it on our simulation environment described in section IV. Section V summarizes the report and provides future direction for extension of the project.

The HSA is a population based meta-heuristic algorithm which is inspired by the musicians. As the musicians tries to improvise the harmonies by practicing over years, the HSA enhance the quality of the solutions over iterations. At the end, all the musician play aesthetic harmony which can excite the audience. Similarly, HSA find the optimum solution for the problem after so many iterations.

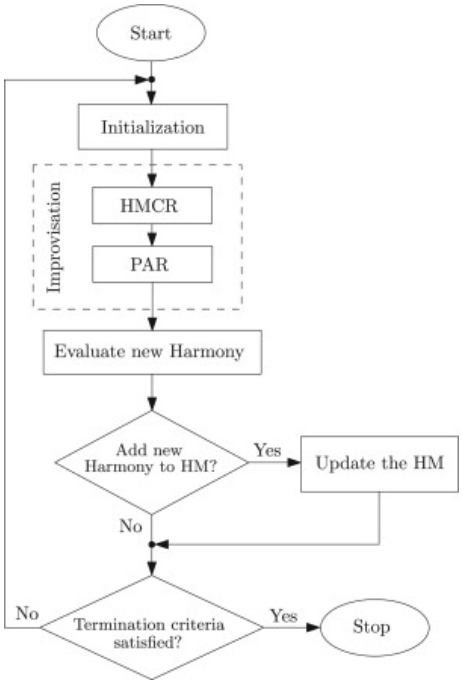

Fig. 1. Flow chart of HSA

The similarities between the improvisation of jazz musicians and optimization task can be summarizes as following [1].

- In improvisation process of musicians, the quality of a harmony is determined by aesthetic estimation. In an optimization process, the quality of a solution vector is determined by the objective function value(Fitness).
- In improvisation process of musicians, the ultimate goal is to obtain the best harmony. In an optimization process, the ultimate goal is to find the global optimum(maxima or minima).
- In improvisation process of musicians, musicians change the pitch of their instruments. An optimization algorithm changes the values of the decision variables.
- In improvisation process of musicians, practice is done to improve the results. In an optimization, each attempt to update a solution vector is called iteration.

$$HM = \begin{bmatrix} Harmony\ 1 \\ Harmony\ 2 \\ \vdots \\ Harmony\ N \end{bmatrix} = \left[\begin{array}{cccc|c} x_{1,1} & x_{1,2} & \cdots & x_{1,d} & f_1 \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} & f_2 \\ \vdots & \vdots & & & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,d} & f_N \end{array}\right]$$

Fig. 2. Harmony Memory Structure

II. Why Harmony Search Algorithm over Genetic Algorithm

The main parameter to compare the meta-heuristic algorithms is their exploration-exploitation strategies. The GA has two exploration-exploitation operators which are cross over and mutation. Whereas the HSA has three exploration-exploitation operators named Harmony Memory Consideration Rate, Bandwidth, and Pitch Adjustment Rate. They drastically affect the performance as they control the speed of the convergence and maintains the exploration and exploitation ratio. Harmony search does not exhibit divergence and it can distinguish the high-performance zone in the solution range within a short time. Therefore, HSA is better than traditional GA.

III. Algorithm

Fig. 1 explain the flowchart of HSA which can be summarized in the following steps:

1) Initialization of Harmony Memory(HM)
2) Improvisation of new harmony
3) Evaluate the new harmony by its fitness
4) If new harmony is better than the worst member in HM, then replace the worst member with new harmony
5) Repeat steps 1-4 until the termination criteria is matched

The detailed pseudo code is explained in Algorithm 1.The structure of Harmony Memory is shown in Fig. 2. The i^{th} Harmony can be represented as [$x_{i,1}$, $x_{i,2}$,...,$x_{i,d}$] and fitness is represented by f_i.

A. Parameter Evaluation

There are mainly three parameters which plays a major role in the improvisation steps named Harmony Memory Consideration Rate, Pitch Adjustment Rate and Bandwidth.

1) **The Harmony Memory Consideration Rate**(HMCR) $\in$ [0,1]. It establish the probability of the solution selecting from the existing list or choosing it randomly. HMCR decides the amount of elitism i.e. it decides how much portion of the previous solution will remain same in the new solution. If the considering rate is set to a low value, the algorithm will converge slowly to the optimum solution; however, a high HMCR value will favor the intensification based on the knowledge contained in the present harmony memory, but at the risk of trapping the algorithm into local optima [5]. As such, it controls the rate of convergence of the algorithm and is typically configured between the range [0.7,0.95].

2) **The Pitch Adjustment Rate** (PAR) $\in$ [0,1]. It establishes the provability of variation of the decision variable.

$$X_{new} = X_{old} + BW.rand(L, U) \quad (1)$$

Using Eq. 1 the decision variable is manipulated by adding some random digit in the old decision variable. In this equation L and U are lower-bound and upper-bound for the given problem which is generally -1 and 1 respectively . BW is the pitch bandwidth. The pitch adjustment can be primarily seen as a refinement procedure of local solutions, while the randomization procedure allows examining the search space in a more explorative fashion.We can assign PAR to control the degree of the adjustment. A low pitch adjusting rate with a narrow bandwidth can slow down the convergence of HSA because the limitation in the exploration of only a small subspace of the whole search space. On the other hand, a very high pitch-adjusting rate with a wide bandwidth may cause the solution to scatter around some potential optima as in a random search [5]. Therefore, PAR is selected between the range [0.7,0.95].

3) **Bandwidth**: If the pitch adjustment mechanism is selected, the value of BW controls the step size of movement. By using large values of BW, the distance between the new value and the HM value increases. Indeed, we can tune the global and local search by BW value [1].

By analyzing these parameters in detail, we can say that HSA has a great exploration and exploitation strategy and it is very necessary to set the parameter values accurately to get good performance of algorithm.

IV. Implementation Details

A. Environment

We have used a virtual robotics experimentation platform (V-REP) [2] for our simulations and results for this project. Our environment is a football field designed in V-REP with a football, a goal post and our striker robot “RoboSparky”. We have set specific dynamics in the V-REP environment for the football and the striker robot which allows it to hit the ball with a defined force towards the goal. We have also put some texture on the objects to make the environment look more realistic (Please refer Fig. 3). Here the agent is our player and the action is the angle with which the agent should hit the ball. Different angles (A,B,C,D,E,F and G in Fig. 4) are achieved by changing the velocities in x directions with respect to the agent (V_x, -V_x, V_y and -V_y in Fig. 4). For the agent to hit the ball, we just assign velocity values in the x and y directions which makes it kick the ball in a specific direction. The agent eventually stops due to the ground friction.

Algorithm 1: Harmony Search Algorithm [4]

1) initialize the harmony memory with HMS randomly generated solutions
2) **repeat**
3) create a new solution in the following way
4) **for** all decision variables **do**
   with probability HMRC use a value of one of the solutions in the harmony memory (selected uniformly at random) and additionally change this value slightly with probability PAR
   Otherwise with probability (1 - HMCR) use a random value for this decision variable
5) **end for**
6) **if** The new solution is better than the worst solution in HM **then**
   Replace the worst Solution by the new one
7) **end if**
8) **until** the maximum number of iterations has been reached
9) **return** the best solution in the harmony memory

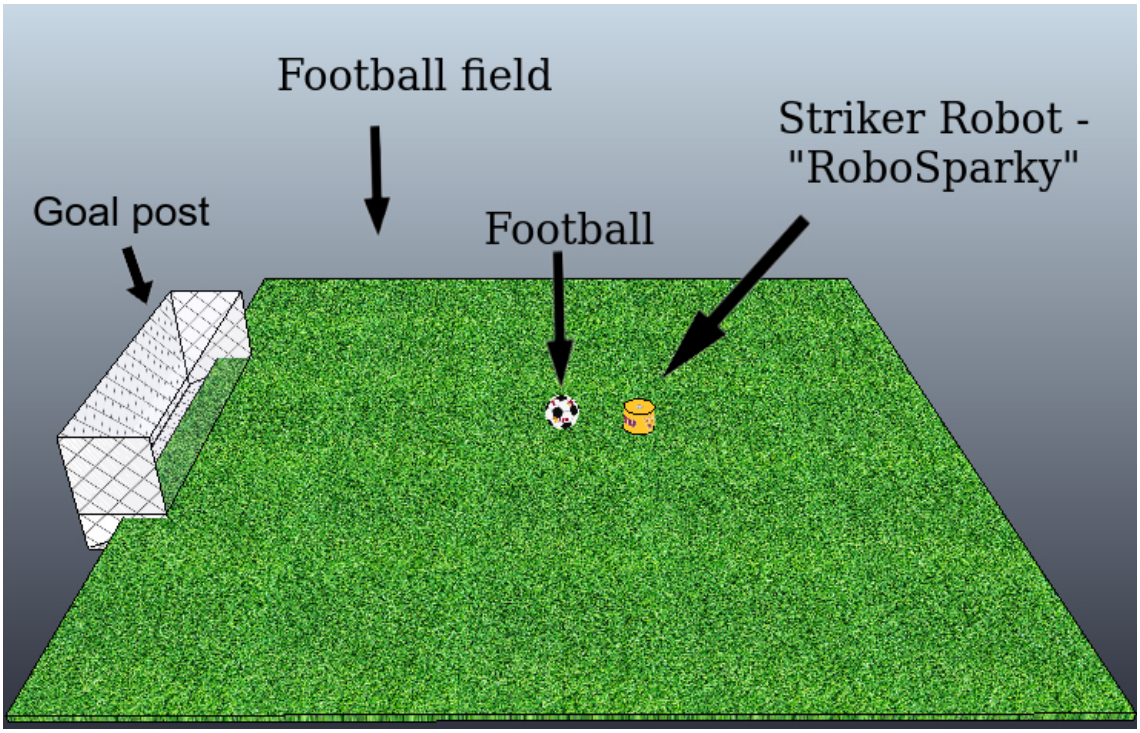

Fig. 3. Our V-REP simulation environment

B. Learning to score via Harmony Search

We can apply harmony search in such a manner that helps the agent to maximize the rewards. Initially, keeping the problem simple, agent gets +1 reward for scoring a goal and -1 reward for missing a goal. It looks like a reinforcement learning scenario but it gets tricky to work on continuous action space in reinforcement learning [3]. Moreover, in our implementation we keep velocity along y direction (V_y) constant and use harmony search to come up with real values of V_x which enables our agent to score a goal. As a result, our agent can explore continuous action space and also exploit the region returning maximum reward. So as described in the algorithm section, Harmony memory is initialized, simulation environment is reset and run for each decision vector and quality of solution is determined.

In order to score a goal, V_y has a constant value of 75 units and we obtain V_x from harmony search. We begin by initializing harmony memory consisting of 10 real numbers ranging from -27 to +27 units. Here, agent can score a goal if value of V_x ranges from -22 to +22. After initializing harmony memory we set our hyper-parameters HMCR and PAR. As explained in the algorithm earlier, the exploration and exploitation is done using HSA by iterating over 50 generations and manipulating decision variables to obtain decision vector that enables the agent to score a goal. So, we begin calculating quality of solution by assigning value of each

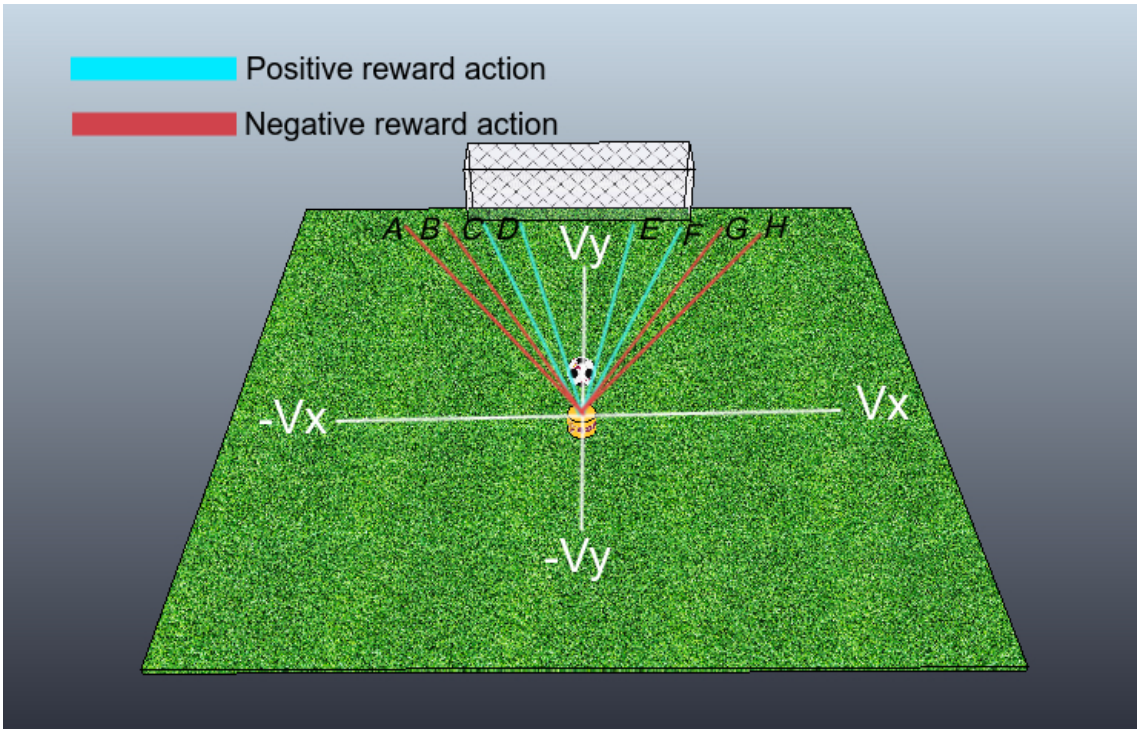

Fig. 4. Action space of the robot

decision vector in harmony memory to V_x and performing the simulation. If a particular member of harmony memory enables the agent to score a goal then fitness is recorded as +1 or else it's -1. There after depending upon random values a new decision vector or new harmony is generated by mixing and matching of decision variables. Again simulation is run with V_x equal to the new harmony. If quality of new solution is better than quality of worst solution in harmony memory, worst harmony is replace by new harmony. In this way, gradually HSA converges to values giving good rewards.

In addition to this we also did one more experiment. We tweaked the reward function so that it helps the agent to learn velocity in y direction such that ball takes least amount of time to reach the goal. In order to take time into account, agent gets *+1 + 1/t* reward for scoring a goal and *-1 + 1/t* reward for missing a goal. In this reward function t is the time taken by ball to reach the goal from the beginning of the simulation process. Center region would be the most optimal region for the agent. This can be inferred as two extreme ends of the goal post are at greater distance while the distance between the ball and the goal post reduces as we move towards the center. Incorporating time component also helps us understand working of harmony search with different optimization function and we can enable the agent to focus on particular region which yields more rewards compared to other areas.

V. Conclusion and Future Work

In summary, evolutionary strategies can be simple and fast to optimize simple reward functions. It also enables us to incorporate continuous action space. Especially harmony search delivers results that can demonstrate exploration and exploitation such that agent actually learns to act as HSA converges towards optimal reward. Experimenting with reward function and analysing the output in simulation enables us to conclude HSA performs very well for this simulation.

In future, there can be multi-modal optimization problem in such a simulation environment. Experimenting with multi agent systems and complex reward function can also be achieved in future.

References

[1] Alireza Askarzadeh and Esmat Rashedi. *Harmony Search Algorithm*. 03 2017.

[2] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.

[3] Hado Van Hasselt and Marco A Wiering. Reinforcement learning in continuous action spaces. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 272–279. IEEE, 2007.

[4] Dennis Weyland. A critical analysis of the harmony search algorithm—how not to solve sudoku. *Operations Research Perspectives*, 2:97–105, 2015.

[5] Xin-She Yang. Harmony search as a metaheuristic algorithm. In *Music-inspired harmony search algorithm*, pages 1–14. Springer, 2009.