EPiC Series in Computing
Volume 69, 2020, Pages 297–306
Proceedings of 35th International Conference on Computers and Their Applications

Implementing Timed Petri net for Modeling and Simulation in Card Gameplay

Garrett Hope, Paul Brodhead, and Seung-yun Kim
Department of Electrical and Computer Engineering
The College of New Jersey
Ewing, NJ 08628, USA
{hopeg1, brodhep1, kims}@tcnj.edu

Abstract

Petri nets (PNs) are a form of directed graph that can be used to model and simulate systems. They are very useful tool for developing and analyzing algorithms, prior to implementation. Adding the component of time, allows for systems with prioritized actions to be modeled more effectively. This paper assesses a Texas Hold'em algorithm using Petri nets and uses them to develop an improved version of this algorithm. Both are implemented in Python scripts to obtain the results to show which is more effective.

1 Introduction

Human Robot Interaction (HRI) is a greatly expanding field within robotics that addresses the interaction between people and robots through communication. Generally, autonomy is required for this type of interaction to exist. The robots must be able to act independently to a certain degree, in order to establish this communicative link. The field of robotics is heavily associated with autonomy, as robots are typically equipped with the necessary technology to perform actions on their own. These robots typically have sensors that gather information, and based upon that information, they make a decision to execute a task. A prominent challenge that persists is designing an artificial intelligence capable of making decisions in an efficient and effective manner. Some common techniques for accomplishing this are machine learning, computer vision, and kinematics [3]. To plan and design the algorithms for these techniques, Petri nets (PN) can be employed. Petri nets are a modelling and simulation tool that have a mathematical and graphical representation. Graphically, they take the form of a finite state automata, and they intuitively represent data flow. Petri nets allow for complex algorithms to be partitioned into a series of "if-then" statements, which is optimal for rule-based systems [1].

While Petri nets are very useful, many variations of them have been developed for more specialized tasks. The element of time has been added to Petri nets so that systems with time constraints can be modeled more accurately. These are known as Time Petri nets (TPN), and Timed Petri nets (TdPN). TPNs introduce intervals in which the net operates, allowing for probability to impact the decisions in the net. TdPNs operate at specific time durations and are more useful in systems with prioritized actions [7].

This paper provides a background on Human Robot Interaction and possible platforms that can be used to study it, such as the Baxter Research Robot, the use of Timed Petri nets (TdPNs) in Section 2.

The selected algorithm is designed for implementation on the Baxter Robot to play Texas Hold'em against a human. Pre-existing algorithms will be identified, modeled, and compared with this new algorithm to demonstrate its effectiveness.

2 Background

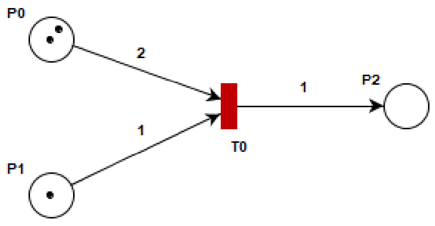
Extensive research has been performed on Human Robot Interaction, and two common platforms for studying it are Nao Robots and the Baxter Research Robot, both of which are humanoids [11-12]. The Nao robots also have received a considerable amount of attention for being used to help teach children with autism [6]. Research conducted by Li et al. used Baxter robot to demonstrate how humans could teach a robot how to act more human-like [5]. Also Baxter has been used as a tool for implementing and testing gameplay algorithms. Chen et al. have employed Baxter to play chess with a human, with the assistance of computer vision [8]. All of these instances of Human Robot Interaction feature some level of autonomy within the robot. A useful tool for modeling these autonomous behaviors are PNs. PNs, and specifically TPNs, excel in modeling systems with ordered data flow and timing constraints

2.1 Petri nets

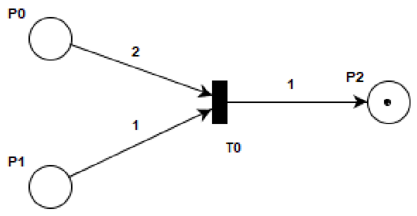
Petri nets (PN) are a modeling and simulation tool that can be represented both mathematically and graphically. They are a valuable tool for analyzing systems, specifically ones with well-defined data flow.

Mathematically [1-3],

- Petri nets represented as a 5-tuple, $PN = (P, T, F, W, M_0)$
- P is a finite set of places, $P = \{p_1, p_2, \ldots, p_k\}$
- T is a finite set of transitions, $T = \{t_1, t_2, \ldots, t_n\}$
- F is a finite set of arcs, $F = \{f_1, f_2, \ldots, f_m\}$
- W is a weight function that describes the number of tokens that an individual arc can move: $W = F \rightarrow \mathbb{N}$
- M_0 is the initial marking, which describes the number of tokens in each place when the Petri net is in its original state, $M: P \rightarrow N$



(a) Before t0 firing



(b) After t0 firing

Figure 1: Graphical representation of Petri net

M_i represents the number of tokens in each place after *i* firings. For example, in Figure 1, $M_0 = (2, 1, 0)$ and $M_1 = (0, 0, 1)$.

Graphically, places are represented with circles, transitions with rectangles, and arcs with arrows. These arcs link places to transitions and transitions to places. Arcs are responsible for producing or deleting tokens, which are represented with dots. The weight of an arc is a positive integer value that is displayed next to the arc. If this is not specified, it is assumed to be 1. In Figure 1, for example, A(P0, T0) = 2, A(P1, T0) = 1, and A(T0, P2) = 1. The tokens in the Petri net are fired by the transitions, but this can only happen when the transition in question is enabled. For a transition to be enabled, all places leading to it must have at least as many tokens as the weight of their respective connecting arcs [2]. In Figure 1a, T0 is colored red because it meets the conditions of this firing rule. In Figure 1b, the result of the Petri net can be seen after T0 fires.

2.2 Time Petri nets

While Petri nets on their own have proven to be a very powerful and useful tool, there are certain situations that they lack the ability to model. This has led to the development of many variations of Petri nets with specialized use cases. An example of this is the addition of fuzzy logic to PNs to create Fuzzy Petri nets (FPNs). FPNs simulate decisions based upon certainty and threshold values [3]. Another example of a specialized PNs are Time Petri nets (TPNs) and Timed Petri nets (TdPNs). All real-world events occur over time, meaning it would be useful to introduce a time variable to model systems with ordered constraints or prioritized actions. TPNs are defined as PNs with time intervals and are expressed as a six-tuple, $TPN = (P, T, F, W, M_0, I)$.

Each interval is associated with a transition and is defined as being between two real, positive numbers, a and b, where $a \leq b$. This means that the earliest a transition is able to fire is at time a, and the latest time is must fire by is b [7]. This variation in initial fire times can be used to incorporate probability into the net's decisions. Timed Petri nets are also represented with a six-tuple: $TdPN = (P, T, F, W, M_0, k)$. In this case, k is a positive integer associated with each transition that determines when the transition can fire [12]. This is more useful when modeling systems with parallel components that must be executed in a certain order.

A popular Petri net editor for Timed Petri nets is TINA (TIme petri Net Analyzer) [13]. TINA allows for a time interval, or timestamp to be associated with each transition. If neither is specified TINA assumes the timestamp is 0, meaning the transition will fire immediately, once its firing rules are satisfied. In Figure 2, each type of these transitions can be seen. Transition t0 has no time marking, so it will fire immediately. Transition t1 has a firing time so it will always fire 5 timesteps after the transition's firing rules are satisfied. Transition t2 has a firing time interval, so it will fire between 1 and 5 timesteps after the transition's firing rules are satisfied.

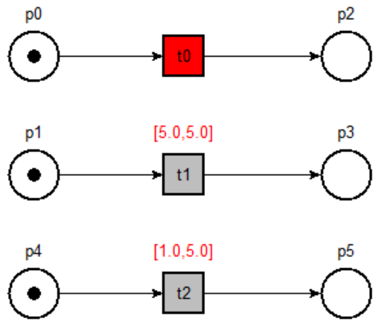


Figure 2: Distinction between timed and time transitions

Once the net is constructed, TINA can simulate the modeled algorithm by moving tokens throughout the PN. The simulator can be single-stepped through, with the user choosing which transition each token goes to, or randomly simulated.

2.3 Baxter Robot

The system that this algorithm is designed for and will eventually be implemented on is the Baxter Research Robot [12]. Baxter has three cameras, two infrared sensors, and one ultrasonic sensor which can all be used to gather information about its surroundings. The robot also has two arms, each with seven degrees of freedom, that are tipped with interchangeable grips. There is also an LCD display capable of providing feedback to the human Baxter is interacting with.

Robot Operating System (ROS) [14] and Python are used to interface with Baxter. The programs that Baxter executes are run on a host machine that the robot communicates with. This means the complexity of the algorithm developed for implementation is only limited by the workstation it is running on.

3 Chen Formula and Petri net Modeling

William Chen is a famous poker player and quantitative analyst who developed an algorithm for determining whether to bet or fold in the first round of Texas Hold'em. The formula takes into account several things: the card values, the gap between the cards, and the suit of the cards. In the formula the function suit can return four possible values:

$$suit \rightarrow \{Clubs, Diamonds, Hearts, Spades\}$$

The value function returns the numerical value associated with the card:

$$value(Ci) = \begin{cases} 10 & if\ Ci = Ace \\ 8 & if\ Ci = King \\ 7 & if\ Ci = Queen \\ 6 & if\ Ci = Jack \\ Ci/2 & else \end{cases}$$

The *max* function returns the larger value of the two cards. This value is used as the initial score of the player's hand. The formula for determining the final score of one's hand can be viewed in Figure 3. The final score is then used to make the decision of whether to bet or fold based upon a predetermined threshold. This score threshold varies based upon several things such as number of players, position at table, and betting limit. For this paper, three-person, limit Texas Hold'em will be used, as three different algorithms will be tested. A score threshold of 4 is used for Chen's formula to determine whether the player should bet or fold.

Chen Formula

```
Chen (C1, C2)
        score = max(value(C1, C2))
        if suit(C1) == suit(C2): score = score + 2
         if |value(C1) – value(C2)| == 0: score = score * 2
        else if |value(C1) – value(C2)| == 1:  score = score + 1
        else if |value(C1) – value(C2)| == 2:  score = score - 1
        else if |value(C1) – value(C2)| == 3:  score = score - 2
        else if |value(C1) – value(C2)| == 4:  score = score - 4
        else: score = score – 5
    End
```

Figure 3: Chen Formula

To model the algorithm using Timed Petri net, five interconnected modules were developed: Initialization Module, Gap Identification Module, Suit Identification Module, Score Identification Module, and the Decision Module.

The Initialization Module, as shown in Figure 4, handles the cards being dealt. At this point the token, which starts in 'Deck' is used to call the Gap, Score, and Suit Identification Modules.

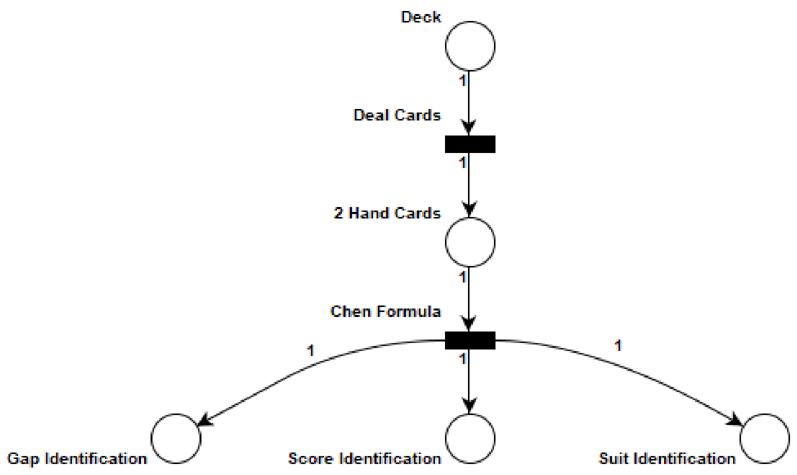


Figure 4: Initialization Module

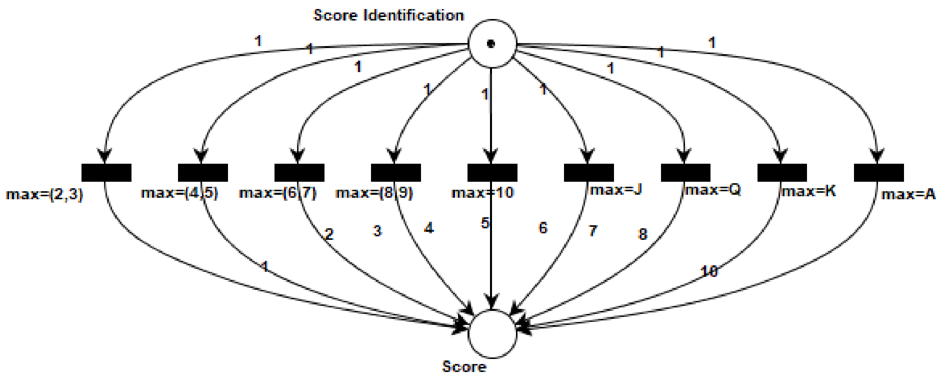


Figure 5: Score Identification Module

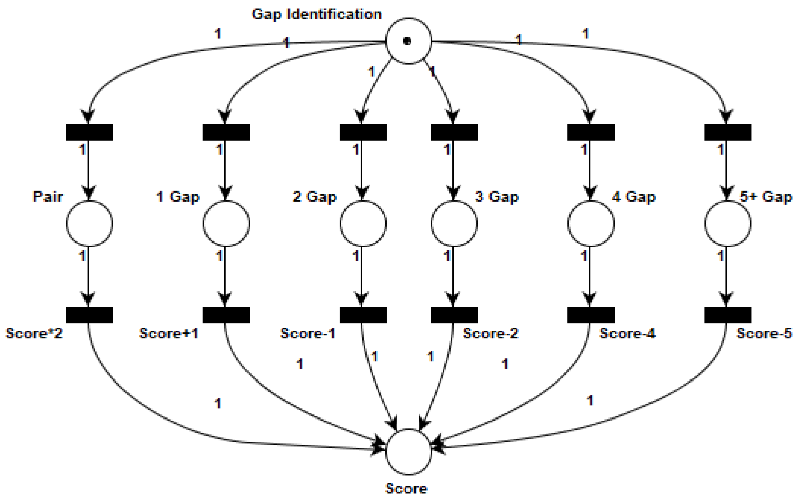


Figure 6: Gap Identification Module

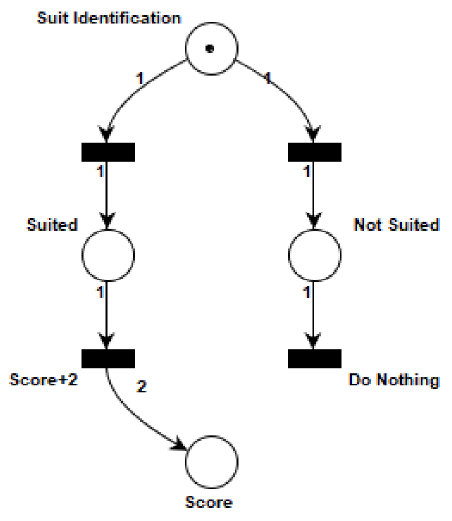


Figure 7: Suit Identification Module

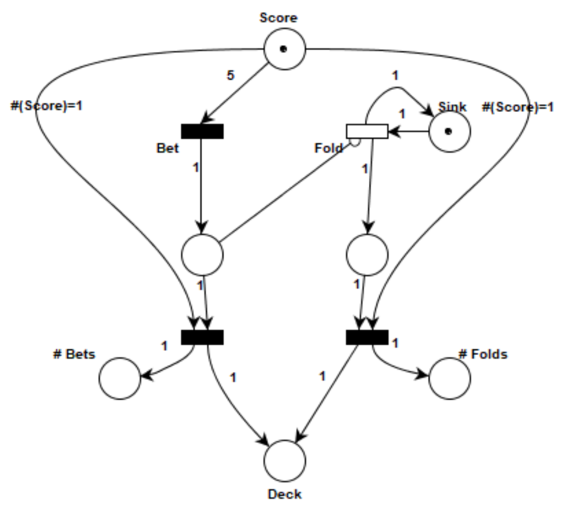


Figure 8: Decision Module

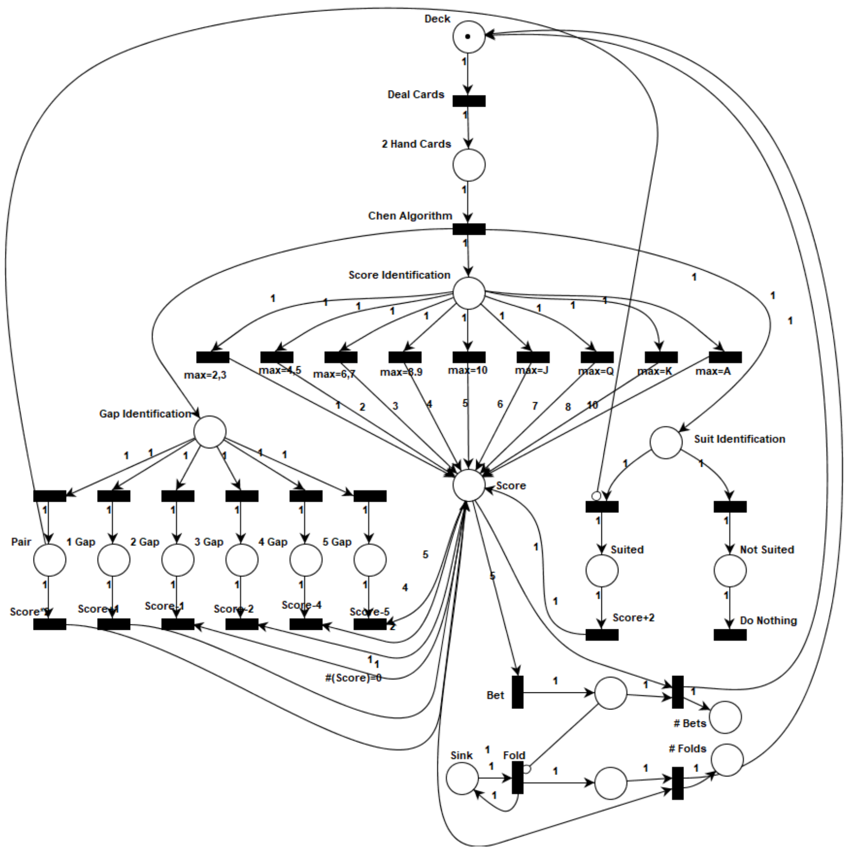


Figure 9: Chen Formula Petri net

By using timed transitions, the next component to activate is the Score Identification Module. The Petri net for this segment is displayed in Figure 5, and demonstrates the process of generating the initial score, based off the values of the two cards in the player's hand. The quantity of tokens in the 'Score' place is then modified by the other modules, and then used to make the decision of whether to bet or fold. The Gap Identification Module is next to execute and is shown in Figure 6. This module detects the how far apart the values of the player's cards are and modifies the number of tokens in 'Score' accordingly.

The Suit Identification Module, Figure 7 then determines whether or not the two hand cards are of the same suit. If they are, three tokens are added to the score. Once all the other modules are executed, the Decision Module, Figure 8, determines whether to bet or fold, adds that decision to a counter, and then returns a token to the 'Deck' so another round can be simulated. The total PN is shown in Figure 9. There is some interaction between modules, for example 'Pair' place inhibits the transition that leads to the 'Suited' place. This is because if the cards are of the same value, they cannot be suited.

4 Algorithm Development

Chen's algorithm is very useful for determining the effectiveness of one's starting hand in a round of Texas Hold'em, however, there are some aspects of the formula that could result in missed opportunities. One dilemma that may become apparent in use is the shortage of weight placed on matching suits. Another concern presented in Chen's formula is the initial value assigned to the cards. Hands containing Queens, Kings, or Aces typically have high expected values for a game of Texas Hold'em. It would be useful to develop a system that categorizes starting hands, and then assigns a score value. One technique for this process is to use the binary representation of each value.

A standard deck of cards contains thirteen different card values. Enumerating these thirteen different values, and then converting to their binary representation allows for the cards to be divided into several different groups. The binary representation of each card can be seen in Table 1. Performing a bitwise AND operation with the binary value '1100' (12) on these 4-bit numbers will result in four subdivisions which can be used to establish an initial score for the algorithm. The results of this operation for each card can be seen in Table 1. This improved algorithm will be similar to the original, in the sense that the initial score will be based off of the card with the higher face value. The score for each card can be seen in Table 1.

Table 1: Binary Representation of Card Values

Card	Decimal	4-Bit Binary	& 1100	Initial Score
2	2	0010	0000	2
3	3	0011	0000	2
4	4	0100	0100	4
5	5	0101	0100	4
6	6	0110	0100	4
7	7	0111	0100	4
8	8	1000	1000	7
9	9	1001	1000	7
10	10	1010	1000	7
J	11	1011	1000	7
Q	12	1100	1100	10
K	13	1101	1100	10
A	14	1110	1100	10

The expected value of a game of Texas Hold'em is typically higher for suited starting hand cards, than if those same cards were not suited. While Chen's Formula does take card suits into account, there is not enough emphasis place on this. The improved algorithm will add 3 points to the score value rather than 2 if the cards are of the same suit.

Chen's Formula also handles the differences between cards effectively. 80% of the final hands in a game of poker are determined based off the values of the cards with respect to one another. Therefore, it is extremely important to adjust the algorithm score based upon this card gap. Cards of the same value typically have high expected values as well, and Chen's Formula prioritizes this very well by multiplying the score value by 2.

Using the methods discussed, the outline for the improved algorithm can be developed, and modeled and simulated with Petri nets. The outline for the algorithm can be seen in Figure 5. The value function used in the Chen Formula will be modified as well:

$$value(Ci) = \begin{cases} 10 & if\ Ci = Ace, King, Queen \\ 7 & if\ Ci = Jack, 10, 9\ 8 \\ 4 & if\ Ci = 7, 6, 5, 4 \\ 2 & if\ Ci = 3, 2 \end{cases}$$

iChen Formula

```
iChen (C1, C2)
        score = max(value(C1, C2))
        if suit(C1) == suit(C2):
            score = score + 3
        if |value(C1) - value(C2)| == 0:
            score = score * 2
        else if |value(C1) - value(C2)| == 1:
            score = score + 2
        else if |value(C1) - value(C2)| == 2:
            score = score - 2
        else if |value(C1) - value(C2)| == 3:
            score = score - 3
        else if |value(C1) - value(C2)| == 4:
            score = score - 4
        else:
            score = score - 5
    End
```

Figure 10: iChen Formula

The Petri net for the iChen Formula follows the same general structure as the Petri net in Figure 9. The differences lie in the Score Identification Module and the arc weights that modify the score.

5 Implementation

To compare the Chen and iChen algorithms, both are implemented in a Python script, along with another algorithm that randomly chooses to bet or fold. Each algorithm is dealt two cards and are required to make the bet/fold decision. The remainder of the game commences (Flop, River, and Turn) and the hands of the players that decided to stay in the game are compared. The graph, shown in Figure 11, displays the results of 10,000 trials. The iChen Formula performed the best, achieving a win percentage of about 39%. The Chen Formula achieved 32% and the Random Formula achieved 29%.

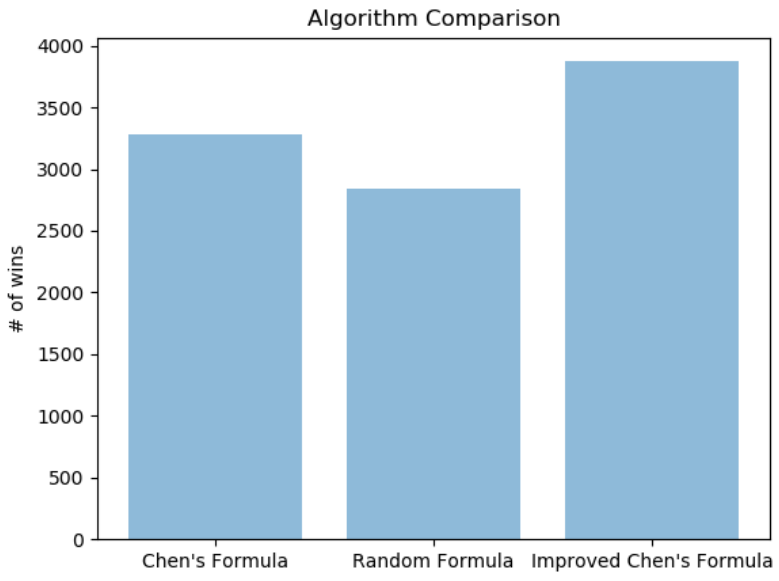


Figure 11: Algorithm Comparisons

6 Conclusion

This paper demonstrates the Petri nets ability to model, simulate, and improve upon complex algorithms. The iChen formula attaining the highest win percentage indicates that it is a more effective strategy to use at the start of a game of Texas Hold'em. The first round of betting is when a player must decide if the cards they've been dealt are likely to win them that hand. The iChen formula is a statistically reliable method for making this decision, and can be implemented effectively in an artificial intelligence application.

7 Acknowledgement

The work was supported by the Office of Academic Affairs at The College of New Jersey, through the Mentored Undergraduate Summer Experience (MUSE) grant.

References

[1] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proc. of the IEEE*, Vol. 77, 2013, pp. 541-574

[2] S-y. Kim, D. Ponsini, and Y. Yang, “Towards a Versatile Opportunity Awareness Algorithm for Humanoid Soccer Robots using Time Petri nets,” *International Journal of Computer Techniques*, Vol. 4, 2017, pp. 82-94

[3] P. Hansen, P. Franco and S. Kim, "Soccer Ball Recognition and Distance Prediction Using Fuzzy Petri Nets," 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, 2018, pp. 315-322.

[4] M. A. Goodrich and A. C. Schultz. Human-Robot Interaction: A Survey. Foundations and Trends in Human-Computer Interaction, 1(3), 27, pp 23-275.

[5] W. Li and M. Fritz, "Teaching robots the use of human tools from demonstration with non-dexterous end-effectors," 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, 2015, pp. 547-553.

[6] M. A. Miskam, S. Shamsuddin, M. R. A. Samat, H. Yussof, H. A. Ainudin and A. R. Omar, "Humanoid robot NAO as a teaching tool of emotion recognition for children with autism using the Android app," 2014 International Symposium on Micro-NanoMechatronics and Human Science (MHS), Nagoya, 2014, pp. 1-5.

[7] J. R. Silva and P. M. G. D. Foyo, “Timed Petri Nets,” Petri Nets - Manufacturing and Computer Science, Aug. 2012.

[8] A. T. Y. Chen and K. I. K. Wang, "Computer vision based chess playing capabilities for the Baxter humanoid robot," 2016 2nd International Conference on Control, Automation and Robotics (ICCAR), Hong Kong, 2016, pp. 11-14.

[9] B. Chen and J. Ankenman, The Mathematics of Poker, 1st editio. Conjelco,

[10] L. F. Teófilo, L. P. Reis and H. L. Cardoso, "Estimating the Odds for Texas Hold'em Poker Agents," 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Atlanta, GA, 2013, pp. 369-374.

[11] Nao Aldebaran Documentation, Available at: http://doc.aldebaran.com/2-1/nao/index.ht

[12] Baxter Research Robot Documentation, Available at: http://sdk.rethinkrobotics.com/wiki/Home

[13] TINA, Time petri Net Analyzer Software, Available at: http://projects.laas.fr/tina//

[14] Robot Operation System, ROS, Available at: http://www.ros.org/