# Survey of Adaptive Algorithms for Intelligent Agents

Nurzhan Sakenov
*Department of Computer Science*
*Nazarbayev University*
Astana, Kazakhstan
nurzhan.sakenov@nu.edu.kz

Ben Tyler
*Department of Computer Science*
*Nazarbayev University*
Astana, Kazakhstan
btyler@nu.edu.kz

***Abstract*—Optimal control and decision making is essential for a wide range of problems, such as resource optimization, real-time scheduling, making decisions based on inaccurate or incomplete data, and reasoning under uncertainty. Examples of such problems include allocation of resources during wildfires, theater missile defense or control of unmanned combat aerial vehicles. One approach to these kinds of problems is adaptive intelligent agents. When mathematically optimal exact solutions are not suitable (for example, dynamic programming is slow, offline and requires perfect knowledge), adaptive agents can approximate optimality with greater speed and ability to handle uncertainty and limited knowledge. Furthermore, using a distributed decision making approach can improve robustness of the overall system in question.**

**In this paper, we will investigate several approaches to designing intelligent agents using neural networks. Neuro-dynamic programming (NDP) is a method of approximate dynamic programming using neural networks. Neuro-Fuzzy dynamic programming (NFDP) is a variation of NDP with incorporated fuzzy logic. There are other methods, such as Genetic Fuzzy Trees and Neuro-Fuzzy Inference Systems. They all differ in their ways of handling the state space and minimizing the relevant objectives. In this paper, we will look at several problems that people have tried to solve using these approaches. We will discuss the types of these problems and features that make these problems well-suited for adaptive agents.**

***Index Terms*—optimal control, adaptive agents, neuro-dynamic programming, reinforcement learning**

## I. Introduction

Optimal control theory is concerned with finding ways to act in a dynamic system so that some optimality criterion is achieved. For example, one such system may be a drone in a maze, and the problem would be to find the sequence of actions that would move the drone to some goal in the maze as fast as possible. Another example is a backpack with limited space and a set of items with size and value, and the problem is to fit as many items in the backpack as possible such that their cumulative value is maximized. These types of problems have traditionally been tackled using Dynamic Programming approaches developed by Bellman in the 1950s. This is an example of optimal control in a deterministic system. However, most real-life problems are inherently stochastic, e.g. resource allocation during forest fires, traffic control, or control of aerial combat drones. There may be elements of uncertainty present; it may not be possible to know the state of the system to its full extent, or the state space might be too large to be computationally feasible.

At this point, we cannot talk about optimal control *solutions* anymore, because we may not be able to find the exact optimum at all. Yet, there are *approaches* that may allow us to find control laws that will approximate optimality. We will discuss a promising approach, which we call Adaptive Intelligent Agents. Where exact methods such as Dynamic Programming fall short due to their inability to handle uncertainty, large dimensionality and partial data, adaptive agents tend to be more flexible and robust via approximate computation and data compression, performing faster and requiring less space. Furthermore, such agents can work together as distributed decision makers, which may improve the effectiveness of the approach.

In this paper, we will investigate several approaches to designing intelligent agents using neural networks. Neuro-Dynamic Programming (NDP) - also known as Approximate Dynamic Programming - is a methodology that combines classical dynamic programming and neural networks, or other approximation structures, to overcome the limitations of DP. Neuro-Fuzzy dynamic programming (NFDP) is a variation of NDP where fuzzy logic is incorporated to make the model more granular and abstract. Other methodologies for finding optimal control strategies include Genetic Fuzzy Trees, based on genetic algorithms and fuzzy logic, and Neuro-Fuzzy Inference Systems, an ensemble of fuzzy inference systems and neural networks. In this paper, we will look at several problems that can be approached using these methods. Moreover, we will discuss the types of these problems and features that make these problems well-suited for adaptive agents.

## II. Motivation Behind Adaptive Agents

For this paper, we consider discrete-time dynamic systems $s_{t+1} = \mathbb{F}(s_t, a_t)$, where $\mathbb{F}$ is the system function. Usually, such systems are modeled as Markov Decision Processes (MDP). If the state at time $t$ is $s_1 \in S$, then an action $a$ can move the system to state $s_2$ with probability $P(s_1, a, s_2)$ and a cost $g(s_1, a, s_2)$. The problem is to find the optimal policy, which is a function $\pi^*$ mapping states to actions such

that the expected cost function

$$J^*(s_1) = \arg\min_{a \in A(s_1)} \mathbb{E}\{g(s_1, a, s_2) + J^*(s_2) | s_1, a\}$$

is minimized. The cost function is some variant of Bellman's Equation. Problems involving such systems are usually solved using Dynamic Programming, which uses tabulation (memoization). [1] However, the complexity of DP calculations grows exponentially as the dimensionality of the state vector increases. Richard Bellman called it "the Curse of Dimensionality": the objective function has to be computed and tabulated for every combination of state (which is usually high-dimensional) and action (which may be a vector), and even the most powerful computers cannot keep up with the growth rate of such problems. [2]

We usually use Dynamic Programming when there is perfect knowledge of the environment, and the system is deterministic. For example, the mosquito growth model in [3] is well-defined by differential equations. However, in real life, we rarely know the dynamics of a system perfectly, if at all. In order to use classical control methods, we need a well-specified model of the system. Real-life problems are often highly complex and have many corner cases, making it difficult to model them adequately. Furthermore, in such systems, an action does not necessarily yield a unique and sure response, due to their stochastic nature. We might extinguish a region of fire with a single burst of water once in a while, but other times we would need more. Finally, Dynamic Programming is inherently off-line, since computing the current optimal policy requires knowing the optimal policy at the next time step, hence it is backwards [4]. Thus, if the environment itself is variable, it is computationally infeasible to recompute the policy, because we would need to recompute the whole table. At the same time, in most real-life problems, we need real-time control.

*General Features of Adaptive Agents*

In situations where classical methods of optimal control, such as Dynamic Programming, are infeasible, we may have to resort to approximate methods. One such approach is Adaptive Intelligent Agents. These agents should operate well with high-dimensional data, i.e. quickly and accurately. Furthermore, should be able to handle uncertainty and noise in the data, as well as potential variations in the environment itself.

## III. Neuro-Dynamic Programming

Neuro-Dynamic Programming (NDP) - also known as Adaptive/Approximate Dynamic Programming - is a class of approaches to complex stochastic decision problems, introduced by Bertsekas and Tsitsiklis in 1996. NDP is supposed to alleviate the Curse of Dimensionality by approximating the cost function $J^*$ via the use of neural networks (or any nonlinear approximator) and simulation. Since we are no longer concerned with exact solutions, we may use techniques of dimensionality reduction, such as feature extraction: representing as vectors containing the most useful information about it. This enables us to approximate these feature vectors by smooth functions, which is not as computationally hard. We describe one of the core NDP algorithms based on policy iteration, as proposed in [1].

**Algorithm 1:** CORE-NDP-POLICY-ITERATION

**Data:** stabilizing suboptimal policy $\pi$
**Result:** approximation of the optimal policy $\pi^*$
**repeat**
  simulate the system trajectories using $\pi$;
  $\tilde{J}(\cdot, r) \leftarrow$ least-squares fit of the simulations;
  obtain a new $\pi$ by minimizing in the Bellman's equation, where $J^*$ is replaced by $\tilde{J}$;
**until** $\pi$ *close enough to* $\pi^*$;

We need Monte Carlo simulation in NDP to generate the training data for the approximator to find the best fit. Furthermore, it allows us to find optimal costs even if the underlying system model is unknown - as long as we can simulate the system. We cannot even apply classical methods (DP) in such cases. Finally, during simulation, some states are going to appear more frequently. This indicates that those states are more "important", and NDP will approximate the optimum better for those particular states.

*Core NDP is Off-Line*

The core version of NDP proposed in [1] computes the tables backwards-in-time, meaning that the algorithm is inherently off-line. The classic paper on Theater Missile Defense [5] describes two off-line policy iteration algorithms. While they were able to successfully train the controller to make nearly optimal decisions, it would not be robust to changes in the environment. Therefore, it is not truly adaptive.

Unfortunately, this is far from enough for real-time adaptive control. However, it is shown in [6] that the solution of Bellman's Equation is a fixed point of some contraction mapping: an admissible policy $a_k = h(s_k)$ converges to its fixed $J_h(s_k)$ by iteration with any initial value $J^0(s_k)$. Therefore, we can reformulate our previous approach with this fact by introducing Temporal Difference (TD). Let $e_k$ be a time-varying residual equation error:

$$e_k = g(s_k, h(s_k)) + \gamma J_h(s_{k+1}) - J_h(s_k)$$

Since our system obeys Bellman's Equation, $e_k = 0$. Therefore, to achieve the best approximation, we would need to solve $g(s_k, h(s_k)) + \gamma J_h(s_{k+1}) - J_h(s_k) = 0$. Solving this equation involves solving a so-called nonlinear Lyapunov equation at each time step, which is difficult to do on-line. Fortunately, we may use nonlinear approximators (actor-critic neural network architectures, for example) to perform a least-squares fit. Incorporating TD into policy iteration allows to perform calculations forwards-in-time [5]. Furthermore, we have to keep in mind that in real life, we can only optimize during a finite time interval. The classical MDP formulation considers the infinite horizon, but we obviously cannot apply this to real-life problems. Thus, we have to use some variant of a finite-horizon control. In [7], it is shown that such an architecture achieves rapid convergence to the optimum.

*True Adaptive Control*

Forward-time policy calculation is the core of real-time adaptive control. This idea is explored in [8], where NDP (here called Reinforcement Learning) is applied to the problem of traffic signal control to solve congestion. To achieve real-time learning, Q-learning is used, which is based on one-step policy iteration (value iteration). The agent perceives the queue lengths and delay time in real time, and it is not aware of the underlying model of the system. Such agent is robust to uncertainty and variability of the environment, since it calibrates the policy continuously (allowed by one-step iteration). In Deep Resource Management (DeepRM), the on-line adaptive agent beat the state of art $Tetris^*$ algorithm by learning a strategy of saving space for small jobs from experience. Furthermore, it was shown that DeepRM can be trained with virtually any objective function, as long as we can provide the agent with relevant percepts [9]. This approach may be crucial for real-time control in ad-hoc systems, such as vehicle platooning in [10], where the controller is not adaptive but trained off-line.

*Distributed Control*

Up to this point, we have considered only single-agent systems. However, many real-life problems are best modeled as multi-agent systems. In [8], multiple traffic signals along a road may cooperate and share percepts to improve their cumulative effectiveness. While cooperation seems beneficial for the system, it comes with the cost of dimensionality, as well as implementation issues. However, we know that distributed systems can be less computationally complex and more robust from examples such as cryptocurrency or peer-to-peer file sharing.

One framework for distributed control problems is based on Game Theory - more specifically, differential graphical games, where the dynamics of the system are represented as a graph [11]. Such systems may be efficiently implemented with NDP, where each agent uses only one neural network, instead of two, to find its optimal policy and synchronize with the other agents. Furthermore, such architecture is guaranteed to be stable regardless of what initial policies are used, and it achieves convergence to so-called Nash Equilibrium. A similar approach may be used to design a controller for multiple cooperating drones in wildfire management in [12].

*Performance*

Neuro-Dynamic Programming was designed to overcome the Curse of Dimensionality, while being as close as possible to the exact method of Dynamic Programming. Off-Line NDP performs nearly as badly as Dynamic Programming in general (in terms of speed), but it allows to operate within larger state spaces. In Theater Missile Defense, performance of NDP in terms of remaining asset health is compared to a heuristic solution, and the exact solution via DP (only small problem sizes were considered). NDP produced well-enough results, meaning the deviation was not large, and while the heuristic sometimes outperformed NDP on small problem sizes, it is not guaranteed to consistently perform better in general [5]. In Forest Stand Management, NDP handles a huge state space and produce better granularity of policy compared to the state-of-art methods [13]. In Post-Hazard Planning, rollout algorithm is used to improve upon the base NDP policy, achieving significant increase in performance [14].

It is more informative to compare on-line NDP with traditional non-adaptive algorithms. DeepRM achieves performance better that the state-of-art $Tetris^*$ after just 200 iterations, converging to the optimum at 1000 iterations, with each iteration taking about 80 seconds on a 24-core CPU. Furthermore, in Traffic Signal Control, the adaptive controller outperformed the traditional pretimed controllers, especially with highly variable traffic flows after about 200-400 epochs, where one epoch is equivalent to a 2-hour peak period [8]. Finally, in distributed graphical games, the proposed NDP algorithm improves the state of art significantly both in execution time and objective function minimization [11].

## IV. Neuro-Fuzzy Dynamic Programming

Off-line NDP operates poorly under uncertainty. Neuro-Fuzzy Dynamic Programming (NFDP) is introduced in [12] to make the agent robust to uncertainty. The number of neural networks is increased to account for the uncertainty in the system, and a fuzzy term is added to the optimality equation. The fuzzy term is essentially a Fuzzy Inference Controller, operating on a set of IF-THEN rules, mapping them (defuzzifying) to a fractional number between 0 and 1, in order to provide granularity compared to binary logic.

In Wildfire Resource Allocation, NFDP performs dramatically better than both DP and NDP in situations with and without various sources of uncertainty. Furthermore, it is significantly faster than DP and NDP, making it possible to perform real-time control. However, NFPD is an off-line approach and is not truly adaptive.

## V. Genetic Fuzzy Trees

Standard Fuzzy Inference Systems suffer from the curse of dimensionality. Furthermore, creating such systems is computationally hard. Genetic Algorithms may be used as a meta-learning tool to learn the parameters of fuzzy systems. However, Genetic Algorithms bring huge computational costs due to searching in a potentially infinite solution space.

Genetic Fuzzy Trees (GFT) group together several Fuzzy Inference Systems, each trained by a Genetic Algorithm in a directed graph. By doing this, a GFT breaks down a big decision into smaller interconnected decisions, shrinking the solution space. Furthermore, this kind of architecture has an additional advantage of being able to incorporate any algorithm as a node in the graph, making it very flexible. This approach was used in Unmanned Combat Aerial Vehicle control [15] and prediction of brain response to Lithium in patients with bipolar disorder [16].

In Aerial Combat Drones, the input is 150-dimensional, which is intractable with traditional algorithms, but Genetic Fuzzy Trees makes it possible for the controller to be highly

dynamic and responsive even on small computers such as Raspberry Pi. Furthermore, the controller defeated all the state-of-art AI agents. Furthermore, due to its fuzzy nature, the system provides transparency needed for easy verification of its requirements and validation of its goals by human beings. The system was extensively tested by an experienced fighter pilot, who could not even land a single kill on the AI aircrafts. In Lithium Response Prediction, the agent outperformed eight other comparator models and showed nearly perfect classification accuracy in twenty subjects.

## VI. Conclusion

In this paper, we provided a survey of existing approaches to designing Adaptive Intelligent Agents. They should be beneficial when dealing with high-dimensional decision problems. Moreover, they may be the only way to get an acceptable policy if the dimensionality is too large, if there is a lot of uncertainty in the system, or if the dynamics of the system in questions are partially or completely unknown. Adaptive Agents can be on-line or off-line, and one may need to choose between them depending on the nature of the problem. If the system is expected to change a lot, then it is better to use on-line approaches.

## VII. Acknowledgments

I want to say big thanks to Professor Mona Rizvi for being there for me when I had my emotionally dark moments during the semester when I wrote this paper. I made it through Spring 2019 alive, and that's nice. Also, beer.

## References

[1] D. P. Bertsekas, "Neuro-dynamic programming: An overview," in *Encyclopedia of Optimization*, 1st ed., C. A. Floudas and P. M. Pardalos, Eds. Boston, MA: Springer, 2001, pp. 1687–1692.

[2] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 2015, vol. 2045.

[3] "Neuro-dynamic programming approach to optimal control of spreading of dengue viruses." *2018 International Conference on Computer, Control, Informatics and its Applications (IC3INA), Computer, Control, Informatics and its Applications (IC3INA), 2018 International Conference on*, p. 169, 2018.

[4] F.-Y. Wang, H. Zhang, D. Liu *et al.*, "Adaptive dynamic programming: An introduction," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 39–47, 2009.

[5] D. P. Bertsekas, M. L. Homer, D. A. Logan, S. D. Patek, and N. R. Sandell, "Missile defense and interceptor allocation by neuro-dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 30, no. 1, pp. 42–51, 2000.

[6] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[7] D. Wang, D. Liu, and Q. Wei, "Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach," *Neurocomputing*, vol. 78, no. 1, pp. 14 – 22, 2012.

[8] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[9] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16. New York, NY, USA: ACM, 2016, pp. 50–56.

[10] V. Turri, O. Flardh, J. Martenssont, and K. Johansson, "Fuel-optimal look-ahead adaptive cruise control for heavy-duty vehicles," 06 2018, pp. 1841–1848.

[11] M. Mazouchi, M. B. Naghibi-Sistani, and S. K. H. Sani, "A novel distributed optimal adaptive control algorithm for nonlinear multi-agent differential graphical games," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 331–341, 1 2018.

[12] N. Hanlon, B. Tyler, M. Kumar, and K. Cohen, "Fuzzy-based approaches to decision making and resource allocation during wildland fires," in *AIAA Infotech at Aerospace Conference and Exhibit 2011*, vol. AIAA 2011, no. 1450, 03 2011.

[13] J. Comeau and E. Gunn, "A neuro-dynamic programming approach to the optimal stand management problem." *Canadian Journal of Forest Research*, no. 6, p. 808, 2017.

[14] S. Nozhati, Y. Sarkale, B. Ellingwood, E. K. Chong, and H. Mahmoud, "Near-optimal planning using approximate dynamic programming to enhance post-hazard community resilience management," *Reliability Engineering & System Safety*, vol. 181, pp. 116–126, 2019.

[15] N. Ernest, D. Carroll, C. Schumacher, M. Clark, K. Cohen, and G. Lee, "Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions," *Journal of Defense Management*, vol. 6, no. 1, pp. 1–7, 3 2016.

[16] D. E. Fleck, N. Ernest, C. M. Adler, K. Cohen, J. C. Eliassen, M. Norris, R. A. Komoroski, W.-J. Chu, J. A. Welge, T. J. Blom *et al.*, "Prediction of lithium response in first-episode mania using the lithium intelligent agent (lithia): Pilot data and proof-of-concept," *Bipolar Disorders*, vol. 19, no. 4, pp. 259–272, 6 2017.