Survey on Optimal Control Algorithms

Ulugbek Adilbekov
Computer Science Department
Nazarbayev University School of Science and Technology
Astana, Kazakhstan
ulugbek.adilbekov@nu.edu.kz

I. Introduction

There exist millions of systems in the world which are controlled by humans. Those might include: logistics; piloting aircraft; delivery services; transport traffic control and others. However, a human cannot always optimally control a system, especially if a system is somewhat large. In such cases, it becomes necessary to hire and train dozens of operators who will cooperate and collectively try to make system work efficiently. Also, there might not exist an option of using humans in system due to the working environment being too dangerous. Examples may include: future systems which will operate in space; reconnaissance systems and etc.

In such cases it is very good to have a system which could operate and control itself in an efficient way, which could be better or at least comparable to human. Using optimal control algorithms is superior in many aspects:

- It is cheap. If a system is controlled by itself then trained operators become redundant.
- Its capabilities are enormous. An efficient algorithm may achieve optimal/nearly optimal solutions.
- It can work in almost any environment: warfare, underwater, space.

These reasons inspired much work to be done in this field. This survey aims to cover and give a brief overview of the state-of-the-art of optimal control, as well as give some insights of inner processes which are key to success in the field.

Section II covers formal definition of an optimal control problem, discusses what are some of the common types of it and their limitations. Section III introduces Dynamic Programming, the "gold standard" of optimality and its role in optimal control both in past and now. Section IV talks about Metaheuristic search algorithms which are successfully used in optimal control problems. Section V gives an overview of Fuzzy Logic and presents its advantages when applying to optimal control problems.

II. Optimal control theory

To have a good image of optimal control it is necessary to formalize how an optimal control problem looks like. The system can be either discrete-time or continuous-time. Discrete-time system can be described with the following equation

$$x(k+1) = F[x(k), u(k), k], k = 0, 1, \dots \quad (1)$$

In here, $x \in \mathbb{R}^n$ represents a state vector and is a function of k, which stands for number of an event, in other words time, while $u \in \mathbb{R}^m$ denotes control action vector. $F: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a function which takes a state and control action as input and produces a new state as output. Clearly to optimize the system it is necessary to wisely pick u and thus a need for performance index is needed, which will help in distinguishing between good and bad control actions:

$$J[x(i), i] = \sum_{k=i}^{\infty} \gamma^{k-i} U[x(k), u(k), k] \quad (2)$$

where J is the performance (cost-to-go) function, which assesses cost of all remaining states starting from $x(i)$, U is utility function assesses goodness of a particular state and control action and γ is discount factor, which controls how the future decisions affect current performance, as it may range from problem to problem, with $0 < \gamma \leq 1$. Given such a setup for optimal control problem the goal is to find such a sequence of $u(k)$ $\forall$ $k \geq 0$ to minimize J, particularly, given $x_0 \in \mathbb{R}^n$ as initial state, $J(x_0, 0)$.

The formulation of J depends on "life expectancy" of the system [11]. Problems can be either finite horizon where the cost is accumulated over finite number of steps or infinite horizon where costs accumulate indefinitely. There are several approaches of how to accumulate costs in infinite horizon problems, such as discounted (example above), stochastic shortest path, and average cost per stage.

If a system is modeled by nonlinear dynamics or the cost-to-go function is not quadratic in state and control, then to find sequence of optimal control actions it will be necessary to find solutions to Hamilton-Jacobi-Bellman equations (HJB) equation [1] which is a nonlinear partial differential equation in most cases. But, when a problem scales it is and/or there is a need for a fast real-time computation running dynamic programming in order to obtain HJB solutions may be computationally unfeasible, due to the "curse of dimensionality" [2], [3].

This survey omits equations for continuous-time system for the sake of simplicity, but any of the proposed approaches below can be easily adapted from discrete-time case, by dividing the timeline into a number of small time periods.

A number of various technics and methodologies are used to be able to efficiently solve and/or approximate optimal control problems.

III. Dynamic Programming

Dynamic programming (DP) is considered as gold standard is solving problems which include any kind of optimization.

In 1954, Richard Bellman introduced theory behind dynamic programming [9]. It was proposed because at the time mathematicians in problems such as optimal control problems would assume checking every possible sequence of controls for being the most optimal one, which was not the most efficient way.

DP break a multi-period planning problem into simple steps at different points of time. When making a control decision, it is not necessary to know what decisions were made before or what decisions will be made next. It is sufficient to maintain a general prescription which determines at any given stage the decision to be made in terms of current state of the system. In other words, if information described by state is enough to make a decision then it is never necessary to have any knowledge of other decisions [9].

A classic example of DP is knapsack problem. Given a knapsack having C capacity (say in kilograms) and a number of items, which can be put into knapsack. Each item i has a weight w_i and value v_i. The goal is to pick a set of items which maximizes total value.

The state in this problem would be represented by (LC, i), where LC is capacity left in the knapsack and i is the item being considered. There are two possible decisions in this state, either pick it or not. No need for storing previous decision as the only information used to come up with decision is state variables.

Dynamic programming follows Bellman's Principle of Optimality, which says "An optimal policy [sequence of decisions] has the property that whatever initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" [9].

Although, dynamic programming used to be the state-of-art, it is not usually used solving optimal control problems, because it is computationally expensive as the system may be nonlinear or the cost-to-go function may not be quadratic in state and control. But, more often than not, it is not necessary to obtain the most optimal solution, close to optimal works fine in most of the cases.

For these cases Adaptive Dynamic Programming (ADP) was introduced. It is a synonym for "Neuro-dynamic programming", "Heuristic Dynamic Programming", "Reinforcement Learning" [10]. Main idea of ADP is to approximate DP solutions by using function approximation structure such as neural networks to approximate the cost function.

Artificial neural network (ANN) is a computational model which is not explicitly programmed, but rather trained. In the context of this survey ANNs are given samples from utility function U and are expected to produce a good and computationally fast approximation for cost-to-go function

ADP replaces the $J(x)$ cost-to-go function with $\hat{J}(x)$ scoring (approximate cost-to-go) function, where x is a state vector. For example, in Heuristic Dynamic Programming which is one of the simplest versions of ADP neural network minimizes the following error measure over all possible states

$$E_h = \frac{1}{2}\sum_{x(k)}[\hat{J}\big(x(k)\big) - U\big(x(k)\big) - \gamma\hat{J}\big(x(k+1)\big)]^2 \quad (3)$$

There exist more sophisticated versions of ADP, such as Dual Heuristic Programming, Globalized Dual Heuristic Programming, Single Network Adaptive Critic and more.

Overall, ADP shows great capabilities given little computational power and is a very promising in solving optimal control problems.

IV. Metaheuristics

An optimization problem is the problem of finding the best solution among set of all possible solutions. In such formulation, optimal control problem is basically a combinatorial optimization problem.

Metaheuristic algorithms are successfully used for optimization problems, so it is an obvious choice for optimal control problems when dynamic programming is not computationally possible. Metaheuristics are inspired by concepts from different fields of science such as biology, genetics, physics and others. Examples of metaheuristics include simulated annealing, tabu search, iterated local search, variable neighborhood search, ant colony optimization, genetic algorithms and etc. There is a number of works, where metaheuristics are successfully applied to optimal control problems [4], [5], [6].

This survey will only overview genetic algorithms, as it is one of the most used metaheuristic used in optimal control problems.

Genetic algorithms (GA) are a family of computational models inspired by evolution and natural selection. These algorithms encode a potential solution to a problem in chromosome-like form, and apply recombination operators to the solution in a way as to preserve critical information and add up to it. In this paper, GA are only viewed as function optimizers, although range of problem to which GA can be applied is broader.

There are several main components of GA. Those are: solution encoding, selection and breeding functions.

Solution encoding is a form of data which will contain whole information about a single solution. In case of Travelling Salesman Problem (TSP) it may be a list of cities given by their numbers, a route. It is highly desirable to have a good solution encoding in order to minimize the solution state space. A bad example of solution encoding for a TSP would be storing a solution as a bit string, each consecutive n bits denoting next city on a route. In this case, there would be many useless states which would harden the task for GA.

Procedure for GA will be as following. First, a set of initial solutions (population) must be generated, usually randomly. After creation of initial population GA is executed for several iterations.

Iteration starts with current population. Each solution is given a corresponding fitness from a cost function. Selection is applied to current population to produce intermediate population. Most common selection methods include: choosing top X% by fitness; roulette wheel style polling, when some number of randomly chose solutions is eliminated with low fitness solutions having higher chances of being eliminated; or tournament style polling. Then breeding functions are applied to intermediate population to create next population.

There are two common breeding procedures: crossover and mutation, but of course one can come up with a completely different breeding procedure, depending on the problem. Crossover is applied to two solutions which are interchanged in some way and then one or more offspring solutions are produced. As their parent solutions survived selection and possess good fitness it is likely that the offspring solution will have decent fitness too. It is also possible to apply crossover to more than two solutions at a time. It is also possible to successfully apply a self-crossover to a solution [7].

Mutation is applied to all solutions produced in the new population with usually low probability. When solution is represented with a bit string, a bit may flip with some probability. Mutation is necessary for 'freshness', it ensures that GA does not get stuck in local minima.

GA was successfully exploited for solving optimal control problems [8] and, with integration with other methods is a very powerful tool.

V. Fuzzy logic

Conventional binary (crisp) logic describes much of systems today. But when it comes to optimal control and decision making it does not seem natural, because people do not think binary. For example, a person in a room does not measure temperature to say it is below $10°C$, but rather he would say it is cold.

Fuzzy logic [12] extends common binary logic from $\{0, 1\}$ to $[0, 1]$. That means that a variable in fuzzy logic may have some degree of membership to a classification group. Fuzzy logic also allows a variable to be included into several classification groups with possibly different degrees of membership. This type of classification allows to apply a combination of if-then rules based on membership degrees in these classification group. This rule base is developed from expert knowledge of the system and can solve many problems without investigation into complex and computationally expensive equations.

Each classification group is usually represented as a linguistic term and several linguistic terms form a linguistic variable. For example, temperature in the room is a linguistic variable and linguistic terms may be $\{cold, warm, hot\}$. The process of breakdown of crisp input value into its levels of membership to each classification group is called fuzzification. For example, depending on the implementation of membership functions, $20°C$ can be fuzzified as hot for 0.8 and warm for 0.2.

Membership functions may have different forms, such as triangular, trapezoidal, Gaussian, piecewise linear and etc. The most common are triangular and trapezoidal forms. The choice of membership function is often crucial part of Fuzzy Logic Systems (FLS).

When FLS produces output it is necessary to defuzzify it into crisp value. There exist different types of defuzzification, too [13]. The most common one is called Center of Gravity. It works as if degrees of membership to classification groups were physical objects on a plane, and the output value would be point where center of gravity would be.

To sum, FLS has 4 main parts: fuzzification, rule base, defuzzification and inference engine. Inference engine is an inner system which takes fuzzy inputs and matches it against rules producing fuzzy output.

One of the most appreciated properties of fuzzy logic is its robustness. When there is even large noise in the input it still performs well. But it is very important to have proper membership functions and rule base. Sometimes Fuzzy Logic Systems are treated as cost function to some optimization algorithms, such as GA. Input variables to GA are parameters of FLS. It allows achieving maximum performance of an FLS.

This all allows Fuzzy Systems to be successfully approximation technique used in optimal control. While it is not optimal in every situation it is most useful when fast computation is needed, which is crucial in real-time optimal control problems.

Fuzzy Systems are often tightly integrated with other techniques such as Dynamic Programming and Genetic Algorithms. In such cases Fuzzy Systems add more robustness to the system and increase speed of obtaining solutions, while maintaining 'goodness' of solutions on the same level.

Fuzzy Systems also successfully serve in solving techniques such as cluster-first route-second serving as intermediate stage between input and main solving algorithm. Ernest [14] successfully used Fuzzy c-means algorithm for that purpose.

More complex Fuzzy Systems such as Fuzzy Tree are successfully used in very complex problems such as control of squad of air fighters [15].

VI. Conclusion

This paper's aim was to give an overview on the state-of-art in optimal control. The most relevant tools were introduced as well as examples of their applications. Symbioses of presented techniques were also covered.

The property all presented state-of-art tools share in common is approximation. Luckily, optimal control problems do not inquire solutions to be perfect, but rather close to optimal, which made techniques presented viable.

The field still has much space to grow and more researches to be done, especially in area of combinations of different techniques.

VII. References

[1] F. L. Lewis and V. L. Syrmos, *Optimal Control*. New York, NY: Wiley, 1995.

[2] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.

[3] S. E. Dreyfus and A. M. Law, *The Art and Theory of Dynamic Programming*. New York, NY: Academic, 1977.

[4] A. H. Borzabadi and H. H. Mehne, *Ant Colony Optimization for Optimal Control Problems*. Journal of Computing and Information Science in Engineering, 2009

[5] Michalewicz, Z., Janikow, C. Z., & Krawczyk, J. B. (1992). A modified genetic algorithm for optimal control problems. Computers & Mathematics with Applications, 23(12), 83-94.

[6] Huang, Hsu-Chih. "A Taguchi-based heterogeneous parallel metaheuristic ACO-PSO and its FPGA realization to optimal polar-space locomotion control of four-wheeled redundant mobile robots." IEEE Transactions on Industrial Informatics 11, no. 4 (2015): 915-922.

[7] Ernest, Nicholas, and Kelly Cohen. "Self-crossover based genetic algorithm for performance augmentation of the traveling salesman problem." In Infotech@ Aerospace 2011, p. 1633. 2011.

[8] Ernest, Nicholas D. "UAV Swarm Cooperative Control Based on a Genetic-Fuzzy Approach." PhD diss., University of Cincinnati, 2012.

[9] Bellman, Richard. The theory of dynamic programming. No. RAND-P-550. RAND CORP SANTA MONICA CA, 1954.

[10] Wang, Fei-Yue, Huaguang Zhang, and Derong Liu. "Adaptive dynamic programming: An introduction." IEEE computational intelligence magazine 4, no. 2 (2009).

[11] Bertsekas, Dimitri P., and John N. Tsitsiklis. "Neuro-dynamic programming: an overview." In Decision and Control, 1995., Proceedings of the 34th IEEE Conference on, vol. 1, pp. 560-564. IEEE, 1995.

[12] Zadeh, Lotfi A. "Fuzzy sets." Information and control 8, no. 3 (1965): 338-353.

[13] Mendel, Jerry M. "Fuzzy logic systems for engineering: a tutorial." Proceedings of the IEEE 83, no. 3 (1995): 345-377.

[14] Ernest, Nicholas D. "UAV Swarm Cooperative Control Based on a Genetic-Fuzzy Approach." PhD diss., University of Cincinnati, 2012.

[15] Ernest, Nicholas, David Carroll, C. Schumacher, M. Clark, and K. Cohen. "Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions." J Def Manag 6, no. 144 (2016): 2167-0374.