Received: February 3, 2025; **Accepted:** May 10, 2025; **Published:** July 1, 2025.
DOI. 10.30473/coam.2025.74474.1306
Summer-Autumn (2025) Volume 10, Issue 2, (255-271)
Research Article

Open Access

Control and Optimization in Applied Mathematics - COAM

Novel Schemes for Approximate Solutions of Optimal Control Problems via a Hybrid Evolutionary and Clustering Algorithm

Maria Afsharirad

Department of Applied Mathematics, University of Science and Technology of Mazandaran, Behshahr, Iran.

Correspondence:
Maria Afsharirad
E-mail:
m.afsharirad@mazust.ac.ir

How to Cite
Afsharirad, M. (2025). "Novel schemes for approximate solutions of optimal control problems via a hybrid evolutionary and clustering algorithm", Control and Optimization in Applied Mathematics, 10(2): 255-271, doi: 10.30473/coam.2025.74474.1306.

Abstract. This paper presents a hybrid scheme for solving optimal control problems. Discretizing the time interval and assuming a constant control value on each sub-interval transforms the optimal control problem into an assignment problem. To cluster feasible solutions, a novel method is proposed in this paper, which applies metaheuristic algorithms— specifically genetic algorithms and particle swarm optimization— to generate a large number of solutions. Subsequently, the K-means clustering method is employed to classify these solutions into clusters. Enhancing the median of each cluster, using metaheuristic techniques, ultimately results in improved medians. The best median from the final iteration of the algorithm serves as an acceptable solution for the optimal control problem. In some cases, it even succeeds in discovering a new best solution.

Keywords. Clustering, K-means algorithm, Optimal control problem, Genetic algorithm, Particle swarm optimization.

MSC. 49J22; 68T20.

1 Introduction

In the mid-20th century, few anticipated that the emerging control theory would play such a pivotal role across various scientific and industrial fields. Resource constraints prompted the integration of optimization techniques with control theory, leading to the development of Optimal Control Problems (OCPs), which remain among the most extensively explored interdisciplinary topics. The ever-increasing advancements in computing power combined with the invention of novel computational algorithms, often intertwined with various branches of mathematics, have driven researchers to develop innovative, efficient, resilient, algorithms. These advancements have facilitated the discovery of approximate solutions to complex OCPs.

Traditionally classified into direct and indirect methods, solutions to OCPs have increasingly relied on evolutionary algorithms, particularly as a form of direct methods. Despite their limitations for large-scale problems, evolutionary algorithms have demonstrated their capability to provide suitable solutions, even for highly nonlinear problems, making them widely applicable for tackling OCPs.

An ant colony optimization (ACO) framework is introduced in [3] to optimize control trajectories, leveraging ACO's strength in effectively exploring complex solution spaces. The study demonstrates the algorithm's effectiveness in handling nonlinear dynamics and constrained optimization problems.

In [6], innovative numerical approaches are proposed to utilize evolutionary computations for solving OCPs, emphasizing the efficient optimization of control trajectories. By addressing computational challenges and validating results on test problems, the study highlights the robustness of evolutionary methods as valuable tools for complex optimal control problems.

The application of evolutionary algorithms to multi-robot interaction in OCPs is explored in [13], where the authors focus on enhancing collaborative dynamics among mobile robots through optimized control strategies. Similarly, in [17], evolutionary algorithms are employed to estimate control input ranges, demonstrating their utility in designing feasible and adaptable control systems, with detailed analyses on performance and precision. More recently, [11] applies numerical methods to convert nonlinear OCPs into quadratic forms, simplifying the solution process.

A common approach to tackle OCPs involves transforming them into Quasi-Assignment Problem (QAP). In [10], a Tabu Search matheuristic is proposed for solving the Generalized QAP, integrating Tabu Search with mathematical programming techniques to enhance solution quality and convergence speed. A Genetic Algorithm (GA) tailored to the Generalized QAP is introduced in [7]. Authors employ genetic operators to address its combinatorial complexity effectively, obtaining competitive results. When generally discussing QAP, successful heuristic methods for such problems, can inspire successful combined methods.

The relationship between OCPs and the QAPs is further explored in [8], where a GA framework is proposed. This highlights the adaptability of GAs in navigating nonlinear, constrained solution spaces, underscoring their versatility in broader optimization challenges.

For a comprehensive survey of algorithms addressing the Generalized Assignment Problem (GAP), one can refer to [5]. This work categorizes and critically evaluates exact and heuristic methods, offering valuable insights into their computational efficiency and practical applicability, serving as a foundational reference for researchers exploring solution techniques for GAP and related combinatorial optimization problems.

Recently, hybrid methods combining various optimization strategies have gained traction for tackling nonlinear optimal control problems. In [19], a parallel hybrid variable neighborhood descent algorithm is proposed, leveraging variable neighborhood search and parallel processing to improve exploration and convergence. This method dynamically adapts neighborhood structures to address nonlinearity, demonstrating effectiveness across multiple benchmark problems. Additionally, [18] introduces a hub-location-based approach, modeling the optimal control problem as a network design task and identifying critical "hubs" to simplify computation while preserving solution quality. This structured yet flexible mechanism is particularly advantageous for high-dimensional decision spaces.

This paper focuses on OCPs modeled as QAPs. It employs evolutionary methods, such as GA and Particle Swarm Optimization (PSO), to generate a community of feasible solutions for the QAP. Subsequently, the problem is transformed into a P-Median Problem (PMP), and clustering techniques are applied to refine the solution set toward higher efficiency. The best solution is then chosen among these clusters. Incorporating K-means clustering, which is widely used in heuristic approaches for network distribution problems such as PMP, Vehicle routing Problem (VRP) offers notable advantages (see e.g., [1, 9]). Unlike approaches requiring large initial solution sets, this method uses metaheuristics to generate a limited number of initial solutions and iteratively improves them with optimization techniques like PSO and GA, enabling high-quality results in relatively short time.

The remainder of this paper is organized as follows: Section 2 defines the optimal control problem and details the discretization process that converts it into an assignment problem. Section 3 elaborates on the proposed algorithms, with two subsections dedicated to the details of the two algorithms. Section 4 presents the numerical implementations and results. Finally, Section 5 concludes the paper.

2 Notations and Preliminaries

The optimal control problem, in its general form, is defined as follows:

$$\text{Min} \quad I(x(T), u(T)) = \int_0^T f_0(t, x(t), u(t))\, dt \tag{1}$$

$$\text{s.t.} \quad \dot{x} = f(t, x(t), u(t)), \tag{2}$$

$$x(0) = x_0, \tag{3}$$

$$x(T) = x_T, \tag{4}$$

where T is a known time horizon, $x(\cdot) : [0, T] \to \mathbb{R}^n$ and $u(\cdot) : [0, T] \to \mathbb{R}^m$ are state and control vectors. Additionally, $f_0 : \mathbb{R}^{1+n+m} \to \mathbb{R}$ and $f : \mathbb{R}^{1+n+m} \to \mathbb{R}^n$ are continuous functions. The objective of this problem is to find an admissible pair (x, u) of state and control functions, minimizing objective functional (1), while satisfying (2) and initial conditions (3)-(4).

The OCP defined in (1)-(4), find applications across a wide range of scientific and industrial fields, offering powerful tools to optimize processes and systems governed by dynamic behavior. See [15] and [14] for two recent applications in robotics and aerospace engineering, respectively.

One of the most common approaches through OCP defined in (1)-(4), is to discretize it. To this end, the time interval $[0, T]$ is partitioned to n sub-intervals $[t_{i-1}, t_i], i = 1, \ldots, n$, where $t_0 = 0$. Since we

tend to detect an approximate optimal control for problem (1)-(3), we focus on finding an optimal piece-wise control function $u(\cdot)$ for the problem. Therefore, the range of feasible values $[u_{min}, u_{max}]$ for control vector u is discretized to the set $\{u_0, \ldots, u_m\}$, where $u_0 = u_{min}$ and $u_m = u_{max}$. Simply, one value from $\{u_0, \ldots, u_m\}$ will be assigned to a sub-interval $[t_{i-1}, t_i], i = 1, \ldots, n$, in the final solution. First, we consider the space of time and control as piecewise constant segments, see Figure 1. Finding the best value from $\{u_0, \ldots, u_m\}$ to be assigned to each interval $[t_{i-1}, t_i], i = 1, \ldots, n$ is a quasi-assignment problem. Note that there are n^m piece-wise constant functions that must be checked in this assignment. The aim of this paper is to provide a method to apply evolutionary techniques for detecting the best approximate piece-wise constant control function. Assume that $u(t) = \sum_{k=1}^{n} u_k \zeta_{[t_{k-1}, t_k]}(t)$ is a piece-wise constant function, then a numerical method as Euler method or Rung-Kutta, is applicable to find the trajectory corresponding $u(t)$ from (2) with initial condition $x(0) = x_0$. Thus, $(\hat{x}, \hat{u})$ is called an approximate solution to (1)-(4), if it satisfies (2) and (3), and $\| \hat{x}(T) - x_T \| \leq \varepsilon$, for a given $\varepsilon > 0$.

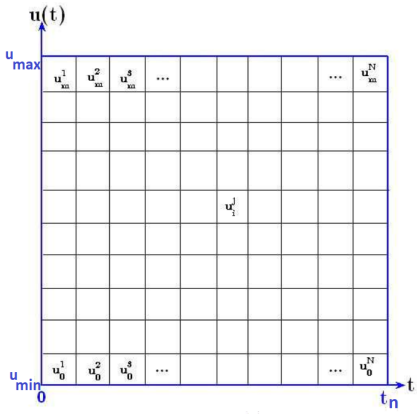

Figure 1: A discrete form of control space, [8].

The main idea to convert the OCP to a QAP, is to assign a constant $u_i \in \{u_0, \ldots, u_m\}$ to any interval $[t_{i-1}, t_i]$, $i = 1, 2, \ldots, n$. Initially, these constants could be selected randomly with $u_0 = u_{min}$ and $u_m = u_{\max}$. This paper suggests two evolutionary-based algorithms to improve this initial assignment, till a stopping rule holds. In other word, after discretizing the OCP, the problem is converted to a QAP in which a term is added to the objective function as the penalty for violating constraint (4). In more details, the multiplier of the term $\| \hat{x}(T) - x_T \|$ is added to the original objective function (1). Finally, the problem (1)-(4) is converted to the following problem:

$$\text{Min} \quad I(\hat{x}, \hat{u}) = \sum_{i=0}^{n} f_0(t_i, \hat{x}(t_i), \hat{u}(t_i)) + M \| \hat{x}(T) - x_T \| \tag{5}$$

$$\text{s.t.} \quad \hat{x}(t_{i+1}) = \hat{x}(t_i) + g(t_i, \hat{x}(t_i), \hat{u}(t_i)), \quad i = 0, \ldots, n-1, \tag{6}$$

$$\hat{x}(0) = x_0, \tag{7}$$

where, M is a large enough integer positive number. In order to provide an evolutionary-based approach for the problem (5)-(7), it suffices to focus on the 2-dimensional space of time and control, since state vectors are obtained by (6).

3 The Proposed Algorithm

This section explains two proposed algorithms, each of them is a hybrid of an evolutionary algorithm and a clustering method in order to find an approximate solution for OCP described by (1)-(4). To be more precise, each proposed method provides a solution for QAP described by (5)-(7). The main idea of both algorithms is to apply clustering techniques, here K-means method, to find the best solution among all available feasible solutions, based on a given objective function. Any control vector satisfying (6) and (7) is considered as a feasible solution. From here onward, by solution, we mean a feasible control vector, satisfying (6) and (7) along with its associated state vector.

First of all, we explain the general framework of the K-means clustering algorithm. In a general clustering problem, a set of data points with their property matrix is given. The goal is to cluster data points into K clusters, such that each point belongs to its most similar median, serving as a prototype of the cluster. K-means is one of the most applied clustering methods, [12]. The general scheme of the K-means algorithm is as follows:

Algorithm 7 Generic K-means algorithm

Input: A set of data points, an integer K: number of clusters.

Output: K clusters of data points, each with a median point.

1. Randomly select K data points as initial medians.
2. Repeat until medians do not change:

   - Construct a property matrix D, where each element D_{ij} is the distance or dissimilarity between data point i and median j.
   - Assign each data point to the cluster associated with the closest (or most similar) median based on D.
   - Update each median by selecting the most suitable point within each cluster.

In order to apply K-means for OCP, all feasible solutions (data points) along with their properties are needed. Here, a data point is any assignment $\zeta : \{t_0, \dots, t_n\} \longrightarrow \{u_0, \dots, u_m\}$, which is a control vector $[u(t_0), \dots u(t_n)]$. It's property will be discussed later. Initially, there is no control vector at hand as the solution of OCP. So, we start with the part of the set of all feasible solutions and improve it step by step. Therefore a pool of various control vectors is also needed in median updating step. Therefore the challenge of applying clustering methods for OCP lies in not having enough feasible solutions for (1) creating initial population of feasible solutions, and (2) median updating step.

Evolutionary algorithms are leveraged to address this challenge. To be more precise, some initial feasible control vectors are randomly generated, i.e. in order to form the initial medians, we randomly select values from $\{u_0, \dots, u_m\}$ for any interval $[t_{i-1}, t_i]$, for $i = 1, \dots, n$, for predetermined number of times. Then, evolutionary techniques are applied to generate new solutions out of initial ones. Now, a population of solutions are at hand for clustering. In any iteration of the clustering algorithm, medians should be updated. Here, since the K-means do not start with the whole feasible solutions, median

updating step changes to *median improving step*. Evolutionary techniques are again utilized to improve the current medians. Finally, The best median is selected as the final solution of the algorithm. It might be questioned that why we do not randomly generate all solutions of the whole initial population, and instead, a smaller number of solutions are generated as initial medians and an evolutionary technique is then applied to generate the whole population for clustering. To address this question, it should be noted that the quality of solutions would not be guaranteed in that case. In other words, there will be no control on randomly generated population of solutions, while evolutionarily generated solutions promotes diversity, as well as the probability of generating high quality solutions out of random ones. Indeed, from the perspective of generating new solution method, evolutionary algorithms might be classified into two main groups:

1. Evolutionary algorithms which utilize totally random methods in order to generate new solutions. GA is one of the most outstanding algorithms in this group, which has shown good results in various optimization problems.
2. Evolutionary algorithms which have a customized rule to generate new solutions and improve them toward a special direction to find better solutions. Particle Swarm Optimization (PSO) algorithm is one of the most successful methods in this group.

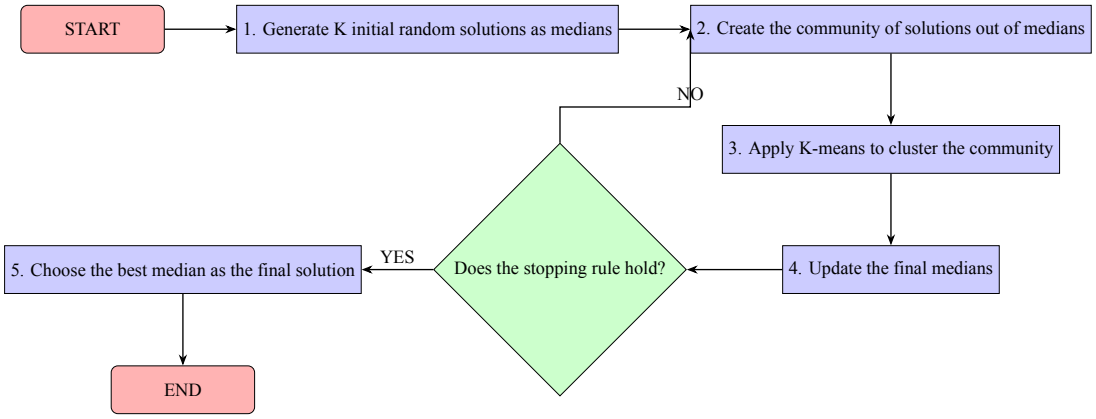

Figure 2: Schematic representation of the proposed algorithm

Figure 2 shows the general process of two proposed algorithms. Initial solutions are generated randomly at the first step as initial medians. These medians are the basis of generating the whole population of solutions at the second step. The third step is the clustering step which is accomplished by the K-means clustering method. The forth step improve the last set of medians. Finally, after predetermined number of iterations the algorithm chooses the best median as the final solution.

Based on the two mentioned challenges in applying clustering methods for OCP, two hybrid algorithms are proposed in this paper. The first algorithm addresses on the challenge of not having enough solutions at hand as initial population for clustering, through crossover operations of GA. In this algorithm, the initial random medians are treated as parents to generate offspring. This is done in the second step of Figure 2. K-means algorithm is then applied to cluster the whole solutions at hand. Mutation techniques are applied to address the second challenge and to possibly improve the final medians, if possible, at step 4. Finally, after repeating the process for the determined number of times, the best median

is selected as the final solution. The reason for applying GA for generating solutions is the high diversity that it provides, due to its random methods.

The second algorithm focuses just on the challenge of not having enough solutions for improving medians in step 4, through PSO. The initial random medians are moving toward some directions applying PSO in the second step of Figure 2 and new solutions are generated. The clustering step is the same as the first method. Median updating in the forth step is implemented by PSO. This algorithm moves former medians toward it's own built directions. The reason that we apply PSO is that it enables us to generate medians with the memory of the previous generation, which provides high quality solutions. Both algorithms are elaborated in the following two subsections.

3.1 Genetic Optimal Control Clustering Algorithm (GenOptClust)

GenOptClust algorithm is applied on the QAP described in (5)-(7). Table 1 shows the parameters of the algorithm.

Table 1: Notations.

Notation	Description
n	No. of divisions of interval $[0, T]$, which is $\{t_0 = 0, \ldots, t_n = T\}$,
m	No. of divisions of the range of feasible values for u, which is $\{u_0 = u_{min}, \ldots, u_m = u_{max}\}$,
K	No. of clusters in K-means algorithm,
U^k	$= (u_0^k, \ldots, u_n^k)$, is the median of cluster k, in which $u(t) = u_i^k$, for $t \in [t_i, t_{i+1})$ $i = 0, \ldots, n-1$,
IT	No. of iterations of the whole algorithm,
N	Size of the whole population including medians and their offspring,
$f(u)$	The value of the objective function for control vector u in (5).

Moreover, assume that $x_T = x(T)$ and $x_0 = x(t_0)$. As it has been mentioned before, a pool of solutions is needed initially in order to apply the K-means algorithm for clustering it. First, K initial medians are generated randomly. In more details, any initial median is generated by $n+1$ times random choices of feasible values $\{u_0, \ldots, u_m\}$. So, $U^k = (u_0^k, \ldots, u_n^k)$ is the median of cluster k, for $k = 1, \ldots, K$. It is clear that $u_i^k \in \{u_0, \ldots u_m\}$. Figure 3 depicts a schematic solution.

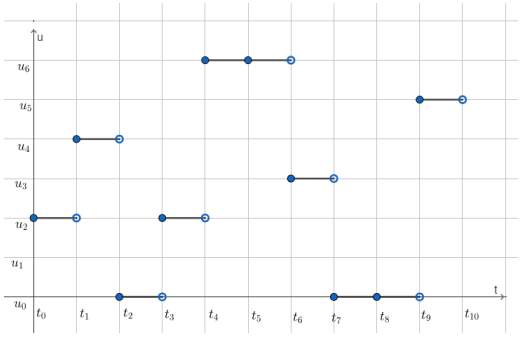

Figure 3: A schematic solution of control vector for QAP.

The state vector of each control vector is then calculated by recursive formulations (6) and its value of the objective function is obtained by (5), which is also called penalty. The next step is to form the whole population based on four crossover operations, using K initial medians as parents as follows:
For any $U^k = (u_0^k, \ldots, u_n^k)$ and $U^{k+1} = (u_0^{k+1}, \ldots, u_n^{k+1})$, for $k = 1, \ldots, K-1$, four following crossover operations are implemented:

1. **1-point crossover:** The values to the right of a random position are swapped between parents and offspring of f_1^k and of f_2^k are generated:

$$\text{of } f_1^k = (u_0^k, \ldots, u_L^k, u_{L+1}^{k+1}, \ldots, u_n^{k+1}),$$
$$\text{of } f_2^k = (u_0^{k+1}, \ldots, u_L^{k+1}, u_{L+1}^k, \ldots, u_n^k),$$

   where L is a random number from $\{0, \ldots, K-1\}$.

2. **2-point crossover:** The values between two random positions are swapped between parents and offspring off_3^k and off_4^k are generated:

$$\text{of } f_3^k = (u_0^k, \ldots, u_{L-1}^k, u_L^{k+1}, \ldots, u_H^{k+1}, u_{H+1}^k, \ldots, u_n^k),$$
$$\text{of } f_2^k = (u_0^{k+1}, \ldots, u_{L-1}^{k+1}, u_L^k, \ldots, u_H^k, u_{H+1}^{k+1}, \ldots, u_n^{k+1}),$$

   where L and H are random numbers from $\{1, \ldots, K-1\}$ and $\{L+1, \ldots, K-1\}$, respectively.

3. **Uniform crossover:** The values of three random positions are swapped between parents and offspring of f_5^k and of f_6^k are generated.

4. **Max-Min crossover:** Offspring off_7^k and off_8^k are maximum and minimum of their parents, respectively.

The last crossover operation is especially done in order to extract more smooth offspring out of their parents, since of f_7^k is the upper part and of f_8^k is the lower part of their parents.

Suppose that N is the size of the whole population of solutions at hand, including K medians as parents and generated offspring. The third step of the algorithm in Figure 2 is to cluster them applying K-means algorithm. Let D be the distance/property matrix, where D_{ik} is the difference between the penalty of k^{th} median and i^{th} solution, for $k = 1, \ldots, K$ and $i = 1, \ldots, N$. Forming property matrix in this form, put the control vector (solutions) with the closest penalty to each other, in the same cluster.

The stop criteria of the K-means algorithm is that the medians remain un-changed in two subsequent steps. The final medians are improved in step 4 of Figure 2, after meeting this condition. The median $U^k = (u_0^k, \ldots, u_n^k)$ is improved, based on 4 mutation methods as the following:

1. Choose the maximum element of U^k and a neighbour of it, with the maximum difference with it. Change both elements to their average. This is done to put down the high jump in vector U^k and convert it to a smoother vector.

$$U^k = (0, 0.1, 0.4, 0.3, 0, \underline{0.2, 0.8}, 0.4) \Longrightarrow Imp^k = (0, 0.1, 0.4, 0.3, 0, \underline{0.5, 0.5}, 0.4),$$

   then median U^k is replaced by Imp^k, if it has a lower penalty.

2. Exchange the values of two randomly selected elements of U^k. This is done to beget diversity in medians. Replace it, if it has a lower penalty.

3. Randomly choose two elements of U^k. Reduce the first one by Δ and increase the second one by the same value, where Δ is the random number out of feasible values of u. If it causes the elements to exceed the allowable range, change them to the minimum/maximum value. Replace it, if it has a lower penalty.

4. Exchange the value of the before and after value of a random position of vector U^k. Replace it, if it has a lower penalty.

The algorithm is repeated with the improved medians. The whole procedure is repeated IT times. Finally, the median with the lowest penalty is chosen as the final approximate solution. GenOptClust algorithm is explained in Algorithm 8.

Algorithm 8 Genetic Optimal Control Clustering (GenOptClust)

Input: K (number of clusters), IT (number of iterations)
Generate K initial medians (U^k) randomly for $k = 1, \ldots, K$
Repeat for each of the IT iterations:
  For all U^k and U^{k+1}, $k = 1, \ldots, K-1$:
    Generate 8 offspring using the 4 specified methods
  End For
  Let M be the set of all medians and their offspring, with size N
  Compute the relative state vector for each $u \in M$
  Calculate the penalty $f(u)$ for each $u \in M$ using equation (5)
  Compute the distance matrix D, where $D_{ik} = |I(U^i, x^i) - I(U^k, x^k)|$, for $i = 1, \ldots, N$, $k = 1, \ldots, K$
  Cluster the solutions in M based on matrix D, applying K-means
  For each final median U^k, $k = 1, \ldots, K$:
    Improve it if possible using the 4 mutation methods
  End For
End Repeat
Select the vector among $U^1, \ldots, U^K$ with the lowest penalty as the final solution

3.2 Particle Swarm Optimal Clustering (PSOptClust) Algorithm

The PSOptClust algorithm is elaborated in this subsection. Indeed, the algorithm is implemented on (5)-(7). The parameters are the same as in Table 1. This algorithm also starts with K initial random solutions. The relative state vectors and penalties are obtained by (6), and (5), respectively. Now, in order to create the population of solutions out of initial medians $U^1, \ldots, U^K$, GA techniques are no more applied, instead we simply multiply four randomly chosen positions of any vector U^k by a random

number in $[0, 1]$. Four new solutions are extracted out of any median. Now, the generated population is clustered by the K-means algorithm, based on the previously mentioned property matrix D, as before. The PSO method is applied in step 4 of Algorithm 2, in order to improve medians. For any $k = 1, \ldots, K$, suppose that U^k is the median of cluster k in the current iteration of the algorithm and U^k_{best} is the best median of k^{th} cluster up to the current iteration. U_{best} also shows the best median all over the clusters, up to the current iteration. In the p^{th} iteration of the algorithm, the direction V^k_p for moving the median of the k^{th} cluster toward it, is calculated as the following:

$$V^k_p = wV^k_{p-1} + cr_1(U^k_{\text{best}} - U^k) + cr_2(U_{\text{best}} - U^k). \tag{8}$$

V^k_0 is set to zero. The values w, cr_1 and cr_2 are parameters of the algorithm, which will be discussed in numerical result section. cr_1 shows the importance of the best median of cluster k, while cr_2 is influencing on the importance of best median by now. Finally, medians U^k are improved by:

$$U^k \longleftarrow U^k + V^k_p, \quad k = 1, \ldots, K$$

The procedure stops after IT iterations. PSOptClust algorithm is explained in Algorithm 9.

Algorithm 9 Particle Swarm Optimal Control Clustering (PSOptClust)

Input: K (number of clusters), IT (number of iterations), w, cr_1, cr_2 (algorithm parameters)
Generate K initial medians (U^k) randomly for $k = 1, \ldots, K$
Repeat for each of the IT iterations:
  For all U^k and U^{k+1}, $k = 1, \ldots, K-1$:
    Generate 4 offspring with the specified method
  End For
  Calculate the relative state vector for each $u \in M$
  Calculate $f(u)$ for each $u \in M$ and D_{ik} for each $i = 1, \ldots, N$, $k = 1, \ldots, K$
  Cluster solutions in M according to matrix D, applying K-means from Algorithm 8
  For each final median U^k, $k = 1, \ldots, K$:
    $U^k \leftarrow U^k + V^k_p$, where V^k_p is obtained by (8)
  End For
End Repeat
Choose the vector among $U^1, \ldots, U^K$ with the lowest penalty as the final approximate solution

4 Numerical Results

Two proposed algorithms, GenOptClust and PSOptClust, are implemented in this section, and their performance is benchmarked against comparable methods reported in the literature. Seven distinct optimal

control examples are solved using MATLAB R2020. All experiments run on an Intel Core M-5Y71 CPU @ 1.20–1.40 GHz with 8 GB RAM. All Each example is solved by two proposed algorithms for clustering counts $K \in \{50, 100, 200\}$, and time-interval partitions $h \in \{10, 30\}$. After 30 runs per algorithm, the parameter settings $cr_1 = 0.2$ and $cr_2 = 0.8$ yielded the best performance. These values are fixed in PSOptClust. All remaining examples were solved with both GenOptClust and PSOptClust across the specified ranges of K and N.

Example 1. Consider the following OCP:

$$J = Min\ \frac{1}{2}\int_0^1 (3x(t)^2 + u(t)^2)dt$$
$$\text{s.t.}$$
$$u(t) = \dot{x}(t) + x(t),$$
$$x(0) = 0, \quad x(1) = 2.$$

The best found value by [16] is $J = 6.0830$. GenOptClust resulted in the best performance for $h = 10$, $K = 200$, and the final value is $J^* = 5.8919$. Moreover, GenOptClust algorithm achieved a final state vector of $\bar{x}_T = 2.0002$ at time $T = 1$ within 16.1 seconds.

Example 2. Consider the following OCP:

$$J = Min\ \int_0^1 u(t)^2 dt$$
$$\text{s.t.}$$
$$u(t) = \dot{x}(t) - x^2(t)\sin(x(t)),$$
$$x(0) = 0,\ x(1) = 0.5.$$

The best found value by [16] is $J = 0.2274$. GenOptClust resulted in the best performance for $h = 10$, $K = 200$, and the final value is $J^* = 0.197$. Moreover, GenOptClust algorithm achieved a final state vector of $\bar{x}_T = 0.500007$ at time $T = 1$ within 10.02 seconds.

Example 3. Consider the following OCP:

$$J = Min\ \int_0^1 u(t)^2 dt$$
$$\text{s.t.}$$
$$\dot{x}(t) = \frac{1}{2}x^2(t)\sin(x(t)) + u(t),$$
$$x(0) = 0,\ x(1) = 0.5.$$

The best found value by [4] is $J = 0.2103$. PSOptClust resulted in the best performance for $h = 10$, $K = 100$. The final value is $J^* = 0.2227$. Moreover, PSOptClust algorithm achieved a final state vector of $\bar{x}_T = 0.4994$ at time $T = 1$ within 6.18 seconds.

Example 4. Consider the following OCP:

$$J = Min\ \int_0^1 (x(t) - e^t)^2 + (u(t) - t)^2 dt$$

s.t.

$$\dot{x}(t) = (2tx(t) - 2u(t)e^t + 1)e^t,$$
$$x(0) = 1,\ x(1) = e.$$

The optimal value for this problem is achieved by $x^*(t) = e^t$ and $u^*(t) = t$ and $J^* = 0$.The best found value by [4] is $J = 0.00011$. PSOptClust resulted in the best performance for $h = 10$, $K = 100$. The final value is $J^* = 0.1148$. Moreover, PSOptClust algorithm achieved a final state vector of $\bar{x}_T = 2.17189$ at time $T = 1$ within 8.03 seconds.

Example 5. Consider the following OCP:

$$J = Min \int_0^1 (tx(t) - u(t)e^t)^2 dt$$

s.t.

$$\dot{x}(t) = (tx(t) - 2u(t)e^t + 1)e^t,$$
$$x(0) = 1, \quad x(1) = e.$$

The optimal value for this problem is achieved by $x^*(t)e^t$ and $u^*(t) = t$ and $J^* = 0$. The best found value by [4] is $J = 0.0012$. GenOptClust resulted in the best performance for $h = 10$, $K = 100$. The final value is $J^* = 0.0542$. Moreover, GenOptClust algorithm achieved a final state vector of $\bar{x}_T = 2.71848$ at time $T = 1$ within 2.18 seconds.

Example 6. Consider the following OCP:

$$J = Min \int_0^1 (x(t) - t)^2 + (u(t) - t)^2 dt$$

s.t.

$$\dot{x} = x^2 - u^2 + 1,$$
$$x(0) = 0,\ x(1) = 1.$$

The optimal value for this problem is achieved by $x^*(t) = u^*(t) = t$ and $J^* = 0$. GenOptClust resulted in the best performance for $h = 10$, $K = 200$. The final value is $J^* = 0.0252$. Moreover, GenOptClust algorithm achieved a final state vector of $\bar{x}_T = 1.00002$ at time $T = 1$ within 8.25 seconds.

Example 7. Consider the following OCP:

$$J = Min \int_0^\pi (x(t) - \sin(t))^2 + (u(t) - t)^2 dt$$

s.t.

$$\dot{x} = x^2 + \cos(u(t)) - \sin^2(u(t)),$$
$$x(0) = 0,\ x(\pi) = 0.$$

The optimal value for this problem is achieved by $x^*(t) = \sin(t)$ and $u^*(t) = t$ and $J^* = 0$. GenOptClust resulted in the best performance for $h = 10$, $K = 100$. The final value is $J^* = 0.2615$. Moreover, GenOptClust algorithm achieved a final state vector of $\bar{x}_T = -0.0025$ at time $T = 1$ within 2.42 seconds.

Table 2 summarizes the results of all seven examples mentioned earlier. Highlighted blocks show the best solution achieved among two algorithms in six different cases. It can be seen that the proposed algorithms managed to improve the best found solution in the first three examples.

Figures 4 and 5 show the diagram of optimal trajectory of control vector by two proposed algrithms. It can be seen that proposed algorithms succeeded in achieving better values than the best values found so far, moreover genetic algorithm performs better compare to PSO. These better values are highlighted in Table 1. Also $h = 10$ is performing better than $h = 30$, and $K = 50$ never success to surpass other choices for K. Figure 4 compares the performance of two proposed algorithms by different parameters with best found values. In some examples, there is a significant difference between the optimal control trajectories derived from the GA-based and PSO-based methods. However, in others, this difference is negligible. Overall, it can be said that the general trend of the optimal control trajectory in both methods is somewhat similar.

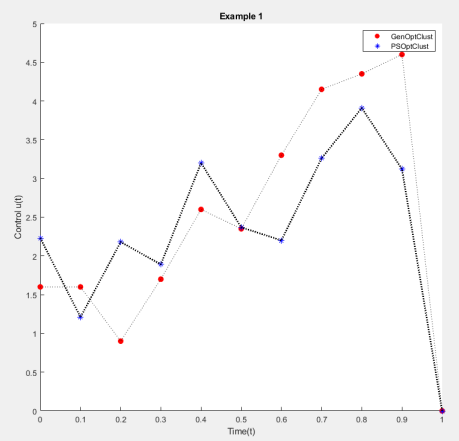

Figure 4: Approximate optimal controls by two proposed algorithms in Example 1.

Figure 6 presents a bar chart comparing the solution values obtained by the two proposed algorithms with the best-known solutions for each example. As evident from the chart, the GenOptClust algorithm has generally performed better than PSOptClust.

5 Conclusion

In this paper, we presented two methods based on metaheuristics, Genetic and PSO algorithms, each of which was used to generate and improve solutions for the optimal control problem. Additionally, we employed the K-means clustering algorithm to categorize the solutions and identify better ones. Finally, we implemented our algorithms on a set of optimal control problems. In this implementation, we utilized three different values for the number of clusters and two different values for the time interval subdivisions. Ultimately, in 3 out of the 7 examples, we achieved better solutions than those found in the literature. Among these, the Genetic-based algorithm with a number of clusters $K = 100$ and time interval subdivisions $h = 10$ outperformed the PSO-based algorithm.

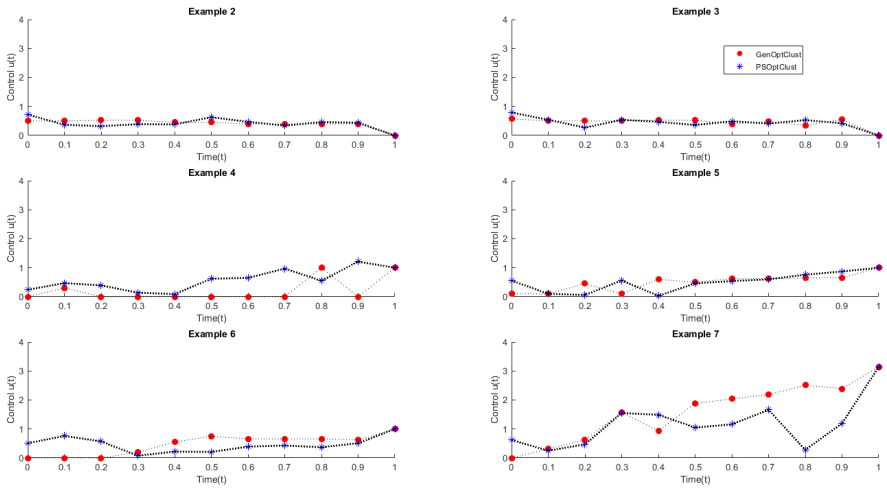

Figure 5: Approximate optimal controls by two proposed algorithms in Examples 2-7.

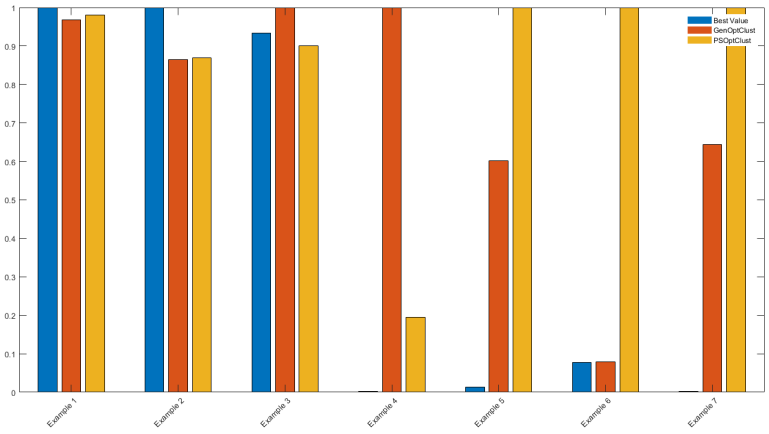

Figure 6: Comparison of Best Value, GenOptClust, and PSOptClust.

Declarations

Availability of Supporting Data

All data generated or analyzed during this study are included in this published paper.

Funding

The author conducted this research without any funding, grants, or support.

Competing Interests

The author declares that there are no competing interests relevant to the content of this paper.

References

[1] Alfiyatin, A.N., Mahmudy, W.F., Anggodo, Y.P. (2018). "K-Means clustering and genetic algorithm to solve vehicle routing problem with time windows", *Indonesian Journal of Electrical Engineering and Computer Science*, 11(2), 462-468, doi:http://doi.org/10.11591/ijeecs.v11.i2.pp462-468.

[2] Borzabadi, A.H., Mehne, H.H. (2009). "Ant colony optimization for optimal control problems", *Journal of Information and Computing Science*, 4(4), 259-264, doi:https://doi.org/2024-JICS-22734.

[3] Borzabadi, A.H., Mirassadi, S., Heidari, M. (2014). "Population based algorithms for approximate optimal distributed control of wave equations", *Iranian Journal of Numerical Analysis and Optimization*, 4(2), 31-41, doi:https://doi.org/10.22067/ijnao.v4i2.40230.

[4] Borzabadi, A.H., Ghasemi, S.G., Fard, O.S. (2011). "A hybrid iterative scheme for optimal control problems based on L-M and penalty techniques", *Journal of Advanced Research in Dynamical and Control Systems*, 3(2), 53-65.

[5] Cattrysse, D.G., Van Wassenhove, L.N. (1992). "A survey of algorithms for the generalized assignment problem", *European Journal of Operational Research*, 60(3), 260-272, doi:https://doi.org/10.1016/0377-2217(92)90077-m.

[6] Diveev, A., Sofronova, E., Konstantinov, S. (2021). "Approaches to numerical solution of optimal control problem using evolutionary computations", *Applied Sciences*, 11(15), 7096, doi:https://doi.org/10.3390/app11157096.

[7] Farahani, M.A., McKendall, A. (2023). "A Genetic Algorithm Meta-Heuristic for a Generalized Quadratic Assignment Problem", Preprint available at arXiv:2308.07828, doi:http://doi.org/10.48550/arXiv.2308.07828.

[8] Fard, O.S., Borzabadi A.H. (2007). "Optimal control problem, quasi-assignment problem and genetic algorithm", Enformatika, Transaction on Engineering Computations and Technologies, 19, 422-424.

[9] Gaon, T., Gabay, Y., Cohen, M.W. (2025). "Optimizing electric vehicle routing efficiency using K-means clustering and genetic algorithms", *Future Internet*, 17(3), 97, doi:https://doi.org/10.3390/fi17030097.

[10] Greistorfer, P., Staněk, R., Maniezzo, V. (2022). "A tabu search matheuristic for the generalized quadratic assignment problem", In: *Di Gaspero, L., Festa, P., Nakib, A., Pavone, M. (eds) Metaheuristics. MIC 2022. Lecture Notes in Computer Science, vol 13838. Springer, Cham*, doi:https://doi.org/10.1007/978-3-031-26504-4_46.

[11] Hatamian, R., Samareh Hashemi, S.A. (2025). "A hybrid numerical approach for solving nonlinear optimal control problems", *Control and Optimization in Applied Mathematics*, 10(1), 125-138, doi:https://doi.org/10.30473/coam.2025.72875.1273.

[12] Jain, A.K. (2010). "Data clustering: 50 years beyond K-means", *Pattern Recognition Letters*, 31(8), 651-666, doi:https://doi.org/10.1016/j.patrec.2009.09.011.

[13] Konstantinov, S., Diveev, A. (2021). "Evolutionary algorithms for optimal control problem of mobile robots group interaction", In: *Olenev, N.N., Evtushenko, Y.G., Jaćimović, M., Khachay, M., Malkova, V. (eds) Advances in Optimization and Applications. OPTIMA 2021. Communications in Computer and Information Science, vol 1514. Springer, Cham*, doi:https://doi.org/10.1007/978-3-030-92711-0_9.

[14] Lucrezia, M. (2024). "Application of optimal control techniques to the parafoil flight of space rider", *Aerotecnica Missili & Spazio*, 103(2), 117-128, doi:https://doi.org/10.1007/s42496-023-00176-3.

[15] Massaro, M., Lovato, S., Bottin, M., Rosati, G. (2023). "An optimal control approach to the minimum-time trajectory planning of robotic manipulators", *Robotics*, 12(3), 64, doi:https://doi.org/10.3390/robotics12030064.

[16] Mehne, H.H., Borzabadi, A.H. (2006). "A numerical method for solving optimal control problems using state parametrization", *Numerical Algorithms*, 42, 165-169, doi:https://doi.org/10.1007/s11075-006-9035-5.

[17] Mînzu, V., Arama, I. (2022). "Optimal control systems using evolutionary algorithm-control input range estimation", *Automation*, 3(1), 95-115, doi:https://doi.org/10.3390/automation3010005.

[18] Salimi, M., Borzabadi, A.H., Mehne, H.H., Heydari, A. (2021). "The hub location's method for solving optimal control problems", *Evolutionary Intelligence*, 14, 1671-1690, doi:https://doi.org/10.1007/s12065-020-00437-1.

[19] Salimi, M., Borzabadi, A.H., Mehne, H.H., Heydari, A. (2025). "A parallel hybrid variable neighborhood descent algorithm for non-linear optimal control problems", *Iranian Journal of Numerical Analysis & Optimization*, 15(2), doi:https://doi.org/10.22067/ijnao.2024.86859.1389.

Table 2: Comparison of Genetic and PSO methods with different steps and K values. Highlighted cells indicate notable values.

Best found value	Example	GenOptClust						PSOOptClust					
		step=10			step=30			step=10			step=30		
		50	100	200	50	100	200	50	100	200	50	100	200
6.083	Ex1	6.15	6.088	5.8919	6.6568	6.5789	6.45	5.9704	6.0455	5.9670	6.0209	6.8302	7.6823
0.2274	Ex2	0.2274	0.1994	0.197	0.2148	0.2094	0.2040	0.2073	0.21002	0.1980	0.2118	0.2139	0.2092
0.2103	Ex3	0.2252	0.2255	0.2256	0.2486	0.2431	0.2432	0.2381	0.2027	0.2286	0.2480	0.2525	0.2413
0.00011	Ex4	0.07462	0.07159	0.05885	0.07594	0.07451	0.07405	0.01679	0.01148	0.01509	0.2622	0.2348	0.1426
0.0012	Ex5	0.0754	0.0542	0.0577	0.3567	0.3099	0.1603	0.0900	0.1454	0.0566	0.3608	0.2140	0.1746
0.021	Ex6	0.0442	0.0328	0.0212	0.080	0.0783	0.0725	0.5498	0.4668	0.3765	0.2693	0.6690	0.3699
0.0051	Ex7	1.56	1.615	0.98	1.59	1.3165	2.29	4.548	2.5397	2.5106	4.1890	5.9574	4.1091