Iranian Journal of Numerical Analysis and Optimization
Vol. 15, No. 2, 2025, pp 424–456
https://doi.org/10.22067/ijnao.2024.86859.1389
https://ijnao.um.ac.ir/

Research Article

A parallel hybrid variable neighborhood descent algorithm for nonlinear optimal control problems

M. Salimi, A.H. Borzabadi*, H.H. Mehne and A. Heydari

Abstract

In this paper, a numerical method for solving bounded continuous-time nonlinear optimal control problems (NOCPs) that based on variable neighborhood descent (VND) algorithm is proposed. First, the genetic algorithm

*Corresponding author

Received 13 February 2024; revised 13 August 2024; accepted 19 August 2024

Mitra Salimi
Department of Mathematics, Payam Noor University, Tehran, Iran. e-mail: salimi2012a@gmail.com

Akbar Hashemi Borzabadi
Department of Applied Mathematics, University of Science and Technology of Mazandaran, Behshahr, Iran. e-mail: borzabadi@mazust.ac.ir

Seyed Hamed Hashemi Mehne
Khayyam Institute, Tehran, Iran. e-mail: hmehne@ari.ac.ir

Aghileh Heydari
Department of Mathematics, Payam Noor University, Tehran, Iran. e-mail: a_heidari@pnu.ac.ir

How to cite this article
Salimi, M., Borzabadi, A.H., Mehne, H.H. and Heydari, A., A parallel hybrid variable neighborhood descent algorithm for nonlinear optimal control problems. *Iran. J. Numer. Anal. Optim.*, 2025; 15(2): 424-456. https://doi.org/10.22067/ijnao.2024.86859.1389

(GA) is combined with an improved VND that uses efficient neighborhood interchange. Then, to improve the efficiency of the algorithm for practical and large-scale problems, the parallel processing approach is implemented for discrete form of NOCP. It performs the required complex computations in parallel. The resulting parallel algorithm is applied to a benchmark of nine practical problems such as Van Der Pol problem and chemical reactor problem. For large-scale problems, the parallel hybrid variable neighborhood descent algorithm (PHVND) is capable of obtaining optimal control values effectively. Our experimentation shows that PHVND outperforms the best-known heuristics in terms of both solution quality and computational efficiency. In addition, computational results indicate that PHVND produces superior results compared to sequential quadratic programming or GA.

AMS subject classifications (2020): Primary 49M25; Secondary 65Y05, 68W50.

Keywords: Parallel algorithm; Neighborhood descent algorithm; Optimal control problem.

1 Introduction

The main goal of optimal control is to find a control for a linear or nonlinear system that will makes the system satisfy physical constraints while also minimizing or maximizing some performance measure. This involves determining the best control inputs over a given time horizon to achieve the desired system behavior [6]. Optimal control problems (OCPs) are utilized to provide an explanation for problems in various industrial processes, such as the control of robots, aircraft, and electrical systems [4]. An optimal control strategy was proposed for dynamic transfer from a simple half-circular legged monopod model in [27]. The development of the nonlinear optimal control problems (NOCP) method for reverse osmosis desalination devices with an induction motor (IM) to minimize energy dissipation and reduce the operating cost of the units is reviewed in [29]. In [28], a nonlinear optimal control approach is proposed for the dynamic model of a gas centrifugal compressor driven by an IM. One of the other applications of NOCPs to human health is reported in [2], where NOCP is used to determine the dynamics of hepatitis E diseases.

The study of numerical methods has provided an attractive field for researchers in mathematical sciences who have seen the emergence of different numerical calculation methods and efficient algorithms to solve OCPs [25]. Nowadays, meta-heuristic approaches have gained popularity in solving NOCPs. Researchers applied the particle swarm optimization (PSO), for example, in [12], to optimize the parameters of a proportional integral derivative (PID) controller. The application of ant colony optimization (ACO) is another meta-heuristic for numerically solving NOCPs. The method uses the discrete form of the control values of the set. ACO solved the problems that have transformed into quasi-assignment problems [5, 19]. The genetic algorithm (GA) method is a population-based meta-heuristic. It was widely used to solve many optimization problems. In recent decades, GA has been also proposed to solve NOCP effectively [21, 30]. Continuous GA and improved GA to solve NOCP can be found in [1, 33].

To improve the precision of the solutions and reduce the computational time, hybrid methods were introduced [3, 20]. In [23], a modified hybrid GA has been proposed to solve NOCP numerically. A hybrid improved GA using a simple local search method is also provided to solve NOCP [34].

One of the advantages of the application of meta-heuristics for solving NOCP is their independence to near-optimal initial guesses. So, one approach to solving an NOCP with meta-heuristics is changing the NOCP into a nonlinear programming (NLP) problem with finite dimension. It is performed by discretizing the space-time and control, which leads to a large-scale problem.

After the discretization of NOCP, a new initiative is proposed to solve the problem more easily and faster. Due to the structural similarity of the discretization problem with the uncapacitated single allocation p-hub median problem (USApHMP), we will use this method to solve NOCP [32]. Due to the simple nature of USApHMP, the NOCP optimization procedure is easily programmed and can be solved effectively. It has many benefits, such as ease of programming, short computation time, self-starting, and convergence to the global minimum.

The large dimension of large-scale NLP is a limitation that makes it difficult to find accurate solutions, especially with population-based algorithms. Parallelization algorithms are used to speed up a wide range of algorithms, for example, the ACO method, which is applied to solve NOCP, [19]. GA uses hybridization, strategic search [31], and parallelization to reach high-quality solutions in a low computational time. In [36, 37, 38], a parallel GA for optimization is described. To reduce the computational time in solving large-scale problems, parallelism is proposed in this paper.

One of the effective methods to solve USApHMP is variable neighborhood descent (VND) [17]. It has been applied to solve the maximum clique problem [16], scheduling problem [7], and hub location [26].

To find a promising region of search space, we use four local searches in sequential VND (SVND). Then, to increase the quality of solutions obtained from the phase search, the number of control partitions in NOCP is changed as hubs in USApHMP [32]. Finally, we combine SVND with the GA in mutations and reduce computational time with parallel processing.

The parallel hybrid variable neighborhood descent algorithm (PHVND) is then evaluated by implementing some real-world examples. The results show that the proposed method obtains more accurate solutions compared with methods that search for control values in discrete form.

The remainder of this paper is organized as follows: Section 2 provides a brief overview of the formulation problem. Section 3 describes the basics of the PHVND algorithm. In Section 4, we examine some benchmark examples to compare the results of the proposed algorithm with the other existing methods. Finally, Section 5 presents more analysis of the results and concludes the paper.

2 Mathematical programming formulation

The goal of NOCP is to control a dynamical system from a given initial state to a final optimal state in a desirable way. Find an optimal controller $u^*(t)$ and state variable $x^*(t)$ that minimize the cost function $J(x, u)$, where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ are the state variable and the control variable respectively. Suppose that $g : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are both continuously differentiable functions in x and u, t_0, t_f are the initial and final times, respectively, and that $\phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is a continuous scaler function. The Bolza problem is minimizing

$$J(x(\cdot), u(\cdot)) = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} g(t, x(t), u(t))dt, \tag{1}$$

under the following constraints and boundary conditions:

$$\dot{x} = f(t, x(t), u(t)), \tag{2}$$

$$c(t, x, u) = 0, \tag{3}$$

$$d(t, x, u) \leq 0, \tag{4}$$

$$\psi(t_f, x(t_f)) = 0, \tag{5}$$

$$x(t_0) = x_0, \tag{6}$$

$$u_l \leq u \leq u_b, \tag{7}$$

where all functions are assumed to be smooth enough in suitable open sets. The cost function (1) should be minimized according to the dynamics (2), control and restrictions of state equality (3) and control inequality and state inequality constraints (4), final state constraints (5), initial condition (6), and limitation of the control variable (7).

Minimum time problems, linear quadratic regulator (LQR) problem, tracking problem, minimum energy, and terminal control problem are another special case of NOCPs.

3 Parallel hybrid algorithm

Representation scheme. At first, the problem of minimizing (1) subject to (2)–(7) is changed to a discrete form. To do this, the time interval is divided to a number of equidistant nodes $\{t_0, t_1, \ldots, t_{N_t-1}\}$. For any interval $[t_{i-1}, t_i], i = 1, 2, \ldots, N_t - 1$, the interval $[u_l, u_b]$ is divided into m subinterval $[u_{j-1}, u_j], j = 1, 2, \ldots, m$ with the same length. Then, p values are determined as hub $(u_0, u_1, \ldots, u_p)$ in USApHMP from m subinterval. Therefore, in direct form of the problem, we have to find an assignment

$$\{t_0, t_1, \ldots\} \rightarrow \{u_0, u_1, \ldots\} \tag{8}$$

with minimum performance index. The new problem is an assignment type of optimization problems with a finite dimension. Since the control interval is divided into m partitions and the time interval is divided into $N_t - 1$ sections, and one control must be selected for each time. Then, this problem can be extended to USApHMP, where every time is assigned to only one hub from the set of $u_0, u_2, \ldots, u_p$. (see details in [32])

Fitness function. Now, a piecewise control function is constructed as follows:

$$u(t) = \sum_{j=1}^{N_t-1} \chi[t_{j-1}, t_j](t) u_j, \tag{9}$$

where χ is the indicator function:

$$\chi[t_{j-1}, t_j](t) = \begin{cases} 1, & t \in [t_{j-1}, t_j], \\ 0, & otherwise. \end{cases} \tag{10}$$

The time nodes are defined by

$$t_j = t_0 + jh, \qquad j = 0, \ldots, N_t - 1, \tag{11}$$

where $h = (t_f - t_0)/N_t - 1$. This is a finite-dimensional optimization problem with $u_1, \ldots, u_p$ as unknowns.
Let $\psi = [\psi_1, \psi_2, \ldots, \psi_{n_\psi}]$ is the vector form of constraints, defined in (5). Then satisfaction of constraints can be relaxed as:

$$\varphi_f = \|\psi\| < \epsilon, \tag{12}$$

where $\|\cdot\|$ is the Euclidean norm and ϵ is a given small number indicating the accuracy. The term integral of (1) can be calculated by a numerical integration such as Runge-Kutta or Newton algorithms for each control input $u(t_j)$, and the corresponding state variable $x(t_j), j = 0, 1, \ldots, N_t - 1$, which satisfy in (2). The notations $x(t_j)$ and $u(t_j)$ are briefly indicated by x_j and u_j respectively.

Then, the discrete form of the performance index (1) is as follows:

$$\text{minimize} \quad J(x(\cdot),u(\cdot)) = \sum_{j=0}^{N_t-1} w_j g(t_j, x(t_j), u(t_j)) + M\varphi_f, \tag{13}$$

where $w_j, j = 0,\dots,N_t-1$ are weights and M is a large enough positive number. In the case of equality or inequality constraints (3) or (4), penalty constants $M_1, M_2, M_{31},\dots,M_{3n_\psi}$ are added to the fitness function:

$$\begin{aligned} I = \tilde{J} &+ \sum_{j=0}^{N_t-1} M_1 \max\{0, d(x_j,u_j,t_j)\} + \sum_{j=0}^{N_t-1} M_2 c^2(x_j,u_j,t_j) \\ &+ \sum_{k=1}^{n_\psi} M_{3k}\psi_k^2(x_{N_t-1}, t_{N_t-1}), \end{aligned} \tag{14}$$

where $\tilde{J}$ is an approximation of J.

In the following, the GA is introduced in the first subsection and the SVND algorithm in the next subsection. Then, the structure of the hybrid PHVND algorithm is explained and then the parallelization process is mentioned.

3.1 Genetic algorithm

Initialization. The initial population is chosen randomly from candidate solutions in the search space. Then, p control input values are selected from the m control subintervals at random. This can be done in the following stages. In our experiment, the population size is fixed at a value of 300 ($N_{pop} = 300$). Then, each time $t_j, j = 0,\dots,N_t-1$ is randomly assigned to the selected control values. With this selection pattern, initial chromosomes are produced. When this repeats N_{pop} times, then N_{pop} feasible chromosomes $\{C_1, C_2,\dots,C_{N_{pop}}\}$ are generated.

Selection. For NOCP, the better chromosome is one with a smaller fitness value. In the first, a certain number of individuals in the population are randomly selected. Using this method of selection, we will have a great chance to find better random chromosomes to produce children. This phase can be also performed in parallel.

Elitism. The incumbent from generation to generation is maintained. If in crossover or mutations, the best chromosome (individual) within the population is lower than the existing solution, the better answer is replaced by the worse one in the current population.

Crossover. The crossover is a combination of two or more parental solutions to provide better solutions. There are several ways to achieve this, and good performance depends on a well-designed recombination mechanism [35]. Here, the standard one-point crossover operator is proposed, where a crossover operator creates a child solution for selected parents. The crossover parents are selected from $\left\{C_1, C_2, \ldots, C_{N_{pop}}\right\}$ and random j from $\{0, \ldots, N_t - 1\}$.

To construct a child solution, its first j genes are taken from the first parent, and the remaining N_t-1-j genes are taken from the second parent. After applying the crossover on the arrays, if the number of assignments control is more than p in the offsprings, we keep the feasible one, then for the remaining gens if a time node t_i was assigned to another control from $\{u_{p+1}, \ldots, u_{N_u}\}$, then the node t_i is randomly assigned to a selected $\{u_1, \ldots, u_p\}$.

Mutation. The modification of the mutation method preserves the entire population. Repeat the selection to the mutation process until a termination condition is met.

3.2 Local searches

The fundamental neighborhood structures can also be extended for mutations in GA exchanges and node interpolation moves. Local search methods are as follows:

1- First, we randomly choose one person in the population, $pop(i)$ and randomly choose gen position $j_1 \in \{1, \ldots, N_t - 1\}$ and $j_2 \in \{j_1, \ldots, N_t - 1\}$. Then, exchange position of j_1 and j_2 and replace if the new offspring is better than the worst person in the population. This means that we select

(t_{j_1}, u_{j_1}) and (t_{j_2}, u_{j_2}) then exchange u_{j_1} with u_{j_2} and vice versa.

2- The second, randomly choosing one person, *pop*(i), then the local search method is as follows. This local modification does not change the location of the selected controls $(u_1, \dots, u_p)$. For all time nodes, we try to change their membership. This means that we have to process every selected control and try to replace the assignment time node with a new assignment. This does not change other elements.

3- The location of selected controls u_j (which in control array $\{u_1, \dots, u_p\}$) is changed in *pop*(i) that choose randomly and move to this neighborhood solution if the cost function is reduced. First, a control u_k that is not in control array, is selected. From the representation of the current assignment, all nodes times that assignment to u_j can generate and then change its assignment to u_k. But if the *pop*(i) has only one control, do nothing. Otherwise, the change in the cost function for all populations is calculated. We developed four neighborhood structures independently, although one of them is essentially the same as the second local search method (Greassign and Greplace) in [39] and one and the second are the same as local search in [17], but these neighborhoods are applied in a different order.

4- The one parent *pop*(i), u_j from the control array and u_k that is not in the control array was selected, randomly. The u_j was deleted and u_k was replaced and randomly reassigned all time nodes in *pop*(i) to new control that includes u_k. Thus all assignments in *pop*(i) are changed.

3.3 SVND

Among available strategies for the exploration of several neighborhoods within deterministic local search, the sequential strategy is more common. In VND, the neighborhood structures are explored one by one in the given sequence that is specified in Algorithm 1 (SVND) [14, 15]. SVND uses all $l_{\max}$ neighborhood structures within VND. Then it performs local searches with them, considering their orders. SVND stops whenever no better solution in the last neighborhood exits.

Algorithm 1 Sequential variable neighborhood descent

```
Require: Make an order of all $l_{\max} \geq 2$ neighborhoods that will be used in
    the search, select the set of neighborhood structures $N_l, l = 1, \dots, l_{\max}$,
    an initial solution $x$.
1:  $x_{opt} = x$, $f_{opt} = f(x)$, $l = 1$
2:  repeat
3:      $i = 0$
4:      repeat
5:          $i = i + 1$
6:          find best neighbor $x^{'} = argmin\{f(x), f(x_i)\}, x_i \in N_l(x)$
7:      until all solutions from neighborhood $l$ are visited set
8:      if solution $x^{'}$ thus obtained is better than $x$ then
9:          $x = x^{'}$, $x_{opt} = x$, $f_{opt} = f(x)$ and $l = 1$
10:     else
11:         $l = l + 1$
12:     end if
13: until $l > l_{\max}$
14: return $x$
```

The last obtained solution is a local minimum with considering all $l_{\max}$ neighborhood structures. After construction or update of the population, the fitness values of the candidate solutions are evaluated using (14). Here, four local search neighborhoods are used for each generated individual to refine the allocation of time nodes to the control array. It will be explained in the next subsection.

3.4 Parallelization

The size of the population is an important factor that affects the scalability and performance of the genetic algorithm. However, increasing the size of the population will increase the computational time. To solve this problem, a parallel implementation is proposed. It will reduce the time and improves the ability of solving complex problems. Due to independent operations in a large amount of data, population-based optimization algorithms can run in parallel. The first stages of discretization and population generation are serial. Then, the iteration of GA including mutation, fitness evaluation and crossover are implemented as loop parallelization.

The process is then repeated until the maximum allowed number of iterations is reached. The parallel algorithm proposed by PHVND is effective in solving the test criteria for the large-scale NOCP algorithm. The algorithm of PHVND to solve NOCP is given as Algorithm 2.

In PHVND, GA is combined with the SVND method in the mutation step as local search to improve the solutions.

Algorithm 2 The algorithm of the PHVND method

```
Require: Select the number of time steps, $N_t$, the number of control functions, $N_u$, the size of the population, $N_{pop}$, the maximum number of iterations $Maxiter$, the mutation factor, $p_m$, the crossover constant, $p_c$, and the number of the available processors, $N_p$.
    Generate a random initial population $P_A$ and distribute it among processors by master processors.
    let $iter = 0$
    while $(iter < Maxiter)$ do
        Selection: Each processors receives its population, $P_B$, and calculates the fitness function of its population.
        Elitism: Each processors selects the best individual among $P_B$ and send them the master processor.
        The master processor determines the best global individuals, send them to processors, and updates the $P_A$.
        Each processor receives the global best individuals and replace them with the worst individual in the $P_B$.
        Crossover: Each processors performs a crossover for $P_B$
        Mutation: Each processors executes SVND for $P_B$
        let $iter = iter + 1$.
    end while
    Master processor returns the best individual in the final population $(P_A)$ as an approximate solution of NOCP.
```

4 Computational results for NOCP

To examine the efficiency of the proposed algorithm, some benchmark problems of NOCPs are considered in some examples. The absolute error of the performance index, satisfaction of the problem's constraints, and the effect of particle size of control times and intervals are parameters for evaluating the algorithms.

The codes are developed using MATLAB software and the OpenMP programming interface. They were executed on a parallel computer at the Aerospace Research Institute with 48 (1.8 GHz) cores.
Let J be the performance index obtained by an algorithm, let φ_f define the error of final state constraints, and let J^* be the best solution obtained among all implementations, or the exact solution (when exists). Now, let N_t and m be defined as the division of time and control interval, and let the absolute error of J, gap_J, of the algorithm be defined by

$$gap_J = |J - J^*|. \tag{15}$$

The parameters of PHVND are $l_{\max} = 4$, u_{left}, u_{right}, t_0, and t_f. We consider $p_m = 0.4$ and $p_c = 0.6$. The other parameters of PHVND are given in what follows. To find the best value for the algorithm parameters, we ran the algorithm with different parameter values and then chose the best. However, the sensitivity of the parameters N_t, m are studied in the tables. Table 1 indicates the details of each case.

4.1 Comparison of results

The results on instances of partition time and control function are summarized in Table 12 where PHVND is compared with Modified Hybrid Genetic (see [24]), GA (see [11]), sequential unconstrained minimization technique (SUMT) (see [10]), sequential quadratic programming (SQP) (see [10]), hybrid algorithm by integrating an improved PSO with improved particle swarm optimization (IPSO-SQP)(see [22]), novel continuous genetic algorithm (CGA) (see [1]), numerical method proposed in [13] called Bézier, ho-

motopy perturbation method (HPM) (see [9]), parallel numerical method based on whale optimization algorithm (WOA) (see [20]), Linear interpolation (LI)(see [24]), and Spline interpolation (SI)(see [24]).
Comparison the best numerical results in 300 different runs of the PHVND method for different partitions of time and control function for each exam-

Table 1: The problem parameters for NOCPs examples

	$g=\frac{1}{2}(x_1^2+x_2^2+u^2)$	$x_0=(1,0)$	
Example 1	$f=\left[x_2,-x_2+(1-x_1^2)x_2+u\right]^T$	$t=[0,5]$	$M=10^2$
	$\psi=x_1-x_2+1$	$u=[-0.5,2]$	
	$g=\frac{1}{2}(x_1^2+x_2^2+0.1u^2)$	$x_0=(0.05,0)$	
Example 2	$f=[x_1-2(x_1+0.25)+(x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right)$	$t=[0,0.78]$	$M=1$
	$-u(x_1+0.25),0.5-x_2-(x_2+0.5)\exp\left(\frac{25x_1}{x_1+2}\right)]^T$		
	$\psi=[x_1,x_2]^T$	$u=[-1.5,2]$	
	$g=u^2$	$x_0=0$	
Example 3	$f=\frac{1}{2}x^2\sin(x)+u$	$t=[0,1]$	$M=10^6$
	$\psi=x-0.5$	$u=[0,1]$	
	$\phi=-x_3$	$x_0=(10,-2,10)$	
Example 4	$f=[x_2,-2\frac{u}{x_3},-0.01u]^T$	$t=[0,5]$	$M=1$
	$\psi=[x_1,x_2]^T$	$u=[-30,30]$	
	$\phi=x_1$	$x_0=(0,1)$	
Example 5	$f=[-620000e^{\left(\frac{-5000}{u}\right)}x_1+4000e^{\left(\frac{-2500}{u}\right)}x_2^2,$	$t=[0,1]$	
	$-4000e^{\left(\frac{-2500}{u}\right)}x_2^2]^T$	$u=[298,398]$	
	$\phi=x_3$	$x_0=(0,-1,0)$	
Example 6	$f=[x_2,-x_2+u,x_1^2+x_2^2+0.005u^2]^T$	$t=[0,1]$	$M=1$
	$\psi=(x_1,x_2)$	$u=[-5,5]$	
	$d(x,t)=x_2+0.5-8(t-0.5)^2$		
	$g=2x_1$	$x_0=(2,0)$	
Example 7	$f=[x_2,u]^T$	$t=[0,3]$	$M=1$
	$d(x,t)=-(6+x_1)$	$u=[-2,2]$	
	$g=x_1^2+x_2^2+0.005u^2$	$x_0=(0,-1)$	
Example 8	$f=[x_2,-x_2+u]^T$	$t=[0,1]$	$M=10^2$
	$d(x,t)=-(8(t-0.5)(t-0.5)-0.5-x_2)$	$u=[-5,5]$	
	$g=\frac{1}{2}(x_1^2+x_2^2+u^2)$	$x_0=(1,0)$	
Example 9	$f=[x_2,-x_1+(1-x_1^2)x_2+u]^T$	$t=[0,5]$	$M=1$
	$d(x,t)=-(x_2+0.25)$	$u=[-1,1]$	

ples are proposed in Table 2 and Tables 4–11. In the tables related to each example, such as Table 2 or Tables 4–11, the value of gap_J is the result of comparing the value of J for different partitions with the optimal value of PHVND. In Table 12, the value of gap_J is the result of comparing the best value of PHVND with other algorithms for each example.

4.2 Numerical Experiments

Example 1. This example is Van Der Pol Problem (VDP) [1], which has to be minimized. After scaling by taking the best-known value reached by PHVND, we obtain the following values: $J^* = 1.5404$ for $N_t = 100$ and $m = 100$. The numerical result for different N_t, m is given in Table 2. We have also compared our result with other algorithms in Table 12. The results objective function of implementing PHVND is less than the result of CGA, which equals $J^* = 1.7404$ and less than LI, $J^* = 1.5949$, and SI, $J^* = 1.5950$.

The error of proposed method is small around $\epsilon = 10^{-4}$. The value of speedup in Table 3 for parallel computing, shows the performance of parallelization and reduction time in terms of number of CPUs. The resulting control and states of Example 1 obtained from using the PHVND method for $(N_t, m) = (100, 100)$ are given in Figures 1 (a), 1(b). Figure 1 (c) shows the performance index for 300 iterations and indicates the convergence rate.

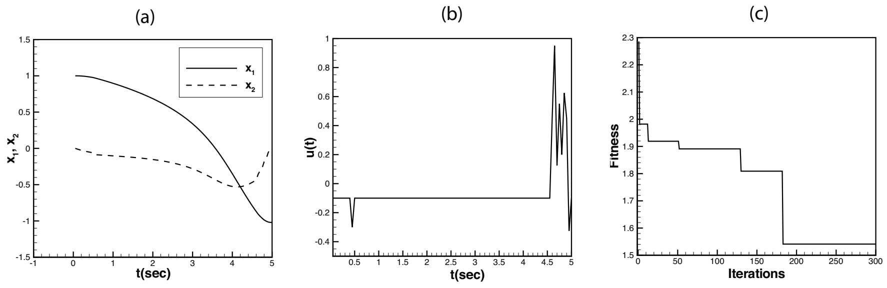

Figure 1: (a) The resulting optimal trajectories for $N_t = 100$, $m = 100$ for Example 1, (b) The resulting optimal control for $N_t = 100$, $m = 100$ for Example 1, (c) Performance index for iterations for Example 1 ($N_t = 100$, $m = 100$)

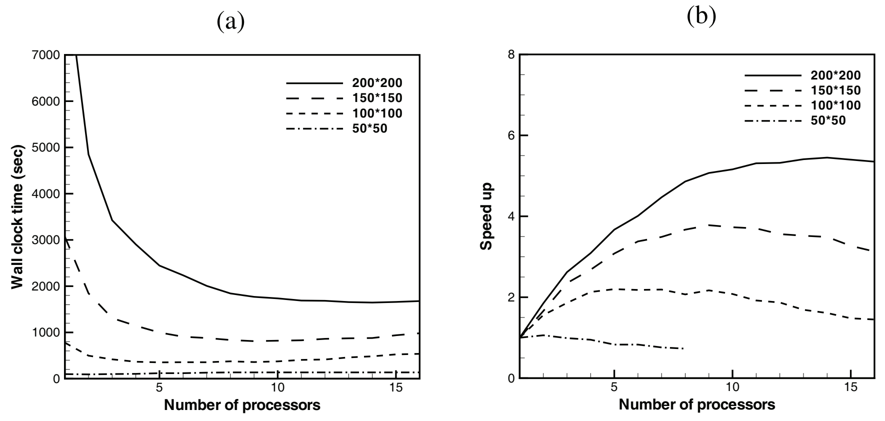

Figure 2: (a) The wall clock time of execution for Example 1, and (b) The speedup curve for Example 1.

In addition, Figure 2(a) shows the wall clock time of running, which shows that run time is decreased with increase the number of processors. The speed of the performance curve is also given in Figure 2(b) and compared with the ideal speedup. The resulting speedup curve is close to the ideal line up to 8 threads and its divergent behavior is normal after 8 threads because of the serial fraction of the algorithm and other overheads.

Based on the figures, the efficiency of the method in controlling the system to desired final states, rate of convergence and parallel performance can be deduced.

Example 2. The Chemical Reactor Problem (CRP) [1] is solved in this example. The proposed algorithm was applied to this problem where the results are given in Table 4.

The results have been compared with CGA, LI, and SI algorithm in Table 12. Although, in Table 12, the norm of final state constraints, φ_f, for CGA, equals 7.57×10^{-10} and for the LI and SI methods, which equal 1.15×10^{-9} and 5.99×10^{-9}, respectively, is less than φ_f of the PHVND methods, which equals 3.60×10^{-3}, but the performance index gap is very small, about 0.008 for CGA and 0.01 for LI and SI.
After solving with the proposed method, we obtain the following results: the best value $J^* = 0.033$ and the error function $\varphi_f = 3.60 \times 10^{-3}$ that obtained

Table 2: Comparison between the best numerical results of different partitions for Example 1

N_t	m	J	φ_f	gap_J
10	10	1.5659	2.19×10^{-3}	0.025
30	50	1.6299	2.85×10^{-3}	0.089
30	100	1.5811	3.12×10^{-3}	0.047
50	50	1.5966	9.06×10^{-4}	0.056
50	100	1.5689	4.14×10^{-4}	0.028
75	75	1.661	7.15×10^{-4}	0.120
100	100	1.5404	1.75×10^{-4}	0.000
150	150	1.7587	1.00×10^{-3}	0.218

Table 3: Comparison the results of PHVND for different partitions of time and control function for 16 number of threads for Example 1

Number of	$N_t = 100$, m=100		$N_t = 150$, m=150		$N_t = 200$, m=200	
threads	time (sec)	Speed up	time (sec)	Speed up	time (sec)	Speed up
1	778	1.00	3070	1.00	8967	1.00
2	497	1.56	1846	1.66	4849	1.85
3	418	1.86	1309	2.35	3425	2.62
4	366	2.13	1144	2.68	2904	3.09
5	353	2.20	998	3.08	2445	3.67
6	356	2.18	907	3.38	2235	4.01
7	355	2.19	879	3.49	2006	4.47
8	375	2.07	836	3.67	1843	4.86
9	359	2.17	811	3.78	1769	5.07
10	374	2.08	821	3.73	1736	5.16
11	406	1.92	830	3.70	1689	5.31
12	415	1.87	861	3.56	1684	5.32
13	460	1.69	873	3.52	1656	5.41
14	483	1.61	879	3.49	1646	5.45
15	525	1.48	939	3.27	1659	5.40
16	537	1.45	982	3.13	1678	5.35

Table 4: Comparison between the best numerical results of different partitions for Example 2

N_t	m	J	φ_f	gap_J
10	10	0.041	7.93×10^{-3}	0.007
30	50	0.033	3.60×10^{-3}	0.000
30	100	0.034	7.46×10^{-4}	0.000
50	50	0.045	5.09×10^{-3}	0.011
50	100	0.042	1.53×10^{-3}	0.008
75	75	0.059	1.02×10^{-2}	0.025
100	100	0.058	1.25×10^{-2}	0.024
150	150	0.060	1.11×10^{-2}	0.026

for $N_t = 30$ and $m = 50$. The resulting control and states of Example 2 obtained by the PHVND method are given in Figures 3(a) and 3(b). Furthermore, the speed-up curve is also shown in Figure 3(c) and compared with the ideal speedup. Figure 4(a) indicates the convergence rate of the performance index for 300 iterations.

The results shown in Figures 3 and 4(a) specify that the method has the ability to find nearly optimal control-trajectories with a reduction in time with the aid of parallel processing.

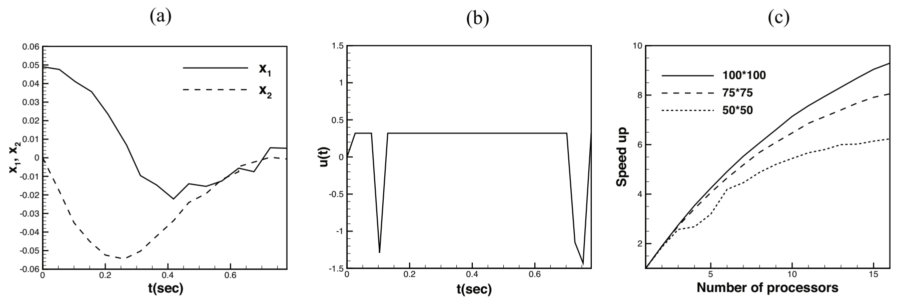

Figure 3: (a) The resulting optimal trajectories ($N_t = 16$, $m = 20$), (b) The resulting optimal control ($N_t = 30$, $m = 50$), and (c) The speedup curve of Example 2.

Example 3. This example [9], is a constraint nonlinear model. After solving this problem, the following results: the best value $J^* = 0.1825$ and the error function $\varphi_f = 6.81 \times 10^{-7}$ that obtained for $N_t = 100$ and $m = 100$. The convergence rate of the performance index for 300 iterations is shown in Figure 4(b). The numerical results for different N_t, m are given in Table 5.

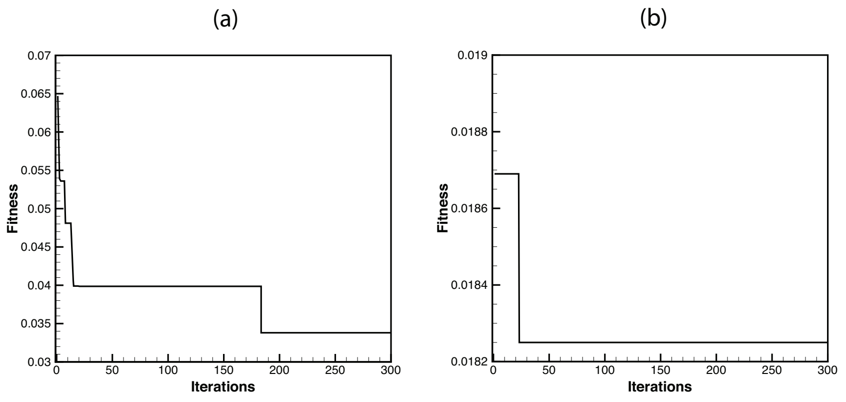

Figure 4: (a) Convergence rate of the performance index for Example 2 ($N_t = 30$, $m = 50$), and (b) Convergence rate of the performance index for Example 3 ($N_t = 100$, $m = 100$).

Table 5: Comparison between the best numerical results of different partitions for Example 3

N_t	m	J	φ_f	gap_J
10	10	0.2547	2.21×10^{-7}	0.072
30	50	0.1952	1.2×10^{-6}	0.012
30	100	0.2100	3.17×10^{-6}	0.027
50	50	0.1877	1.33×10^{-7}	0.005
50	100	0.1906	8.06×10^{-7}	0.013
75	75	0.1869	1.52×10^{-6}	0.004
100	100	0.1825	6.81×10^{-7}	0.000

From Table 5, it is obvious that the gap between the cost function and the best one for different values of m and N_t is less than 0.07, indicating that the objective functions in PHVND do not differ a lot. Although, from Table 5, the norm of final state constraints, φ_f, for our method, is equal to 10^{-7} or 10^{-6}. The numerical results of the proposed algorithm are compared with algorithms: GA in [11], LI, SI, and HPM. For different n, m, the performance index, J,in our method is also less than $J^* = 0.2015$ which was proposed in [24] and $J^* = 0.3526$ which was taken by GA in [11].

Example 4. This problem [10] is to control the performance index with minimum effort. The problem is compared with several numerical methods, see Table 12. The best optimal objective was $J^* = -8.8157$ for $N_t = 30, m = 50$ (see Table 6) which is close to the result of LI and SI,-8.8692 or SQP and SUMT, -8.8690. The norm of final state constraint for LI and SI equals 1.45×10^{-10} and 2.79×10^{-10} and this criterion for the PHVND methods equals 4.8×10^{-4}. By increasing the repetition in the algorithm, better solutions can be obtained, as the convergence rate of the performance index for 300 iterations is shown in Figure 5(a).

Table 6: Comparison between the best numerical results of different partitions for Example 4

N_t	m	J	φ_f	gap_J
10	10	-8.3931	0.586	0.422
30	50	-8.8157	0.036	0.000
30	100	-8.7597	0.075	0.056
50	50	-8.7945	0.063	0.021
50	100	-8.7829	0.051	0.032
75	75	-8.8074	0.036	0.008
100	100	-8.8099	0.042	0.005

Example 5. The problem of dynamic optimization of consecutive reaction batch reactor ($A \rightarrow B \rightarrow C$) is a classical problem studied by several researchers [41, 8, 34]. The aim is to find the optimal temperature profile that maximizes the product performance of B temperature at the result of the process in the batch reactor. The state constraints and initial conditions are given in Table 1, where x_1 shows the concentration of B, x_2 represents the concentration of A, and u is the temperature.

To compare the performance of PHVND with existing methods, the comparison results are given in Table 7. The maximum value by applying PHVND is $J^* = 0.717$ was obtained very rapidly. The computation time with a discretization of $N_t = 9, m = 10$ subintervals, was 27.7 seconds.

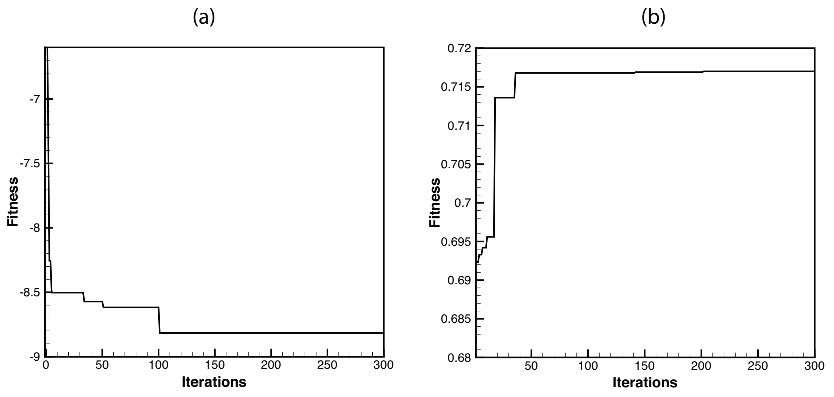

Figure 5: (a) Convergence rate of the performance index for Example 4 ($N_t = 30$, $m = 50$), and (b) Performance index versus iterations number in Example 5 ($N_t = 9$, $m = 10$).

Table 7: Comparison between the best numerical results of different partitions for Example 5

N_t	m	J	gap_J
9	10	0.717	0.000
10	10	0.677	0.04
30	50	0.612	0.105
30	100	0.612	0.105
50	50	0.609	0.108
50	100	0.609	0.108
75	75	0.608	0.109
100	100	0.607	0.11

In addition, we present our approach to the optimal control in Figure 5(b) and the results from the approach of [40] and [18] are 0.6107 and 0.6128 that exceeded by our method to 0.717 that shown in Table 12.

Example 6. This problem [22] contains an inequality constraint with minimum control effort. After solving with the proposed method, the best value $J^* = 0.2746$ was obtained for $N_t = 10$ and $m = 10$.

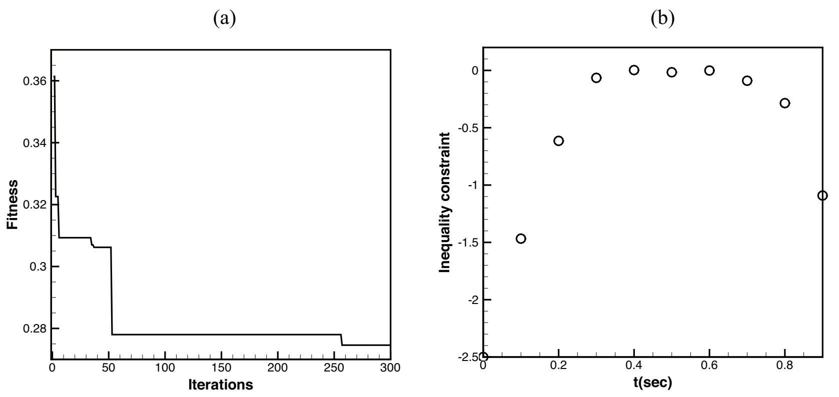

Figure 6: (a) Convergence rate of the performance index, and (b) the inequality constraint of Example 6 ($N_t = 10$, $m = 10$).

To see the convergence of the example in the 300 iterations, Figure 6(a) is presented, and Figure 6(b) shows an inequality constraint for 300 iterations. Table 8 shows the numerical results for different N_t, m.

Table 8: Comparison between the best numerical results of different partitions for Example 6

N_t	m	J	gap_J
10	10	0.2746	0.000
30	50	0.2797	0.005
30	100	0.2796	0.005
50	50	0.2839	0.009
50	100	0.2771	0.002
75	75	0.2871	0.012
100	100	0.2931	0.018

Example 7. This problem [13] contains an inequality constraint that should be minimized. After solving with the proposed method, the best value $J^* = -8.999$ was obtained for $N_t = 75$ and $m = 75$.

The convergence rate of the performance index for 300 iterations is given in Figure 7(a) and inequality constraint d is displayed in Figure 7(b). The numerical results for different N_t, m are compared in Table 9. The perfor-

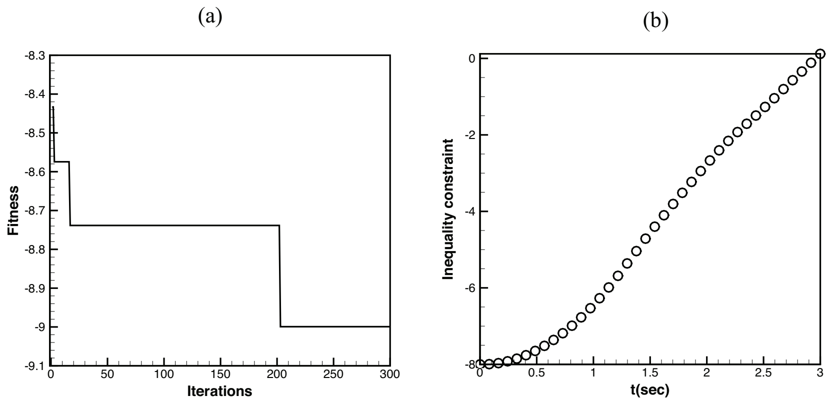

Figure 7: (a) The performance index versus iterations number, and (b) The inequality constraint of Example 7 ($N_t = 75$, $m = 75$).

Table 9: Comparison between the best numerical results of different partitions for Example 7

N_t	m	J	gap_J
10	10	-7.391	1.608
30	50	-8.885	0.114
30	100	-8.574	0.425
50	50	-8.917	0.082
50	100	-8.843	0.156
75	75	-8.999	0.000
100	100	-8.860	0.139

mance index in Table 12, J, for different N_t, m in our method is also low than $J = -5.3898$ by Bézier and $J = -5.4309$ by LI and SI.

Example 8. This problem [10] contains an inequality constraint,that J has to be minimized. After solving with the proposed method, the best value $J^* = 0.2646$ was obtained for $N_t = 30$ and $m = 50$.

The convergence rate of the performance index for 300 iterations is shown in Figure 8(a) and inequality constraint d for 300 iterations is shown in Figure 8(b). The numerical result for different N_t, m is given in Table 10. From Table 10, it is obvious that for different values of m and N_t, the gap of the

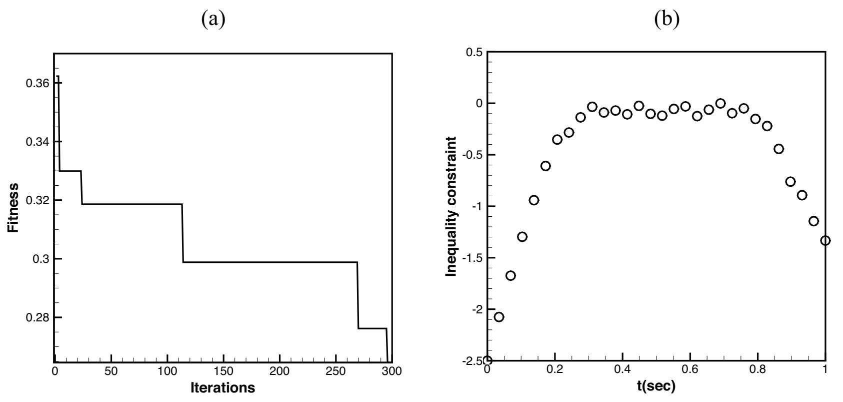

Figure 8: (a) The performance index versus iterations number, and (b) The inequality constraint of Example 8 ($N_t = 30$, $m = 50$).

Table 10: Comparison between the best numerical results of different partitions for Example 8

N_t	m	J	gap_J
10	10	0.2818	0.017
30	50	0.2646	0.000
30	100	0.3019	0.037
50	50	0.2846	0.02
50	100	0.2748	0.01
75	75	0.2755	0.01
100	100	0.2818	0.017

cost function from the best one, is less than 0.03, indicating that the objective functions in PHVND do not differ a lot. The performance index, J, for different N_t, m in our method is also do not differ a lot than $J = 0.2163$ by SQP and $J = 0.1703$ which was obtained by SUMT and $J = 0.1549$ by LI and SI. This difference will decrease with increasing repetition and will converge to the optimal solution.

Example 9. The aim of this problem [10], is minimizing the performance index. After solving with the proposed method, the best value $J^* = 1.476$ that obtained for $N_t = 30$ and $m = 50$.

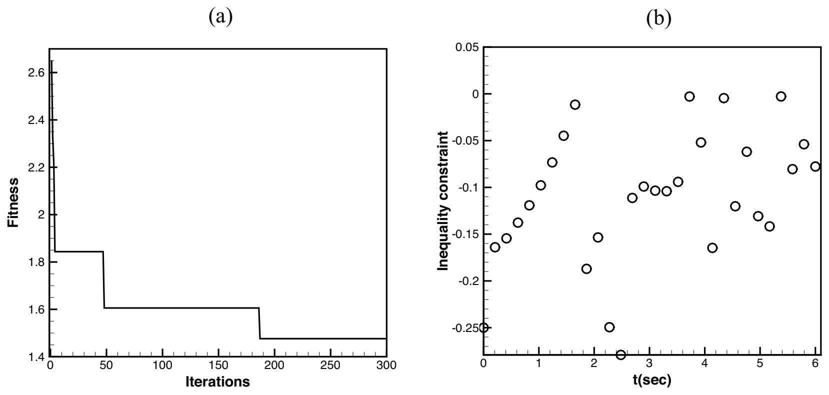

Figure 9: (a) The performance index versus iterations number, and (b) the inequality constraint of Example 9 ($N_t = 30$, $m = 50$).

The performance index convergence rate for 300 iterations is shown in Figure 9(a) and inequality constraint d for 300 iterations is shown in Figure 9(b). The numerical result for different N_t, m is presented in Table 11. According to Table 11, we find that increasing the number of partitions m and N_t, does not necessarily improve the solution. The performance index

Table 11: Comparison between the best numerical results of different partitions for Example 9

N_t	m	J	gap_J
10	10	1.822	0.346
30	50	1.476	0.000
30	100	1.546	0.07
50	50	1.795	0.319
50	100	1.596	0.12
75	75	1.993	0.517
100	100	2.118	0.642

in Table 12, J, for different N_t, m in our method is also less than $J = 1.7950$ by SQP and $J = 1.7980$ which was obtained by SUMT and $J = 1.6509$ by LI and SI. In general, its performance is influenced by the size of the problem.

Table 12: Comparison of the best value of the PHVND with some algorithms for nine examples

Problem	Algorithm	J	φ_f	gap_J
Example 1	CGA	1.7404	2.67×10^{-11}	0.2
	LI	1.5949	1.08×10^{-13}	0.054
	SI	1.5950	9.99×10^{-14}	0.054
	PHVND	1.5404	1.75×10^{-4}	0.000
Example 2	CGA	0.0163	7.57×10^{-10}	0.003
	LI	0.0127	1.15×10^{-9}	0.000
	SI	0.0127	5.99×10^{-9}	0.000
	PHVND	0.033	3.60×10^{-3}	0.021
Example 3	HPM	0.2353	4.2×10^{-6}	0.052
	LI	0.2015	2.35×10^{-9}	0.019
	SI	0.2015	2.82×10^{-10}	0.019
	GA	0.3526	4.09×10^{-5}	0.17
	PHVND	0.1825	6.81×10^{-7}	0.000
Example 4	SQP	-8.8690	0	0.000
	SUMT	-8.8690	0	0.000
	LI	-8.8692	1.45×10^{-10}	0.000
	SI	-8.8692	2.79×10^{-10}	0.000
	PHVND	-8.8157	3.6×10^{-2}	0.053
Example 5	Method of [18]	0.6107	-	0.106
	method of [40]	0.6128	-	0.104
	PHVND	0.717	-	0.000
Example 6	IPSO-SQP	0.1727	-	0.002
	LI	0.1699	-	0.000
	SI	0.1700	-	0.000
	PHVND	0.2746	-	0.104
Example 7	LI	-5.4309	-	3.56
	SI	-5.4309	-	3.56
	Bézier	-5.3898	-	3.6
	PHVND	-8.999	-	0.000
Example 8	SQP	0.2163	-	0.061
	LI	0.1549	-	0.000
	SI	0.1549	-	0.000
	SUMT	0.1703	-	0.015
	PHVND	0.2646	-	0.109
Example 9	SQP	1.7950	-	0.319
	LI	1.6509	-	0.174
	SI	1.6509	-	0.174
	SUMT	1.7980	-	0.322
	PHVND	1.476	-	0.000

4.2.1 Analysis of the numerical examples

By analyzing the number of times and controls that an approach used to reach its best solution, we can conclude that PHVND is more reliable than other heuristics for problems. The results obtained by the algorithm are similar for small instances of the problem, but when the size increases, the PHVND approach improves the results in terms of quality of solutions and running times. Moreover, we improve the best-known solution for benchmark data sets presented in examples.

According to our computational results, PHVND is an efficient method in general since only a little computational time is added to the algorithm in each iteration. Besides, it does not require large RAM for relatively medium instances. Note that the PHVND can find optimal solutions (or feasible solutions in some cases), however, others cannot load the problem due to the shortage of memory. Whereas, the gap solution of others are larger than those of our algorithm for most cases (see Table 12).

Thus, the PHVND together with efficient partitioning for solving trajectory corresponding significantly improves our ability to tackle large instances of NOCP. On the other hand, by comparison of computational results of the number of partitions in Tables 2–12, our choice of the objective function has a significant impact on the number of iterations. In general, the proposed method can solve complex problems with accurate solutions as shown in Table 12. Note that the method outperforms control problems in average computational effort or the best solution gap for large instances. Tables 2–12 present results for network designs of NOCP for different partitions. Naturally, the selection of the number of nodes in the partitions resulting from the division of time and control space may depend on the dimensions of the problem, e.g. the number of state variables, but in any case, the numerical results show that the selection of finer partitions reduces the possibility of the solution getting worse. Basically, by dividing the time interval and control into more nodes, we can determine that in most cases, a worse criterion does not occur.

5 Conclusion

This paper proposes a PHVND algorithm to solve large-scale NOCPs. Comparing the results of the proposed method with existing methods for NOCP reveals that the proposed method leads to solutions closer to the optimal solutions. The algorithm also increases the efficiency of the solution process, improves the scalability of performance by using the SVND process, and increases the diversification of solutions at the same time while reducing the execution time compared to the genetic algorithm. A set of large-scale NOCPs was used to test this algorithm. Examining the parallel implementation of the method shows that this method uses parallelism, which reduces the execution time. The proposed algorithm can significantly reduce execution time compared to other well-known algorithms in lower-scale problems. In most cases, our algorithm has achieved good results. In other cases, we have found a very close to the optimal solution, which, as seen in the convergence figures, will improve with the increasing number of repetitions. For future research, methods can be used to solve the minimum-optimal-time control problem, where the end of the time interval is not fixed. OCPs in dynamically distributed systems may also be genetically analyzed. From a computational point of view, this method can be used in computers with GPUs to take advantage of the acceleration of graphics processing and compare parallel performance on shared memory structures.

Abbreviations

ACO	Ant Colony Optimization
CGA	Continuous Genetic Algorithm
CRP	Chemical Reactor Problem
GA	Genetic Algorithm
HPM	Homotopy Perturbation Method
IM	Induction Motor
PSO	Particle Swarm Optimization
IPSO	Improved Particle Swarm Optimization
LI	Linear Interpolation
LQR	Linear Quadratic Regulator
NLP	Nonlinear Programming
NOCP	Nonlinear Optimal Control Problem
OPC	Optimal Control Problem

PHVND	Parallel Hybrid Variable Neighborhood Descent
PID	Proportional Integral Derivative
SI	Spline Interpolation
SQP	Sequential Quadratic Programming
SUMT	Sequential Unconstrained Minimization Technique
VND	Variable Neighborhood Descent
SVDN	Sequential Variable Neighborhood Descent
USApHMP	Uncapacitated Single Allocation p-hub Median problem
VDP	Van Der Pol
WOA	Whale Optimization Algorithm
C_j	jth feasible chromosome
f	A function representing the system dynamic
f_{opt}	Current optimal objective

Nomenclature

gap_J	Absolute error of the resulting objective value
J	Objective function
$\tilde{J}$	Discrete form of J
J^*	Optimal objective value
$l_{\max}$	Maximum number of iterations in Algorithm 1
m	Number of divisions of the control interval
$Maxiter$	Maximum number of iterations in Algorithm 2
M_1, M_2, M_{3k}	Penalty constants
N_p	Number of processors
N_{pop}	Number of population in GA
N_t	Number of time interval divisions
p_m	Mutation factor
p_c	Crossover constant
$pop(i)$	A randomly chosen individual
t	Time variable
t_j	jth time node

t_0	Initial time
t_f	Final time
u_l	Lower bound on the control values
u_b	Upper bound on the control values
$u(t), u$	Control variable
$u^*(t)$	Optimal control
$x(t), x$	State variable
$x^*(t)$	Optimal state
x_{opt}	Current optimal state
x_0	Initial state
x_f	Final state
φ_f	Total error in constraints
$\chi(t)$	Indicator function

References

[1] Abo-Hammour, Z.S., Asasfeh, A.G., Al-Smadi A.M. and Alsmadi, O.M., *A novel continuous genetic algorithm for the solution of optimal control problems*, Optim. Control Appl. Methods, 32(4) (2011), 414–432.

[2] Alzahrani, E.O. and Khan, M.A., *Modeling the dynamics of Hepatitis E with optimal control*, Chaos Solit. Fractals, 116 (2018), 287–301.

[3] Babaie-Kafaki, S., Ghanbari, R. and Mahdavi-Amiri, N., *Two effective hybrid metaheuristic algorithms for minimization of multimodal functions*, Int. J. Comput. Math. 88(11) (2011), 2415–2428.

[4] Betts, J.T., *Practical methods for optimal control and estimation using nonlinear programming*, Society for Industrial and Applied Mathematics, 2010.

[5] Borzabadi, A.H. and Mehne, H.H., *Ant colony optimization for optimal control problems*, Journal of Information and Computing Science, 4(4) (2009), 259–264.

[6] Chachuat, B., *Nonlinear and dynamic optimization: From theory to practice*, Automatic Control Laboratory, 2007.

[7] Costa, W.E., Goldbarg, M.C. and Goldbarg, E.G., *New VNS heuristic for total flowtime flowshop scheduling problem*, Expert Syst. Appl. 39(2) (2012), 8149–8161.

[8] Dadebo, S.A. and Mcauley, K.B., *Dynamic optimization of constrained chemical engineering problems using dynamic programming*, Comput. Chem. Eng. 19(5) (1995), 513–525.

[9] Effati, S. and Nik, H.S., *Solving a class of linear and non-linear optimal control problems by homotopy perturbation method*, IMA J. Math. Control Inf. 28(4) (2011), 539–553.

[10] Fabien, B.C., *Some tools for the direct solution of optimal control problems*, Adv. Eng. Softw. 29(1) (1998), 45–61.

[11] Fard, O.S. and Borzabadi, A.H., *Optimal Control Problem, Quasi-Assignment Problem and Genetic Algorithm*, World Academy of Science, Engineering and Technology, 33 (2007), 46–48.

[12] Gaing, Z.L., *A particle swarm optimization approach for optimum design of PID controller in AVR system*, IEEE Trans. Energy Convers. 19(2) (2004), 384–391.

[13] Ghomanjani, F., Farahi, M.H. and Gachpazan, M., *Bézier control points method to solve constrained quadratic optimal control of time varying linear systems*, Comput. Appl. Math. 31(3) (2012), 433–456.

[14] Hansen, P. and Mladenović, N., *Developments of Variable Neighborhood Search*, Essays and Surveys in Metaheuristics, Springer, 2002.

[15] Hansen, P., Mladenović, N., Moreno, P. and érez, J.A., *Variable neighbourhood search: methods and applications*, 4OR, 6(4) (2008), 319–360.

[16] Hansen, P., Mladenović, N. and Urošević, D., *Variable neighborhood search for the maximum clique*, Discret. Appl. Math. 145 (2004), 117–125.

[17] Ilić, A., Uroševic, D., Brimberg, J. and Mladenović, N., *A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem*, Eur. J. Oper. Res. 206(2) (2007), 289–300.

[18] Lazutkin, E., Geletu, A., Hopfgarten, S. and Li, P., *An Analytical Hessian and Parallel-Computing Approach for Efficient Dynamic Optimization Based on Control-Variable Correlation Analysis*, Ind. Eng. Chem. Res. 54(48) (2015), 12086–12095.

[19] Mehne, H.H., *Evaluation of parallelism in ant colony optimization method for numerical solution of optimal control problems*, J. Electr. Eng. Electron. Control Comput. Sci. Science, 1(2) (2015), 15–20.

[20] Mehne, H.H. and Mirjalili, S., *A parallel numerical method for solving optimal control problems based on whale optimization algorithm*, Knowl.-Based Syst. 151 (2018), 114–123.

[21] Michalewicz, Z., Krawczyk, J.B. and Kazemi, M., *Genetic algorithms and optimal control problems*, 29th IEEE Conference on Decision and Control, 3 (1990), 1664–1666.

[22] Modares, H. and Naghibi Sistani, M.B., *Solving nonlinear optimal control problems using a hybrid IPSO–SQP algorithm*, Eng. Appl. Artif. Intell. 24(3) (2011), 476–484.

[23] Nezhadhosein, S., Heydari, A. and Ghanbari, R., *A Modified Hybrid Genetic Algorithm for Solving Nonlinear Optimal Control Problems*, Math. Probl. Eng. 2015 (2015), 1–21.

[24] Nezhadhosein, S., Heydari, A. and Ghanbari, R., *Integrating Differential Evolution Algorithm with Modified Hybrid GA for Solving Nonlinear Optimal Control Problems*, Iran. J. Math. Sci. Inform. 12(1) (2017), 47–67.

[25] Nezhadhosein, S., Ghanbari, R. and Ghorbani-Moghadam, K., *Article Solving a Class of Nonlinear Optimal Control Problems Using Haar Wavelets and Hybrid GA*, Control and Optim. Appl. Math. 8 (2023), 1–17.

[26] Oliveira, F.A., de Sá, E.M., d. Souza, S. R. and Souza, M.J.F., *ILS-based algorithms for the profit maximizing uncapacitated hub network design problem with multiple allocation*, Comput. Oper. Res. 157 (2023), 0305–0548.

[27] Ozkan, Y., Aydin, Y.O., Saranli, A., Yazicioglu, Y., Saranli, U. and Leblebicioğlu, K., *Optimal control of a half-circular compliant legged monopod*, Control Eng. 33 (2014), 10–21.

[28] Rigatos, G., Abbaszadeh, M., Sari, B., Siano, P., Cuccurullo, G. and Zouari, F., *Nonlinear optimal control for a gas compressor driven by an induction motor*, Results Control. Optim. 11(2023) 100226.

[29] Rigatos, G., Siano, P., AL-Numay, M., Abbaszadeh, M. and Sari, B., *Nonlinear optimal and multi-loop flatness-based control of induction motor-driven desalination units*, Results Control. Optim. 14 (2024), 100360.

[30] Roy, T. and Chakraborty, D., *Optimal vibration control of smart fiber reinforced composite shell structures using improved genetic algorithm*, J. Sound Vib. 319 (2009), 15-40.

[31] Said, S.M. and Nakamura, M., *Asynchronous parallel algorithms for strategic hybrid on a mixture gaussin model, International Journal of Innovative Computing*, Inf. Control, 10(2) (2014), 459-479.

[32] Salimi, M., Borzabadi, A.H., Mehne, H.H. and Heydari, A., *The hub location's method for solving optimal control problems*, Evol. Intell. 14 (2021), 1671-1690.

[33] Shi, X.H., Wan, L.M., Lee, P.H., Yang, X.W., Wang, L.M. and Liang, Y.C., *An improved genetic algorithm with variable population-size and a PSO-GA based hybrid evolutionary algorithm*, Int. J. Syst. Sci. 3 (2003), 1735-1740.

[34] Sun, F., Du, W., Qi, R., Qian, F. and Zhong, W., *A Hybrid Improved Genetic Algorithm and Its Application in Dynamic Optimization Problems of Chemical Processes*, Chin. J. Chem. Eng. 21(2) (2013), 144-154.

[35] Tavakolpour, A.R., Mat Darus, I.Z., Tokhi, O. and Mailah, M., *Genetic algorithm-based identification of transfer function parameters for a rectangular flexible plate system*, Eng. Appl. Artif. Intell. 23(8) (2010), 1388-1397.

[36] Van Soest, A.J.K. and Casius, L.J.R.R., it The Merits of a Parallel Genetic Algorithm in Solving Hard Optimization Problems, J. Biomech. Eng.125(1) (2003), 141–146.

[37] Wang, Z. and Ju, G., *A parallel genetic algorithm in multi-objective optimization*, 2009 Chinese Control and Decision Conference, 3497–3501.

[38] Roberge, V., Tarbouchi, M. and Labonte, G., *Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning*, IEEE Trans. Ind. Inform. 9(1) (2013), 132–141.

[39] Wolf, S. and Merz, P., *Evolutionary local search for the super-peer selection problem and the p-Hub median problem*, Hybrid Metaheuristics, (2007), 1–15.

[40] Wu, X., Lei, B., Zhang, K. and Cheng, M., *Hybrid stochastic optimization method for optimal control problems of chemical processes*, Chem. Eng. Res. Des. 126 (2017), 297–310.

[41] Zhang, B., Chen, D. and Zhao, W., *Iterative ant-colony algorithm and its application to dynamic optimization of chemical process*, Comput. Chem. Eng. 29(10) (2005), 2078–2086.