SPC

International Journal of Applied Mathematical Research , 3 (4) (2014) 496-507
©Science Publishing Corporation
www.sciencepubco.com/index.php/IJAMR
doi : 10.14419/ijamr.v3i4.3538
Research Paper

Integrating PSO with modified hybrid GA for solving nonlinear optimal control problems

R. Ghanbari[1], **S. Nezhadhosein**[2*], **A. Heydari**[3]

[1]*Department of Applied Mathematics, Faculty of Mathematical science, Ferdowsi University of Mashhad, Mashhad, Iran*
[2,3]*Department of Applied Mathematics, Payame Noor University, Tehran, Iran*
**Corresponding author E-mail:s_ nezhadhossin@yahoo.com*

Abstract

Here, a two-phase algorithm based on integrating particle swarm optimization (PSO) with modified hybrid genetic algorithm (MHGA) is proposed for solving the associated nonlinear programming problem of a nonlinear optimal control problem. In the first phase, PSO starts with a completely random initial swarm of particles, where each of them contains two random matrices in time nodes. After phase 1, to achieve more accurate solutions, the number of time nodes is increased. The values of the associated new control inputs are estimated by linear or spline interpolations using the curves computed in the phase 1. In addition, to maintain the diversity in the population, some additional individuals are added randomly. Next, in the second phase, MHGA, starts by the new population constructed by the above procedure and tries to improve the obtained solutions at the end of phase 1. MHGA combines a GA with a successive quadratic programming, SQP, as a local search.
Finally, we implement the proposed algorithm on some well-known nonlinear optimal control problems. The numerical results show that the proposed algorithm can find almost better solution than other proposed algorithms.

Keywords: *Optimal control problem, Hybrid genetic algorithm, particle swarm optimization, Spline interpolation.*

1. Introduction

Nonlinear optimal control problems (NOCP) are dynamic optimization problems with many applications in industrial processes such as airplane, robotic arm, bio-process system, biomedicine, electric power systems, plasma physics and etc., [17]. Many methods for solving NOCP, either direct or indirect, rely upon gradient information and therefore may converge to a local optimum [1]. In direct methods [17], the original continuous-time problem is approximated by a finite-dimensional nonlinear programming problem using discrete states and control variables. The major drawback of these methods is shortage of accuracy. In indirect approaches, the problem through the use of the Pontryagins minimum principle (PMP), is converted to two boundary value problems (TBVP) that can be solved by numerical methods such as shooting method [17]. These methods have two major disadvantages. First, they may converge to a local optimum. Next, they require good initial guesses that lie within the domain of convergence.
Metaheuristics as the global optimization methods can overcome these problems. They differ from classic methods. They don't really need good initial guesses and deterministic rules. Some of these methods are; Genetic algorithm (GA), see [1, 23, 24], Genetic programming (GP), see [18], Particle swarm optimization (PSO), see [3, 4, 21], Ant colony optimization (ACO), see [27] and Differential evolution (DE), see [8, 19, 28].
Many authors proposed many types of metaheuristics for solving NOCPs. Wang and Chiou [28] proposed a DE to solve NOCPs described by differential-algebraic systems with nonlinear constraints. Lee et al. [19] used a modified DE algorithm for dynamic optimization of a continuous polymer reactor. Sim et al. [24] combined a GA and the shooting method for solving NOCP. Cruz et al. [8] used efficient DE algorithms for solving multi-modal NOCPs. Arumugam and Rao [4] considered the

popular GA operator, cross-over and root mean square variants into PSO algorithm to make a faster convergence. Arumugam et al. [3] used various optimization algorithms, including PSO, with time varying inertia weight methods, and PSO with globally and locally tuned parameters to solve the NOCPs for steel annealing processes. van Ast et al. [27] proposed a novel ACO approach to solve NOCP. Kumar and Balasubramaniam [18], using GA, solved NOCP for a linear system with quadratic performance.

Shi et al. [23] presented an improved GA with variable population-size inspired by the natural features of the variable size of the population used to continuous optimization problems. Ghosh et al. [13] used an ecologically inspired optimization technique for solving NOCPs. They used Bézier curves to parameterize the control functions. Abo-Hammour et al. [1] applied the continuous GA for solving NOCPs. They used smooth genetic operators to solve NOCPs.

To increase the quality of solutions and decrease the running time, hybrid methods were introduced, which used a local search in the implementation of a population-based metaheuristics [5]. Modares and Naghibi-Sistani [21] proposed a hybrid algorithm by integrating an improved PSO with successive quadratic programming (SQP) for solving NOCPs. Recently, Sun *et al.* [25] proposed a hybrid improved GA, which used Simplex method (SM) to perform a local search, for solving NOCPs and applied it for chemical processes.

Based on the success of the hybrid methods for solving NOCPs, mentioned above, we here use a modified hybrid Genetic algorithm (MHGA), which combines GA with SQP, see [7], as a local search. SQP is an iterative algorithm for solving nonlinear programming (NLP) problems, which uses gradient information. It can moreover be used for solving NOCPs, see [10, 14, 26]. For decreasing the running time in the early generations (iterations) of MHGA, a less number of iterations for SQP was used and then, when the promising region of search space was found, we increase the number of iterations of SQP, gradually.

To perform the new algorithm, the time interval is uniformly divided by using a constant number of time nodes. Next, in each of these time nodes, the control variable is approximated by a scaler vector of control input values. Thus, an infinite dimensional NOCP is changed to a finite dimensional NLP. Now, we encounter two conflict situations: the quality of the global solution and the needed computational time. In other words, when the number of time nodes is increased, then we expect the quality of the global solution is also increased, but we know that in this situation the computational time is increased dramatically. In other situation, if we consider less number of time nodes, then the computational time is decreased but we may find a poor local solution. To conquer these problems, a two-phase algorithm is proposed. In the first phase (exploration phase), to decrease the computational time and to find a promising regions of search space, PSO uses a less number of time nodes, which is showed to converge rapidly to a near optimum solution [21]. After phase 1, to increase the quality of solutions obtained from phase 1, the number of time nodes is increased. Using the population obtained in the phase 1, the values of the new control inputs are estimated by Linear or Spline interpolations. Next, in the second phase (exploitation phase), MHGA uses the solutions constructed by the above procedure, as an initial population. Finally, the best individual in the last population is considered as an approximation of the optimal solution.

The paper is organized as follows: in Section 2, problem formulation is introduced. In Section 3, overview of the PSO and GA are presented. In Section 4, we introduce the proposed algorithm for solving NOCP. In Section 5, we provide some numerical NOCPs to examine the proposed algorithm. Results are compared with some numerical and heuristic methods. We conclude in Section 6.

2. Formulation of problem

The bounded continuous-time NOCP is considered as finding the control input $u(t) \in \mathbb{R}^m$, over the planning horizon $[t_0, t_f]$, that minimizes the cost functional:

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

subject to :

$$\dot{x}(t) = f(x(t), u(t), t), \tag{2}$$
$$c(x, u, t) = 0, \tag{3}$$
$$d(x, u, t) \leq 0, \tag{4}$$
$$\psi(x(t_f), t_f) = 0, \tag{5}$$
$$x(t_0) = x_0. \tag{6}$$

where $x(t) \in \mathbb{R}^n$ denotes the state vector for the system and $x_0 \in \mathbb{R}^n$ is the initial state. The functions $f : \mathbb{R}^{n+m} \times \mathbb{R} \to \mathbb{R}^n$, $g : \mathbb{R}^{n+m} \times \mathbb{R} \to \mathbb{R}$, $c : \mathbb{R}^{n+m} \times \mathbb{R} \to \mathbb{R}^{n_c}$, $d : \mathbb{R}^{n+m} \times \mathbb{R} \to \mathbb{R}^{n_d}$, $\psi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{n_\psi}$ and $\phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ are assumed to be sufficiently smooth on appropriate open sets. The cost function (1) must be minimized subject to dynamic (2), control

and state equality constraints (3) and control and state inequality constraints (4), the final state constraints (5) and the initial condition (6). A special case of the NOCPs is the linear quadratic regulator (LQR) problem where the dynamic equations are linear and the objective function is a quadratic function of x and u. The minimum time problems, tracking problem, terminal control problem and minimum energy are another special cases of NOCPs.

3. Overview of the PSO and GA

Here, we introduce PSO and MHGA as subprocedures for the main algorithm. Firstly the control variables are parameterized. Next, NOCP is changed into a finite dimensional NLP (See [11]). Now, we can imply an optimization algorithm to find the global solution of the corresponding NLP.

3.1. PSO

PSO, introduced by Kennedy and Eberhart [16] in 1995, is an evolutionary computation technique motivated by simulation of social individual and uses a population of random particles which flown through the problem space. Each particle has two components: position and velocity.
Here, the underling PSO has the following steps:
Initialization: The time interval is divided into N_t-1 subintervals using time nodes $t_0,\ldots,t_{N_t-1}$ and then control input values are computed (or selected randomly). This can be done by the following stages:

1. Let $t_k = t_0 + kh$, where $h = \frac{t_f - t_0}{N_t-1}, k = 0,1,\ldots,N_t-1$ and t_0 and t_f are the initial and final times, respectively.
2. The corresponding control input value at each time node t_k is an $m\times 1$ vector, u_k, as the position of particle with the corresponding velocity, v_k, which is similarly an $m\times 1$ vector. So, the k-th particle in the swarm has two $m\times N_t$ matrices, by the following components:

$$(U_{(k)})_{ij} = u_{left,i} + (u_{right,i} - u_{left,i}).r_{ij} \tag{7}$$
$$(V_{(k)})_{ij} = \alpha.(u_{left,i} + (u_{right,i} - u_{left,i}).r_{ij}) \tag{8}$$

where $i = 1,2,\ldots,m, j = 1,2,\ldots,N_t, k = 1,2,\ldots,N_p$, $\alpha \in [0,1]$ is constant parameter for confine the velocity, r_{ij} is a random number in $[0,1]$ with a uniform distribution and $u_{lelt}, u_{right} \in R^m$ are the lower and the upper bound vectors of control input values, which can be given by the problem's definition or the user (e.g. see the NOCPs no. 5 and 6 in Appendix). Next, we let $U_{(k)} = [u_0,u_1,\ldots,u_{N_t-1}] = (u_{ij})_{m\times N_t}, k = 1,2,\ldots,N_p$ as k-th particle (individual) of the population, control input matrix, which N_p is the size of the population.

Evaluation: For each control input matrix, $U^{(k)}, k = 1,2,\ldots,N_p$, the corresponding state variable is a $n\times N_t$ matrix, $X^{(k)}$, and it can be computed by the forth Runge-Kutta method on dynamic system (2) with the initial condition (6), approximately. Then, the performance index, $J(U^{(k)})$, is approximated by a numerical method (denoted by $\tilde{J}$). If NOCP includes equality or inequality constraints (3) or (4), then we add some penalty terms to the corresponding fitness value of the solution. Finally, we assign $I(U^{(k)})$ to $U^{(k)}$ as the fitness value as follows:

$$I(U^{(k)}) = \tilde{J} + \sum_{l=1}^{n_d}\sum_{j=0}^{N_t-1} M_{1l} max\{0, d_l(x_j,u_j,t_j)\} + \sum_{h=1}^{n_c}\sum_{j=0}^{N_t-1} M_{2h} c_h^2(x_j,u_j,t_j) + \sum_{i=p}^{n_\psi} M_{3p}\psi_p^2(x_{N_t-1},t_{N_t-1}) \tag{9}$$

where $M_1 = [M_{11},\ldots,M_{1n_d}]^T$, $M_2 = [M_{21},\ldots,M_{2n_c}]^T$ and $M_3 = [M_{31},\ldots,M_{3n_\psi}]^T$ are big numbers, as the penalty parameters , $c_h(.,.)$, $h = 1,2,\ldots,n_c$, $d_l(.,.)$, $l = 1,2,\ldots,n_d$, and $\psi_p(.,.)$, $p = 1,2,\ldots,n_\psi$ are defined in (3), (4) and (5), respectively.
Update: The new velocity and the new position for the k-th particle, in $iter$-th iteration, $V_{(k)}^{iter+1}$ and $U_{(k)}^{iter+1}$, are calculated as following:

$$V_{(k)}^{iter+1} = wV_{(k)}^{iter} + c_1r_1(Pbest_{(k)}^{iter} - U_{(k)}^{iter}) + c_2r_2(Gbest^{iter} - U_{(k)}^{iter}) \tag{10}$$
$$U_{(k)}^{iter+1} = U_{(k)}^{iter} + V_{(k)}^{iter+1} \tag{11}$$

where $k = 1,2,\ldots,N_p$, w is inertia weight, c_1 and c_2 are positive constants called cognitive and social parameters (or acceleration coefficients), respectively, r_1 and r_2 are random numbers in $[0,1]$ and $Pbest_{(k)}^{iter}$ is the best previous position for the k-th particle (personal best) and $Gbest^{iter}$ is the best previous position among all the particles in $iter$-th iteration (global best), $U_{(k)}^1 = U_{(k)}$, $V_{(k)}^1 = V_{(k)}$.

Also, we use time-varying inertia weight, TVIW, and time-varying acceleration coefficients, TVAC, for updating the inertia weight, w, and acceleration coefficients, c_1 and c_2, which are done by the following equations [4]:

$$w = w_f + (w_i - w_f)\frac{Maxiter - iter}{Maxiter} \tag{12}$$

$$c_k = c_{kf} + (c_{ki} - c_{kf})\frac{Maxiter - iter}{Maxiter}, \quad k = 1,2 \tag{13}$$

where w_i and w_f are the initial and final values of the inertia weight, respectively, c_{ki} and c_{kf}, for $k = 1,2$, are initial and final values of the acceleration coefficients, respectively, and $iter$ is the current iteration number and $Maxiter$ is the maximum number of allowable iterations.
Terminal conditions: The algorithm is terminated when one of the following conditions are met: the number of iterations is equal to $Maxiter$, running time is equal to $CPUTIME$.
The PSO algorithm is given in Algorithm 1.

Algorithm 1 PSO algorithm

{**Initialization**} Input the number of time nodes N_t, the size of population N_p, the initial and final values of the inertia weight, w_i and w_f, the initial and final values of the acceleration coefficient, c_{ki} and c_{kf}, for $k = 1,2$, the parameter for control velocity, α, the maximum number of iterations $Maxiter$, the maximum running time $CPUTIME$ and an initial population consist of initial position, U, and initial velocity of particles, V.
Let $iter = 1$.
while stopping conditions are not satisfied **do**
{**Evaluation**} Evaluate the fitness value of each particle by (9).
{**Update**} Update the inertia weight and acceleration coefficients, w and $c_k, k = 1,2$, by (12) and (13), new $Gbest^{iter}$ and new position of particles by (11).
Let $iter = iter + 1$
end while
Return the best particle of swarm as an approximate solution.

3.2. GA

GAs introduced by Holland in 1975, are a class of heuristics and probabilistic methods. These algorithms start with an initial population of solutions. This population is evaluated by using genetic operators that include selection, crossover and mutation. In the following, we introduce GA operators.

3.2.1. GA operators

Here, in MHGA, the underling GA has the following steps:
Initialization: The initial population is sequence random input matrices, similar to initialization in PSO, from previous section.
Evaluation: Similar to evaluation of particles of swarm in PSO, see (9).
Selection: To select two parents from population, we use a tournament operator with size 8 [9].
Crossover: When two parents $U_{(1)}$ and $U_{(2)}$ are selected, we use the following stages to construct an offspring:

1. Select the following numbers:

$$\lambda_1 \in [0,1], \lambda_2 \in [-\lambda_{max}, 0], \lambda_3 \in [1, 1+\lambda_{max}] \tag{14}$$

randomly, where λ_{max} is a random number in $[0,1]$.

2. Let:

$$of^k = \lambda_k U_{(1)} + (1-\lambda_k)U_{(2)}, \quad k = 1,2,3 \tag{15}$$

where $\lambda_k, k = 1,2,3$ is defined in (14). For $i = 1 \ldots m$ and $j = 1, \ldots, N_t$, if $(of^k)_{ij} > u_{right,i}$, then let $(of^k)_{ij} = u_{right,i}$ and if $(of^k)_{ij} < u_{left,i}$, then let $(of^k)_{ij} = u_{left,i}$.

3. Let $of = of^*$, where of^* is the best $of^i, i = 1,2,3$ constructed by (15).

Mutation: We apply a perturbation on each component of the offspring as follows:

$$(of)_{ij} = (of)_{ij} + r_{ij}.\beta, \quad i = 1,2,\ldots,m, j = 1,2,\ldots,N_t \tag{16}$$

where r_{ij} is selected randomly in $\{-1,1\}$ and β is a random number in $[0,1]$. If $(of)_{ij} > u_{right,i}$, then let $(of)_{ij} = u_{right,i}$ and if $(of)_{ij} < u_{left,i}$, then let $(of)_{ij} = u_{left,i}$.
Replacement: Here, in the underling GA, we use a traditional replacement strategy. The replacement is done, if the new offspring has two properties: First, it is better than the worst person in the population. Second, it isn't very similar to a person in the population.
Termination conditions: Underlying GA is terminated when at least one of the following conditions is occurred: over a specified number of generations, N_i, we don't have any improvement (the best individual is not changed), the maximum number of generations, N_g, is reached, or a predefined running time, $CPUTIME$, is achieved.

3.2.2. MHGA

In MHGA, GA uses a local search method to improve solutions. Here, we use SQP as a local search [7]. Using SQP as a local search in the hybrid metaheuristic is common for example see [21].
In the beginning of MHGA, a less number of iterations for SQP was used. Then, when the promising regions of search space were found, we increase the number of iterations of SQP gradually. Using this approach, we may decrease the needed running time (in [6] the philosophy of this approach is discussed).
Finally, we give the MHGA, in Algorithm 2.

Algorithm 2 MHGA algorithm

{**Initialization**} Input the number of time nodes N_t, the size of population N_p, the maximum number of generations without improvement N_i, the maximum number of generations N_g, the maximum running time $CPUTIME$, the mutation implementation probability P_m, the initial value of the maximum number of iterations in SQP, $sqpmaxiter$ and an initial population.
{**Evaluation**} Evaluate the fitness of each individual by (9).
{**Local search**} Perform SQP on each individual of the population when the maximum number of iteration is $sqpmaxiter$.
while stopping conditions are not satisfied **do**
  {**Selection**} Select two parents $U_{(1)}$ and $U_{(2)}$ by using an eight tournament from the population.
  {**Crossover**} Construct a new offspring, of, using (14) and (15).
  {**Mutation**} Apply (16) on of with probability P_m.
  {**Local search**} Perform SQP on of when the maximum number of iteration is $sqpmaxiter$.
  {**Replacement**}
  if replacement conditions are satisfied (see Section 3.2.1) **then** replace of with the worst individual of the papulation.
  end if
  Let: $sqpmaxiter := sqpmaxiter + 1$
end while
Return the best individual in the final population as an approximate solution of NOCP.

4. The proposed algorithm

Here, we propose a two-phase algorithm based on PSO and MHGA for solving NOCP, which is a direct approach. The main idea of the algorithm is to find promising regions of search space (converge rapidly to a near optimum solution) with a few number of time nodes, using PSO. Then, after finding good solutions, we increase the number of time nodes to improve the approximation of the optimal solution.
In the first phase, we perform PSO (Algorithm 1) with a completely random initial particles. Each particle has two matrices, a position matrix which contain control input values and a velocity matrix. Since the main goal in the first phase is to find the promising regions of the search space in a less running time, we use a few numbers of time nodes, here. In addition, to have a faster converged PSO, the size of the population of PSO in the first phase is usually less than the size of the population.
After phase 1, to maintain the property of individuals in the last iteration of the phase 1 and to increase the accurately of solutions, we add some additional time nodes. Thus, we increase time nodes from N_{t_1} in the phase 1 to N_{t_2} in the phase 2. The corresponding control input values of the new time nodes are added to individuals. To use the information of the obtained solutions from phase 1 in the construction of the initial population of the phase 2, we use either linear or spline interpolations to estimate the value of the control inputs in the new time nodes in each individual of the last population of phase 1. Moreover,

to maintain the diversity in the initial population of the phase 2, we add new random individuals to the population using (7). In the second phase, MHGA starts with this population and new parameters.
Finally, the proposed algorithm is given in Algorithm 3.

Algorithm 3 The proposed algorithm

{**Initialization**} Input $CPUTIME$ and let $CPUTTIME_1 := \frac{CPUTIME}{2}$
{**Phase 1**} Perform PSO (Algorithm 1) with random particles, position, U, velocity, V, and the parameters N_{t_1}, N_{p_1}, w_i, w_f, c_{ki}, c_{kf}, $k = 1,2$, α, $CPUTIME_1$ and $Maxiter$.
{**Construction of the initial population of the phase 2**} Increase time nodes uniformly to N_{t_2} and estimate the corresponding control input values of the new time nodes in each individual obtained from phase 1, using either linear or spline interpolations.
Create $N_{p_2} - N_{p_1}$ new different individuals with N_{t_2} time nodes, randomly.
Let $CPUTIME_2 := CPUTIME$ - the running time of the **Phase 1**.
{**Phase 2**} Perform MHGA (Algorithm 2) with the constructed random population and N_{t_2}, N_{p_2}, N_i, N_g, $CPUTIME_2$, P_m and $sqpmaxiter$.

5. Numerical experiments

In this section, in order to show the feasibility of the proposed algorithm, 23 NOCPs, which are described in Appendix in terms of eqns (1)-(6), are considered. These NOCPs are selected with single control signal and multi control signals, which will be demonstrated in a general manner. Numerical results for these problems, are summarized in Tables 1. The parameters, for each problem, are reported in Table 2.
The algorithm was implemented in Matlab R2011a environment on a Notebook with Windows 7 Ultimate, CPU 2.53 GHz and 4.00 GB RAM. Also, to implement SQP in the proposed algorithm, we used 'fmincon' in Matlab when the 'Algorithm' was set to 'SQP'. To set the parameters of the proposed algorithms, we ran them with different values of parameters and selected the best of them.
The notation φ_f, in Table 1, shows the norm of error in the final condition, i.e. $\varphi_f = \|\psi(x^*(t_{N_t}-1), t_{N_t}-1)\|_\infty$, where $\psi = [\psi_i]_{i=1}^{n_\psi}$; see (6). Moreover, we used Gap to compare the cost function's values, as follows:

$$Gap(J) = |\frac{J - J^*}{J^*}| \tag{17}$$

where J^* is the best value obtained for the cost function.

Remark 5.1 *We use the following abbreviations to show the used interpolation method in the proposed algorithm:*

1. *LI: linear interpolation.*
2. *SI: spline interpolation.*

Remark 5.2 *In first phase of Algorithm 3, in PSO, we let* $Maxiter = 1000$, $\alpha = 0.1$, $c_{1i} = c_{2f} = 2.5$, $c_{1f} = c_{2i} = 0.5$, $w_i = 0.9$, $w_f = 0.4$ *and in second phase, in MHGA, we let* $sqpmaxiter = 4$, $P_m = 0.8$. *Also, we use the composite Simpson method to approximate* (1).

Table 1, shows the numerical results (the cost function, J, the norm of error in the final condition, φ_f and Gap) for NOCPs given in Appendix. the proposed algorithm is compared with some recently proposed algorithms. These methods consist of SUMT and SQP, proposed in [11], continuous genetic algorithm, CGA [1], (better than SUMT), IPSO-SQP [21], (more accurate than some heuristic algorithms such as GA [22], DE [8], and PSO [15]) and some numerical methods [12, 14, 29].
Because of the stochastic nature of the proposed algorithm, 10 different runs were made for each result. From Table 1, the following observations can be achieved:

1. The proposed algorithm could find the best cost function's value. It is seen that SI could find the best solution among all algorithms in 52 percent of test problems, also LI could find the best solution among all algorithms in 17 percent of test problems and for the other cases are equal.

2. Unfortunately, the final condition didn't report in many previous works. Thus, we only used the available results of other algorithms. Only in 70 percent of NOCPs given in Appendix the final condition is exist (see (5)), and among them

only in 37 percent it is reported. The proposed algorithm in 87 percent of them give better results.
Table 1, also shows the proposed algorithm was a robust algorithm from the final condition perspective with respect to the other algorithm on the available data.

3. The value of Gap for the proposed algorithm, LI or SI methods, is better than other methods. From this point of view, the proposed algorithm could find the best cost function.

Table 1: Numerical results for NOCPs described in Appendix.

Problem	**Algorithm**	J	φ_f	Gap
VDPO	Abo-Hammour et al. [1]	1.7404	$\mathbf{2.67 \times 10^{-11}}$	0.0687
	LI	1.6322	1.01×10^{-4}	0.0023
	SI	**1.6284**	1.02×10^{-4}	**0**
CRP	Abo-Hammour et al. [1]	0.0163	$\mathbf{7.5 \times 10^{-10}}$	0.0251
	LI	0.0160	1.45×10^{-7}	0.0062
	SI	**0.0159**	1.31×10^{-8}	**0**
FFRP	Abo-Hammour et al. [1]	83.63	3×10^{-4}	0.6213
	LI	**51.58**	$\mathbf{2 \times 10^{-4}}$	**0**
	SI	54.14	0.0014	0.0496
CSTCR	Modares and Naghibi-Sistani [21]	0.1354	0.0025	0.0407
	Ali et al. [2]	$J \in [0.135, 0.245]$	NR[a]	0.0376[b]
	Cruz et al. [8]	$J \in [0.1358, 0.1449]$	NR	0.0438[b]
	LI	0.1332	$\mathbf{2 \times 10^{-4}}$	0.007
	SI	**0.1301**	0.0029	**0**
MSNIC	Modares and Naghibi-Sistani [21]	0.1727	—	0.0069
	Goh and Teo [14]	0.1816	—	0.0588
	Mekarapiruk and Luus[20]	0.1769	—	0.0314
	LI	0.1719	—	0.0023
	SI	**0.1715**	—	**0**
NOCP no. 6	Zhang and He-ping [29]	0.0266	—	0.0114
	LI	**0.0263**	—	**0**
	SI	**0.0263**	—	**0**
NOCP no. 7	Ghomanjani et al. [12]	-5.3898	0.1387	0.0358
	LI	**−5.5902**	0.0617	**0**
	SI	-5.4926	**0.0360**	0.0174
NOCP no. 8	Ghomanjani et al. [12]	0.1713	0.0021	0.0076
	LI	**0.1700**	**0.0007**	**0**
	SI	0.1717	0.0009	0.01
NOCP no. 9	SUMT [11]	5.15×10^{-6}	—	1.33×10^{-14}
	SQP [11]	6.57×10^{-6}	—	1.70×10^{-14}
	LI	7.19×10^{-8}	—	8.72×10^{-17}
	SI	$\mathbf{3.84 \times 10^{-8}}$	—	**0**
NOCP no. 10	SUMT [11]	1.7980	—	0.0697
	SQP [11]	1.7950	—	0.0680
	LI	1.7342	—	0.0268
	SI	**1.6807**	—	**0**
NOCP no. 11	SUMT [11]	0.1703	—	2.0410
	SQP [11]	0.2163	—	2.8625
	LI	0.0561	—	0.0017
	SI	**0.0560**	—	**0**
NOCP no. 12	SUMT [11]	**3.2500**	NR	**0**
	SQP [11]	**3.2500**	NR	**0**
	LI	**3.2500**	$\mathbf{7.62 \times 10^{-7}}$	**0**
	SI	**3.2500**	1.57×10^{-6}	**0**
NOCP no. 13	SUMT [11]	−0.2490	NR	0.0016
	SQP [11]	−0.2490	NR	0.0016

Continued on next page

Table 1 – *Continued from previous page*

Problem	**Algorithm**	J	φ_f	Gap
	LI	$\mathbf{-0.2494}$	4.11×10^{-8}	**0**
	SI	$\mathbf{-0.2494}$	$\mathbf{4.11 \times 10^{-9}}$	**0**
NOCP no. 14	SUMT [11]	0.0167	NR	0.2189
	SQP [11]	0.0168	NR	0.2262
	LI	**0.0137**	$\mathbf{4.91 \times 10^{-8}}$	**0**
	SI	**0.0137**	7.52×10^{-7}	**0**
NOCP no. 15	SUMT [11]	3.7700	NR	0.1406
	SQP [11]	3.7220	NR	0.1261
	LI	**0.3052**	$\mathbf{5.34 \times 10^{-7}}$	**0**
	SI	0.3053	1.03×10^{-5}	**0**
NOCP no. 16	SUMT [11]	9.29×10^{-4}	NR	5.09×10^{-10}
	SQP [11]	1.01×10^{-3}	NR	1.42×10^{-9}
	LI	$\mathbf{8.84 \times 10^{-4}}$	$\mathbf{5.73 \times 10^{-9}}$	**0**
	SI	8.91×10^{-4}	4.84×10^{-8}	7.91×10^{-11}
NOCP no. 17	SUMT [11]	2.2080	NR	0.0420
	SQP [11]	2.2120	NR	0.0439
	LI	2.1313	10^{-5}	0.0058
	SI	**2.1189**	$\mathbf{10^{-6}}$	**0**
NOCP no. 18	SUMT [11]	−8.8690	NR	2.25×10^{-5}
	SQP [11]	−8.8690	NR	2.25×10^{-5}
	LI	**−8.8692**	$\mathbf{1.96 \times 10^{-7}}$	**0**
	SI	**−8.8692**	8.46×10^{-6}	**0**
NOCP no. 19	SUMT [11]	0.0368	—	0.1288
	SQP [11]	0.0368	—	0.1288
	LI	**0.0326**	—	**0**
	SI	**0.0326**	—	**0**
NOCP no. 20	SUMT [11]	76.83	NR	0.1477
	SQP [11]	77.52	NR	0.1581
	LI	70.03	**0.0170**	0.0462
	SI	**66.94**	0.7001	**0**
NOCP no. 21	SUMT [11]	0.3428	NR	13.1069
	SQP [11]	0.3439	NR	13.1522
	LI	0.0377	**0.0023**	0.0134
	SI	**0.0243**	0.0071	**0**
NOCP no. 22	SUMT [11]	5.27×10^{-3}	NR	0.1285
	SQP [11]	5.19×10^{-3}	NR	0.1133
	LI	4.84×10^{-3}	**3.7780**	0.0364
	SI	$\mathbf{4.67 \times 10^{-3}}$	3.8508	**0**
NOCP no. 23	SUMT [11]	5.22×10^{-3}	NR	0.0610
	SQP [11]	5.19×10^{-3}	NR	0.0549
	LI	$\mathbf{4.92 \times 10^{-3}}$	2.4898	**0**
	SI	5.01×10^{-3}	**2.4105**	0.0183

[a] Not reported.
[b] Gap is calculated by lower bound of interval.

6. Conclusions

In this paper, we gave a two-phase algorithm based on integrating PSO with MHGA for solving the associated NLP of a NOCP. In the first phase, PSO started with a completely random initial swarm of particles, where each of them had two random matrices; a position matrix, or solution, which contain control input values in time nodes and a velocity matrix. After phase 1, to achieved more accurate solutions, we increased the number of time nodes. The values of the associated new control inputs were estimated by linear or spline interpolations using the curves computed in the phase 1. In addition, to maintain

Table 2: The parameters of the proposed algorithm for the NOCPs in Appendix.

Problem	Parameters								
	N_{p_1}	N_{p_2}	N_{t_1}	N_{t_2}	N_g	N_i	u_{left}	u_{right}	$CPUTIME$
1	25	30	51	121	7000	5500	-2	2	60
2	12	15	31	41	5000	4000	-1.5	2	80
3	9	15	21	35	5000	2800	-15	10	90
4	12	15	31	151	2000	1300	-7	7	70
5	12	15	41	51	2000	1200	-20	20	60
6	11	15	41	51	2000	1200	-1	1	20
7	11	15	71	251	2000	1200	-2	2	40
8	11	15	31	131	3000	2200	-5	15	150
9	25	35	21	91	7000	5500	0	2	30
10	35	45	21	51	5000	3500	-1	1	30
11	25	35	21	91	7000	5500	-20	20	30
12	35	45	21	51	5000	3500	-3	3	30
13	15	25	31	71	7000	5500	-1	1	30
14	15	20	31	61	7000	5500	-2	2	30
15	25	35	21	51	7000	5500	$-\pi$	π	30
16	42	55	9	13	7000	5500	-1	1	30
17	15	25	31	131	5000	4000	-3	3	40
18	25	35	31	131	5000	4000	-30	30	40
19	9	15	51	91	5000	4000	-2	2	50
20	9	15	21	35	5000	2800	-15	10	90
21	9	12	9	11	5000	2800	-2	2	90
22	9	12	15	21	5000	3600	$\begin{bmatrix} -2.8 \\ -0.8 \end{bmatrix}$	$\begin{bmatrix} 2.8 \\ 0.7 \end{bmatrix}$	130
23	9	15	11	15	6000	4600	$\begin{bmatrix} -2.8 \\ -0.8 \end{bmatrix}$	$\begin{bmatrix} 2.8 \\ 0.7 \end{bmatrix}$	150

the diversity in the population, some additional individuals were added randomly. Next, in the second phase, MHGA, started by the new population constructed by the above procedure and tried to improve the obtained solutions at the end of phase 1. MHGA combined a GA with a SQP, as a local search. In MHGA, to decrease the running time in the early iterations, a less number of iterations of SQP was used. Then, after finding the promising regions of the search space, we increased the number of iterations for SQP gradually. Finally, we implemented the proposed algorithm on some well-known NOCPs. The numerical results showed that we can find almost better solution than other proposed algorithms.

References

[1] Zaer S. Abo-Hammour, Ali Ghaleb Asasfeh, Adnan M. Al-Smadi, and Othman M. K. Alsmadi, *A novel continuous genetic algorithm for the solution of optimal control problems*, Optimal Control Applications and Methods **32** (2011), no. 4, 414–432.

[2] M.M. Ali, C. Storey, and A. Törn, *Application of stochastic global optimization algorithms to practical problems*, Journal of Optimization Theory and Applications **95** (1997), no. 3, 545–563 (English).

[3] M. Senthil Arumugam, G. Ramana Murthy, and C. K. Loo, *On the optimal control of the steel annealing processes as a two stage hybrid systems via PSO algorithms*, International Journal Bio-Inspired Computing **1** (2009), no. 3, 198–209.

[4] M. Senthil Arumugam and M. V. C. Rao, *On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems*, Application Soft Computing **8** (2008), no. 1, 324–336.

[5] Saman Babaie-Kafaki, Reza Ghanbari, and Nezam Mahdavi-Amiri, *Two effective hybrid metaheuristic algorithms for minimization of multimodal functions*, International Journal Computing Mathematics **88** (2011), no. 11, 2415–2428.

[6] Saman Babaie-Kafaki, Reza Ghanbari, and Nezam Mahdavi-Amiri, *An efficient and practically robust hybrid metaheuristic algorithm for solving fuzzy bus terminal location problems*, Asia-Pacific Journal of Operational Research **29** (2012), no. 2, 1–25.

[7] J.J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal, *Numerical optimization: Theoretical and practical aspects*, Springer London, Limited, 2006.

[8] I.L. Lopez Cruz, L.G. Van Willigenburg, and G. Van Straten, *Efficient differential evolution algorithms for multimodal optimal control problems*, Applied Soft Computing **3** (2003), no. 2, 97 – 122.

[9] A.P. Engelbrecht, *Computational intelligence: An introduction*, Wiley, 2007.

[10] Brian C. Fabien, *Numerical solution of constrained optimal control problems with parameters*, Applied Mathematics and Computation **80** (1996), no. 1, 43 – 62.

[11] Brian C. Fabien, *Some tools for the direct solution of optimal control problems*, Advances Engineering Software **29** (1998), no. 1, 45–61.

[12] F Ghomanjani, M.H Farahi, and M Gachpazan, *Bézier control points method to solve constrained quadratic optimal control of time varying linear systems* , Computational and Applied Mathematics **31** (2012), 433 – 456 (en).

[13] Arnob Ghosh, Swagatam Das, Aritra Chowdhury, and Ritwik Giri, *An ecologically inspired direct search method for solving optimal control problems with Bézier parameterization*, Engineering Applications of Artificial Intelligence **24** (2011), no. 7, 1195 – 1203.

[14] C.J. Goh and K.L. Teo, *Control parametrization: A unified approach to optimal control problems with general constraints*, Automatica **24** (1988), no. 1, 3 – 18.

[15] Fernando Herrera and Jie Zhang, *Optimal control of batch processes using particle swam optimisation with stacked neural network models*, Computers and Chemical Engineering **33** (2009), no. 10, 1593 – 1601.

[16] J. Kennedy and R. Eberhart, *Particle swarm optimization*, Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 4, 1995, pp. 1942–1948.

[17] Donald E. Kirk, *Optimal control theory: An introduction*, Dover Publications, 2004.

[18] A. Vincent Antony Kumar and P. Balasubramaniam, *Optimal control for linear system using genetic programming*, Optimal Control Applications and Methods **30** (2009), no. 1, 47–60.

[19] Moo Ho Lee, Chonghun Han, and Kun Soo Chang, *Dynamic optimization of a continuous polymer reactor using a modified differential evolution algorithm*, Industrial and Engineering Chemistry Research **38** (1999), no. 12, 4825–4831.

[20] Wichaya Mekarapiruk and Rein Luus, *Optimal control of inequality state constrained systems*, Industrial and Engineering Chemistry Research **36** (1997), no. 5, 1686–1694.

[21] Hamidreza Modares and Mohammad-Bagher Naghibi-Sistani, *Solving nonlinear optimal control problems using a hybrid IPSO - SQP algorithm*, Engineering Applications of Artificial Intelligence **24** (2011), no. 3, 476 – 484.

[22] Debasis Sarkar and Jayant M. Modak, *Optimization of fed-batch bioreactors using genetic algorithm: multiple control variables.*, Computers and Chemical Engineering **28** (2009), no. 5, 789–798.

[23] X. H. Shi, L. M. Wan, H. P. Lee, X. W. Yang, L. M. Wang, and Y. C. Liang, *An improved genetic algorithm with variable population-size and a PSO-GA based hybrid evolutionary algorithm*, Machine Learning and Cybernetics, 2003 International Conference on, vol. 3, 2003, pp. 1735–1740.

[24] Y. C. Sim, S. B. Leng, and V. Subramaniam, *A combined genetic algorithms-shooting method approach to solving optimal control problems*, International Journal of Systems Science **31** (2000), no. 1, 83–89.

[25] Fan SUN, Wenli DU, Rongbin QI, Feng QIAN, and Weimin ZHONG, *A hybrid improved genetic algorithm and its application in dynamic optimization problems of chemical processes*, Chinese Journal of Chemical Engineering **21** (2013), no. 2, 144 – 154.

[26] K.L. Teo, C.J. Goh, and K.H. Wong, *A unified computational approach to optimal control problems*, Pitman monographs and surveys in pure and applied mathematics, Longman Scientific and Technical, 1991.

[27] Jelmer Marinus van Ast, Robert Babuška, and Bart De Schutter, *Novel ant colony optimization approach to optimal control*, International Journal of Intelligent Computing and Cybernetics **2** (2009), no. 3, 414–434.

[28] Feng-Sheng Wang and Ji-Pyng Chiou, *Optimal control and optimal time location problems of differential-algebraic systems by differential evolution*, Industrial and Engineering Chemistry Research **36** (1997), no. 12, 5348–5357.

[29] Wen Zhang and He-ping Ma, *Chebyshev-legendre method for discretizing optimal control problems*, Journal of Shanghai University (English Edition) **13** (2009), no. 2, 113–118.

Appendix

The following NOCPs are described using (1)-(6).

1. [1, 10] (Van Der Pol oscillator problem (VDPO)) $g=(x_1^2+x_2^2+u^2)/2, t_0=0, t_f=5, f=[x_2,-x_2+(1-x_1^2)x_2+u]^T, x_0=[1,0]^T, \psi=x_1-x_2+1$.
2. [1, 17] (Chemical reactor problem (CRP)) $g=(x_1^2+x_2^2+0.1u^2)/2, t_0=0, t_f=0.78, f=[x_1-2(x_1+0.25)+(x_2+0.5)exp(25x_1/(x_1+2))-(x_1+0.25)u, 0.5-x_2-(x_2+0.5)exp(25x_1/(x_1+2))]^T, x_0=[0.05,0]^T, \psi=[x_1,x_2]^T$.
3. [1, 11] (Free floating robot problem (FFRP)) $g=(u_1^2+u_2^2+u_3^2+u_4^2)/2, t_0=0, t_f=5, f=[x_2,((u_1+u_2)\cos x_5-(u_2+u_4)\sin x_5)/M, x_4, ((u_1+u_3)\sin x_5+(u_2+u_4)\cos x_5)/M, x_6, (D(u_1+u_3)-L_e(u_2+u_4))/I]^T, x_0=[0,0,0,0,0,0]^T, \psi=[x_1-4, x_2, x_3-4, x_4, x_5, x_6]^T, M=10, D=5, I=12, L_e=5$.
4. [21] (Continuous stirred-tank chemical reactor (CSTCR)) $g=x_1^2+x_2^2+0.1u^2, t_0=0, t_f=0.78, f=[-(2+u)(x_1+0.25)+(x_2+0.5)exp(25x_1/(x_1+2)), 0.5-x_2-(x_2+0.5)exp(25x_1/(x_1+2))]^T, x_0=[0.09,0.09]^T$.
5. [21] (Mathematical system with nonlinear inequality constraint (MSNIC)) $\phi=x_3, t_0=0, t_f=1, f=[x_2,-x_2+u,x_1^2+x_2^2+0.005u^2]^T, d=[-(20-u)(20+u), x_2+0.5-8(t-0.5)^2]^T, x_0=[0,-1,0]^T$.
6. [29] $g=0.39(x_1^2+x_2^2+0.1u^2), t_0=-1, t_f=1, f=[0.39(-2(x_1+0.25)+(x_2+0.5)exp(25x_1/(x_1+2))-(x_1+0.25)u), 0.39(0.5-x_2-(x_2+0.5)exp(25x_1/(x_1+2)))]^T, x_0=[0.05,0]^T$.
7. [12] $g=2x_1, t_0=0, t_f=3, f=[x_2,u]^T, d=[-(2-u)(2+u),-6-x_1]^T, x_0=[2,0]^T$.

8. [12] $g = x_1^2 + x_2^2 + 0.005u^2, t_0 = 0, t_f = 1, f = [x_2, -x_2 + u]^T, d = x_2 + 0.5 - 8(t-0.5)^2, x_0 = [0,-1]^T$.
9. [11] $g = x^2 \cos^2 u, t_0 = 0, t_f = \pi, f = \sin u/2, x_0 = \pi/2$.
10. [11] $g = (x_1^2 + x_2^2 + u^2)/2, t_0 = 0, t_f = 5, f = [x_2, -x_1 + (1-x_1^2)x_2 + u]^T, d = -(x_2 + 0.25), x_0 = [1,0]^T$.
11. [11] $g = x_1^2 + x_2^2 + 0.005u^2, t_0 = 0, t_f = 1, f = [x_2, -x_2 + u]^T, d = [-(20-u)(20+u), 0.5 + x_2 - (8(t-0.5)^2]^T, x_0 = [0,-1]^T$.
12. [11] $g = u^2/2, t_0 = 0, t_f = 2, f = [x_2, u]^T, x_0 = [1,1]^T, \psi = [x_1, x_2]^T$.
13. [11] $g = -x_2, t_0 = 0, t_f = 1, f = [x_2, u]^T, d = -(1-u)(1+u), x_0 = [0,0]^T, \psi = x_2$.
14. [11] $g = (x_1^2 + x_2^2 + 0.1u^2)/2, t_0 = 0, t_f = 0.78, f = [-2(x_1 + 0.25) + (x_2 + 0.5)exp(25x_1/(x_1+2)) - (x_1 + 0.25)u, 0.5 - x_2 - (x_2 + 0.5)exp(25x_1/(x_1+2))]^T, x_0 = [0.05, 0]^T, \psi = [x_1, x_2]^T$.
15. [11] $g = u^2/2, t_0 = 0, t_f = 10, f = [\cos u - x_2, \sin u]^T, d = -(\pi - u)(\pi + u), x_0 = [3.66, -1.86]^T, \psi = [x_1, x_2]^T$.
16. [11] $g = (x_1^2 + x_2^2)/2, t_0 = 0, t_f = 0.78, f = [-2(x_1 + 0.25) + (x_2 + 0.5)exp(25x_1/(x_1+2)) - (x_1 + 0.25)u, 0.5 - x_2 - (x_2 + 0.5)exp(25x_1/(x_1+2))]^T, d = -(1-u)(1+u), x_0 = [0.05, 0]^T, \psi = [x_1, x_2]^T$.
17. [11] $\phi = x_3, t_0 = 0, t_f = 1, f = [x_2, u, u^2/2]^T, d = x_1 - 1.9, x_0 = [0,0,0]^T, \psi = [x_1, x_2 + 1]^T$.
18. [11] $\phi = -x_3, t_0 = 0, t_f = 5, f = [x_2, -2 + u/x_3, -0.01u]^T, d = -(30-u)(30+u), x_0 = [10, -2, 10]^T, \psi = [x_1, x_2]^T$.
19. [11] $\phi = (x_1 - 1)^2 + x_2^2 + x_3^2, g = u^2/2, t_0 = 0, t_f = 5, f = [x_3 \cos u, x_3 \sin u, \sin u]^T, x_0 = [0,0,0]^T$.
20. [11] $g = (u_1^2 + u_2^2 + u_3^2 + u_4^2)/2, t_0 = 0, t_f = 5, f = [x_2, ((u_1 + u_3)\cos x_5 - (u_2 + u_4)\sin x_5)/M, x_4, ((u_1 + u_3)\sin x_5 + (u_2 + u_4)\cos x_5)/M, x_6, (D(u_1 + u_3) - L_e(u_2 + u_4))/I]^T, x_0 = [0,0,0,0,0,0]^T, \psi = [x_1 - 4, x_2, x_3 - 4, x_4, x_5 - \pi/4, x_6]^T, M = 10, D = 5, I = 12, L_e = 5$.
21. [11] $g = 4.5(x_3^2 + x_6^2) + 0.5(u_1^2 + u_2^2), t_0 = 0, t_f = 1, f = [9x_4, 9x_5, 9x_6, 9(u_1 + 17.25x_3), 9u_2, -9(u_1 - 27.0756x_3 + 2x_5x_6)/x_2]^T, x_0 = [0, 22, 0, 0, -1, 0]^T, \psi = [x_1 - 10, x_2 - 14, x_3, x_4 - 2.5, x_5, x_6]^T$.
22. [11] Same as problem 21 with $d = [-(2.83374 - u_1)(2.83374 + u_1), -(0.71265 - u_2)(0.80865 + u_2)]^T$.
23. [11] Same as problem 21 with $d = [-(2.83374 - u_1)(2.83374 + u_1), -(0.71265 - u_2)(0.80865 + u_2), -(2.5 - x_4)(2.5 + x_4), -(1 - x_5)(1 + x_5)]^T$.