Differential Ant-Stigmergy Algorithm for Optimal Control Problems

Adrian SERBENCU
Department of Automation and Electrical Engineering, “Dunarea de Jos” University of Galati, Galati ,Romania
adrian.serbencu@ugal.ro

Viorel MINZU
Department of Automation and Electrical Engineering, “Dunarea de Jos” University of Galati, Galati ,Romania
viorel.minzu@ugal.ro

***Abstract*— The objective of this paper is to show how the Differential Ant-Stigmergy Algorithm (DASA) can be used to solve an Optimal Control Problem (OCP). A version of this algorithm devoted to search a good solution for an OCP is presented. Because this kind of problems has usually a large computational complexity, a technique to reduce it is presented. This one exploits a particularity of DASA related to the coding of the problem's solution. In the first phase, DASA find out a good solution in a small number of iteration by adopting a rough coding of the control input. The second phase is a kind of zoom of the search space around this solution, which becomes initial solution, with a finer coding of the control input.**

Keywords*— *optimal control problem, metaheuristic, Ant Colony Optimization

I. INTRODUCTION

Many papers have treated metaheuristics in order to solve optimization problems. Ant Colony Optimization algorithm was created for solving a combinatorial optimization problem that can be modeled as an optimal path in a graph [1]. Because ACO has proved to be an efficient algorithm, it is a challenge for many authors to adapt this algorithm to continuous optimization problems. The key aspect on this subject is how to model the depositation of the pheromone by the artificial ants. Differential Ant Stigmergy Algorithm (DASA) [2] has turned out to be a success adaptation of ACO to continuous problems. This algorithm takes a solution of the problem and modifies the values of its components using some variations (offsets) in order to create another solution. The offsets belong to a discretized domain. The paradigm of solutions construction using a path in a graph and updating the pheromone distribution acquired an efficient extension. DASA was successfully applied to solve benchmark problems [2] and diverse engineering applications [3]. Improvements of DASA's feats through an elitist strategy have been proposed and analyzed in [4] and [5]

Among the optimization problems there is a special category: the optimal control problems. Metaheuristic algorithms have already been developed in order to solve an optimal control problem (OCP) concerning a dynamic environment. An optimal solution is a sequence of values for the control variables, covering the control horizon that optimizes a performance index. The first metaheuristics used to solve OCPs have been the Simulated Annealing, Genetic Algorithms and Evolutionary Programming [6], [7]. [8]. Particle Swarm Optimization has been also used by the authors in this context either in its continuous formulation [9], or in its discrete version [10]. In the book [11], besides many engineering problems, applications solving OCP that are using metaheuristic algorithms are presented and analyzed.

This paper presents DASA and gives a description that is devoted to optimal control. Because OCPs have usually a large computational complexity, a method to decrease it is proposed. This one consists in a technique in two phases that can globally reduce the number of iterations until finding the optimal solution and implicitly the number of objective function evaluations.

Remark 1: DASA yields an offline "optimal" solution, which is a sequence of values for the control variables, covering the control horizon. This control sequence may only be used within an open-loop control structure. This is a possible but unusual situation (for known reasons). Firstly, the implementer has to verify that DASA yields good solutions, has good convergence and its computational complexity can justify the possibility of using it in a closed-loop structure. The goal of this paper is *only* to give to the reader a tool to make this analysis. If DASA passes these tests, further work has to be done by the implementer, in order to design the controller and integrate it in a closed loop structure. An effective approach is presented in [12] using Model Predictive Control.

The section II describes minimally an OCP and DASA in order to fix the scientific framework. The lecturer will find more details in [3]. The way of coding the control input is emphasized. The description of DASA is adapted to an OCP solving. An example of using DASA to solve an OCP is given in section III, where the concentration of a component in a Batch Reactor has to be maximized. In section IV, an evaluation of the computational complexity is presented in connection with the number of iterations until the convergence is apprehended. In this context, a technique that can reduce the computational complexity of DASA is proposed. A number of tests with MATLAB system are carried out and their results are described in section V. These results proved that the technique proposed to reduce the complexity turned out to be efficient. Some conclusions are drawn in section VI.

II. DASA FOR OPTIMAL CONTROL PROBLEMS

A given OCP considers the environment evolution on a control horizon $[t_0, t_N]$, with discrete moments $t_i=t_0+i\cdot T$, $i=0,\ldots, N$, where T is the sampling period and t_0 is the initial moment. If the value $X(t_0)$ of the initial state and the sequence of control inputs $U(t_0)$, $U(t_1)$, …, $U(t_{N-1})$ are known, then the sequence of state variables $X(t_1),\ldots, X(t_{N-1})$, $X(t_N)$ and the sequence of controlled (output) variables $Y(t_1),\ldots, Y(t_{N-1})$, $Y(t_N)$ can be calculated using a process model. In this work, we consider that the process model is a set of differential algebraic equations.

Let Π be the structure of the OCP defined as

$$\Pi =<f, \text{constraints}, t_0, N, X(t_0), J(t_0, N, X(t_0))>, \qquad (1)$$

where f is the function appearing in the state equation

$$dX/dt =f(X(t), Y(t),U(t),\ldots,t) \qquad (2)$$

$J(t_0, N, X(t_0))$ is the objective function, and "constraints" is the set of all algebraic and differential constraints imposed by the dynamic environment. To solve Π means finding the control sequence that optimizes (maximizes or minimizes) the objective function $J(.)$ on the control horizon, starting from the initial state $X(t_0)$.

For different reasons, especially when a deterministic algorithm is not known, we may decide to solve this problem using an approaching algorithm based on a metaheuristic, such as Genetic Algorithm, Particle Swarm Optimization, Ant Colony Systems, Simulated Annealing, etc. The main reason is the ability of such algorithms to cope with the high complexity of Π.

Let U be an optimal solution of the considered OCP that DASA algorithm will try to find out. It can be coded as a vector

$$U = [U_1, U_2, \ldots, U_N]$$

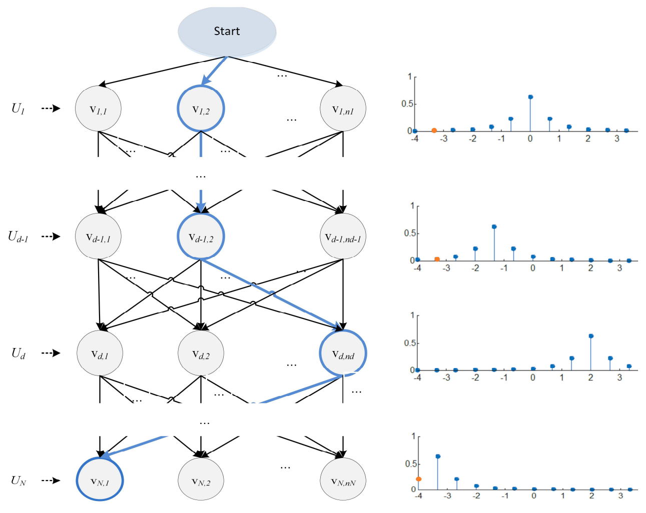

Fig.1. (a) Ant's search graph for an OCP
(b) Example of pheromone distribution over vertices based on Cauchy distributions sampling

that corresponds to the N sampling periods of the control horizon.

The basic elements of DASA, as described in [3], are reviewed hereafter and adapted to PCO. DASA uses a single current solution U' that is improved along the iterations based on the difference vectors proposed by ants. Let *n_ants* be the number of ants. At each iteration, each ant proposes a control sequence

$$U_m = U' + \omega\cdot \Delta^m \,;\, m=1,\ldots, n\_ants \qquad (3)$$

DASA uses a fine-grained discrete form of the search step. The vector used to update the current control sequence is $\Delta^m = [\delta_1, \delta_2, \ldots, \delta_i, \ldots, \delta_N]^T$. Each element δ_i has the form $\delta_i= b^x$ with $x\in \mathbf{Z}$ and b is a discrete base. ω is a weighting factor randomly chosen as an integer number having values between 1 and $b-1$. Therefore δ_i is an element of the set Δ_i of positive and negative differences that can be applied to the command U_i :

$$\begin{aligned}\Delta_i = &\{\delta^-_{i,k} \mid \delta^-_{i,k} = -b^{k+L_i-1}, k=1,2,\ldots,d_i\} \cup \\ &\cup 0 \cup \{\delta^+_{i,k} \mid \delta^+_{i,k} = +b^{k+L_i-1}, k=1,2,\ldots,d_i\}\end{aligned}. \qquad (4)$$

The values d_i are calculated based on the maximum and minimum values (U_i and L_i) of the admissible control inputs. A search graph $G = (V, E)$ is created, where V is the set of nodes and E is the set of edges, using the elements of the sets Δ_i , $1\leq i \leq N$, according to Fig.1. Each set Δ_i from (4) is represented by a set of nodes,

$$V_i= \{v_{i,1}, v_{i,2}, \ldots, v_{i,2di+1}\};\; V = V_1 \cup V_2 \cdots \cup V_n$$

In this way, a direct mapping is defined between the discrete values of the search steps and the nodes.

The search graph is fully connected, i.e. each node in the V_i set is connected to all the nodes belonging to the V_{i+1} set. We have an oriented graph in which any path p from the start node to any node belonging to V_N has the same length N. We shall represent such a path as a sequence of nodes:

$$p= (v_1,v_2,\ldots,v_N) \text{ with } v_i \in V_i,\; 1\leq i\leq N$$

DASA associates to each set V_i a Cauchy probability distribution that models the pheromone. In Fig.1a, the path in blue corresponds to a possible sampling of the Cauchy distributions given in Fig.1b.

The pseudocode description of the implemented version of DASA is presented in Fig. 2. The DASA iteratively improves the current control sequence $U_{current}$ by applying (3). In lines 1-2 an initial solution is generated based on available information. The call *Graph_initialization*(ε) generates the search graph based on the desired precision ε (the minimum value which can modify a control input) and the dimension of the search space. The call *Pheromone_initialization*(G) associates an initial amount of pheromone to each sets V_i whose value corresponds to a standard Cauchy distribution with probability density

$$C(x) = 1 \Big/ \left[s \cdot \pi \left(1 + (x - l/s)^2 \right) \right], \quad (5)$$

where l is the local offset with initial values $l_0=0$, s is the scale factor, $s= s_{global}-s_{local}$ with initial values $s_{global,0}=1$, $s_{local,0}=0$. The nodes are equidistantly arranged between $x = [-4,4]$.

Until a stop criterion is met, the ants construct in parallel *n_ants* paths (lines 9-18). The call *Find_path(G)* is made by each ant. Each ant begins from the start node (Fig. 1a) and adds to its partial path a node in accordance with the probabilistic rule specific to ACO. The ant #m at time i moves from the node $v_{i-1,m} \in V_{i-1}$, to a node $v_{i,j} \in V_i$ with the probability

$$P(m, \mathbf{v}_{i,j}) = \tau(\mathbf{v}_{i,j}) \Big/ \sum\nolimits_{1 \le k \le 2^* d_i + 1} \tau(\mathbf{v}_{i,k}),$$

where $\tau(v_{i,j})$ is the pheromone of node j from V_i, given by sampling (5). In pheromone implementation just s and l are memorized for each set V_i. The ants sample a node $v_{i,j}$ by using the inverse of cumulative distribution function(the quantile) of the Cauchy distribution (see [3]). If the search is trapped in a local optimum and the pheromone directs the ants to construct null difference paths(cycle lines 11-18), then the search is restarted from a new initial solution (line 15) and the pheromone is also reset (line 16).

After each ant constructed its path, new control sequences are computed based on these paths (lines 19-20). The quality of the new solutions is evaluated. Let M be the best ant of the current iteration (lines 21-25). If it improves the current solution $U_{current}$ then this one will be replaced (lines 26-27). The call *Scale_update(s_{global}, s_{local})* increases s_{global} according to s_+, the global scale-increasing factor, and s_{local} is set to half of s_{global} (line 28). The call *Pheromon_redistribution(p_M)* updates the offsets' location of the Cauchy distribution according to the path p_M (line 29). Furthermore if the new $U_{current}$ sequence control improves the global best solution, U_{best}, then this one will be replaced (lines 30-31). If no improvement is found then the call *Scale_update(s_{global})* decreases s_{global} according to s_-, the global scale-decreasing factor, (line 33) and the pheromone is evaporated by moving the offset location l towards 0 (line 34). When the *stop_criterion* is met, the best found solution is returned (line 36-37). [2] details on how s_{global} and s_{local} balance between exploration and exploitation in DASA.

III. EXAMPLE OF USING DASA FOR SOLVING AN OCP

In this section it is considered an OCP taken from [6], where it is solved using the simulated annealing algorithm. The optimization problem is to maximize an intermediate product after a fixed reaction time. The reaction model involves a component A that is consumed by chemical reaction producing the product B. At high temperatures, B further reacts to the undesired by-product C:

$$2A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

The reaction rate is a function of temperature and concentration. The objective is to maximize $c_B(t_f)$ the concentration of component B at the end of the optimization time horizon tf of 1 h, i.e. $t = [0,3600]$:

The Batch Reactor continuous model is based on the Arrhenius approach and consists in three differential equations

$$\frac{dc_A}{dt} = -k_{1,0} \cdot e^{-E_1/RT} c_A^2$$

$$\frac{dc_B}{dt} = k_{1,0} \cdot e^{-E_1/RT} c_A^2 - k_{2,0} \cdot e^{-E_2/RT} c_B$$

$$\frac{dc_C}{dt} = k_{2,0} \cdot e^{-E_2/RT} c_B^2 - k_{2,0} \cdot e^{-E_2/RT} c_C$$

c_j , are the concentrations of components $j=A, B, C$ expressed in (mol mol^{-1}), $k_{1,0} = 4000\, mol^{-1} s^{-1}$ and $k_{2,0} = 6.2 \times 10^5\, s^{-1}$ are the rate coefficient constants, $E_1 = 5000$ cal/mol and $E_2 = 10000$ cal/mol are the activation energies, R=1.98721 cal mol^{-1}K^{-1} is the universal gas constant and T is the temperature in (K). T is chosen as the control variable and the concentrations c_A, c_B, c_C are the state variables. At the beginning the reactor is filled with component A, giving the following initial conditions:

$$c_A(t_0) = 1\, ;\; c_B(t_0) = 0\, ;\; c_C(t_0) = 0\, .$$

The temperature has to meet the bound constraints

$$298K \le T \le 398K$$

The objective function that has to be maximized is

$$J(U) = c_B(t_f)$$

For our problem, the samples of the control variable are the values of the control temperature:

$$U_i = T_i\, ,\;\; i=1,\ldots,N$$

Generally speaking, the sampling period is determined considering control engineering aspects. Because the control horizon is already set, the value of N used in coding the control variable is implicitly determined. In our implementation we considered N=50, that corresponds to a sampling period of 72 s. The implementation of DASA devoted to solve this OCP uses *n_ants*=10. Other parameters of the algorithm are set as follow: b=10, desired precision $\varepsilon=10^{-6}$, $s_+ = 0.02$ (global scale-increasing factor), $s_- = 0.01$ (global scale-decreasing factor)

In the case of solving an OCP, the evaluation of the objective function for a given control input U involves the numerical integration of the dynamic system over the control horizon. In our tests we used MATLAB system for implementing DASA that has some functions devoted to the numerical integration. *These functions make also the implicit discretization of the process model.* In the first tests, the stop criterion was a maximum number if iterations equal to 500. The results obtained after a typical run - an execution is a random process - are illustrated in Fig. 3 and 4.

```
$U_{current}$ =Initial_solution()
$U_{best}$= $U_{current}$
G = Graph_initialization(ε)
Pheromone_initialization(G)
iter=0;
repeat
    iter = iter +1
    k = 0
    for m=1,...,n_ants
        $p_m$= 0
        repeat
            $p_m$ = Find_path(G)
            k = k+1
            if k≥N then
        $U_{current}$=Initial_solution()
        Pheromone_initialization(G)
                Restart()
        until $p_m \neq 0$ ■
    ω=Random(1,b-1)
    $U_m = U_{current} + ω*δ(p_m)$
    $J_{best\_of\_iter}$=inf
    for m=1,...,n_ants
        if $J(U_m) < J_{best\_of\_iter}$ then
            $J_{best\_of\_iter} = J(U_m)$
            M=m■
    if $J_{best\_of\_iter} < J( U_{current} )$ then
        $U_{current} = U_M$
        s= Scale_update($s_{global}$, $s_{local}$)
        Pheromon_redistribution($p_M$)
        if $J(U_{best}) < J(U_{current})$ then
            $U_{best} = U_{current}$
    else
        s = Scale_update($s_{global}$)
        Pheromone_evaporation(G,ρ) ■
until stop_criterion ■
$J_{best} = J(U_{best})$
return $U_{best}$, $J_{best}$
```

Fig. 2. Differential ant-stigmergy algorithm

The iterative process begins with an initial solution U_{init}.The best solution, U_{best}, depicted in Fig. 3 generates the state variables evolution presented in Fig. 4. Let's note that the maximum value for J found out using DASA is 0.61067 that corresponds to the optimum known from [6].

IV. IMPROVING THE COMPUTATIONAL COMPLEXITY

In the case of solving an OCP, the evaluation of the objective function J for a given control input U has an important computational complexity, because it involves the numerical integration of the dynamic system over the control horizon. That is way the computational complexity of DASA may be estimated starting from the number of evaluations of $J(U)$. If the number of iterations until the run stops is denoted by *Iter*, the total number of evaluations is *Iter·n_ants*. Hence, *Iter* has to be as small as possible.

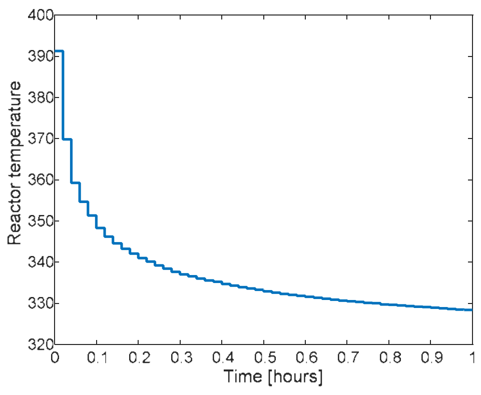

Fig. 3. The best solution found out in a typical run of DASA

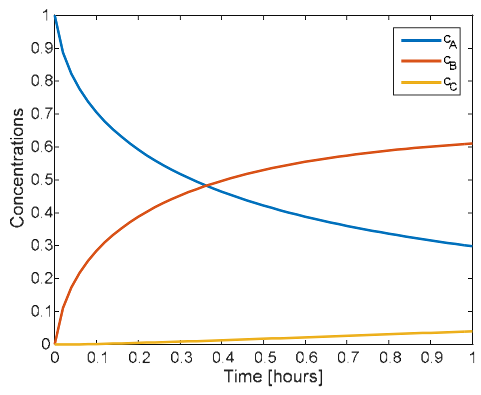

Fig. 4. State variables evolution with the best solution

In the implemented version, the DASA algorithm stops the iterative procedure as early as possible, in other words when the convergence is apprehended. For this purpose, a constant value is predefined: the number of iterations without improvement of J denoted by *no_imp*. If the algorithm already evolved along at list *n_min* iterations (*iter* > *n_min*) and the best solution has not been improved in the last *no_imp* iterations, one may consider that the algorithm has converged. For example, we used *n_min*=200 and *no_imp*=10.

On the other side, the complexity of the numerical integration of the dynamic system is obviously directly proportional to N. If we denote by $C(N)$ the complexity of the dynamic system's integration, then the computational complexity of DASA may be approximated by

$$\mathrm{Iter}\cdot \mathrm{n\_ants}\cdot \mathrm{C(N)}.$$

Obviously, it holds

$$N_1<N_2 \Rightarrow \mathrm{C}(N_1) < \mathrm{C}(N_2).$$

As a characteristic of OCPs, the control horizon is the given data of the problem, while N is a derived parameter that takes into consideration the sampling period. Let's consider DASA(N, $X(t_0)$, U_{init}) as the call of a procedure with 3 parameters, where $X(t_0)$ is the vector of the initial state values and U_{init} is the initial solution used in the iterative process. Let's also suppose that DASA(N_i, $X(t_0)$, U_{init}) starts

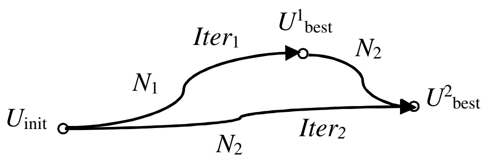

Fig. 5. Improving the complexity of DASA

from the same initial solution and converges after $Iter_i$ iterations to the best solution U^i_{best} and the best value of the objective function

$$J^i_{best} = J(U^i_{best}),\ i=1 \text{ or } 2\ .$$

At the first sight, we would be interested to choose a lower value for N. Our tests executed with $N_1 < N_2$ come to the conclusion that:

- $Iter_1 < Iter_2$, on condition that N_1 ensures the convergence;

- J^1_{best} is relatively close to the optimal value that is practically J^2_{best} .

Remark 2: U^1_{best} may be a very good initial solution for a new iterative process produced by DASA(N_2, $X(t_0)$, U^1_{best}) with a greater number of sampling periods as illustrated in Fig. 5. Our expectation is to find out the same performance index(J^2_{best}) in a number of iterations smaller than $Iter_2$-$Iter_1$.

Because the numerical tests confirmed this hypothesis, we propose hereafter a technique that can reduce the computational complexity of DASA in the context of a given OCP:

```
DASA(N_1, X(t_0), U_init)                                   1.
N_2 ← k·N_1                                                 2.
$U^2_{init} \leftarrow \text{expand}\left(U^1_{best}\right);$         3.
$\text{DASA}\left(N_2, X(t_o), U^2_{init}\right);$                   4.
$U^* \leftarrow U^2_{best}\,;$                                       5.
$J^* \leftarrow J(U^*)\,;$                                           6.
```

Fig. 6. Complexity improving technique

In Fig. 6, the step 3 makes an adjustment of the length of control input. Because the length of U^1_{best} is N_1, each element of this vector will be replicated k times as in the outline below:

$$[U^1_{\text{best}}(1)\ldots\ U^1_{\text{best}}(1)\ldots\ldots U^1_{\text{best}}(N_1)\ldots\ U^1_{\text{best}}(N_1)].$$

Let's recall that the control horizon is the same for the two iterative processes from steps 1 and 4. Finally the optimal solution U^* and the performance index J^* are set by the second iterative process.

TABLE I. RESULTS OF DASA FOR N=50

Trial	***Iter***	J_{best}
1	437	0,610641
2	552	0,610667
3	524	0,610631
4	477	0,610660
5	459	0,610653
6	472	0,610651
7	390	0,610529
8	447	0,610649
9	528	0,610648
10	481	0,610671
11	508	0,610646
12	383	0,610620
13	492	0,610649
14	539	0,610665
15	550	0,610666
Mean	483	0,610643

V. SIMULATION RESULTS

Simulation is an effective way to verify that DASA is appropriate for solving the given OCP (good solutions are obtained), it has good convergence and its computational complexity can justify the possibility of using it in a closed-loop structure (see *Remark* 1).

In a first group of tests, DASA was executed 15 times with an initial solution U_{init} and has generated the results shown in TABLE I. For each iterative process the column *Iter* gives the number of iterations until convergence and the column J_{best} indicates the best performance index. The computational complexity may be approximated by the value

$$Iter\cdot n\_ants\cdot C(N)=4830\cdot C(50). \qquad (6)$$

This means that the evaluation of objective function – which involves a numerical integration procedure with 50 sampling periods – is called 4830 times.

In order to apply the proposed technique reducing the complexity, the algorithm presented in Fig. 6 was executed 15 times for N_1=25 and N_2=50. It is important to notice that the initial solution is basically the same as that one used in the first group of tests. Because in the first execution of DASA the number of samplings is 25, the initial solution was set to $[U_{init}(1),\ U_{init}(3),\ldots,\ U_{init}(49)]$.

The results are shown in TABLE II where the columns $Iter_1$ and $Iter_2$ give the number of iterations until convergence for the two calls of DASA and the columns J_1 and J_2 indicate the best performance index.

The results confirm our expectation. On average, the computational complexity may be considered as being

$$Iter_1\cdot 10\cdot C(25)+ Iter_2\cdot 10\cdot C(50). \qquad (7)$$

Because of (12), the complexity is smaller than

$$(Iter_1+ Iter_2)\cdot 10\cdot C(50)=4070\cdot C(50) \qquad (8)$$

Comparing (6) and (8), it can be stated that the proposed technique has diminished the computational complexity. In TABLE III, the results of our technique with k=5 are presented. The complexity is not better than in TABLE II because the best solution U^1_{best} is farther from the optimal solution and the initial coding of the control variable is too rough, but is better that in TABLE I.

VI. Conclusion

This paper proposes to solve a given Optimal Control Problem using the Differential Ant-Stigmergy Algorithm. This is a metaheuristic that searches a good solution evolving from a single initial solution. A very important particularity of an OCP is that the evaluation of the objective function for a given control input has a substantial computational complexity, because it needs a numerical integration of the dynamic system over the control horizon (even if the problem has a Mayer type objective function, like in our example). This characteristic involves an important computational complexity.

Other characteristic of an OCP is that the control horizon is a given data. On the other hand the sampling period is chosen taking into account considerations related to control engineering. In this way the number N used to encode the control input variable is derived. But a big value for N involves a big or unacceptable complexity of DASA. That is why we have proposed a technique in two phases able to decrease the complexity. In the first phase, DASA find out a good solution in a small number of iteration by adopting a rough coding of the control input. The second phase is a kind of zoom of the search space around this solution, which becomes initial solution, with a finer coding of the control input. In this way DASA will converge relatively fast to the optimal solution.

Nevertheless, the value of N_1 has to be enough large in order to ensure the convergence to a good solution that would be sufficiently close to the optimum solution.

For the OCP treated as example in this paper and whose solution is known from a previous paper, the tests proved that the technique proposed to reduce the complexity turned out to be efficient.

TABLE II. Technique in two steps with N1=25, N2=50

Trial	***Iter₁***	***J₁***	***Itet₂***	***J₂***	***Iter₁+ Iter₂***
1	251	0.610496	137	0.610695	388
2	296	0.610491	132	0.610686	428
3	299	0.610521	122	0.610697	421
4	298	0.610516	160	0.610708	458
5	305	0.610528	128	0.610705	433
6	257	0.610494	101	0.610688	358
7	254	0.610505	101	0.61069	355
8	333	0.610503	128	0.610696	461
9	327	0.610515	115	0.610704	442
10	277	0.610499	111	0.610688	388
11	236	0.61051	118	0.610698	354
12	349	0.61053	102	0.610707	451
13	314	0.610512	110	0.610692	424
14	294	0.61048	121	0.610686	415
15	229	0.61045	110	0.610677	339
Mean	287	0.610503	119	0.610694	407

TABLE III. Technique in two steps with N_1=10, N_2=50

Trial	***I₁***	***J₁***	***I₂***	***J₂***	***I₁+ I₂***
1	247	0.610478	117	0.610697	364
2	264	0.610499	140	0.6107	404
3	292	0.610518	131	0.610697	423
4	289	0.610482	102	0.610696	391
5	263	0.610517	105	0.61069	368
6	235	0.610487	134	0.610697	369
7	264	0.610498	153	0.6107	417
8	218	0.610485	110	0.610679	328
9	266	0.610488	111	0.61069	377
10	310	0.610521	126	0.610665	436
11	280	0.610499	520	0.610712	800
12	237	0.610466	114	0.610693	351
13	266	0.610491	157	0.6107	423
14	238	0.610478	121	0.610679	359
15	327	0.610509	140	0.610703	467
Mean	266	0.610494	152	0.610693	418

References

[1] M. Dorigo, C.Blum, “Ant colony optimization theory: A survey,” Theoretical Computer Science, vol. 344, Issues 2-3, November 2005

[2] P. Korošec, J. Šilc, “Using stigmergy to solve numerical optimization problems,” Computing and Informatics, vol. 27, pp. 377–402, 2008

[3] P. Korošec, J. Šilc, K. Oblak, F. Kosel, “The Differential Ant-Stigmergy Algorithm: An experimental evaluation and a real-world application,” Proc. 2007 IEEE CEC, pp. 157-164, 2007

[4] A.E. Şerbencu, V. Mînzu, A. Şerbencu, D. Cernega, “Two elitist variants of Differential Ant-stigmergy Algorithm,” Proc. 8th ICINCO vol.1, pp. 136-141, 2011

[5] A.E. Şerbencu, A. Şerbencu,. “A comparison of particle swarm optimization and differential ant stigmergy algorithm,” Proc. 16th International Conference on System Theory, Control and Computing (ICSTCC),IEEE., pp. 1-6, October 2012

[6] R. Faber, T. Jockenhövelb, G. Tsatsaronis, “Dynamic optimization with simulated annealing,” Computers and Chemical Engineering, vol. 29, pp. 273–290, 2005

[7] D.B. Fogel, “Applying Evolutionary Programming to Selected Control Problems.” *Computers Math. Applic*. Vol. **27**, No. 11, pp. 89-104, Pergamon., 1994

[8] Z. Michalewicz, C. Janikow, J. Krawczyk, “A Modified Genetic Algorithm for Optimal Control Problems.” Computers Math. Applic. Vol. 23, No. 12, pp. 83-94, 1992

[9] V. Mînzu,.”Optimal Control Using Particle Swarm Optimization”, The 5th IEEEE International Symposium on Electrical and Electronics Engineering, 20-22 October, Galati, Romania, 2017

[10] V. Mînzu, et al., “A Binary Hybrid Topology Particle Swarm Optimization Algorithm for Sewer Network Discharge.” Proc. of 19th ICSTCC 2015, Romania, October 14-16, IEEE, pp 627-634. 2015

[11] J. Valadi and P. Siarry -editors, *Applications of Metaheuristics in Process Engineering*, ISBN 978-3-319-06507-6, Springer, 2014.

[12] V. Mînzu, and A. Serbencu. Control structure for the optimal sewer network discharge. In System Theory, Control and Computing (ICSTCC), 2016 20th International Conference on (pp. 61-66). IEEE