Cooperative Particle Swarm Optimization-Based Predictive Controller for Multi-robot Formation

Seung-Mok Lee and Hyun Myung

KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea
seungmok@kaist.ac.kr, hmyung@kaist.ac.kr

Abstract. In this paper, cooperative particle swarm optimization (CPSO)-based model predictive control (MPC) scheme is proposed to deal with the formation control problem of multiple nonholonomic mobile robots. In distributed MPC framework, control input of each robot needs to be optimized over a finite prediction interval considering control inputs of the other robots, where the objective function is coupled by the state variables of the neighboring robots. To solve the optimization problem on a prediction interval, we present a modified CPSO algorithm which finds a Nash equilibrium between multiple robots. Simulations are performed on a group of nonholonomic mobile robots to demonstrate the effectiveness of the proposed MPC scheme incorporating CPSO.

1 Introduction

Model predictive control (MPC) has been successfully applied to control complex systems in industry as one of the most popular optimal control techniques. The control technique is derived on the basis of the prediction of the future behavior estimated by the explicit model of the system. Thus, the application to nonlinear system is not easy because the nonlinear optimization process should be completed within a limited time.

Recently, some researchers have studied the possibility of applying evolutionary algorithms (EAs) to solve the optimization problem in MPC. Onnen et al. [1] suggested a genetic algorithm (GA) in order to optimize a control sequence, and they showed the effectiveness of the GA comparing to a branch-and-bound discrete search algorithm. Similar algorithms applying GA to MPC were presented in [2,3,4]. More recently, modified particle swarm optimization (PSO) algorithms have been combined with MPC as presented in [5,6,7,8] due to the fact that PSO algorithms provide quick results even with multiple objectives and constraints [6]. In particular, the PSO-based nonlinear MPC controller developed in [7] showed its better performance compared to a MPC controller using quadratic programming.

Cooperative particle swarm optimization (CPSO) algorithm is a variant of PSO, employing multiple swarms to optimize different variables of the solution in a cooperative coevolution (CC) framework. An early attempt to apply the CC framework to PSO was made by Bergh and Engelbrecht [9], resulting in two CPSO algorithms, namely CPSO-S$_K$ and CPSO-H$_K$. Recent studies by Li and Yao [10,11] suggest cooperative coevolving PSO (CCPSO) and CCPSO2 incorporating *random grouping* and *adaptive weighting* schemes, and their performance was validated on benchmark functions of up to 1,000 dimensions.

The problem of interest here is cooperative control between subsystems in MPC framework by including coupling terms in the objective function. Subsystems which are coupled in the objective function are referred as neighbors. In this paper, a distributed MPC scheme incorporating CPSO algorithm is presented where each subsystem is assigned its own optimization problem and communicates information only with neighboring subsystems. Thus, each subsystem has a particle swarm to optimize its objective function value, and the optimization problem is solved according to a Nash equilibrium strategy, i.e., in the optimization process, the swarms coevolve to reach a Nash equilibrium state by exchanging the best particles between the subsystems. In general, it has been found that finding the Nash equilibrium is very difficult given that the system is nonlinear [12]. In [13], a receding horizon Nash controller was developed for multi-agent systems, but it has been limited to linear systems. Thus, this paper proposes a distributed MPC incorporating CPSO having multiple swarms. The proposed MPC scheme is simulated for a formation control problem of multiple nonholonomic robots.

2 Modified CPSO for Distributed MPC

A Nash equilibrium strategy is a collection of strategies of all subsystems, and each strategy is the best response regarding the others' strategies. When the system reaches a Nash equilibrium, no subsystem can further improve their cost by changing its strategy given that all the others keep their strategies fixed or stationary. A Nash equilibrium strategy $(u_1^*, u_2^*, ..., u_M^*)$ is defined by the condition

$$J_i(u_1^*, u_2^*, ..., u_{i-1}^*, u_i^*, u_{i+1}^*, ..., u_M^*) \leq J_i(u_1^*, u_2^*, ..., u_{i-1}^*, u_i, u_{i+1}^*, ..., u_M^*) \quad (1)$$

for $i = 1, ..., M$ where J_i is the objective function of the i-th subsystem and u_i is an arbitrary control input.

In order to reach the Nash equilibrium, each subsystem needs first to know what the others' strategies are. However, the others' strategies are not available at the time when a subsystem computes its strategy since the state variables and control inputs of the multiple subsystems are coupled together in the objective functions, which results in a chicken and egg problem.

To deal with the problem, the concept of cooperative particle swarms is adapted such that all subsystems reach a Nash equilibrium at the same time in a distributed way. Since CPSO coevolves the multiple particle swarms by exchanging the global best particles with neighboring swarms, we have selected CPSO as the optimizer for multiple subsystems.

2.1 Cooperative Particle Swarms for Nash Equilibrium

For each subsystem, an objective function defined as a coupled form by the state variables and control inputs of the neighboring subsystems is given, and a particle swarm is assigned to each subsystem in order to optimize the given objective function. To reach the Nash equilibrium, each subsystem evolves its particle swarm with respect to its objective function by exchanging the global best particle with its neighboring swarms, while the neighboring swarms are also evolving.

The proposed cooperative PSO algorithm incorporates two new schemes to improve its performance. First, we put the best particle from the neighboring swarms together into the particle swarm to be optimized at the initialization step. Second, the best particles to be used to evaluate the objective function are updated at every generation. The details are given below.

2.1.1 Particle Initialization

At every update time step, particles of each swarm should be initialized in order to research optimal control input. When initializing the particles of i-th swarm, the global best particles from j-th swarm are used where $j \in \mathcal{N}_i$ and $\mathcal{N}_i$ is the set of the swarms neighboring with j-th swarm. The fact that the optimization process starts with the best particles from itself and neighboring swarms leads to improved convergence performance.

2.1.2 Objective Evaluation

Let $P_j.\mathbf{x}_{l-1}^i$ be the current position of the i-th particle of the j-th swarm at generation $l-1$, $P_j.\mathbf{y}_{l-1}^i$ the personal best of the i-th particle of the j-th swarm, and $P_j.\hat{\mathbf{y}}_{l-1}$ the global best particle of the j-th swarm. For M subsystems, at generation l, j-th subsystem optimizes its objective function J_j using the j-th swarm $P_{j,l-1}$ and the global best particles $P_i.\hat{\mathbf{y}}_{l-1}$ from $\mathcal{N}_j$ where $i \in \mathcal{N}_j$, and then the same process iterates through all swarms from $j=1$ to M at the same time. After the optimization process of a generation, each subsystem sends the global best particle to neighboring subsystems. In order to evaluate the objective, each particle $P_j.\mathbf{x}_i$ constructs a context vector $\hat{\mathbf{s}}_{\mathcal{N}_j}(P_j.\mathbf{x}_i)$ consisting of the particle $P_j.\mathbf{x}_l^i$ and the received best particles from its neighbor $\mathcal{N}_j$. For example, in three-subsystem case where their objective functions are coupled with each other, i.e., $\mathcal{N}_1 = \{2,3\}$, $\mathcal{N}_2 = \{1,3\}$, and $\mathcal{N}_3 = \{1,2\}$, the context vectors can be constructed as $\hat{\mathbf{s}}_{\mathcal{N}_1}(P_1.\mathbf{x}_i) = (P_1.\mathbf{x}_l^i, P_2.\hat{\mathbf{y}}_l, P_3.\hat{\mathbf{y}}_l)$, $\hat{\mathbf{s}}_{\mathcal{N}_2}(P_2.\mathbf{x}_i) = (P_1.\hat{\mathbf{y}}_l, P_2.\mathbf{x}_l^i, P_3.\hat{\mathbf{y}}_l)$, and $\hat{\mathbf{s}}_{\mathcal{N}_3}(P_3.\mathbf{x}_i) = (P_1.\hat{\mathbf{y}}_l, P_2.\hat{\mathbf{y}}_l, P_3.\mathbf{x}_l^i)$, at generation l. The subsystems reach a Nash equilibrium when there are no subsystems which can further improve their objective evaluation.

2.1.3 Update Rule

The update rule for the i-th particle in the j-th swarm is given by

$$P_j.\mathbf{x}_{l+1}^i = P_j.\mathbf{x}_l^i + P_j.\mathbf{v}_{l+1}^i, \tag{2}$$

$$P_j.\mathbf{v}_{l+1}^i = w_l P_j.\mathbf{v}_l^i + c_1 r_1 (P_j.\mathbf{y}_l^i - P_j.\mathbf{x}_l^i) + c_2 r_2 (P_j.\hat{\mathbf{y}}_l - P_j.\mathbf{x}_l^i), \tag{3}$$

where w_l is the inertia coefficient, which decrease linearly to maintain a balance between exploration and exploitation, c_1 and c_2 are acceleration coefficients, and r_1 and r_2 are random values uniformly and independently generated at every generations within $[0,1]$.

2.2 Nash Equilibrium-Based Predictive Control

The Nash equilibrium-based predictive control is based on the proposed CPSO as shown in Algorithm 1. There are two main loops, inner and outer, in order to perform Nash equilibrium-based predictive control. The outer loop is for the process of MPC, and the inner loop is for the optimization over a finite prediction interval through the CPSO at each control time step t. In the inner loop, for $P_j.\mathbf{x}_l^i$, its personal best $P_j.\mathbf{y}_l^i$ is checked, and then the global best $P_j.\hat{\mathbf{y}}_l$ is checked for update. When the subsystems reach a Nash equilibrium, the states of the subsystems are updated using $P_j.\hat{\mathbf{y}}_l$ in the time interval $[t, t+\delta t)$ where $\delta t < T$. These procedures are repeated until a termination condition is satisfied.

Algorithm 1. The pseudocode of the Nash equilibrium-based MPC incorporating CPSO algorithm. In this psudocode, the j-th subsystem is considered.

```
Create and initialize swarm;
repeat
    for each particle $i \in [1, ..., \texttt{swarmSize}]$ do
        Initialize particle;
    end for
    repeat
        for each particle $i \in [1, ..., \texttt{swarmSize}]$ do
            if $J_j(\hat{\mathbf{s}}_{\mathcal{N}_j}(S_j.\mathbf{x}_l^i)) < J_j(\hat{\mathbf{s}}_{\mathcal{N}_j}(S_j.\mathbf{y}_{l-1}^i))$ then
                $S_j.\mathbf{y}_l^i \leftarrow S_j.\mathbf{x}_l^i$
            else
                $S_j.\mathbf{y}_l^i \leftarrow S_j.\mathbf{y}_{l-1}^i$
            end if
            if $J_j(\hat{\mathbf{s}}_{\mathcal{N}_j}(S_j.\mathbf{y}_l^i)) < J_j(\hat{\mathbf{s}}_{\mathcal{N}_j}(S_j.\hat{\mathbf{y}}_{l-1}))$ then
                $S_j.\hat{\mathbf{y}}_l \leftarrow S_j.\mathbf{y}_l^i$
            else
                $S_j.\hat{\mathbf{y}}_l \leftarrow S_j.\hat{\mathbf{y}}_{l-1}$
            end if
        end for
        Update position and velocity of each particle in $P_j$ using (2) and (3);
        Send $P_j.\hat{\mathbf{y}}_l$ to neighboring robots;
        Receive $P_i.\hat{\mathbf{y}}_l$ from neighboring robots where $i \in \mathcal{N}_j$;
        $l \leftarrow l+1$;
    until Termination condition is satisfied;
    Perform state update of plant $j$ using $P_j.\hat{\mathbf{y}}_l$ in the time interval $[t, t+\delta t)$;
    $t \leftarrow t+\delta t$;
until Termination condition is satisfied;
```

3 Application to Multi-Robot Formation Control

3.1 Multi-Robot Formation Control Problem

Consider a group of M identical differential drive mobile robots. The motion state of j-th robot defined by $X_j = [x_j, y_j, \theta_j]^T$ can be described by

$$\begin{bmatrix} x_j(k+1) \\ y_j(k+1) \\ \theta_j(k+1) \end{bmatrix} = \begin{bmatrix} x_j(k) + \Delta T \cos\theta_j(k) v_j(k) \\ y_j(k) + \Delta T \sin\theta_j(k) v_j(k) \\ \theta_j(k) + \Delta T \omega_j(k) \end{bmatrix} \tag{4}$$

where X_j is described by its position (x_j, y_j) and orientation θ_j; v_j and ω_j are the linear and angular velocities of each robot, respectively, k is a discrete time step, and ΔT is a sample time.

A desired formation pattern $\mathscr{P}$ consisting of M vertices satisfies the relationships, $\sum_{i=1}^{M} p_{ix} = 0$ and $\sum_{i=1}^{M} p_{iy} = 0$, where p_{ix} and p_{iy} are defined by orthogonal coordinate such that the center of the formation pattern $\mathscr{P}$ is placed at the origin of the orthogonal coordinate.

Now, we define the multi-robot formation control problem in a similar way to [14] as follows:

Problem Definition 1. *Consider a group of nonholonomic mobile robots, given a reference path X_r and a desired formation patterns $\mathscr{P}$. For each robot j, using its own state $[x_j, y_j, \theta_j]^T$, its neighbors' state $[x_i, y_i, \theta_i]^T$ for $i \in \mathscr{N}_j$, the reference path X_r, and the desired formation pattern $\mathscr{P}$, find a predictive controller such that*

$$\lim_{t\to\infty} \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} = \begin{bmatrix} p_{jx} - p_{ix} \\ p_{jy} - p_{iy} \end{bmatrix}, 1 \le j \ne i \le M, \tag{5}$$

$$\lim_{t\to\infty} \begin{bmatrix} \sum_{j=1}^{M}(x_r - \frac{x_j}{M}) \\ \sum_{j=1}^{M}(y_r - \frac{y_j}{M}) \\ \sum_{j=1}^{M}(\theta_r - \frac{\theta_j}{M}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{6}$$

Equations (5) and (6) mean that the formation and tracking error converge to zero as $t \to \infty$, i.e., the center of the group of robots tracks the reference path while maintaining the desired formation pattern $\mathscr{P}$.

3.2 State Description

The global best particle $P_j.\hat{\mathbf{y}}$ represents the predicted control input of subsystem j, $u_j(t) = [v_j(t), \omega_j(t)]^T$, over a prediction interval T as a sequence, i.e.,

$$P_j.\hat{\mathbf{y}} = [v_j(1), v_j(2), ..., v_j(N-1), \omega_j(1), \omega_j(2), ..., \omega_j(N-1)] \tag{7}$$

subject to $|v_j(k)| \le V_{max}$ and $|\omega_j(k)| \le \Omega_{max}$, for $j = 1, 2, ..., M$ and $k = 1, 2, ..., N-1$ where N is a number of prediction steps. Based on v_j and ω_j, the robot state can be produced from the current pose $X_j(1) = [x_j(1),\ y_j(1),\ \theta_j(1)]$ at update time t using (4) as $x_j(k+1) = x_j(1) + \sum_{m=1}^{k} \cos\theta_j(m) v_j(m) \Delta T$, $y_j(k+1) = y_j(1) + \sum_{m=1}^{k} \sin\theta_j(m) v_j(m) \Delta T$, and $\theta_j(k+1) = \theta_j(1) + \sum_{m=1}^{k} \omega(m) \Delta T$.

3.3 Objective Function

The control objective is to achieve two goals: tracking a reference path and maintaining a desired formation in a cooperative and distributed way. Considering system (4) with its neighbors $\mathscr{N}_j$, the objective function J_j, for robot j over a number of finite prediction interval step N can be expressed as follows:

$$J_j = \sum_{k=1}^{N} ||\tilde{X}_j(k)||_P^2 + \sum_{i \in \mathscr{N}_j} \sum_{k=1}^{N} ||\tilde{X}_j(k) - \tilde{X}_i(k)||_Q^2 \tag{8}$$

where $\tilde{X}_j = [\tilde{x}_j\ \tilde{y}_j\ \tilde{\theta}_j]^T$, $\tilde{x}_j = x_r + p_{xj} - x_j$, $\tilde{y}_j = y_r + p_{yj} - y_j$, and $\tilde{\theta}_j = \theta_r - \theta_j$. Also, $P = P^T > 0$, $Q = Q^T > 0$, and $||x||_P^2$ represents $x^T Px$.

Since the multi-robot formation control problem can be considered to be a nonseparable problem (i.e., there are coupling terms in the objective functions), it can be optimized by the proposed CPSO algorithm.

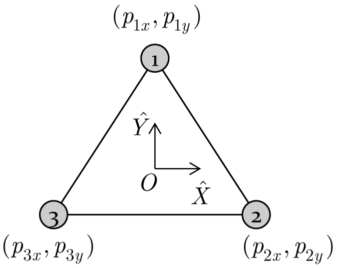

Fig. 1. Desired formation pattern of robots. In simulation, $(p_{x1}, p_{y1}) = (0, \frac{0.1}{\sqrt{3}})$, $(p_{x2}, p_{y2}) = (0.05, -\frac{0.05}{\sqrt{3}})$, and $(p_{x3}, p_{y3}) = (-0.05, -\frac{0.05}{\sqrt{3}})$.

3.4 Simulation Results

The number of prediction horizon steps is selected as $N = 10$, while the control time interval and prediction time interval are selected to be $\delta t = 0.1$s and $\Delta T = 0.1$s. Thus, the prediction horizon is 1s. Since the controller has high computational burden as the prediction interval increase, a long prediction length is not feasible. The weight matrices P and Q are set to be diagonal where $P = \texttt{diag}[0.1,\ 0.1,\ 0.1]$ and $Q = \texttt{diag}[0.1,\ 0.1,\ 0.1]$.

For the optimization process by the modified CPSO, each swarm has a population size of 100, and the maximum number of generations is 100. The inertia weight w_l starts with 0.9 and linearly decrease to 0.4. The search space is limited to real-valued variables within $[-V_{max}, V_{max}]$ and $[-\Omega_{max}, \Omega_{max}]$ for v_j and ω_j, respectively, where $V_{max} = 10$m/s and $\Omega_{max} = \pi$rad/s. The acceleration coefficients are $c_1 = 2.0$ and $c_2 = 2.0$.

Three mobile robots are used with the network graph described in Fig. 1. The reference path is a sinusoidal path given by $x_r(t) = 0.1t$, $y_r(t) = 0.8\sin(t/10)$, and $\theta_r(t) = \arctan 2(\dot{y}_r, \dot{x}_r)$. Initially, the robots are located at $X_1 = [0.0, 0.0, \pi/2]^T$, $X_2 = [0.0, -0.05, \pi/2]^T$, and $X_3 = [0.0, 0.05, \pi/2]^T$, respectively. The desired formation pattern $\mathscr{P}$ is an equilateral triangle formation in which the desired separation between the robots is 0.1m, i.e., $p_{x1} = 0$, $p_{y1} = 0.1/\sqrt{3}$, $p_{x2} = 0.05$, $p_{y2} = -0.05/\sqrt{3}$, $p_{x3} = -0.05$, and $p_{y3} = -0.05/\sqrt{3}$ as shown in Fig. 1. To validate the performance, the tracking error e_T and the formation error e_F are defined by $e_T = \sqrt{(x_r - x_c)^2 + (y_r - y_c)^2}$ and $e_F = \sqrt{||\tilde{X}_1 - \tilde{X}_2||^2 + ||\tilde{X}_2 - \tilde{X}_3||^2 + ||\tilde{X}_3 - \tilde{X}_1||^2}$, respectively.

The resulting trajectories of the group of the robots are shown in Fig. 2. It is shown that the three robots maintain a triangular formation while the center of the

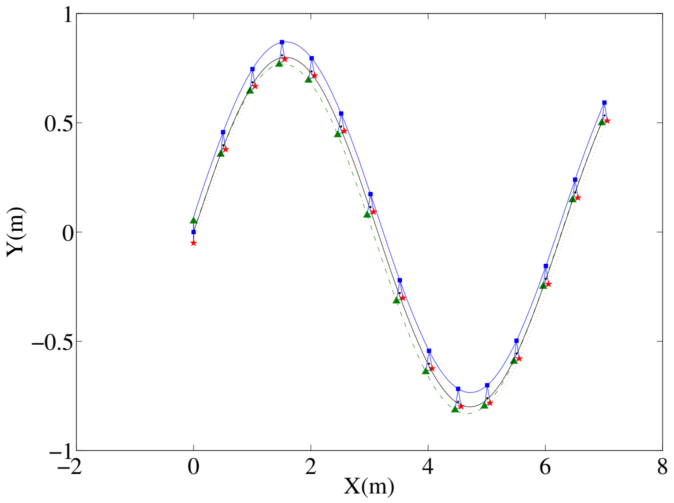

Fig. 2. Triangular formation tracking a sinusoidal reference path. The robot locations sampled at every 5s are indicated by squares, stars, and triangles for $j = 1, 2, 3$ and the black dots denote the center of the formation. The headings are tangential to the robot's path.

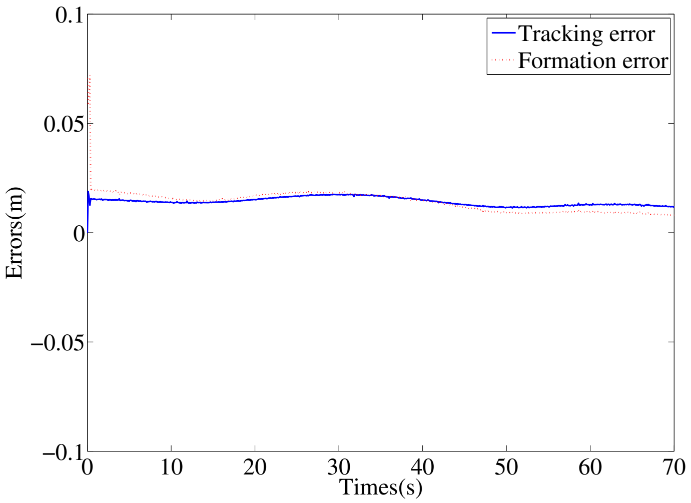

Fig. 3. The tracking error e_T and formation error e_F during tracking a sinusoidal reference path

formation tracks the given reference path using the transmitted information from neighboring robots. Fig. 3 shows the tracking error and formation error which are stable during maneuvering.

4 Conclusion and Future Works

In this paper, a distributed MPC scheme incorporating CPSO was proposed for multi-robot formation control problem. For the optimization process in MPC, a Nash equilibrium strategy was used to solve the optimization problem by exchanging particle information which has the best experience among neighboring subsystems. In the simulation, using the proposed MPC scheme, it was found that the robots moved to track a given reference path, while maintaining a desired formation pattern successfully.

Future works may include investigations of the stability, robustness, improvement of convergence speed, and comparative studies between the proposed method and conventional MPC schemes. The final goal of this research is the development of real-time cooperative MPC scheme according to the Nash equilibrium strategy.

Acknowledgements. This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the Human Resources Development Program for Convergence Robot Specialists support program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2011-C7000-1001-0001). It was also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number 2012-0003897).

References

1. Onnen, C., Babuska, R., Kaymak, U., Sousa, J.M., Verbruggen, H.B., Iserman, R.: Genetic algorithms for optimization in predictive control. Contr. Eng. Practice 5(10), 1363–1372 (1997)
2. Fravolini, M.L., Ficola, A., Cava, M.L.: Real-time evolutionary algorithms for constrained predictive control: Frontiers in Evolutionary Robotics, pp. 139–184. InTech (2008)
3. Martinez, M., Senent, J.S., Blasco, X.: Generalized predictive control using genetic algorithms. Eng. Appl. Artif. Intell. 11(3), 355–367 (1997)
4. Shin, S.C., Park, S.B.: GA based predictive control for nonlinear processes. Electron. Lett. 34(20), 1980–1981 (1998)
5. Newman, A.J., Martin, S.R., DeSena, J.T., Clarke, J.C., McDerment, J.W., Preissler, W.O., Peterson, C.K.: Receding horizon controller using particle swarm optimization for closed-loop ground target surveillance and tracking. In: Proc. SPIE 2009, vol. 7336 (2009)
6. Yousuf, M.S.: Nonlinear predictive control using particle swarm optimization: application to power systems. VDM Verlag Dr. Müller (2010)
7. Mercieca, J., Fabri, S.G.: Particle swarm optimization for nonlinear model predictive control. In: Proc. ADVCOMP, pp. 88–93 (2011)
8. Sedraoui, M., Abdelmalek, S.: Multivariable generalized predictive control using an improved particle swarm optimization algorithm. Informatica 35(3), 363–374 (2011)

9. van den Bergh, F., Engelbrecht, A.: A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. 8(3), 225–239 (2004)
10. Li, X., Yao, X.: Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In: Proc. IEEE CEC, pp. 1546–1553 (2009)
11. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. Evol. Comput. 16(2), 210–224 (2012)
12. Sefrioui, M., Periaux, J.: Nash genetic algorithms: examples and applications. In: Proc. IEEE CEC, pp. 509–516 (2000)
13. Gu, D.: A differential game approach to formation control. IEEE Trans. Control Syst. Technol. 16(1), 85–93 (2008)
14. Dong, W.: Flocking of multiple mobile robots based on backstepping. IEEE Trans. Syst. Man Cybern. Part B-Cybern. 41(2), 414–424 (2011)