2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Cooperative Coevolutionary Approach to Multi-Robot Formation Control

Seung-Mok Lee and Hyun Myung

Abstract— **This paper proposes a cooperative coevolutionary approach to multi-robot formation control. To deal with the formation control problem, the concept of a cooperative coevolution (CC) framework is incorporated with model predictive control (MPC) such that candidates of all robots coevolve toward a Nash equilibrium in a distributed way. Using the Nash-equilibrium strategy, the robots can quickly move to a desired formation from their initial locations. The stability is guaranteed via a novel repair algorithm that enforces each candidate to satisfy a derived condition for asymptotic stability. The cooperative coevolutionary particle swarm optimization (CCPSO) is adopted and modified to fit into the formation control problem. Simulations are performed on a group of nonholonomic mobile robots to demonstrate the effectiveness of the CC-based MPC. Also, the proposed MPC shows a better performance compared to sequential quadratic programming (SQP)-based MPC.**

I. Introduction

MODEL predictive control (MPC) has been successfully applied to control complex systems in industry as one of the most popular optimal control techniques [1]. The control technique is derived on the basis of the prediction of the future behavior estimated by the explicit model of the system. However, application to nonlinear system is not easy because the nonlinear optimization process should be completed within a limited time.

In order to solve the optimization problem in MPC, there have been many conventional techniques such as standard gradient search [2]-[4], quadratic programming [5], and mixed-integer linear programming [6]. However, the essential drawback of the most existing techniques is the computational complexity, which increases the computation time as the dimension of the system increases.

Recently, some researchers have studied the possibility of applying evolutionary algorithms (EAs) to solve the optimization problem in MPC. Onnen et al. [7] applied a genetic algorithm (GA) in order to optimize a control sequence in predictive control. They showed the effectiveness of the GA compared to the branch-and-bound discrete search algorithm. Similar algorithms that applied GA to MPC were presented in [8]-[10]. More recently, the modified particle swarm optimization (PSO) algorithms have been combined with MPC as presented in [11]-[14] because of the fact that PSO algorithms provide quick results even with multiple objectives and constraints. However, there is a lack of research on EAs applied to distributed MPC with its stability analysis.

In distributed MPC of multi-robot systems, one of the key issues is to find conditions guaranteeing stability while reducing the computational burden of optimization processes. To guarantee the stability in the conventional distributed MPC, it is assumed that each subsystem does not deviate too far from its previous computed state trajectory, referred to as the state *compatibility constraint* [15]-[19]. A drawback of this approach is that the system responses can be slow. A sufficiently short update period is used to relax the compatibility constraint, but the closed-loop control performance tends to depend on the update period.

This paper proposes a cooperative coevolution (CC)-based MPC framework to stabilize multi-robot formation. The proposed CC-based MPC approach guarantees the asymptotic stability regardless of the optimality of the solution that cooperative coevolutionary algorithm generates with a small number of individuals and within a limited time while conventional EA-based MPC approaches cannot guarantee the stability. We find a terminal state constraint which can guarantee the stability, and a repair algorithm is proposed such that all candidate solutions satisfy the terminal state constraint. The repair algorithm can handle the terminal state constraint in a short time. In this paper, we adopt cooperative coevolutionary particle swarm optimization (CCPSO) [20]-[22] among CC-based EAs, and the CCPSO is modified to fit on solving the distributed optimization problem.

To demonstrate the effectiveness of the proposed CCPSO-based MPC, the control performance and stability are tested through simulations. Also, the results of the proposed MPC are compared to sequential quadratic programming (SQP)-based MPC, which is one of the popular algorithms that are suitable for solving nonlinear optimization problem.

The rest of this paper is organized as follows. In Section II, the formation control problem is defined. Section III proposes a novel CC framework-based MPC, and then Section IV presents simulation results for the formation control problem. Finally, conclusion is presented in Section V.

II. Formation Control Problem

A. Mathematical Formulation of Formation Control Problem

The formation control problem to be investigated in this paper can be stated as follows: Consider a group of M

Seung-Mok Lee and Hyun Myung are with the Urban Robotics Laboratory, KAIST (Korea Advanced Institute of Science and Technology), Daejeon, 305-701, Republic of Korea (email: {seungmok, hmyung}@kaist.ac.kr).

This research was supported by the MOTIE (The Ministry of Trade, Industry and Energy), Korea, under the Human Resources Development Program for Convergence Robot Specialists support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2013-H1502-13-1001). This research was also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (grant number NRF-2013R1A1A1A05011746).

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

$$\dot{x}_j = v_j \cos\theta_j, \quad \dot{y}_j = v_j \sin\theta_j, \quad \dot{\theta}_j = \omega_j, \tag{1}$$

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.

For each robot j, using its own state $[x_j, y_j, \theta_j]^T$ and its neighboring states $[x_i, y_i, \theta_i]^T$, given a reference trajectory X_r and a desired formation pattern $\mathcal{P}$, find a controller such that a group of robots maintain the desired formation pattern $\mathcal{P}$ while the center of the formation tracks the reference trajectory X_r.

A desired formation pattern defined by $\mathcal{P} = \{[p_{1x}, p_{1y}], [p_{2x}, p_{2y}], ..., [p_{Mx}, p_{My}]\}$, consisting of M nodes, satisfies that $\sum_{i=1}^{M} p_{ix} = 0$ and $\sum_{i=1}^{M} p_{iy} = 0$, where p_{ix} and p_{iy} are defined by an orthogonal coordinate such that the center of the formation pattern is placed at the origin of the orthogonal coordinate. A reference trajectory $X_r = [x_r, \ y_r, \ \theta_r]^T$ is assumed to be continuously differentiable at any time, and θ_r is tangential to the reference trajectory defined by $\theta_r = \arctan 2(\dot{y_r}, \dot{x_r})$.

In order to solve the formation control problem in a distributed way, let us define the formation and tracking error of the robot j as

$$e_j = \begin{bmatrix} e_{jx} \\ e_{jy} \\ e_{j\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta_j & \sin\theta_j & 0 \\ -\sin\theta_j & \cos\theta_j & 0 \\ 0 & 0 & 1 \end{bmatrix} z_{je}, \tag{2}$$

where $z_{je} = \sum_{i\in\mathcal{N}_j}\{(P_j - P_i) - (X_j - X_i)\} + \mu_j(\tilde{X}_j + P_j)$, in which the former summation part is the formation error and the latter part is the tracking error. Also, $\tilde{X}_j = [\tilde{x}_j, \tilde{y}_j, \tilde{\theta}_j]^T = X_r - X_j$, $P_j = [p_{jx}, p_{jy}, 0]^T$, and $\mu_j = 1$ if the reference X_r is available to robot j, and $\mu_j = 0$ if X_r is not available to robot j. The error e_j is obtained by multiplying a rotation matrix in a robot fixed frame with z_{je}.

By differentiating e_j with respect to time, and then substituting (1) into the resulting equation, the error state equation can be obtained as follows:

$$\begin{aligned} \dot{e}_{jx} &= \omega_j e_{jy} + \sum_{i\in\mathcal{N}_j}(v_i\cos\theta_{ij} - v_j) + \mu_j(v_r\cos\tilde{\theta}_j - v_j), \\ \dot{e}_{jy} &= -\omega_j e_{jx} + \mu_j v_r \sin\tilde{\theta}_j + \sum_{i\in\mathcal{N}_j} v_i \sin\theta_{ij}, \\ \dot{e}_{j\theta} &= \sum_{i\in\mathcal{N}_j}(\omega_i - \omega_j) + \mu_j(\omega_r - \omega_j), \end{aligned} \tag{3}$$

where $\theta_{ij} = \theta_i - \theta_j$; v_r and ω_r are the desired linear and angular velocities of a group of robots, which can be derived by differentiating X_r. Equation (3) can be generally rewritten as a nonlinear nominal system as follows:

$$\dot{e}_j(t) = f\left(e_j(t), u_j(t), u_{\mathcal{N}_j}(t)\right) \tag{4}$$

where $u_{\mathcal{N}_j}(t) = (..., u_i, ...)$, $i \in \mathcal{N}_j$, denotes the concatenated vector of the control inputs of the neighbors of robot j, $\mathcal{N}_j$ denotes the set of neighbors of robot j, and $u_j = [v_j, \omega_j]^T$.

The cost function to be minimized for each robot j is designed as follows:

$$\begin{aligned} &J_j(t, e_j(t), u_j(t)) \\ &\quad = g(e_j(t+T)) + \int_t^{t+T} L(\tau, e_j(\tau), u_j(\tau))d\tau \end{aligned} \tag{5}$$

where $g(e_j(t+T)) = \frac{1}{2}e_j(t+T)^T e_j(t+T)$ is a terminal state penalty function, $L(t, e_j(t), u_j(t)) = e_j(t)^T Q e_j(t) + u_j(t)^T R u_j(t)$ is a running cost function, $Q > 0$ and $R \geq 0$ are symmetric weight matrices, and T is a finite prediction horizon.

At time t, the open-loop optimization problem in a distributed MPC framework can be formulated as a constrained nonlinear optimization problem

$$\min_{u_j} J_j\left(t, e_j(t), u_j(t)\right) \tag{6}$$

subject to

$$\begin{aligned} \dot{e}_j(\tau) &= f(e_j(\tau), u_j(\tau), u_{\mathcal{N}_j}(\tau)), \\ 0 \leq v_j(\tau) &\leq V_{\max}, \\ |\omega_j(\tau)| &\leq \Omega_{\max}, \\ e_j(t+T) &\in \Omega, \end{aligned}$$

where $\tau \in [t, t+T]$, Ω is the terminal state region, and $V_{\max}$ and $\Omega_{\max}$ are the maximum control signals.

B. Nash Equilibrium Strategy-based MPC Framework

The formation control problem can be solved in a MPC framework where a Nash equilibrium strategy is used as the formation control strategy. Brief explanations of the distributed MPC and Nash equilibrium strategy are given below.

The MPC [23]-[25] can be characterized by the following features: an explicit model of the system, a finite prediction horizon T, and an optimization process of the system behavior. The explicit model of the system is used to predict the behavior of a plant over a finite prediction horizon starting from the current time t.

The general procedure of the MPC scheme is as follows: 1) Compute the internal model state error $e(t)$ at the current time t; 2) Optimize a control input sequence over the prediction horizon T; 3) Apply the optimized control input $u^*(t)$ to the plant in a time interval $[t, t+\delta t)$, where $\delta t < T$; and 4) Repeat the procedure from step 1) to step 3) at the next time instant, $t \leftarrow t + \delta t$, until a certain termination condition is satisfied.

Our objective is to control a system consisting of multiple robots toward an equilibrium state in a cooperative way using the MPC scheme. In this paper, a Nash equilibrium strategy is applied to the system in order to optimize the control input over a finite prediction horizon. A Nash equilibrium strategy is a collection of strategies of all robots, and each strategy is the best response regarding the others' strategies. When the system reaches a Nash equilibrium, no robot can

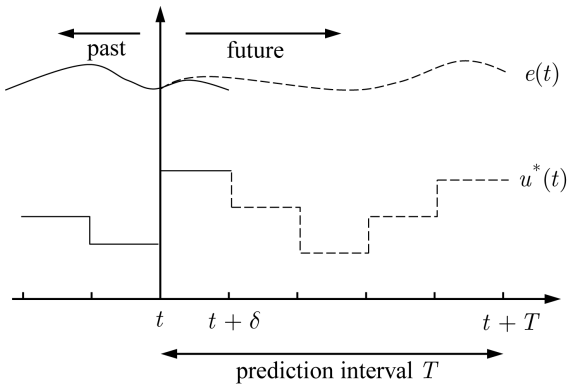


Fig. 1. General model predictive control (MPC) scheme.

further improve its 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 following condition:

$$J_i(u_1^*, u_2^*, ..., u_{i-1}^*, u_i^*, u_{i+1}^*, ..., u_M^*) \leq J_i(u_1^*, u_2^*, ..., u_{i-1}^*, u_i, u_{i+1}^*, ..., u_M^*) \quad (7)$$

for $i = 1, ..., M$, where J_i is the cost function and u_i is an arbitrary control input of the i-th robot, and M is the number of robots. Using the Nash strategy, each robot cooperates with the others in order to move toward the Nash equilibrium state.

Generally, it has been found that finding a Nash equilibrium strategy guaranteeing stability of the system is very difficult when the system is nonlinear and the state vectors are coupled in a cost function [26]. In [27], a receding horizon Nash controller was developed for multi-agent systems, but it has been limited to linear systems.

To cope with this problem, the concept of a cooperative coevolution (CC) framework can be adopted such that swarms of all robots coevolve toward a Nash equilibrium in a distributed way. This paper applies CCPSO to the Nash equilibrium strategy-based MPC framework. Since the CCPSO coevolves the multiple particle swarms by exchanging the global best particles with neighboring particle swarms, Nash equilibrium strategies can be found. The stability also can be simply guaranteed by a repair algorithm that enforces each particle to satisfy a derived terminal state condition using a small number of particles. Thus, the CCPSO can be applied to the formation control problem in real-time while guaranteeing stability. Therefore, we propose to use the CCPSO as the optimizer in the MPC framework for the multi-robot formation control problem.

III. MPC BASED ON COOPERATIVE COEVOLUTION FRAMEWORK

A. CCPSO-based MPC

For each robot, a cost function defined as a coupled form by the state variables and control inputs of the neighboring robots is given, and a particle swarm is assigned to each robot in order to optimize the cost function value.

Algorithm 1 The pseudocode of the proposed CCPSO algorithm. In this pseudocode, the j-th robot is considered.

```
Create and initialize a swarm;
for each particle i ∈ [1, ..., swarmSize] do
  Initialize particle;
end for
repeat
  for each particle i ∈ [1, ..., swarmSize] do
    if J_j(S_j.x_l^i) < J_j(S_j.y_{l-1}^i) then
      S_j.y_l^i ← S_j.x_l^i
    else
      S_j.y_l^i ← S_j.y_{l-1}^i
    end if
    if J_j(S_j.y_l^i) < J_j(S_j.ŷ_{l-1}) then
      S_j.ŷ_l ← S_j.y_l^i
    else
      S_j.ŷ_l ← S_j.ŷ_{l-1}
    end if
  end for
  Update position and velocity of each particle in S_j using (9);
  Send S_j.ŷ_l to neighboring robots;
  Receive S_i.ŷ_l from neighboring robots where i ∈ N_j;
  l ← l + 1;
until Termination condition is satisfied;
```

Let $S_j.\mathbf{x}_l^i$ be the current position of the i-th particle of the j-th swarm at generation l, $S_j.\mathbf{y}_l^i$ the personal best of the i-th particle of the j-th swarm, and $S_j.\hat{\mathbf{y}}_l$ the global best particle of the j-th swarm. Each particle $S_j.\mathbf{x}_l^i$ represents the predicted control input sequence of robot j at update time step t_k, $u_j(\tau; t_k) = [v_j(\tau; t_k), \omega_j(\tau; t_k)]^T$, over a prediction horizon T as a sequence, i.e.,

$$S_j.\mathbf{x}_l^i = [v_j(t_k; t_k), v_j(t_{k+1}; t_k), ..., v_j(t_{k+N-1}; t_k), \omega_j(t_k; t_k), \omega_j(t_{k+1}; t_k), ..., \omega_j(t_{k+N-1}; t_k)] \quad (8)$$

subject to $0 \leq v_j(t_{k+m}; t_k) \leq V_{\max}$ and $|\omega_j(t_{k+m}; t_k)| \leq \Omega_{\max}$ for $m = 0, 1, 2, ..., N-1$, where $N = \frac{T}{\delta t}$ is the number of prediction steps. The update time step is $t_k = t_0 + \delta t \cdot k$, where $k = 0, 1, 2, \cdots$.

The overall process of the proposed MPC consists of estimating robots' current states, predicting neighboring robots' future states using received global best particles, optimization process via PSO, and application of control input. The procedure is similar to that of the conventional PSO-based approach except for the method of exchanging global best particles. While the conventional PSO exchanges global best particles with neighbors only at once for one optimization process, the CCPSO-based approach exchanges global best particles at each generation for one optimization process. The exchanging of global best particles in the CCPSO algorithm is a powerful strategy to obtain optimal control input sequence that does not depend on the state computed

at the previous time step and update period. Using the exchanging strategy, robots can find optimal control input sequences according to a Nash equilibrium strategy.

Algorithm 1 shows the pseudocode of the proposed CCPSO algorithm. There is one loop in order to optimize the control input according to a Nash equilibrium strategy. At generation l, the j-th robot evaluates the cost function value of $S_j.\mathbf{x}_l^i$ for all i using the global best particles $S_i.\hat{\mathbf{y}}_{l-1}$, where $i \in \mathcal{N}_j$. After evaluating the cost of $S_j.\mathbf{x}_l^i$, its personal best $S_j.\mathbf{y}_l^i$ is checked, and then the global best $S_j.\hat{\mathbf{y}}_l$ is checked for update. After one generation of the process, each robot transmits its global best particle to neighboring robots. At each generation, the process of evaluating cost, updating personal best and global best particles, and updating the velocity and position of each particle is repeated based on the updated global best particles of neighboring robots. The update rule for the i-th particle in the j-th swarm is given by

$$S_j.\mathbf{x}_{l+1}^i = S_j.\mathbf{x}_l^i + S_j.\mathbf{v}_{l+1}^i, \tag{9a}$$

$$S_j.\mathbf{v}_{l+1}^i = w_l S_j.\mathbf{v}_l^i + c_1 R_1 (S_j.\mathbf{y}_l^i - S_j.\mathbf{x}_l^i) + c_2 R_2 (S_j.\hat{\mathbf{y}}_l - S_j.\mathbf{x}_l^i). \tag{9b}$$

This process is performed through all robots simultaneously. When all the robots cannot further improve their cost evaluation, the robots are considered to be in a Nash equilibrium. When the robots reach a Nash equilibrium, the states of the robots are updated using $S_j.\hat{\mathbf{y}}_l$ in the time interval $[t_k, t_k + \delta t)$.

B. Repair Algorithm for Stability

In order to assure the closed-loop stability, a terminal state region is derived with its corresponding terminal state controller. As stated in [24], [25], it has been proved that the MPC algorithm is asymptotically stable if the following condition is satisfied

$$\dot{g}(e_j(t)) + L(t, e_j(t), u_j(t)) \leq 0. \tag{10}$$

Considering the terminal state penalty function selected as $g(e_j(t+T)) = \frac{1}{2} e_j(t+T)^T e_j(t+T)$, the stability condition (10) can be rewritten as

$$\begin{aligned} &\dot{g}(e_j(t)) + L(t, e_j(t), u_{Tj}(t)) \\ &= -(1-q_x)e_{jx}^2 - (1-q_\theta)e_{j\theta}^2 + \mu_j v_r \sin\tilde{\theta}_j \left(1 - \frac{e_{j\theta}}{\tilde{\theta}_j}\right) e_{jy} \\ &\quad + \sum_{i \in \mathcal{N}_j} v_i \sin\theta_{ij} \left(1 - \frac{e_{j\theta}}{\theta_{ij}}\right) e_{jy} + q_y e_{jy}^2. \end{aligned} \tag{11}$$

If $\|e_{jy}(t)\| \to 0$, (11) converges to $-(1-q_{jx})e_{jx}^2 - (1-q_{j\theta})e_{j\theta}^2$. Thus the terminal state region Ω can be derived as

$$\Omega = \{q_x \in \mathbb{R}, q_\theta \in \mathbb{R}, e_{jy}(t) | q_x < 1, q_\theta < 1, \|e_{jy}(t)\| = 0\} \tag{12}$$

where N_j is the number of robots belonging to the set of neighbors $\mathcal{N}_j$.

To guarantee the stability, the candidate solutions have to satisfy the terminal state constraint. The terminal state constraint can be handled by constraining the terminal state variable of each particle $S_j.\mathbf{x}_i^l$ since the constraint is simply derived as an equality condition at terminal state. As the terminal state denotes the state at time $t = t_k + T$ predicted at t_k, the terminal state constraint $e_{jy}(t) = 0$ can be rewritten as $e_{jy}(t_k + T; t_k) = 0$.

Now, we introduce a new method that repairs all candidate solutions to satisfy $e_{jy}(t_k + T; t_k) = 0$. For simplicity, let us denote $X_j(t_k + \delta t \cdot m; t_k)$ as $X_{j,m} = [x_{j,m}, y_{j,m}, \theta_{j,m}]^T$ for $m = 1, ..., N$, and the time index t_k will be omitted for brevity of presentation. For $e_{jy}(t_k + T; t_k) = 0$, the desired heading angle at terminal state θ_Ω can be derived as follows:

$$\begin{aligned} e_{jy}(t_k + T; t_k) = 0 &\Leftrightarrow -\sin\theta_\Omega \Delta x_{j,N} + \cos\theta_\Omega \Delta y_{j,N} = 0 \\ &\Leftrightarrow \theta_\Omega = \arctan 2(\Delta y_{j,N}, \Delta x_{j,N}) \end{aligned} \tag{13}$$

where $\Delta x_{j,N} = x_{jd} - x_{j,N}$, $\Delta y_{j,N} = y_{jd} - y_{j,N}$, and $X_{jd} = [x_{jd}, y_{jd}]^T$ denotes the desired position of robot j at time $t_k + T$, which is defined as

$$x_{jd} = \frac{1}{N_j + \mu_j} \left(\sum_{i \in \mathcal{N}_j} (x_{i,N} - p_{ix}) + \mu \tilde{x}_{j,N} \right) + p_{jx}, \tag{14a}$$

$$y_{jd} = \frac{1}{N_j + \mu_j} \left(\sum_{i \in \mathcal{N}_j} (y_{i,N} - p_{iy}) + \mu \tilde{y}_{j,N} \right) + p_{jy}, \tag{14b}$$

where $\tilde{x}_{j,N} = x_r - x_{j,N}$ and $\tilde{y}_{j,N} = y_r - y_{j,N}$. To satisfy $e_{jy}(t_k + T; t_k) = 0$, the heading angle at terminal state $\theta_{j,N}$ should be equal to θ_Ω. As the equality $\theta_{j,N} = \theta_\Omega$ would require time consuming optimization process, a geometric-based approach is used for handling the equality constraint within a short time. The geometric-based algorithm repairs $\omega_{j,m-1}$ in a backward manner from $m = N - 1$ step until the condition $\theta_{j,N} = \theta_\Omega$ is met. Fig. 2 shows a repairing process of $\omega_{j,m-1}$ when the angular control inputs from $\omega_{j,m}$ to $\omega_{j,N-1}$ are saturated by $\pm\Omega_{\max}$. When the condition that $\theta_{j,N} = \theta_\Omega$ is met, $x_{j,N}$ and $y_{j,N}$ can be calculated as

$$x_{j,N} = x_{j,m} + D_1 \cos\alpha, \tag{15a}$$

$$y_{j,N} = y_{j,m} + D_1 \sin\alpha, \tag{15b}$$

where D_1 denotes the distance between $X_{j,m}$ and $X_{j,N}$. The angle α can be obtained by geometric relationship of robot's state trajectory. Using the relationship $\theta_{j,N} = \theta_{j,m} + \sum_{n=m}^{N-1} \omega_{j,n} \delta t$, $\theta_{j,m}$ that satisfies $\theta_{j,N} = \theta_\Omega$ can be obtained as follows:

$$\theta_{j,m} = \theta_\Omega - \sum_{n=m}^{N-1} \omega_{j,n} \delta t. \tag{16}$$

Based on (16) the angular velocity is repaired as

$$\omega_{j,m-1} = \begin{cases} \mathtt{sign}(\omega_{jd}) \cdot \Omega_{\max}, & \text{if } |\omega_{jd}| \geq \Omega_{\max}, \\ \omega_{j,m-1} = \omega_{jd}, & \text{otherwise,} \end{cases}$$

where $\omega_{jd} = \frac{\theta_{j,m} - \theta_{j,,m-1}}{\delta t}$. If $|\omega_{jd}| \geq \Omega_{\max}$, the repairing process is repeated in a backward manner $m \leftarrow m - 1$ until $|\omega_{jd}|$ is less than $\Omega_{\max}$.

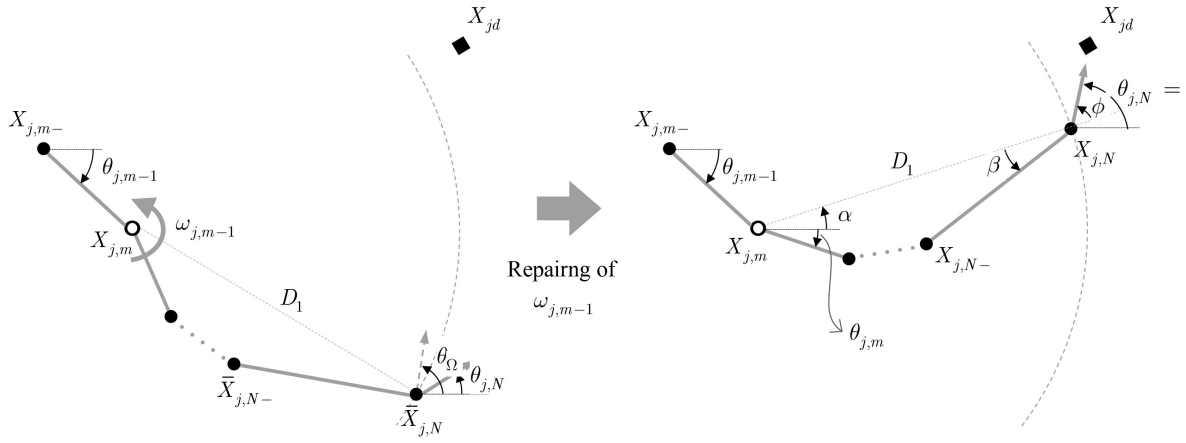


Fig. 2. Geometric representation for state trajectory of robot j. In order to satisfy the condition $\theta_{j,N} = \theta_\Omega$, $\omega_{j,m}$ is repaired in a backward manner from $m = N - 1$ step until the condition is met. This figure shows a repairing process of $\omega_{j,m-1}$ when the angular control inputs from $\omega_{j,m}$ to $\omega_{j,N-1}$ are saturated by $\pm\Omega_{\max}$.

This stability condition does not require the optimal control input. Any feasible control input sequence satisfying the constraint of $e_{jy}(t_k + T; t_k) = 0$ would lead to stability. It means that even though the CCPSO cannot reach an equilibrium state within a limited time, the stability can be achieved by applying the fittest control input sequence found so far.

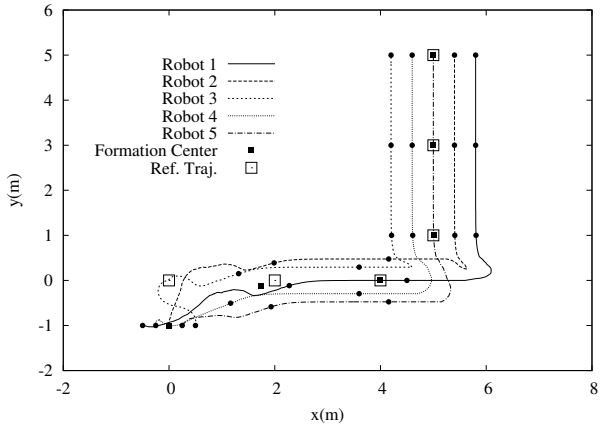
IV. Simulation Results

In this section, an example is given to illustrate the proposed method. A group of five robots is initially located at $X_1 = [-0.5, -3.0, 0.0]^T$, $X_2 = [0.0, -3.0, \pi/2]^T$, $X_3 = [0.5, -3.0, \pi/2]^T$, $X_4 = [-0.25, -3.0, \pi/2]^T$, and $X_5 = [0.25, -3.0, \pi/2]^T$ respectively. The reference trajectory used in the test is defined as follows: $x_r(t) = 0.1t$, $y_r(t) = 0$ for $0 \le t < 50$; $x_r(t) = 5$, $y_r(t) = 0.1(t - 50)$ for $50 \le t < 100$. We assume that the reference trajectory is available only to robots 2 and 5. Also, it is assumed that each robot communicates with two or three neighboring robots to avoid communication traffic increase. The control inputs for the five robots are computed at 5 Hz update rate (i.e., $\delta t = 0.2$ s) including the repair algorithm. The number of prediction steps is selected as $N = 10$, i.e., $T = N \cdot \delta t = 2$ s.

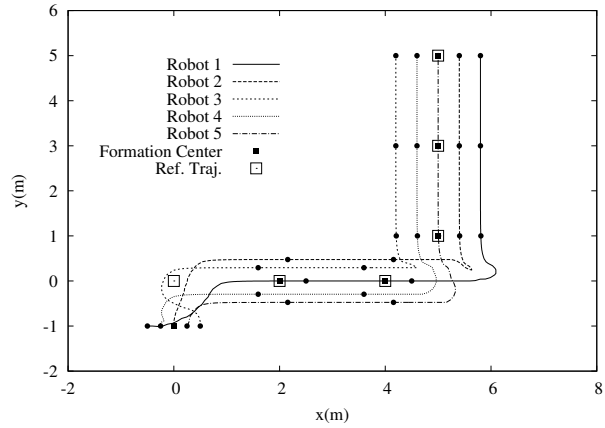
For the optimization processes, each robot has a particle swarm with a population size of 50, and the maximum number of generations $L_{\max}$ is limited to 100. The inertia weight w_l is determined as follows:

$$w_l = w_{\max} - \frac{w_{\max} - w_{\min}}{L_{\max}} \times l \qquad (17)$$

where $w_{\max} = 0.9$ and $w_{\min} = 0.4$ are the upper and lower bounds for w_l, respectively. The search space is limited to real values within $[0, V_{\max}]$ and $[-\Omega_{\max}, \Omega_{\max}]$ for v_j and ω_j, respectively, where $V_{\max} = 0.5$m/s and $\Omega_{\max} = \frac{\pi}{2}$rad/s. The acceleration coefficients are $c_1 = 2.0$ and $c_2 = 2.0$. The weight matrix Q is set to $Q = \mathtt{diag}[0.1,\ 0.1,\ 0.001]$.

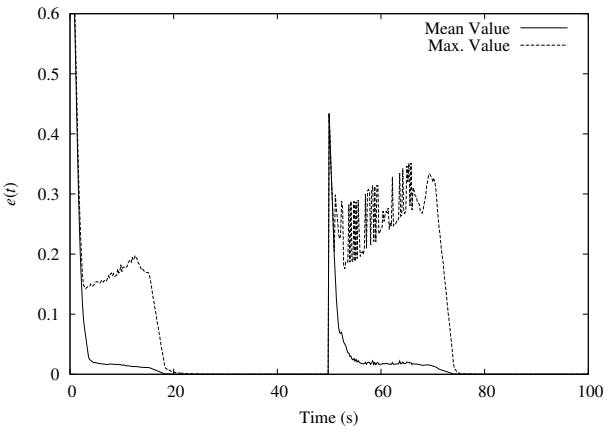


(a) Without repair process

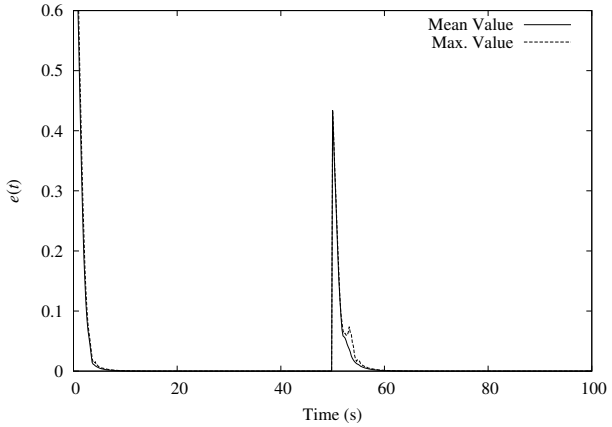


(b) With repair process (proposed)

Fig. 3. Trajectories and sampled positions of five robots. The black dots denote the positions of five robots, the light squares denote the reference trajectory, and the black squares denote the center of the formation. The positions are sampled at every 20 s.



(a) Without repair process



(b) With repair process (proposed)

Fig. 4. Mean and maximum values of $e(t)$ with $\delta t = 0.2$ s for 30 runs.

To test the switching formation, the following two different desired formation patterns are used:

$$\mathcal{P} = \begin{cases} \{[0.5, 0], [0.1545, 0.4755], [-0.4045, 0.2939], \\ \quad [-0.4045, -0.2939], [0.1545, -0.4755]\}, \\ \qquad\qquad \texttt{for } 0 \leq t < 50 \\ \{[0, 0], [0, 0.4], [0, 0.8], [0, -0.8], [0, -0.4]\}, \\ \qquad\qquad \texttt{for } 50 \leq t < 100. \end{cases}$$

To measure the performance of the algorithm, two performance indices are defined as follows:

$$E_T = \int_{t_0}^{t_f} \sqrt{\frac{1}{M} \sum_{j=1}^{M} e_j^T(t) Q e_j(t) dt} \qquad (18)$$

where $e(t)$ refers to the error generated at time t and E_T denotes the total amount of $e(t)$ in the running time from the starting time $t_0 = 0$ s through to the final time $t_f = 100$ s.

A. Control Performance and Stability

To investigate how the repair algorithm affect the control performance and stability, the results of the proposed MPC with the repair process are compared to the results without the repair process. The simulation is run 30 times for each case using the same simulation parameters. Fig. 3 shows the resulting trajectories computed by the CCPSO-based MPC without and with the repair process, respectively. Fig. 4 shows average and maximum values of the performance index defined in (18) for 30 runs. The results show that the repair algorithm is effective in stabilizing multi-robot formation. Without the repair process, the stability cannot be guaranteed when the robots converge to the desired formation from initial positions or the desired formation pattern is changed at 50 s as shown in Fig. 4(a). However, the errors converge to zero for 30 runs when the MPC accompanied by the repair algorithm as shown in Fig. 4(b). Therefore, the asymptotic stability can be guaranteed by the repair algorithm.

B. Comparison with SQP

To show the benefit of the proposed MPC, an MPC with SQP, which is one of the popular algorithms that are suitable for solving nonlinear optimization problems, is applied to the formation control problem and compared with the proposed approach when $\delta t = 0.2$ s. The results are shown in Fig. 5. As shown in the results, $e(t)$ of the proposed algorithm converges to zero faster than the SQP-based MPC. As the SQP methods provide local optimal solutions, the system responses can be slow and the state error may not converge to zero.

Compared with SQP, the advantages of the proposed approach can be summarized as follows: First, the implementation of the proposed algorithm is much easier and simpler than the classical optimization-based approaches. For example, SQP needs to compute the Hessian of the Lagrangian consisting of an objective function and constraints. In the formation control problem, the computation of the Hessian is quite complex due to the high nonlinearity. However, the proposed MPC is very easy and simple to implement without the computation of the Hessian.

Second, the proposed MPC with the repair algorithm can always provide the solution that satisfies the given constraints within a limited time when accompanied by the repair algorithm. The optimization problem for the formation control is formulated as a nonlinear constrained optimization problem with various equality and inequality constraints that should be handled simultaneously. The classical optimization-based approaches have difficulties due to the computational complexity. Also, due to their non-constant computation time for each time step, especially when the robots reconfigure their formation, the irregular delays between the time the control inputs are calculated and the time the control inputs are applied can degrade the performance of the system in practice.

Third, the proposed MPC finds the Nash-equilibrium strategy among multiple robots, which is the best control input of each robot for the formation control, whereas the most classical optimization methods such as SQP provide local optimal solutions. Using the Nash-equilibrium strategy, the robots can thus quickly move into a desired formation from their initial locations and switch their formation during maneuvers.

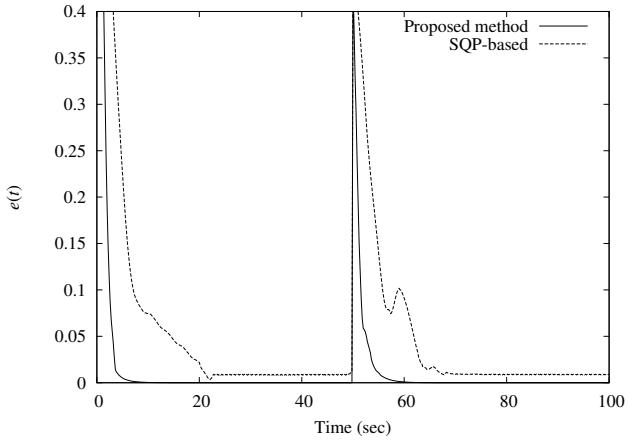


Fig. 5. Comparison with a SQP-based MPC. The result of the proposed method is from Fig. 4(b).

V. Conclusion

In this paper, a novel CCPSO-based MPC was proposed for multi-robot formation control. The stability was guaranteed by a novel repair algorithm that enforces all candidates to satisfy the terminal state constraint. For the optimization process in the MPC, a Nash equilibrium strategy was used to solve the optimization problem by exchanging information that has the best experience among neighboring robots. Thus, the robots can quickly move to a desired formation from their initial locations using the Nash-equilibrium strategy. Through the simulations, it was found that the robots could track a given reference path, while maintaining a desired formation pattern successfully.

References

[1] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 44-52, 2002.

[2] G. J. Sutton and R. R. Bitmead, "Computational implementation of NMPC to nonlinear submarine," *Nonlinear Model Predictive Control*, vol. 26, pp. 461-471, 2000.

[3] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry, "Model-predictive active steering and obstacle avoidance for autonomous vehicles," *Control Engineering Practice*, vol. 17, no. 7, pp. 741-750, Jul. 2009.

[4] J. Shin and H. J. Kim, "Nonlinear model predictive formation flight," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 5, pp. 1116-1125, Sep. 2009.

[5] J. Yan and R. R. Bitmead, "Incorporating state estimation into model predictive control and its application to network traffic control," *Automatica*, vol. 41, no. 4, pp. 595-604, Apr. 2005.

[6] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs," in *Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004, pp. 4286-4291.

[7] C. Onnen, R. Babuska, U. Kaymak, J. M. Sousa, H. B. Verbruggen, and R. Iserman, "Genetic algorithms for optimization in predictive control," *Control Engineering Practice*, vol. 5, no. 10, pp. 1363-1372, 1997.

[8] M. L. Fravolini, A. Ficola, and M. L. Cava, "Real-time evolutionary algorithms for constrained predictive control," *Frontiers in Evolutionary Robotics*, InTech, pp. 139-184, Apr. 2008.

[9] M. Martinez, J. S. Senent, and X. Blasco, "Generalized predictive control using genetic algorithms," *Engineering Applications of Artificial Intelligence*, vol. 11, no. 3, pp. 355-367, 1997.

[10] S. C. Shin and S. B. Park, "GA based predictive control for nonlinear processes," *Electronics Lettetters*, vol. 34, no. 20, pp. 1980-1981, 1998.

[11] A. J. Newman, S. R. Martin, J. T. DeSena, J. C. Clarke, J. W. McDerment, W. O. Preissler, and C. K. Peterson, "Receding horizon controller using particle swarm optimization for closed-loop ground target surveillance and tracking," in *Proceedings of SPIE 2009*, vol. 7336, 2009.

[12] H. B. Duan and S. Q. Liu, "Non-linear dual-mode receding horizon control for multiple unmanned air vehicles formation flight based on chaotic particle swarm optimization," *IET Control Theory & Applications*, vol. 4, no. 11, pp. 2565-2578, 2010.

[13] M. Sedraoui and S. Abdelmalek, "Multivariable generalized predictive control using an improved particle swarm optimization algorithm," *Informatica*, vol. 35, no. 3, pp. 363-374, 2011.

[14] S. M. Lee and H. Myung, "Cooperative particle swarm optimization-based predictive controller for multi-robot formation," *Advances in Intelligent Systems and Computing*, vol. 194, pp. 533-541, 2013.

[15] W. B. Dunbar and R. M. Murray, "Receding horizon control of multi-vehicle formations: a distributed implementation," in *Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004, pp. 1995-2002.

[16] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp. 549-558, 2006.

[17] W. B. Dunbar and D. S. Caveney, "Distributed receding horizon control of vehicle platoons: stability and string stability," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 620-633, Mar. 2012.

[18] T. Keviczky, F. Borrelli, and G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105-2115, Dec. 2006.

[19] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas, "Decentralized receding horizon control and coordination of autonomous vehicle formations," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 19-33, Jan. 2008.

[20] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, Jun. 2004.

[21] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 1546-1553.

[22] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210-224, Apr. 2012.

[23] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205-1217, 1998.

[24] F. A. C. C. Fontes, "A general framework to design stabilizing nonlinear model predictive controllers," *Systems & Control Letters*, vol. 42, pp. 127-143, 2001.

[25] D. Gu and H. Hu, "A stabilizing receding horizon regulator for nonholonomic mobile robots," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 1022-1028, Oct. 2005.

[26] M. Sefrioui, and J. Periaux, "Nash genetic algorithms: examples and applications," in *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, 2000, pp. 509-516.

[27] D. Gu, "A differential game approach to formation control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 85-93, Jan. 2008.