Particle Swarm Optimization-Based Distributed Control Scheme for Flocking Robots

Seung-Mok Lee and Hyun Myung

Dept. of Civil & Environmental Engg. and Robotics Program, KAIST
291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Korea
{seungmok,hmyung}@kaist.ac.kr

Abstract. This paper proposes a Nash equilibrium-based model predictive control (MPC) scheme incorporating a cooperative particle swarm optimization (CPSO) to deal with the control of flocking robots whose state vectors are coupled in a cost function. In conventional distributed MPC, the stability is assured by guaranteeing a bounded error between what a subsystem plans to do and what neighbors believe that the subsystem plans to do over a finite prediction horizon. This condition is referred to as compatibility constraint, and the closed-loop control performance largely depends on the responses computed at the previous time step. As an alternative of the compatibility constraint, the distributed CPSO is suggested in an MPC framework, which guarantees the stability without enforcing the compatibility constraint. A numerical simulation is performed on a group of nonholonomic mobile robots to demonstrate the effectiveness of the proposed MPC scheme incorporating CPSO.

Keywords: cooperative particle optimization (CPSO), model predictive control (MPC), Nash equilibrium, flocking.

1 Introduction

In distributed model predictive control (MPC) of multiple subsystems, one of the key issues is to find conditions guaranteeing stability while reducing the computational burden of optimization processes [1]-[6]. To guarantee the stability in the conventional distributed model predictive control (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 or it is assumed that the updating time is sufficiently short [1],[2]. 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.

CPSO algorithm is a variant of PSO, employing multiple swarms to optimize different variables of the solution in a cooperative coevolution framework. An early attempt to apply the CC framework to PSO was made by Bergh and Engelbrecht [7], resulting in two cooperative PSO algorithms, namely CPSO-S_K and CPSO-H_K. Recent studies by Li and Yao [8],[9] suggested cooperative coevolving PSO (CCPSO) and

J.-H. Kim et al. (Eds.): *Robot Intelligence Technol. & Appl. 2012*, AISC 208, pp. 517–524.
DOI: 10.1007/978-3-642-37374-9_50

CCPSO2, and their performance was validated on benchmark functions of up to 1,000 dimensions. In fact, the CPSO and its variants were originally developed to deal with high-dimensional optimization problems.

This paper proposes a modified version of the CPSO to find optimal strategies for formation control of flocking robots operated by distributed MPC scheme. It is assumed that each robot is assigned with its own optimization problem and communicates information only with neighboring robots. Thus, each robot has a particle swarm to optimize its cost function value, and the optimization problem is solved by the particle swarm.

The rest of this paper is organized as follows. In Section 2, formation control problem is defined in a distributed MPC framework. Section 3 proposes a novel CPSO-based distributed MPC scheme. Section 4 then presents a simulation result for multi-robot formation control problem. Finally, conclusion is presented in Section 5.

2 Problem Formulation

The formation control problem can be stated as follows: Consider a group of nonholonomic mobile robots. 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 path X_r and a desired formation pattern P, find a controller such that a group of robots maintain the desired formation pattern P while the center of the formation tracks the reference path X_r. The motion state of the j-th robot defined by $X_j=[x_j, y_j, \theta_j]^T$ can be described by

$$\begin{bmatrix} \dot{x}_j \\ \dot{y}_j \\ \dot{\theta}_j \end{bmatrix} = \begin{bmatrix} v_j \cos\theta_j \\ v_j \sin\theta_j \\ \omega_j \end{bmatrix} \tag{1}$$

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.

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

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

where

$$z_{je} = \sum \left\{ \left(P_j - P_i \right) - \left(X_j - X_i \right) \right\} + \mu_j \left(\tilde{X}_j + P_j \right)$$

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 μ_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}. P_j is a desired relative position of robot j to form a desired pattern of flocking.

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:

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

where θ_{ij}=θ_i−θ_{ij}; N_j denotes a set of neighbors of robot j; v_r and ω_r are the desired linear and angular velocities, which can be derived by differentiating X_r. The error state equation (3) can be generally rewritten as a nonlinear nominal system as follows:

$$\dot{e}_j(\tau) = f\left(e_j(\tau), u_j(\tau), u_{N_j}(\tau)\right) \tag{4}$$

where u_{Nj} (t)=(...,u_i,...), i∈N_j, denotes the concatenated vector of the control inputs of the neighbors of robot j, and u_j=[v_j, ω_j]^T.

The cost function to be minimized for each robot j in a distributed MPC framework is designed as follows:

$$J_j\left(t, e_j(t), u_j(t)\right) = g_j\left(e_j(t+T)\right) + \int_t^{t+T} L_j\left(\tau, e_j(\tau), u_j(\tau)\right)d\tau \tag{5}$$

where $g_j(e_j(t+T))$=$e_j(t+T)^T e_j(t+T)$ is a terminal state penalty function, $L(t,e_j(t), u_j(t))$= $e_j(t)^T Qe_j(t)$+$u_j(t)^T Ru_j(t)$ is a running cost function, and Q and R are positive definite symmetric weight matrices At time t, the open-loop optimization problem in a distributed MPC framework can be formulated as

$$\min_{u_j} J\left(t, e_j(t), u_j(t)\right)$$

subject to

$$\begin{gathered}
\dot{e}_j(\tau) = f\left(e_j, u_j(\tau), u_{Nj}(\tau)\right)\\
0 \le v_j(\tau) \le V_{\max}\\
\left\|\omega(\tau)\right\| \le \Omega_{\max}
\end{gathered}$$

where τ∈[t,t+T], and V_{max} and Ω_{max} are the maximum control inputs.

3 CPSO-Based MPC Framework for Flocking Robots

In this section, we propose a method that optimizes the control input sequence over a prediction horizon by using CPSO for formation control of flocking robots. A cost function for each robot is defined as a coupled form by the future state trajectories of the neighboring robots, and a particle swarm is assigned to each robot in order to minimize the cost function value.

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 $S_j.\hat{\mathbf{y}}_l^i$ 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 t_k, $u_j(\tau;t_k)=[v_j(\tau;t_k), \omega_j(\tau;t_k)]^T$ over a prediction horizon *T*.

The process of the proposed CPSO algorithm is shown in Fig. 1. Each robot *j* receives global best particles found by neighboring robots and predicts the future states of the neighbors. Based on the predicted states, the future control input sequence $u_j(\tau;t_k)$ for $\tau\in[t_k,t_k+T]$ is optimized by minimizing (5) via CPSO. 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 $S_i.\hat{\mathbf{y}}_{l-1}^i$ where $i\in N_j$. The concatenated vector $u_{Nj}(t)$ in (4) is constructed using the received best particles from neighbors of robot *j*. Then, the cost of $S_j.\mathbf{x}_l^i$ is evaluated by replacing $u_j(t)$ in (5) with $S_j.\mathbf{x}_l^i$. 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 $S_i.\hat{\mathbf{y}}_l^i$ 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 particle, and updating velocity and position of each particle is repeated based on the updated global best particles of neighboring robots. When the robots reach an equilibrium state in which all the robots cannot further minimize their cost, the states of the robots are updated using $S_j.\mathbf{y}_l^i$ in the time interval $[t, t+\delta t)$.

The update rule to determine the particles' position in the next generation can be described as follows:

$$\begin{aligned} S_j.\mathbf{x}_{l+1}^i &= S_j.\mathbf{x}_l^i + S_j.\mathbf{v}_{l+1}^i, \\ S_j.v_{l+1}^i &= w_l S_j.\mathbf{v}_l^i + c_1 r_1 (S_j.\mathbf{y}_l^i - S_j.\mathbf{x}_l^i) + c_2 r_2 (S_j.\hat{\mathbf{y}}_l - S_j.\mathbf{x}_l^i). \end{aligned} \tag{6}$$

A block diagram of the proposed CPSO-based MPC is shown in Fig. 2. After the process of estimating robots' current states, the future control input sequence $u_j(\tau;t_k)$ is minimized by the CPSO process. When the CPSO process is finished, the first part of the control input sequence is applied to its robot.

At each update time step t_k, particles of each robot should be initialized in order to re-search optimal control input sequence. When initializing the particles of the *j*-th robot, the global best particle found at the last update time step is chosen again as one of the candidate solutions for the next time step. The fact that the optimization process starts with the best particle found at the last update time step leads to improved convergence performance.

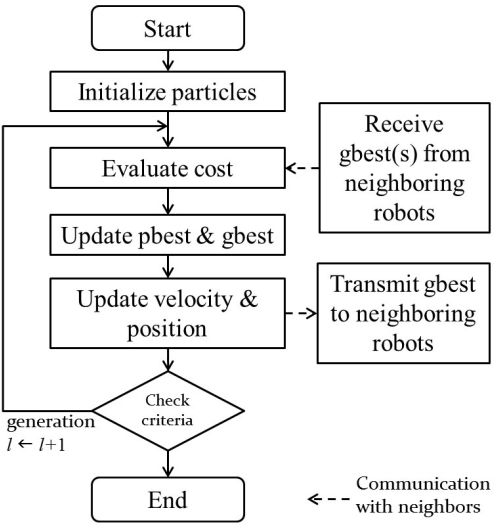

Fig. 1. Flowchart of the CPSO process

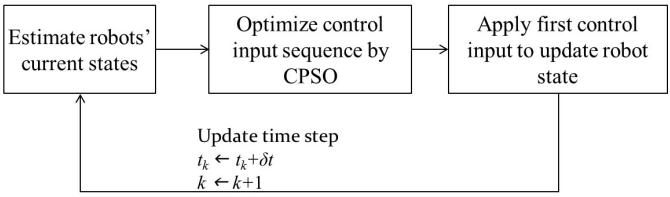

Fig. 2. Block diagram of the CPSO-based MPC Framework

4 Simulation Results

A numerical simulation is performed to validate the effectiveness of the proposed CPSO-based MPC scheme. For the optimization processes by the proposed CPSO, each robot has a particle swarm with a population size of 50, and the maximum number of generations is limited to 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}=0.5m/s and Ω_{max}=1.57rad/s. The acceleration coefficients are c_1=2.0 and c_2=2.0.

The number of prediction horizon steps is 10, while the prediction time interval are selected to be δt=0.1s. Thus, the prediction horizon is 1s. The weight matrices Q and R are set to be diagonal where Q=diag[0.1,0.1,0.01] and R= diag[0.1, 0.1].

Five mobile robots are used to test the algorithm. The reference path is a circle path given by $x_r(t)$=2cos(0.05t), $y_r(t)$=2sin(0.05t), and $\theta_r(t)$=atan2($\dot{y}_r$, $\dot{x}_r$). Initially, the robots are located at $X_1=[2.0, -1.0, 0.0]^T$, $X_2=[2.25, -1.0, 1.57]^T$, $X_3=[2.5, -1.0, 1.57]^T$, $X_4=[1.75, -1.0, 1.57]^T$, and $X_5=[1.5, -1.0, 1.57]^T$, respectively. The desired formation pattern P is a regular pentagon formation, i.e., p_{1x}=0.5, p_{1y}=0.0,

p_{2x}=0.1545, p_{2y}=0.4755, p_{3x}=−0.4045, p_{3y}=0.2939, p_{4x}=−0.4045, p_{4y}=−0.2939 p_{5x}=0.1545, and p_{5y}=−0.4755 as shown in Fig. 3. To measure the performance of the algorithm, an error function of t is defined:

$$E(t) = \sum_{j=1}^{M} e_j^T(t) Q e_j(t). \tag{7}$$

The resulting trajectories of the group of the robots are shown in Fig. 4. It is shown that the five robots maintain a regular pentagon formation while the center of the formation tracks the given reference path using the transmitted information from neighboring robots. Fig. 5 shows the stable total error which converges to zero during maneuvers.

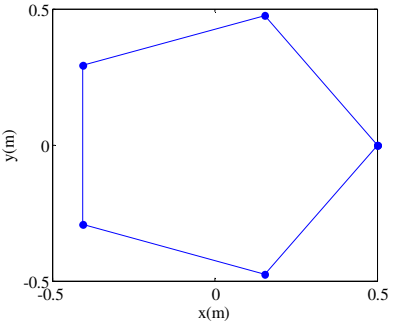

Fig. 3. Desired formation pattern

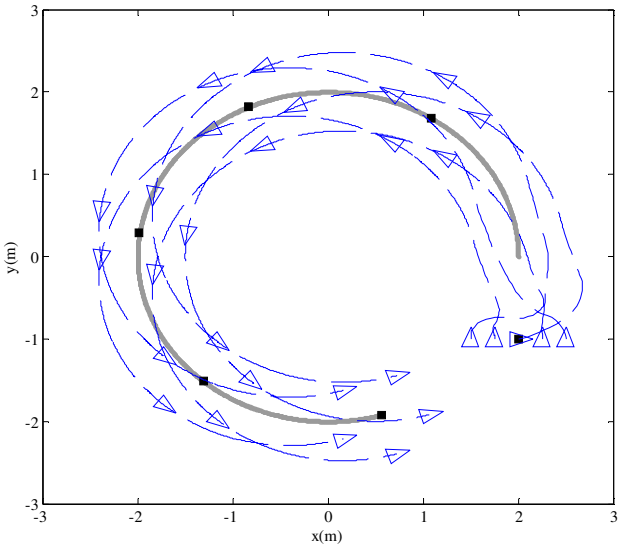

Fig. 4. Trajectories and sampled positions of five robots are indicated with their heading angles. The gray line denotes the reference path and the black squares denote the center of the formation. The positions are sampled at every 20s.

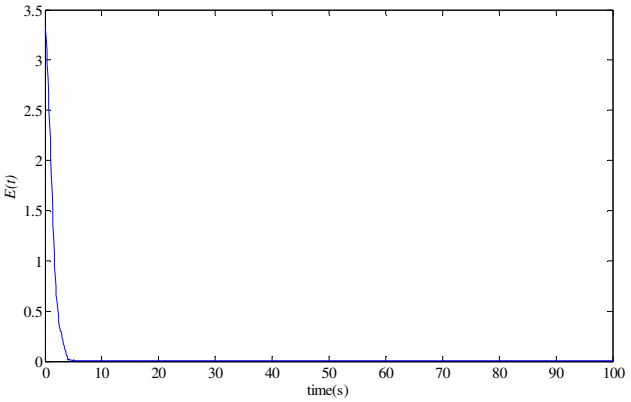

Fig. 5. Total error during maneuvers

5 Conclusion

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-2012-H1502-12-1002). 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). Additionally, it was supported by Korea Ministry of Land, Transport and Maritime Affairs(MLTM) as U-City Master and Doctor Course Grant Program.

References

1. Dunbar, W.B., Murray, R.M.: Distributed receding horizon control for multi-vehicle formation stabilization. Automatica 42(4), 549–558 (2006)
2. Dunbar, W.B., Caveney, D.S.: Distributed receding horizon control of vehicle platoons: stability and string stability. IEEE Trans. Autom. Control 57(3), 620–633 (2012)

3. Fontes, F.A.C.C.: A general framework to design stabilizing nonlinear model predictive controllers. Syst. Control Lett. 42, 127–143 (2001)
4. Keviczky, T., Borrelli, F., Balas, G.J.: Decentralized receding horizon control for large scale dynamically decoupled systems. Automatica 42(12), 2105–2115 (2006)
5. Gu, D., Hu, H.: A stabilizing receding horizon regulator for nonholonomic mobile robots. IEEE Trans. Robot. 21(5), 1022–1028 (2005)
6. Chen, J., Sun, D., Yang, J., Chen, H.: Leader-follower formation control of multiple non-holonomic mobile robots incorporating a receding-horizon scheme. Int. J. Robot. Res. 29(6), 727–747 (2010)
7. van den Bergh, F., Engelbrecht, A.: A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. 8(3), 225–239 (2004)
8. Li, X., Yao, X.: Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In: Proc. IEEE Congress on Evolutionary Computation (CEC), pp. 1546–1553 (2009)
9. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. Evol. Comput. 16(2), 210–224 (2012)