Trajectory Generation for Multiagent Point-to-Point Transitions via Distributed Model Predictive Control

Carlos E. Luis

***Abstract*— This paper introduces a novel algorithm for multi-agent collision-free trajectory generation, based on distributed model predictive control (DMPC). The prediction horizon used in DMPC allows the agents to predict and avoid future collisions, while driving the agents to their desired goals. DMPC is able to rapidly compute trajectories in reduced spaces with high density of agents, where other approaches either failed or required long runtimes. The effectiveness of the algorithm is validated through extensive simulation test cases, and compared with a previous algorithm based on sequential convex programming (SCP).**

I. Introduction

Generating collision-free trajectories when dealing with multiagent systems is a safety-critical task. In missions that require cooperation of multiple agents, such as surveillance [1], crop inspection [2] and warehouse management [3], it is often desirable to safely drive the agents from a starting location to a final location. Solving this problem, known as multiagent point-to-point transitions, is therefore an integral objective of any robust multiagent system.

There are two main variations of the multiagent point-to-point transition problem: the labelled or the unlabelled agent problems. In the former, each agent has a fixed final position that cannot be exchanged with other agents [4], [5]; in the latter, the algorithm is free to assign the goals to the agents, as to ease the complexity of the trajectories [6]. This paper will focus on the labelled agent version of the problem.

A natural way to formulate the problem is as an optimization problem. One of the first approaches proposed the use if Mixed Integer Linear Programming (MILP), modelling collision constraints as binary variables [4]. These methods, although viable, are computationally heavy and not suited for large teams.

More recently, techniques based on Sequential Convex Programming (SCP) [7] were conceived, reducing computational effort of mixed integer approaches. In [5], SCP is used to solve the optimal-fuel trajectories of quadrotor teams midflight. A decoupled version of that algorithm (dec-iSCP) is proposed in [8], which has shorter computation time at the cost of suboptimal solutions. However, dec-iSCP is a sequentially greedy approach that makes the optimization problem harder to solve as more vehicles are added.

Discrete approaches divide the space in a grid and use known discrete seach algorithms [9]. Other mixed approaches combine optimization techniques and enforced

The author is with the Dynamic Systems Lab at the University of Toronto Institute for Aerospace Studies (UTIAS), Canada. Email: carlos.luis@robotics.utias.utoronto.ca

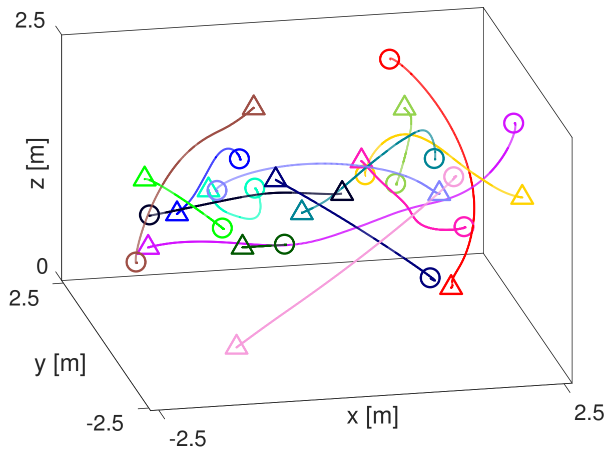

Fig. 1. A 15-vehicle point-to-point transition solved using our DMPC algorithm. The initial and final positions (denoted with triangles and circles, respectively) were chosen randomly within a convex volume of $5 \times 5 \times 2$ m. Triangles denote the initial positions. The agents were required to maintain a minimum distance of 0.75m of each other throughout the whole trajectory. Even in this highly cluttered dynamic environment, the proposed algorithm is capable of quickly finding a collision-free trajectory for all agents.

predefined behaviours to manage collisions [10]. Lyapunov barrier functions have also been used to compute multiagent collision-free trajectories [11].

Distributed optimization approaches can effectively include pair-wise distance nonlinear constraints [12]. Furthermore, the distributed nature reduces the computational effort with respect to centralized approaches. A new solution to the problem is proposed in this paper, based on distributed model predictive control (DMPC) [13]. Synchronous implementation of DMPC [14], [15], allows the agents to solve their optimization problem in parallel. Previous work on DMPC show its capabilities in multiagent tasks, such as formation control [16], [17], but not explicitly for point-to-point transitions as highlighted in Fig. (1).

The rest of the paper is organized as follows: Section II formulates the optimization problem solved by each agent, and Section III formalizes the DMPC algorithm for multiagent point-to-point transitions. Finally, Section IV presents simulation results of DMPC and compares its performance solving the problem with that of dec-iSCP.

II. Problem Statement

This section introduces the different components of the optimization problem to be solved. The formulation must accomplish the following criteria:

1) Drive all agents from their starting position to their desired final positions.
2) Respect a minimum pair-wise distance throughout the trajectory (collision constraint).
3) The position and acceleration of the agents must be within specified boundaries.

A. *The model*

The model used for each individual agent is a simple kinematic model of a unit mass in $\mathbb{R}^3$. For simplicity, we use an abbreviated matrix representation as if the vectors were one-dimensional:

$$\begin{bmatrix} p[k+1] \\ v[k+1] \end{bmatrix} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p[k] \\ v[k] \end{bmatrix} + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} a[k] \tag{1}$$

where vectors p, v and a are the discretized position, velocity and acceleration of the mass at time step $k = 0, 1, \ldots, K-1$, with K being the length of the horizon and h the time step duration in seconds. A compact representation is the following:

$$x[k+1] = Ax[k] + Bu[k] \tag{2}$$

where $x \in \mathbb{R}^6$, $A \in \mathbb{R}^{6\times6}$, $B \in \mathbb{R}^{6\times3}$ and $u[k] \in \mathbb{R}^3$. Expanding the model gives

$$\begin{aligned} x[1] &= Ax[0] + Bu[0] \\ x[2] &= A^2x[0] + ABu[0] + Bu[1] \\ &\vdots \\ x[K] &= A^Kx[0] + A^{K-1}Bu[0] + \ldots + Bu[K-1] \end{aligned} \tag{3}$$

From Eqn. (3), we can write the position sequence $P \in \mathbb{R}^{3K}$ as an affine function of the input sequence $U \in \mathbb{R}^{3K}$.

$$P = A_0X_0 + \Lambda U \tag{4}$$

where $X_0 \in \mathbb{R}^{6K}$ is the repeated sequence of initial states, and $\Lambda \in \mathbb{R}^{3K\times3K}$ is defined as

$$\Lambda = \begin{bmatrix} \Psi B & 0 & \ldots & 0 \\ \Psi AB & \Psi B & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \Psi A^{K-1}B & \Psi A^{K-2}B & \ldots & \Psi B \end{bmatrix} \tag{5}$$

with matrix $\Psi = \begin{bmatrix} I_3 & 0 \end{bmatrix}$ selecting the first three rows of the matrix products (those corresponding the position states). Lastly, $A_0 \in \mathbb{R}^{3K\times6}$ is the propagation of the initial states

$$A_0 = \begin{bmatrix} \Psi A & \Psi A^2 & \ldots & \Psi A^K \end{bmatrix}^\top \tag{6}$$

B. *Objective function*

The objective function has three main components: trajectory error, control effort and input variation. A similar formulation can be found in [18].

1) Trajectory error penalty: this term drives the agents to their goals. We want to minimize the error between the position at the final time step of the prediction horizon $p[K]$, and the desired final position p_d. The error term is given by

$$e = \|p[K] - p_d\|_2 \tag{7}$$

which can be casted into a quadratic cost function of the input sequence, using Eqn. (4):

$$J_e = U^\top(\Lambda^\top\tilde{Q}\Lambda)U - 2(P_d^\top\tilde{Q}\Lambda - A_0X_0\tilde{Q}\Lambda)U \tag{8}$$

Matrix $\tilde{Q} \in \mathbb{R}^{3K\times3K}$ is a positive semidefinite matrix that weights the error at each time step. Given that we want to enforce the trajectory to *end* at the desired reference, then

$$\tilde{Q} = \begin{bmatrix} 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & Q \end{bmatrix} \tag{9}$$

where $Q \in \mathbb{R}^{3\times3}$ is chosen as a diagonal matrix that weights each coordinate of the position vector at the last time step of the horizon.

2) Control effort penalty: we also want to minimize the total control effort throughout the trajectory, through the following quadratic cost function:

$$J_u = U^\top\tilde{R}U \tag{10}$$

Similarly, $\tilde{R} \in \mathbb{R}^{3K\times3K}$ is positive semidefinite and weights the penalty on control effort

$$\tilde{R} = \begin{bmatrix} R & 0 & \ldots & 0 \\ 0 & R & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & R \end{bmatrix} \tag{11}$$

with $R \in \mathbb{R}^{3\times3}$, a diagonal positive definite matrix.

3) Input variation penalty: this term is used to minimize variations of the acceleration, leading to smooth trajectories in position. Define the quadratic cost

$$J_\delta = \sum_{k=0}^{K-1} \|u[k] - u[k-1]\|^2 \tag{12}$$

To transform Eqn. (12) into a quadratic form, define $\Delta \in \mathbb{R}^{3K\times3K}$

$$\Delta = \begin{bmatrix} I_3 & 0 & 0 & \ldots & 0 & 0 \\ -I_3 & I_3 & 0 & \ldots & 0 & 0 \\ 0 & -I_3 & I_3 & \ldots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -I_3 & I_3 \end{bmatrix} \tag{13}$$

and define a vector $U_{-1} \in \mathbb{R}^{3K}$ to include the term $u[-1]$ (which is a known constant at iteration k)

$$U_{-1} = \begin{bmatrix} u[-1] & 0 & \ldots & 0 \end{bmatrix}^\top \tag{14}$$

Finally, we write Eqn. (12) in quadratic form

$$J_\delta = U^\top(\Delta^\top\tilde{S}\Delta)U - 2(U_{-1}^\top\tilde{S}\Delta)U \tag{15}$$

where $\tilde{S} \in \mathbb{R}^{3K\times3K}$ is positive definite of the form

$$\tilde{S} = \begin{bmatrix} S & 0 & \ldots & 0 \\ 0 & S & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & S \end{bmatrix} \tag{16}$$

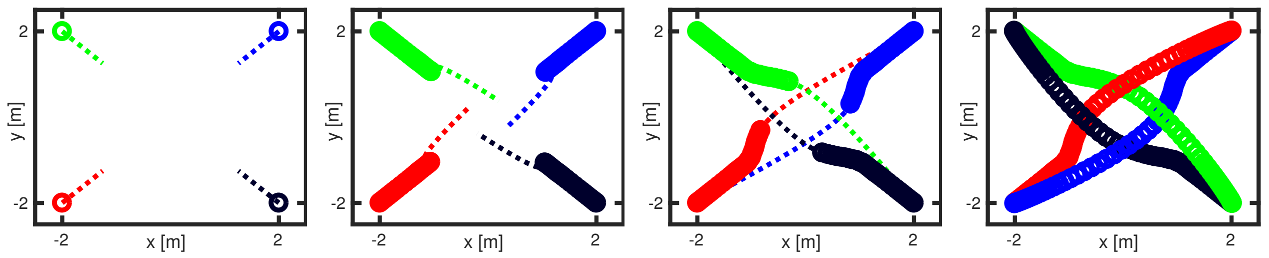

Fig. 2. 4-vehicle position exchange scenario in 2D, solved using DMPC. The vehicles predict future collisions and slow down to safely circumvent the congested area. The result is a successful point-to-point transition, respecting all the constraints of the problem.

with $S \in \mathbb{R}^{3\times3}$, a diagonal matrix.

The total cost function $\mathcal{J}$ is obtained by adding together Eqns. (8,10,15)

$$\begin{aligned}\mathcal{J}(U) &= U^\top(\Lambda^\top\tilde{Q}\Lambda + \tilde{R} + \Delta^\top\tilde{S}\Delta)U \\ &\quad - 2(P_d^\top\tilde{Q}\Lambda - A_0X_0\tilde{Q}\Lambda + U_{-1}^\top\tilde{S}\Delta)U\end{aligned} \tag{17}$$

C. Collision avoidance

To procure collision avoidance, each agent can access information of the most up-to-date prediction horizon of every other agent (p_j). When agent i predicts a future collision at time step k of the current prediction horizon, then it these collision constraints to its optimization problem

$$\|p_i[k] - p_j[k]\|_2 \geq r_{min} \quad \forall i, j = 1, 2, \ldots, N \quad i \neq j \tag{18}$$

The non-convex constraint in Eqn. (18) is linearized using a Taylor series expansion about the previous iterate q of the predicted position horizon of agent i, p_i^q.

$$\|p_i^q[k] - p_j[k]\|_2 + \frac{(p_i^q[k] - p_j[k])^\top}{\|p_i^q[k] - p_j[k]\|_2}(p_i[k] - p_i^q[k]) \geq r_{min} \tag{19}$$

The only variable in Eqn. (19) is $p_i[k]$, then it can can be expressed as

$$(p_i^q[k] - p_j[k])^\top p_i[k] \geq \rho_j \tag{20}$$

with

$$\begin{aligned}\rho_j &= r_{min}\,\|p_i^q[k] - p_j[k]\|_2 - \|p_i^q[k] - p_j[k]\|_2^2 \\ &\quad + (p_i^q[k] - p_j[k])^\top p_i^q[k]\end{aligned} \tag{21}$$

Now, cast Eqn. (20) into an affine function of the input U:

$$\Upsilon_j\Lambda U_i \geq \rho_j - \Upsilon_j A_0 x_i[0] \tag{22}$$

where $\Upsilon_j \in \mathbb{R}^{1\times3K}$ is defined as

$$\Upsilon_j = \begin{bmatrix} 0_{1\times3k} & (p_i^q[k] - p_j[k])^\top & 0_{1\times3(K-1-k)} \end{bmatrix} \tag{23}$$

Finally, introduce $A_{coll,j} \in \mathbb{R}^{1\times3K}$ and $b_{coll,j} \in \mathbb{R}$

$$A_{coll,j} = \Upsilon_j\Lambda; \quad b_{coll,j} = \rho_j - \Upsilon_j A_0 x_i[0] \tag{24}$$

The complete collision constraint matrices for agent i, $A_{coll} \in \mathbb{R}^{(N-1)\times3K}$ and $b_{coll} \in \mathbb{R}^{(N-1)}$, can be obtained by stacking vertically the matrices in Eqn. (24) for all the $N-1$ neighbours of agent i.

D. Physical Limits

The agents must remain within a specified volume and respecting acceleration limits, hence inequality constraints must be enforced both in the position and acceleration at each time step of the prediction horizon. Let $p_{min}, p_{max}, a_{min}, a_{max} \in \mathbb{R}^3$ be the physical limits of the system. Define $P_{min}, P_{max}, U_{min}, U_{max} \in \mathbb{R}^{3K}$:

$$\begin{aligned} P_{min} &= [p_{min}\ldots p_{min}]^\top; \quad P_{max} = [p_{max}\ldots p_{max}]^\top \\ U_{min} &= [a_{min}\ldots a_{min}]^\top; \quad U_{max} = [a_{max}\ldots a_{max}]^\top \end{aligned} \tag{25}$$

Then, the physical limits constraints are formulated as

$$\begin{aligned} P_{min} &\leq \Lambda U \leq P_{max} \\ U_{min} &\leq U \leq U_{max} \end{aligned} \tag{26}$$

E. Convex optimization problem

The complete inequality constraints for agent i are obtained by stacking vertically the collision avoidance and physical limits constraints. At each time step, agent i solves the following convex optimization problem to update its prediction horizon

$$\begin{aligned} \underset{U}{\text{minimize}} \quad & \mathcal{J}(U) \\ \text{subject to} \quad & A_{in}U \leq b_{in} \end{aligned} \tag{27}$$

If collisions are predicted, then the problem would have $6K + N - 1$ inequality constraints, otherwise the number is reduced to $6K$. The formulation in Eqn. (27) results in a Quadratic Programming problem that can be computed efficiently.

III. Distributed Model Predictive Control Algorithm

The proposed DMPC algorithm for point-to-point transitions is outlined in Alg. 1. It requires as input several parameters used in Section II: h, K, N, Q, R, S, p_{min}, p_{max}, a_{min}, a_{max}, r_{min}. Also, a simulation duration T, in seconds, needs to be specified. In line 1, the algorithm initializes a list of prediction horizons for all vehicles by specifying linear trajectories from initial positions to final positions. The algorithm goes through each time step of the trajectory ($k_T = 1, \ldots, K_T$ with $K_T = T/h + 1$), and through every agent, solving the corresponding QP as derived in Eqn. (27). If a solution is found, then it executes a state propagation by applying the first input of the optimization variable U, to the model in Eqn. (1) (lines 6-8). If a solution was not found due to infeasibility of the problem or exceeding the maximum number of iterations of the QP solver, then a heuristic approach was introduced to retry solving the problem.

Algorithm 1: DMPC for Point-to-Point Transitions

```
Input  : Initial and final positions, DMPC parameters
Output: Position, velocity and acceleration trajectories
1  $l \leftarrow$ Initialize predictions
2  while $trials < max$ and not converged do
3      foreach trajectory time step $k_T = 1, ..., K_T - 1$ do
4          foreach agent $i = 1, ..., N$ do
5              $a_i[k_T] \leftarrow$ Solve QP - Eqn. (27)
6              if QP solved then
7                  $p_i[k_T], v_i[k_T] \leftarrow$ Propagate States
8                  $l \leftarrow$ Update Prediction Horizon
9              else
10                 exit loop
11         if QP not solved then
12             $Q_{coll} \leftarrow$ Increase $Q_{coll}$
13             $trials \leftarrow trials + 1$
14             exit loop
15     if QP solved and all vehicles reached goal then
16         converged $\leftarrow$ true;
17     else if QP solved and not all vehicles reached goal
        then
18         $Q_{coll} \leftarrow$ Increase $Q_{coll}$
19         $trials \leftarrow trials + 1$
20 if converged then
21     $[p, v, a] \leftarrow$ Interpolate for higher resolution
22 else
23     $[p, v, a] \leftarrow \emptyset$
24 return [p,v,a]
```

A. Heuristic approach for improved feasibility

The heuristic approach considers two different scenarios for choosing the values of the Q and S matrices.

1) No collisions in the horizon: more aggressive behaviour towards the goal. Use matrices Q_{agg} with higher value and S_{agg} with a lower value (less penalty in acceleration change).
2) Collisions in the horizon: more conservative behaviour is required. Use matrices Q_{coll} with reduced values and S_{coll} with higher values.

Sometimes the conservativeness of the agents is counterproductive to avoid collisions. In those cases, (lines 11-14) a more aggressive behaviour during predicted collisions is enforced, and the algorithm tries to solve the problem with the new matrix Q_{coll}. In other cases, the conservative behaviour is unable to drive the agents to the waypoint within the simulation duration T; in those cases a more aggressive behaviour is also induced (lines 17-19).

Convergence of Alg. 1 is declared when the trajectories for all agents respect all the constraints, and the point-to-point transition was completed within T seconds (vehicles within 5cm of the desired location). If the algorithm coverged, the last step (lines 20-21) is to interpolate the resulting trajectory to obtain a better resolution (say, time step $T_s = 0.01s$).

IV. Main Results

In this section, simulation analysis of the DMPC algorithm is provided. Implementation was done in MATLAB and executed on a PC with an Intel Xeon CPU and 16GB of RAM. The simulation parameters were $T = 15$s, $h = 0.2$s, $K = 15$ (3-second horizon), $r_{min} = 0.75$m and $a_{max} = -a_{min} = 0.7[m/s^2]$. As for the penalty matrices, the values chosen were: $Q_{agg} = 1000I_3$, $Q_{coll} = 10I_3$, $S_{agg} = 10I_3$, $S_{coll} = 100I_3$, $R = I_3$. DMPC was allowed a maximum of 10 tries to converge.

A. Four corner position exchange scenario

To illustrate how DMPC is able to manage colliding trajectories, Fig. (2) shows a transition problem for four vehicles, which were tasked to exchange positions in the plane. Initially, the vehicles follow a direct path towards their desired final locations, until future collisions are detected. In that moment, collision constraints start acting within the optimization problem, replanning the trajectories as to avoid the other vehicles. By the time the vehicles reach the minimum allowed distance, they slow down and are able to coordinate and circumvent the congested area successfully. Such a cooperative behaviour is also observed in more difficult tasks (although harder to visualize), such as the one depicted in Fig. (1), and is the key factor enabling large teams of vehicles to perform successful transitions.

B. DMPC vs dec-iSCP

To test the capabilities of the DMPC algorithm, we compare it to dec-iSCP [8] a state-of-the-art algorithm using sequential convex programming. Scenarios with random sets of initial and final positions were generated and solved using both algorithms. The comparison criteria was the probability of success (finding feasible solutions within T seconds) and the average computation time.

Fig. (3) shows the probability of success as function of the number of vehicles (ranging from 2 to 26). The proposed DMPC algorithm was able to find a solution in each case, whereas for dec-iSCP the amount of succeeded trials tended to decrease as the number of vehicles increased. The sequential solving nature of dec-iSCP makes it more difficult to solve as more vehicles are introduced to the problem, a shortcoming not experienced in DMPC thanks to the parallel nature of the algorithm.

As for the computation time, Fig. (4) reveals that the complexity of the DMPC algorithm scales linearly with the number of vehicles. On the other hand, dec-iSCP seems to scale quadratically. Moreover, the standard deviation bars indicate there is a lot of variance in the computation time of dec-iSCP. The computational complexity of dec-iSCP is

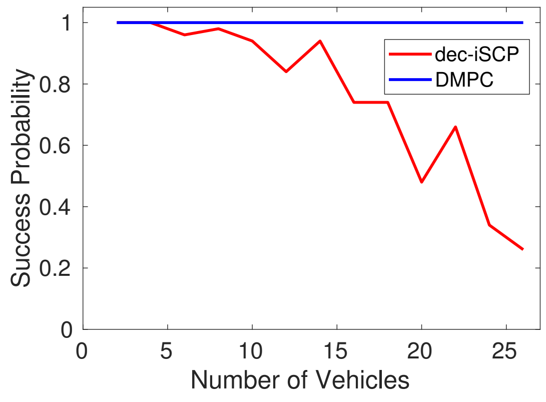

Fig. 3. Probability of success comparison on randomly generated scenarios in a $5 \times 5 \times 2$ workspace. For each one of the number of vehicles cases considered, 50 random initial and final points were generated within the feasible set of the optimization problem.

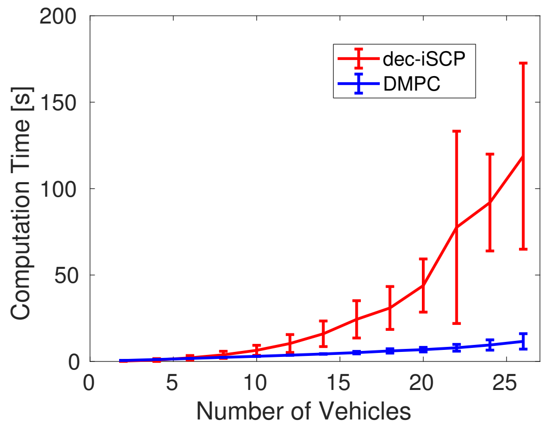

Fig. 4. Average computation time for the same randomly generated trials as in Fig. (3). DMPC scales better than dec-iSCP,

directly related with the number of collision constraints to be added for each vehicle [8], which in turn is correlated with the complexity of the transitions to execute. For DMPC, the variance in execution time is introduced mainly by the use of the heuristic approach, which sometimes force the algorithm to recompute trajectories to find a solution. Naturally, as the number of vehicles increases, the difficulty of the transition also increases, which leads to a higher probability of DMPC requiring to recompute trajectories. This is seen in the increasingly larger standard deviation bars, although the variance is much less prominent than in dec-iSCP. Also, note that the MATLAB implementation of DMPC is unable to exploit the parallelization of the algorithm. A multi-threaded parallel implementation could significantly lower the runtimes shown in Fig. (3).

V. Conclusions

This paper formulated a DMPC algorithm for fast multi-agent point-to-point trajectory generation. By equipping the agents with a prediction horizon, they are able to predict and avoid future collisions. An intuitive heuristic was introduced to enhance the probability of finding feasible solutions, an increasing the usefulness of the algorithm. The parallel execution is a more cooperative paradigm than sequential algorithms such as dec-iSCP, which results in higher probability of success, specially in volumes with high density of agents. The algorithm scales linearly with the number of agents, and it is only slightly dependent on the difficulty of the task. Further reduction in computation time could be achieved by a true parallel implementation of the algorithm (e.g. multithreading in C++), making it a good option for real-time experiments with robots.

References

[1] D. Vallejo, P. Remagnino, D. N. Monekosso, L. Jiménez, and C. González, “A multi-agent architecture for multi-robot surveillance,” in *International Conference on Computational Collective Intelligence*. Springer, 2009, pp. 266–278.

[2] C. Carbone, O. Garibaldi, and Z. Kurt, “Swarm robotics as a solution to crops inspection for precision agriculture,” *KnE Engineering*, vol. 3, no. 1, pp. 552–562, 2018.

[3] E. Guizzo, “Three engineers, hundreds of robots, one warehouse,” *IEEE spectrum*, vol. 45, no. 7, pp. 26–34, 2008.

[4] T. Schouwenaars, B. De Moor, E. Feron, and J. How, “Mixed integer programming for multi-vehicle path planning,” in *European Control Conference (ECC)*. IEEE, 2001, pp. 2603–2608.

[5] F. Augugliaro, A. P. Schoellig, and R. D’Andrea, “Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 1917–1922.

[6] M. Turpin, N. Michael, and V. Kumar, “Decentralized formation control with variable shapes for aerial robots,” in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 23–30.

[7] S. Boyd, “Sequential convex programming,” *Lecture Notes, Stanford University*, 2008.

[8] Y. Chen, M. Cutler, and J. P. How, “Decoupled multiagent path planning via incremental sequential convex programming,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5954–5961.

[9] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme, “Downwash-aware trajectory planning for large quadrotor teams,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 250–257.

[10] S. Tang and V. Kumar, “A complete algorithm for generating safe trajectories for multi-robot teams,” in *Robotics Research*. Springer, 2018, pp. 599–616.

[11] L. Wang, A. D. Ames, and M. Egerstedt, “Safety barrier certificates for collisions-free multirobot systems,” *IEEE Transactions on Robotics*, 2017.

[12] S. Bhattacharya and V. Kumar, “Distributed optimization with pairwise constraints and its application to multi-robot path planning,” *Robotics: Science and Systems VI*, vol. 177, 2011.

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

[14] L. Dai, Q. Cao, Y. Xia, and Y. Gao, “Distributed mpc for formation of multi-agent systems with collision avoidance and obstacle avoidance,” *Journal of the Franklin Institute*, vol. 354, no. 4, pp. 2068–2085, 2017.

[15] P. Wang and B. Ding, “A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance,” *International Journal of Control*, vol. 87, no. 1, pp. 52–63, 2014.

[16] R. Van Parys and G. Pipeleers, “Distributed model predictive formation control with inter-vehicle collision avoidance,” in *Proceedings of the 2017 Asian Control Conference*, 2017.

[17] H. Sayyaadi and A. Soltani, “Decentralized polynomial trajectory generation for flight formation of quadrotors,” *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, vol. 231, no. 4, pp. 690–707, 2017.

[18] P. Ru, “Nonlinear model predictive control for cooperative control and estimation,” Ph.D. dissertation, 2017.