

Distributed Optimization of Multi-Robot Motion with Time-Energy Criterion

Mohamad T. Shahab and Moustafa Elshafei

Abstract

This paper is an application of a special case of distributed optimization problem. It is applied on optimizing the motion of multiple robot systems. The problem is decomposed into L subproblems with L being the number of robot systems. This decomposition reduces the problem to solving a single robot problem. The optimization problem is solved via a distributed algorithm, utilizing subgradient method. A global objective function is set as the sum of individual robot objectives in time and energy. Constraints are divided into two sets, namely, robot-individual constraints and robots' interactions (collision) constraints. The approach is applied for the case of wheeled mobile robots: we are able to generate in parallel for each robot an optimized control input trajectory and then illustrate it in simulation examples.

Keywords: distributed algorithms, multi-robot systems, numerical optimal control, time-energy minimization

1. Introduction

Research in multi-robot systems is motivated by several notions; namely, some motivation can be put as [1]:

- It is complex for one single robot system to fulfill complex tasks. Instead, more than one system would simplify the solution.
- Tasks are generally distributed in nature.
- Multiple limited-resource robot systems are more efficient to deal with than a single powerful robot system.
- Speed of the task process increases through parallelism in multiple robot systems.
- Robustness increases as redundancy is introduced in multiple systems.

Until recently, the number of real-life implementations of multi-robot systems is relatively small. The reason is the complexity associated with the field. Also, the related technologies are relatively new. Emergence of autonomous driving vehicle

technology and market can push the boundaries in the field. As technology develops, new venues for application will open for mainstream use rather than only in research and development labs. Due to its promising applicability, autonomous cars and vehicles (or various intelligent transportation systems in general) sit at the forefront [2, 3]. To name a few, benefits include reducing congestions [4], increasing road safety [5], and, of course, self-driving cars [6]. Another application in civil environments is related to safety and security like rescue missions of searching for missing people in areas hard for humans to operate in [7] or searching for dangerous materials or bombs [8] in an evacuated building. Also, another area of application of multi-robot systems is in military area; research was done heavily in the fields of unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) [9, 10].

Many approaches are developed to tackle the issue of *multiple* robot systems. Under the inspiration of biological systems and the need of technologies, many problems are defined as cooperative motions. Cooperative motion is discussed in [11–16]. Optimization in both time and energy has been tackled in the literature [17–20]. There is an opportunity to incorporate concept of time/energy optimization into the paradigm of multi-robot systems.

This paper investigates the solution of a time-energy optimal control problem for multiple mobile robots; namely, the paper is to study the problem as a nonlinear programming (NLP) problem. The main idea of the solution used here is to utilize distributed optimization techniques to solve the overall optimization problem. Solving for optimal time and energy of more than one robot system adds more burden on the problem; robot interaction with each other is added to the problem. This paper will focus more on the distributed aspect of the problem; more details about the numerical optimal control problem formulation can be found in [21]. In [21], the problem of controlling the motion of a single mobile robot is solved using the direct method of numerical optimal control (see [22]); this showed great flexibility in incorporating physical constraints and nonlinear dynamics of the system.

The rest of this section will define the global problem formulation. Discussion about distributed optimization and associated algorithm is presented in Section 2. Section 3 will apply the method on the multi-robot problem. Application to wheeled mobile robots and simulation examples are discussed in Section 4 followed by the conclusion.

1.1 Global problem formulation

We can present the discrete time global optimization (numerical optimal control) problem for L robots as follows:

$$\begin{aligned}
 & \min_{\{u^i, t_s^i\}_{\forall k, \forall i}} \sum_{\forall i} H(x^i(N)) + z^i(N) \\
 & \text{s.t.} \\
 & z^i(k+1) = z^i(k) + t_s^i(k) \cdot \{L(x^i(k), u^i(k), t_s^i(k))\} \\
 & x^i(k+1) = f_D(x^i(k), u^i(k), t_s^i(k)) \\
 & g(x^i(k), u^i(k), t_s^i(k)) \leq 0 \\
 & \Omega^i(\{x^i(k)\}_{\forall i}, \{u^i(k)\}_{\forall i}, \{t_s^i(k)\}_{\forall i}) \leq 0 \\
 & \forall k, z^i(0) = 0, x^i(0) = x_0^i, i = 1, 2, \dots, L
 \end{aligned} \tag{1}$$

with t being the time-independent variable, k being the time index in discrete domain, t_s^i being the sampling period, and N being the number of time discrete instants across the time horizon, i.e., $k = 0, 1, \dots, N$. The sampling period

corresponds to the length of time the system input $u^i(t)$ is kept constant (zero-order hold): we assume the system input to be

$$u^i(t) = u^i(k), \text{ for } t_k^i \leq t < t_{k+1}^i, t_{k+1}^i = t_k^i + t_s^i(k), t_0^i = 0, \forall i$$

System behavior is governed by the nonlinear dynamic system in $f_D(x^i(k), u^i(k), t_s^i(k))$ with $x^i(k)$ as the robot i states an initial condition of $x^i(0) = x_0^i$. The above optimal control problem has a final state objective of $H(x^i(N))$ with the Lagrangian $L(x^i(k), u^i(k), t_s^i(k))$ information being embedded into a dummy state variable $z^i(k)$.

The above optimization problem can be viewed as having *two* sets of control variables; the first set resembles the discretized system inputs, $\{u^i(k)\}_{\forall k}$, and the other set consists of the variable sampling period, $\{t_s^i(k)\}_{\forall k}$. Let us have the Lagrangian for the problem be

$$L = x^T Q x + u^T R u + \beta,$$

with β being the scalar weight on time. The performance is restricted by a collection of inequality constraints of robot-specific constraints $g(\cdot) \leq 0$ and robot-interaction constraints of $\Omega^i(\cdot) \leq 0$. The objective function is just the summation of individual objectives. In this paper, as it will be explained later, we consider only collision avoidance as robot-interaction requirement; however, the above formulation can also meet other considerations. It can be shown that the objective function in (1) corresponds to objective function of the form

$$\min \sum_{i=1}^L \left[H(x^i(t_f)) + \int_{t_0}^{t_f} x(t)^T Q x(t) + u(t)^T R u(t) + \beta dt \right]$$

2. Distributed optimization

Here in this section, the concept of distributed optimization is explored. This area tackles optimization problems with *distributed* nature. Consider the following optimization problem:

$$\begin{aligned} \min_{\{u^i\}_{\forall i}} & \sum_{i=1}^L J^i(u^i) \\ \text{s.t.} & u^i \in \Omega^i, \text{ for } i = 1, \dots, L \end{aligned} \quad (2)$$

with $J^i(u^i)$ as the objective function and Ω^i the set of constraints. We can easily separate the problem into its corresponding *sub* problems. An i th subproblem is easily put as

$$\begin{aligned} \min_{u^i} & J^i(u^i) \\ \text{s.t.} & u^i \in \Omega^i \end{aligned} \quad (3)$$

Observing the global problem in (2), we can see that it is just equivalent to the combination of all the subproblems; it is easy to see that solving for each subproblem (3) *individually* will result in the solution for the whole global problem. Now, however, consider the following problem:

$$\begin{aligned} \min_{\{u^i\}} \sum_{i=1}^L J^i(u^i) \\ \text{s.t. } g^i(u^1, u^2, \dots, u^L) \leq 0, \forall i \end{aligned} \quad (4)$$

You see clearly that the above problem cannot be trivially separated into some subproblems due to the constraint $g^i(u^1, u^2, \dots, u^L) \leq 0$. This can be called a *complicating* constraint or a *coupling* constraint. In the next subsection, we discuss an optimization method that will help us in solving this kind of problems.

Decomposition in mathematics is the concept of breaking a mathematical problem into smaller subproblems that can be solved independently while not violating the original problem. Primary works of [23, 24] discuss multiple aspects of optimization in general while exploring specific classes as well; these works are excellent resources for reading and understanding. Viewing the applications of distributed optimization will convey the impression that they, however different, are all mostly very similar theoretically. Terms of *networked*, *distributed*, *decentralized*, *cooperative*, and the like are becoming all corresponding to somewhat similar problems. Other works related to this area and the area of multi-agent systems can be found in [25–29].

2.1 Subgradient method

Before going further, we discuss a method used in solving distributed optimization problem which will help us in solving the problem of this paper. This method is called subgradient methods [30]. These methods are similar to the popular optimization algorithms using gradient descent. However, they are extended to escape function differentiation. The works [31–33] also explore the method in the perspective of multi-agent systems.

Consider the typical problem:

$$\min_u J(u) \quad (5)$$

This typical problem can be solved using any *gradient descent* method. At iteration m of an algorithm, a solver, or an optimizer, can be constructed as

$$u_{(m+1)} = u_{(m)} - \alpha_{(m)} d_{(m)} \quad (6)$$

with $\alpha_{(m)}$ as a predefined step size. For a standard gradient method, the vector $d_{(m)}$ contains the gradient information of the problem. The simplest definition is to have

$$d_{(m)} = \nabla J(u_{(m)}) \quad (7)$$

However, for the subgradient method [33], we will have a *definition* of

$$d_{(m)} = p_{(m)} \quad (8)$$

with $p_{(m)}$, called *subgradient* of $J(u)$, being any vector that satisfies the following:

$$J(x) - J(u_{(m)}) \geq p_{(m)}^T [x - u_{(m)}], \forall x \quad (9)$$

The subgradient method is a simple first-order algorithm to minimize a possibly nondifferentiable function. The above definition escapes the requirement of a differentiated objective function. It is defined as finding any vector that makes the

optimization algorithm go to *better* value in a first-order optimality sense. Of course, when a gradient $\nabla J(u_{(m)})$ exists, we can compute the subgradient as the gradient. As it is a first-order method, it could have a lower performance than other second-order approaches. However, the advantage here is that it does not require differentiation. Also, and perhaps more importantly, it gives us flexibility to solve problems in a distributed manner as will be seen later.

Now observe the following constrained optimization problem:

$$\begin{aligned} \min_u J(u) \\ \text{s.t. } g(u) \leq 0 \end{aligned} \quad (10)$$

Let $g(u)$ be a vector of M constraints. Then, we can define the *dual problem*. Let us define the dual function of λ and u as

$$q(\lambda, u) = J(u) + \lambda^T g(u) \quad (11)$$

The vector λ of size M corresponds to the *multipliers* associated with each constraint. The dual problem relaxes the constraints of the original *primal problem* in (10) and solves for λ to maximize the *dual function*:

$$\begin{aligned} \max_{\lambda} q(\lambda, u) \\ \text{s.t. } \lambda \geq 0 \end{aligned} \quad (12)$$

The dual optimization problem is the pair of two optimization problems, namely, a maximization in λ as in (12) and a minimization in u . The pair resembles a maximization-minimization problem. You can visualize the solution of the problem as attacking the effect of constraint violation while solving for the original minimization problem concurrently.

Now, an algorithm for solving the dual problem utilizing subgradient method is discussed. Let us define

$$u(\lambda) = \arg \min_u \{J(u) + \lambda^T g(u)\} \quad (13)$$

The above definition is to clarify the minimum attained at any value of λ . So, with the above definition, at iteration m , with also denoting $u_{(m)} = u(\lambda_{(m)})$, we can safely have

$$q(\lambda, u_{(m)}) = J(u_{(m)}) + \lambda^T g(u_{(m)}) = \min_u \{J(u) + \lambda^T g(u)\} \quad (14)$$

Now, it is obvious from (14) that at iteration m , a subgradient of the dual function in (14) as function of λ can be computed as $p_{(m)} = g(u_{(m)})$. At iteration m of the algorithm, an update for the multipliers is constructed as

$$\lambda_{(m+1)} = P_{\lambda \geq 0} \{\lambda_{(m)} + \alpha_{(m)} g(u_{(m)})\} \quad (15)$$

The projection operator $P_{\lambda \geq 0}\{\cdot\}$ is to ensure that the value of the update $\lambda_{(m)} + \alpha_{(m)} p_{(m)}$ is positive or enforced to zero. Also, observe the *ascent* update with the “+” sign rather than a descent update as it is a maximization. We can assume an initial $\lambda_{(0)} = 0$ or any other positive value. An optimal solution to the original problem in (10) will be attained as $m \rightarrow \infty$, with the optimal solution value of $u_{(m)}$.

2.2 The distributed algorithm

Recall the problem of combination of L subproblems in (2). Now, let us have the following global problem:

$$\begin{aligned} \min_{\{u^i\}} \sum_{i=1}^L J^i(u^i) \\ \text{s.t. } u^i \in \Omega^i, \forall i \\ g^i(u^1, u^2, \dots, u^L) \leq 0, \forall i \end{aligned} \quad (16)$$

As mentioned, the constraints $g^i(u^1, u^2, \dots, u^L) \leq 0$ are the complicating (or coupling) constraints. We can formulate the dual pair problems to be

$$\begin{aligned} \max_{\{\lambda^i\}} q(\{\lambda^i\}) \\ \text{s.t. } \lambda^i \geq 0, \forall i \end{aligned} \quad (17)$$

Put in mind that $\{\lambda^i\} = \{\lambda^1, \lambda^2, \dots, \lambda^L\}$. If we define the notations of

$$u = [u^1 \ u^2 \ \dots \ u^L]^T, \lambda = [\lambda^1 \ \lambda^2 \ \dots \ \lambda^L]^T,$$

then we can apply the *primal-dual* update from (13) and (15) at an iteration m as

$$\begin{aligned} u_{(m+1)}^* &= \arg \min_{\{u^i\} \in \Omega^i} \left\{ \sum_{i=1}^L \left\{ J^i(u^i) + [\lambda_{(m)}^i]^T g^i(u) \right\} \right\} \\ \lambda_{(m+1)} &= \mathcal{P}_{\lambda \geq 0} \left\{ \lambda_{(m)} + \alpha_{(m)} \cdot [g^1(u_{(m)}^*) \ g^2(u_{(m)}^*) \ \dots \ g^L(u_{(m)}^*)]^T \right\} \end{aligned} \quad (18)$$

We can see that the above pair of updates can easily be distributed; after the relaxing of the constraints, the primal problem can be separated. The facility of subgradients lets us propose that any iteration m for subproblem i has

$$\begin{aligned} u_{(m+1)}^i &= \varphi_{\min} \left(J^i(u_{(m)}^i) + [\lambda_{(m)}^i]^T g^i(u_{(m)}^1, u_{(m)}^2, \dots, u_{(m)}^L) \right) \\ \lambda_{(m+1)}^i &= \mathcal{P}_{\lambda^i \geq 0} \left\{ \lambda_{(m)}^i + \alpha g^i(u_{(m)}^1, u_{(m)}^2, \dots, u_{(m)}^L) \right\} \end{aligned} \quad (19)$$

The function $\varphi_{\min}(\cdot)$ is any algorithm minimizer for the primal problem constrained by $u^i \in \Omega^i, \forall i$. Observe that the primal update for each subproblem needs *only* the latest values of the other subproblem updates. During the computations of an iteration, computation of $u_{(m+1)}^i$ and $\lambda_{(m+1)}^i$ is done independent of each other; the updates above can be computed in parallel for each subproblem. The only information shared after each iteration is $u_{(m)}^1, u_{(m)}^2, \dots, u_{(m)}^L$ among all.

3. Distributed algorithm for multi-robot system

3.1 Problem formulation

Now, in this section we can apply the previous discussion into the problem of optimizing the motion of multiple robots. Recall the global optimization problem of motion of L mobile robots from (1)

$$\begin{aligned}
 & \min_{\{u^i, t_s^i\}_{\forall k, \forall i}} \sum_{i=1}^L H(x^i(N)) + z^i(N) \\
 & \text{s.t.} \\
 & z^i(k+1) = z^i(k) + t_s^i(k) \cdot \{L(x^i(k), u^i(k), t_s^i(k))\} \\
 & x^i(k+1) = f_D(x^i(k), u^i(k), t_s^i(k)) \\
 & g^i(x^i(k), u^i(k), t_s^i(k)) \leq 0 \\
 & \Omega^i(\{x^i(k)\}_{\forall i}, \{u^i(k)\}_{\forall i}, \{t_s^i(k)\}_{\forall i}) \leq 0 \\
 & k = 0, 1, \dots, N, i = 1, \dots, L, z^i(0) = 0, x^i(0) = x_0^i
 \end{aligned} \tag{20}$$

You can see that the problem above is just the combination of L subproblems with superscript i corresponding to each robot. The objective function is just the summation of individual objectives. The coupling constraint of $\Omega^i(\{x^i(k)\}_{\forall i}, \{u^i(k)\}_{\forall i}, \{t_s^i(k)\}_{\forall i}) \leq 0, \forall k$ is the only difference to the single robot problem. To simplify the notations, let us define

$$\bar{u}^i = \{u^i(k), t_s^i(k)\}_{k=0}^N$$

The above definition is just to reduce the notation of robot input sequence. If we have, for example, two robot inputs $u^i = [u_1^i \ u_2^i]^T$ (e.g., wheel torques), then we have a total of $3 \times L \times N$ control variables of the optimization problem. We can condense notation of the global problem of multi-robot system without loss of generality to be

$$\begin{aligned}
 & \min_{\{\bar{u}^i\}_{\forall i}} \sum_{i=1}^L J^i(\bar{u}^i) \\
 & \text{s.t. } \bar{u}^i \in \Xi^i \\
 & \Omega^i(\bar{u}^1, \bar{u}^2, \dots, \bar{u}^L) \leq 0, \forall i
 \end{aligned} \tag{21}$$

with objectives

$$J^i(\bar{u}^i) = H(x^i(N)) + z^i(N) \tag{22}$$

which are subject to the set of individual robot i constraints of

$$\Xi^i : \begin{cases} z^i(k+1) = z^i(k) + t_s^i(k) \cdot \{L(x^i(k), u^i(k), t_s^i(k))\} \\ x^i(k+1) = f_D(x^i(k), u^i(k), t_s^i(k)) \\ g^i(x^i(k), u^i(k), t_s^i(k)) \leq 0 \\ \forall k, z^i(0) = 0, x^i(0) = x_0^i \end{cases} \tag{23}$$

3.2 Distributed algorithm

Returning back to the primal-dual problem pair in Section 2, we can establish the algorithm updates according to the defined updates in (19). At each iteration m , we update each robot inputs and multipliers according to

$$\begin{aligned}
 \bar{u}_{(m+1)}^i &= \varphi_{\min} \left(J^i(\bar{u}_{(m)}^i) + [\lambda_{(m)}^i]^T [\Omega_{(m)}^i] \right) \\
 \lambda_{(m+1)}^i &= \mathcal{P}_{\lambda^i \geq 0} \left\{ \lambda_{(m)}^i + \alpha \Omega_{(m)}^i \right\}
 \end{aligned} \tag{24}$$

The minimizer update $\boldsymbol{\varphi}_{min}(\cdot)$ is responsible to solve the single robot optimization (primal) problem according to the objective defined in (22) and subject to constraints in (23). In this paper, the minimizer update $\boldsymbol{\varphi}_{min}(\cdot)$ is selected to be any state-of-the-art nonlinear programming (NLP) algorithm. Let us have the step size α for the dual update to be constant. This is sufficient for converging to a solution of the original problem [33]. You can read the algorithm updates in (24) at iteration m as each robot *independently* optimizes its whole motion throughout the whole time horizon $k = 0, 1, \dots, N$ while at the same time puts in mind the extra *cost* of cooperation/interaction with others introduced by the term $\left[\lambda_{(m)}^i\right]^T \left[\Omega_{(m)}^i\right]$ and so on.

3.3 Algorithm convergence

In this brief section, elaboration is put forth about how to practically use the algorithm. The ultimate goal is to optimize primal problem with no collision violation, i.e., reaching optimal dual (maximum) solution. At each global iteration, we only need to *improve* the primal problem values for the updated extra cost of the interaction constraint, $\left[\lambda^i\right]^T \left[\Omega^i\right]$. In this paper, a perfect solution is to optimize while maintaining $\left[\Omega^i\right]_{\forall k} \leq 0$. A logical property is to monitor the M-element vector of constraints for positive values, i.e., violations. So, a stopping criterion for the algorithm can be chosen to be some minimum change TolJ in the primal problem value:

$$\left|J_{(m+1)} - J_{(m)}\right| \leq \text{TolJ} \text{ with } J_{(m)} = \sum_{i=1}^L J\left(\bar{u}_{(m)}^i\right) \quad (25)$$

We can also *distribute* the stopping decision to individual robots by observing the change in individual objective values.

With condition (25) on its own, we cannot always be satisfying the collision requirement. So, this condition can be accompanied by a condition on the collision constraint violation. For all robots, elements of the complete constraint vector $\left[\Omega^i\right]_{\forall k}$ should be less than a relatively small positive value Tol Ω . So, for each robot, an extra stopping criterion along with criteria in (25) is to have

$$\max\left(\left[\Omega_{(m)}^i\right]_{\forall k}\right) \leq \text{Tol}\Omega \quad (26)$$

Specific values of Tol Ω and TolJ depend on the nonlinear programming algorithm and/or the global desired requirements. Note that the behavior of the two tolerance parameters could be competitive with each other.

4. Application to wheeled mobile robots

4.1 System description

Figure 1 shows the individual robot system considered here. Robot state includes $x^i = [x \ y \ \phi \ \theta_R \ \theta_L \ v_R \ v_L]^T$ with both position (x, y) and orientation ϕ and $(\theta_R, \theta_L, v_R, v_L)$ as the right and left wheel angular positions and velocities, respectively. Robot input includes the respective wheel torques $u^i = [\tau_R \ \tau_L]^T$. You can have the details of the applied nonlinear dynamic model $f_D(x^i(k), u^i(k), t_s^i(k))$ based on the system model developed in [34, 35]. Details of discretization and

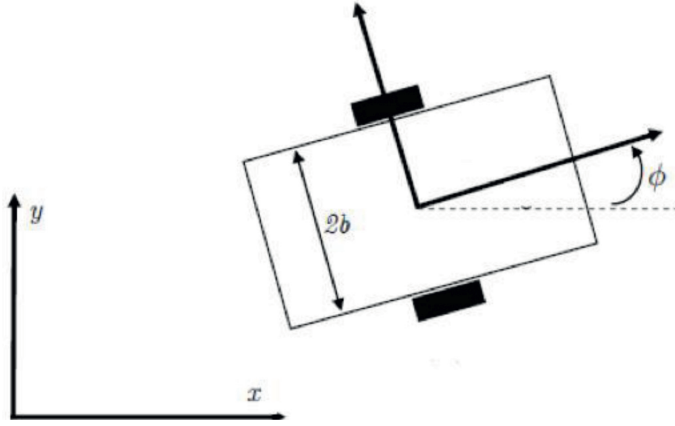


Figure 1.
 Wheeled mobile robot.

choice of parameters of the robot model can be found in [21]. As mentioned before, for choices of Q, R in Section 1, the Lagrangian for the problem is chosen to include the cost for energy spent by the torques of the wheels, the cost for kinetic energy spent by robot body, and the weight on time. Individual robot constraints include final desired configuration tolerance, torque limits, and the ensurance of zero final velocities (see more details in [36]).

4.2 Collision avoidance

Here, we will discuss the formulation and the structure of the coupling constraints. The robots can be designed to perform any cooperative strategy in their motion. Here, we only consider the global goal of optimizing the motion of each robot in time and energy while avoiding colliding with each other during the motion. Let us define the coupling constraint vector across the discrete time indexes as

$$\Omega^i(k) = \Omega^i \left(\{x^i(k)\}_{v_i}, \{u^i(k)\}_{v_i}, \{t_s^i(k)\}_{v_i} \right).$$

For the i th robot, it tries to avoid colliding with the rest of $L - 1$ robots at each of its time indexes k . Let us label elements of the constraint vector as $[\Omega^{ij}(k)]$. Each element is corresponding to a definition of constraint at time index k for all other robots, $j \neq i$. So, for each robot, the constraint vector $[\Omega^i]$ is of size $M = (L - 1) \times N$; of course, the multiplier vector λ^i in (24) is of the same size.

We define the collision avoidance by constraining motion of other robots to be outside a safety circle region around each i robot at the position (x^i, y^i) in the 2D plane:

$$(x^i(k) - \hat{x}^j(k))^2 + (y^i(k) - \hat{y}^j(k))^2 \geq p^2.$$

So, we can define each element of the constraint vector as

$$\Omega^{ij}(k) = p^2 - (x^i(k) - \hat{x}^j(k))^2 + (y^i(k) - \hat{y}^j(k))^2 \quad (27)$$

The radius of the safety region is chosen as p . Because of the definition of the sampling period variable, at each of discrete time step k , the actual time variable does *not* necessarily imply $t^i(k) = t^j(k)$ for all the other $L - 1$ robots. That is why you

see that the x- and y- coordinates in (23) of the j th robot are noted as (\hat{x}^j, \hat{y}^j) which are chosen to be calculated as linear interpolation of positions according to available actual times of $t^i(k)$ and $t^j(k)$.

4.3 Simulation examples

You can follow the whole distributed algorithm for the time-energy optimization of multi-robot system with collision avoidance in the flowchart in **Figure 2**. View the flowchart as the process for each individual robot (subproblem). We implement the algorithm in **Figure 2**. For the primal minimizer update in (24), the nonlinear programming (NLP) function of `fmincon` in MATLAB is used. We solved the problem for motion of $L = 3$ mobile robots. Utilizing the parallel capability in MATLAB, the distributed steps are solved independently utilizing three parallel processors. We choose number of instants $N = 40$; so, we are going to optimize 120 control variables for each robot. More details can be found in [36].

Example 1. In this example, exploration of the behavior of the algorithm is shown. The problem has the following desired values of initial and final positions and orientations for the three robots:

$$(x_0^1, y_0^1, \phi_0^1) = (0, -8, \frac{\pi}{2}), (x_f^1, y_f^1, \phi_f^1) = (0, 5, \pi)$$

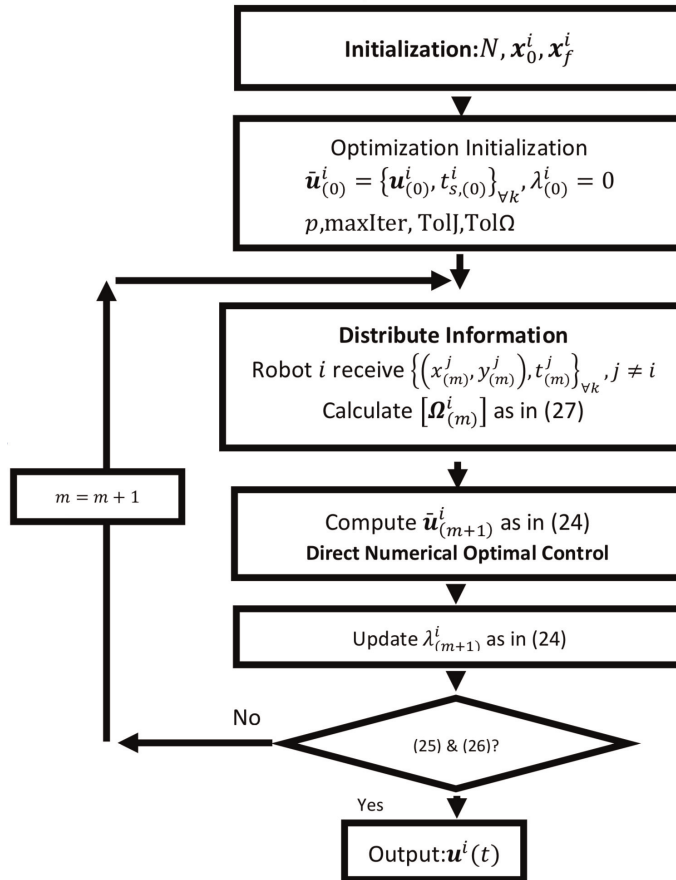


Figure 2.
Distributed algorithm flowchart to optimize multi-robot motion.

$$(x_0^2, y_0^2, \phi_0^2) = (-10, -1, 0), (x_f^2, y_f^2, \phi_f^2) = \left(8, 5, \frac{\pi}{2}\right)$$

$$(x_0^3, y_0^3, \phi_0^3) = (5, 0, \pi), (x_f^3, y_f^3, \phi_f^3) = \left(-8, -1, \frac{\pi}{2}\right)$$

Here, robot 1 has equal objective weights of 5 on both time and energy, robot 2 has weights of 10 on energy and 1 on time, and robot 3 has 10 on time and 1 on energy. The maximum number of internal NLP iterations (primal update) is set to

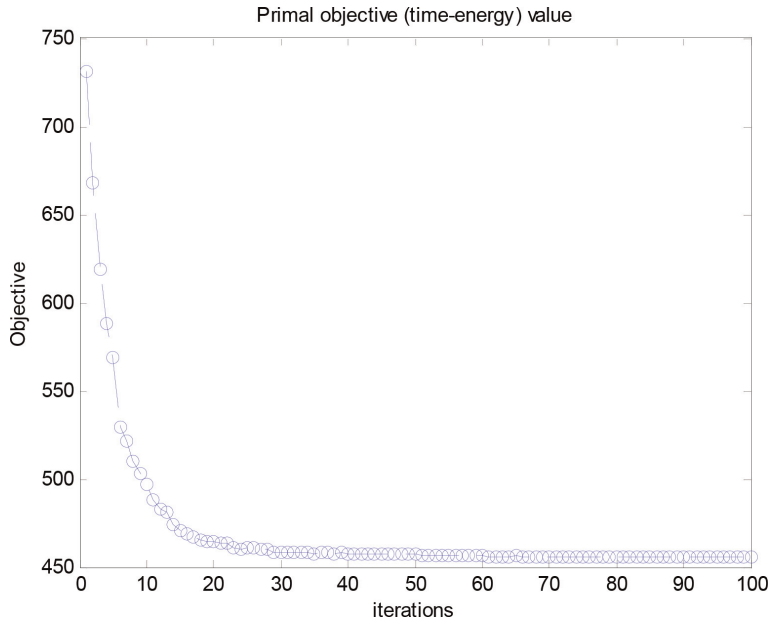


Figure 3.
 The time-energy objective values throughout global iterations (Example 1).

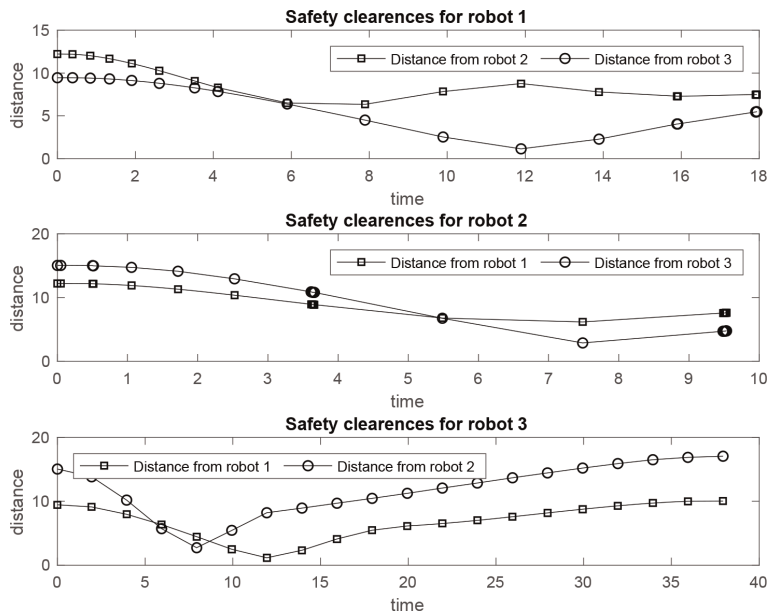


Figure 4.
 Collision avoidance (Example 1).

only 10. The step size is set to $\alpha = 0.1$. Maximum global iterations are allowed for 100 iterations. The safety circle radius is chosen to be $p = 2$.

Figure 3 contains the objective (time-energy) value evolution throughout iterations. You can see the stable convergence as the algorithm progresses. **Figure 4** shows each of the robots' safety clearances during the optimized motion. In **Figure 5**, snapshots of motion of the three robots at different time instants are depicted. This illustrates the collision avoidance attained throughout the optimized motion. Observe also how different are the speeds of each robot because of objective weights; note from **Figure 4** that each robot has a different final time for their motion. The algorithm has shown good performance at eliminating collision constraint violations. **Figures 4** and **5** show an instant (around $t = 12$) where robots 1 and 3 violate collision distance with very small value, but no collision occurs. This is because the maximum number of iterations of the algorithm is exhausted. This

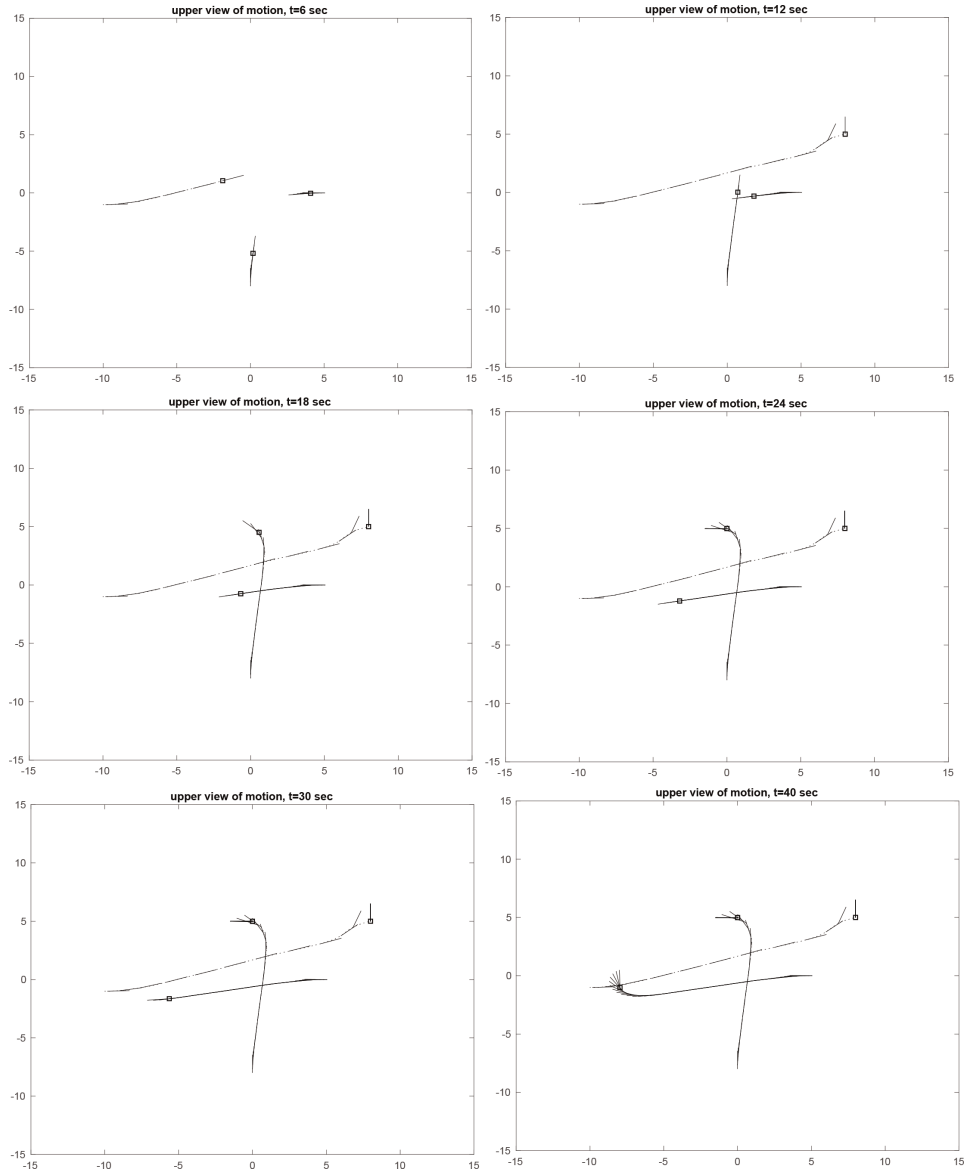


Figure 5. Snapshots of optimized motions at different instants (Example 1).

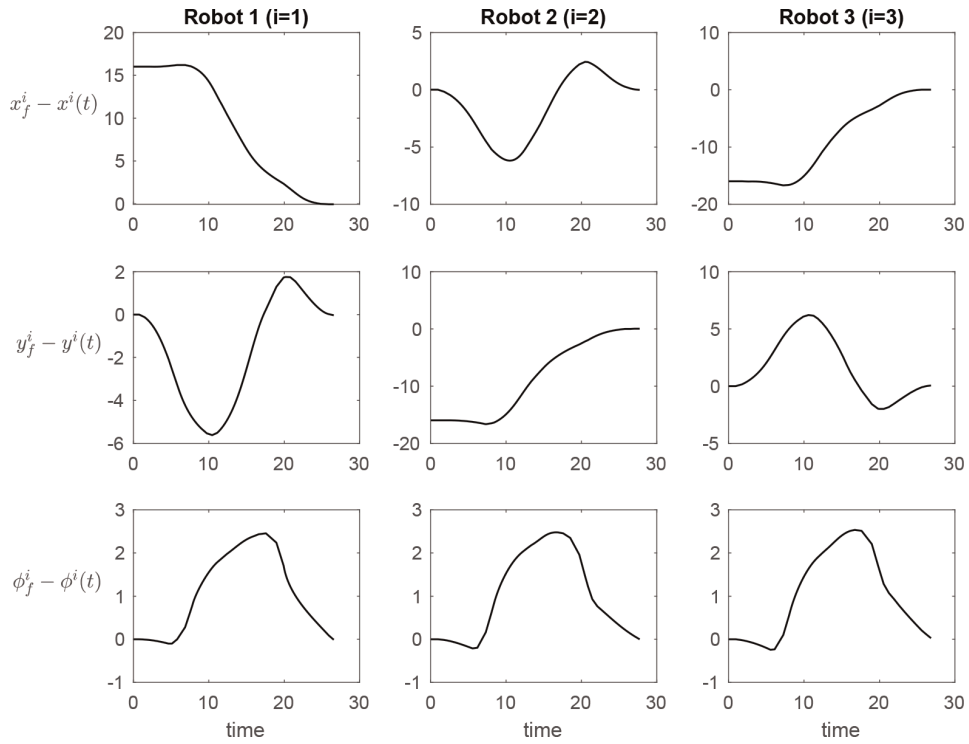


Figure 6.
 Optimized trajectories of the three robots: each row of plots shows x-coordinate error, y-coordinate error, and orientation error, respectively; each column of plots show robots 1, 2, and 3 errors, respectively (Example 2).

indicates the possibility for the motion to be optimized even more if collision constraints were relaxed or if more algorithm iterations were allowed. Initialization of the algorithm also plays a role in algorithm evolution.

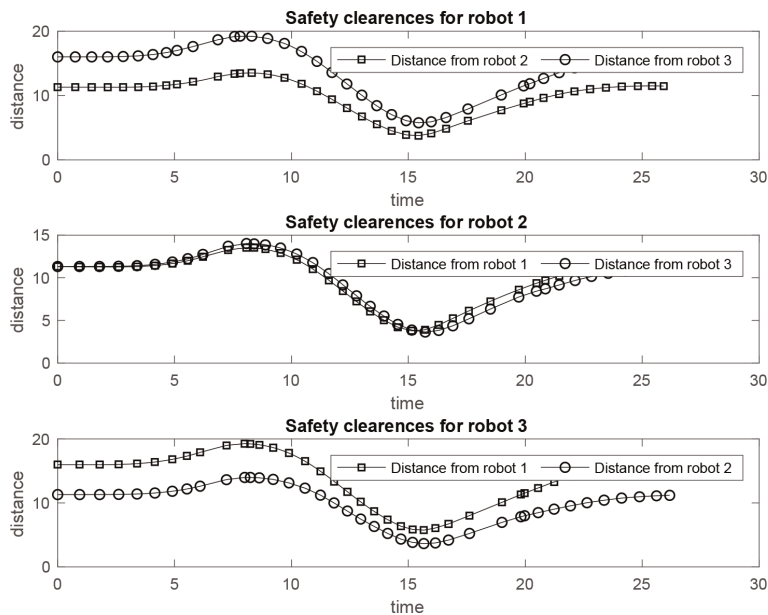


Figure 7.
 Collision avoidance (Example 2).

Example 2. This example illustrates more the satisfaction of the objectives. In this example the safety circle radius is put as $p = 3$. Here we choose the following initial and final positions and orientations for the three robots:

$$\begin{aligned}(x_0^1, y_0^1, \phi_0^1) &= \left(-8, 0, \frac{\pi}{2}\right), (x_f^1, y_f^1, \phi_f^1) = \left(8, 0, \frac{\pi}{2}\right) \\ (x_0^2, y_0^2, \phi_0^2) &= (0, 8, 0), (x_f^2, y_f^2, \phi_f^2) = (0, -8, 0) \\ (x_0^3, y_0^3, \phi_0^3) &= \left(8, 0, -\frac{\pi}{2}\right), (x_f^3, y_f^3, \phi_f^3) = \left(-8, 0, -\frac{\pi}{2}\right)\end{aligned}$$

After applying our approach, you can see the resulting optimized motions in **Figure 6**. In **Figure 6**, errors in x- and y-coordinates and orientation of each robot are shown with respect to time. It is clear that errors of zero are achieved. In **Figure 7** for each robot, constraint evaluations, i.e., safety clearance, are displayed for the other two robots throughout time. You can see that robots come close to each other sometimes but without violating the safety distance. This result is attained maybe because of special structure of initial and final positions and orientations. That could have given flexibility for the algorithm.

5. Conclusion

The paper investigated the time-energy minimization onto the multi-robot case. A global objective function is formulated as the sum of individual robot objectives in time and energy. Constraints are divided into two sets, namely, robot-individual constraints and robots' interaction constraints. The problem is decomposed into L subproblems with L being the number of robot systems. The subproblems are coupled with each other by the collision avoidance information. Applying a distributed algorithm solved the problem iteratively. The overall output gives optimized motions for all robots in time and energy while adhering and not colliding with each other. We applied our approach to the case of three wheeled mobile robots: we generated in parallel for each robot an optimized control input trajectory.

An extension to this study is to generate optimized motion trajectories and apply them experimentally. A possible area for experimentation is full-scale autonomous vehicles. Issues related to communication and distributing information during the parallel algorithm will need to be incorporated and investigated. Also, aspects of state estimation and localization of the robot system will come into the place which were not considered in this work. A possible other investigation is to distribute the problem further onto the time variable k ; this will lead the problem to the domain of distributed model predictive control. This will, possibly, pave the way to faster deployment into autonomous vehicles.

Acknowledgements

The authors would like to thank King Fahd University of Petroleum and Minerals supporting this work.

Author details


Mohamad T. Shahab^{1*} and Moustafa Elshafei²

1 Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada

2 Zewail City of Science and Technology, October Gardens, 6th of October City, Giza, Egypt

*Address all correspondence to: m4shahab@uwaterloo.ca

IntechOpen

© 2019 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Siciliano B, Khatib O, editors. Springer Handbook of Robotics. In: Springer Science & Business Media. Berlin/Heidelberg; 2008
- [2] Thrun S. What we're Driving At. The Official Google Blog. 2011. Available from: <https://googleblog.blogspot.com/2010/10/what-were-driving-at.html> [Accessed: 23 Jan 2019]
- [3] Ioannou P, editor. Automated Highway Systems. Boston, MA: Springer Science & Business Media; 1997
- [4] Kurzhanskiy AA, Varaiya P. Traffic management: An outlook. *Economics of Transportation*. 2015;**4**(3):135-146
- [5] Huang C-L, Fallah YP, Sengupta R, Krishnan H. Adaptive intervehicle communication control for cooperative safety systems. *Network, IEEE*. 2010; **24**(1):6-13
- [6] Guizzo E. How Google's self-driving car works. *IEEE Spectrum Online*. 2011. Available: <https://spectrum.ieee.org/autoton/robotics/artificial-intelligence/how-google-self-driving-car-works>
- [7] Burgard W, Moors M, Stachniss C, Schneider FE. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*. 2005;**21**(3):376-386
- [8] Ferranti E, Trigoni N, Levene M. Brick & Mortar: An on-line multi-agent exploration algorithm. In: *Robotics and Automation, 2007 IEEE International Conference on*. 2007. pp. 761-767
- [9] Beard RW, McLain TW, Nelson DB, Kingston D, Johanson D. Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proceedings of the IEEE*. 2006;**94**(7):1306-1324
- [10] Das AK, Fierro R, Kumar V, Ostrowski JP, Spletzer J, Taylor CJ. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*. 2002;**18**(5): 813-825
- [11] Jadbabaie A, Lin J, Morse AS. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*. 2003;**48**(6): 988-1001
- [12] Murray RM. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*. 2007; **129**(5):571-583
- [13] Fax JA, Murray RM. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*. 2004;**49**(9): 1465-1476
- [14] Olfati-Saber R. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*. 2006;**51**(3):401-420
- [15] Gazi V, Passino KM. Stability analysis of swarms. *IEEE Transactions on Automatic Control*. 2003;**48**(4): 692-697
- [16] Desai JP, Ostrowski JP, Kumar V. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*. 2001;**17**(6):905-908
- [17] Khoukhi A, Baron L, Balazinski M. Constrained multi-objective trajectory planning of parallel kinematic machines. *Robotics and Computer Integrated Manufacturing*. 2009;**25**(4):756-769
- [18] Bobrow JE. Optimal robot plant planning using the minimum-time criterion. *IEEE Journal of Robotics and Automation*. 1988;**4**(4):443-450

- [19] Sergaki ES, Stavrakakis GS, Pouliezios AD. Optimal robot speed trajectory by minimization of the actuator motor electromechanical losses. *Journal of Intelligent and Robotic Systems*. 2002;**33**(2):187-207
- [20] Sun Z, Reif JH. On finding energy-minimizing paths on terrains. *IEEE Transactions on Robotics*. 2005;**21**(1): 102-114
- [21] Shahab MT, Khoukhi A, Al-Sunni F. Time-energy optimal control of a mobile robot using direct numerical method. arXiv preprint arXiv:1312.7088; 2013
- [22] Chachuat B. *Nonlinear and Dynamic Optimization: From Theory to Practice*. Switzerland: Automatic Control Laboratory, EPFL; 2007
- [23] Bertsekas DP, Nedic A, Ozdaglar AE. *Convex Analysis and Optimization*. Belmont, MA: Athena Scientific; 2003
- [24] Stephen B, Xiao L, Mutapcic A, Mattingley J. *Notes on Decomposition Methods*. Notes for EE364B. Stanford, CA: Stanford University; 2007
- [25] Zhu M, Martinez S. An approximate dual subgradient algorithm for multi-agent non-convex optimization. *IEEE Transactions on Automatic Control*. 2013;**58**(6):1534-1539
- [26] Terelius H. *Distributed multi-agent optimization via dual decomposition* [PhD diss.]. Stockholm, Sweden: KTH; 2010
- [27] Bhattacharya S, Kumar V. Distributed optimization with pairwise constraints and its application to multi-robot path planning. *Robotics: Science and Systems VI*. 2011;177
- [28] Johansson B. *On distributed optimization in networked systems* [PhD diss.]. Stockholm, Sweden: KTH; 2008
- [29] Kuwata Y, How J. Decentralized cooperative trajectory optimization for UAVs with coupling constraints. In: *Decision and Control, 2006 45th IEEE Conference on*. 2006. pp. 6820-6825
- [30] Boyd S, Mutapcic A. Subgradient methods. In: *Lecture Notes of EE364b*. Stanford, CA; Stanford University
- [31] Nedic A, Ozdaglar A. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*. 2009;**54**(1):48-61
- [32] Nedic A, Ozdaglar A, Parrilo PA. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*. 2010;**55**(4):922-938
- [33] Nedic A, Ozdaglar A. Cooperative distributed multi-agent optimization. In Eldar Y, Palomar D, editors. *Convex Optimization in Signal Processing and Communications*. Cambridge: Cambridge University Press; 2010. pp. 340-386
- [34] Fukao T, Nakagawa H, Adachi N. Adaptive tracking control of a nonholonomic mobile robot. *IEEE Transactions on Robotics and Automation*. 2000;**16**(5):609-615
- [35] Yamamoto Y, Yun X. Coordinating locomotion and manipulation of a mobile manipulator. *IEEE Transactions on Automatic Control*. 1994;**39**(6): 1326-1332
- [36] Shahab MT. Time-energy optimization of motion of multi-robot system [master thesis]. Saudi Arabia: King Fahd University of Petroleum and Minerals; 2014