UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Distributed Control Systems

**Course Project 3
Distributed Task Assignement for Robotic
Network**

Professor: **Giuseppe Notarstefano**
Tutor: **Andrea Testa**

Students:
Francesco Cerri
Gianmarco Canello
Marco Roncato

Academic year 2020/2021

Contents

List of Figures

1 Introduction

1.1 Problem introduction

The aim is to Model and control of a network of robots that need to self-assign a set of tasks in target locations by running an optimization algorithm. The objective of this algorithm is to choose an assignment that optimizes the total path length between tasks and robots. The optimization problem can be approached in a centralized fashion or in a distributed way. A centralized optimization algorithm works using the definitions of hive and drones, where the drones send to the hive the gathered information to be computed and the hive performs computations and replies to the drones with the orders to be followed. In a distributed algorithm there is no hive and drones communication,instead we rely on agents that communicate among each other in order to decide the best action to perform. The problem is well posed and admits a solution in both the approaches, although the uniqueness and the quality of the solution may vary depending on the domain in which the optimization problem is evaluated.

1.2 Tools used

In order to solve the optimization problem we have used basic MATLAB functionalities and the Optimization toolboxes.Regarding the theoretical part, we have used the knowledge of cost coupled optimization principles presented in the Bullo's book [1] ; various tries were carried out by doing a Montecarlo simulation with the agent and task positions randomized inside an interval decided by the user. The spawn area of the agents and tasks is a square.

1.3 Motivations

Solving the Generalized Assignment Problem (GAP) provides us a solution that could be implemented in any application in which is required to assign tasks to agents with a predefined criteria (e.g. path length, agents capability to solve that task, ecc). Even though the GAP problem is a NP-hard problem, and there are several solution in optimization literature, we are trying to provide a robust solution able to cope possible noise that could interfere inside robot communication

1.4 Applications

This type of algorithm is needed in various fields: robotics, for example the optimization of the route for robots that carry items inside a warehouse or a factory, logistics, for example maximizing fuel efficiency of delivery trucks. In general, this algorithm is used when we want to minimize the energy used to complete a certain task made by a large number of independent agents that are able to communicate between themselves.

1.5 Contributions

We have developed some MATLAB simulations in order to show the algorithm working. As a first effort, we have written a problem generator file that can be configured to perform the generation of matrices for both of the assigned tasks. The execution of the problem generation can be personalized by giving the dimension of the spawn area, the number of agents, the number of tasks and In the first task we simulated the convergence error of agents and tasks under the constraint of ... and compared them to a centralized algorithm solution. In the second task we simulated a similar situation but with an added graphical representation.

2 Constrained coupled linear problems

$$\min_{x_i \in X_i} \sum_{i=1}^{N} f_i(x_i) \tag{1}$$

$$\text{subj.to } h_l(x) - b_l = 0 \qquad\qquad \forall l \in 1, \ldots, m \tag{2}$$

$$x_i \in X_i \subset \mathbb{R}^{n_i}, \qquad\qquad \forall i \in 1, \ldots, N \tag{3}$$

with $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ convex, $h_l : \mathbb{R}^{n_i} \to \mathbb{R}$ convex and $X_i \subset \mathbb{R}^{n_i}$ convex and compact

2.1 Distributed Dual Subgradient algorithm

we can solve the constrained coupled optimization problem via the dual distributed subgradient algorithm

$$v_i^{t+1} = \sum_{j \in \mathcal{N}_i} a_{ij} \lambda_j^t \tag{4}$$

$$x_i^{t+1} = \operatorname*{argmin}_{x_i \in \mathcal{X}_i} f_i(x_i) + (v_i^{t+1})^T h_i(x_i) \tag{5}$$

$$\lambda_i^{t+1} = v_i^{t+1} + \alpha^t h_i(x_i^{t+1}) \tag{6}$$

the optimal solution for the dual problem is given by maximizing $q(\lambda)$ with

$$q(\lambda) = \inf_{x \in P} L(x, \lambda) \tag{7}$$

$$L(z, \lambda) = \sum_{i=0}^{N} f_i(x_i) + \sum_{l=0}^{S} \lambda_l \left(\sum_{i=1}^{N} H_i x_i - b_l \right) \quad b \in \mathbb{R}^S \tag{8}$$

In general we know for sure that weak duality always hold (e.g. $q^* \leq f^*$) , thus

$$q^* = \sup_{\lambda} \leq \inf_{x \in \mathbb{X}} = f^* \tag{9}$$

4

Therefore by maximizing the dual function we can define a lower bound for the optimal cost of the primal problem, Although we need strong duality in order to nullify the duality gap (e.g. $q^* \leq f^*$).
in order to get the Strong duality we need to have a "Convex Problem":

1. $X \subset \mathbb{R}^d$ convex;

2. f finite and convex over X;

3. h_l are affines;

4. $\exists \, \overline{x} \in ri(X)$ s.t. $h_l(\overline{x}) - b = 0$;

if the previous assumption hold for the primal problem. Then there is no duality gap, i.e., $q^* = f^*$. This does not solve our main problem of evaluating the optimal value for the decision valuables x, we need to make some extra evaluation in order to obtain the optimal values of the decision variables by making the

$$\min_{x \in X} \; f(x) \tag{10}$$

$$\text{subj.to } h_l(x) - b = 0 \quad l \in 1, \dots, m \tag{11}$$

$$\tag{12}$$

Assuming that

1. $a_{ij} \; i,j \in 1, \dots, N$ be non negative entries of a weighted adjacency A to the undirected graph $\mathcal{G}$, with $a_{ij} > 0$ and A doubly stochastic;

2. the step-size sequence $\{\alpha^t\}_{t \geq 0}$, satiesies the conditions

$$\alpha^t \geq 0, \; \sum_{t=0}^{\infty} \alpha^t = \infty, \; \sum_{t=0}^{\infty} (\alpha^t)^2 < \infty \tag{13}$$

3. $\forall i \in 1, \dots, N$: each function f_i is convex, each constraint $\mathcal{X}_i$ is non-empty, compact and convex set; each function h_i is a component-wise convex function.
   Moreover, there exist $\overline{x}_1 \in X_1, \dots, \overline{x}_N \in X_N$ such that $\sum_{i=1}^{N} h_i(\overline{x}_i) < 0$

Let assumptions 1, 2 and 3 hold, then the sequence of dual variables $\lambda_1^t, \dots, \lambda_N^t$ with $t \geq 0$ generated by the Distributed Dual Subgradient satisfies:

$$\lim_{t \to \infty} \| \lambda_i^t - \lambda^* \| = 0, \tag{14}$$

$$i \in \{1, \dots, N\}, \tag{15}$$

where λ^* is an optimal solution of the dual problem. Moreover, let for each t,

$$\widehat{x_i}^t = \frac{1}{t} \sum_{\tau=0}^{t} x_i^\tau \tag{16}$$

Then, it holds

$$\lim_{t \to \infty} \sum_{i=1}^{N} f_i(\widehat{x_i}^t) = f^*, \tag{17}$$

$$\lim_{t \to \infty} \|\widehat{x_i}^t - x^*\| = 0, \qquad i \in \{1, \ldots, N\}, \tag{18}$$

where x^* and f^* denote an optimal solution and the optimal cost of the primal problem.

3 Distributed constrained coupled problem

3.1 Problem introduction

In the first task we are considering a constrained coupled linear problem in distributed scenario of a set of N agents in $\mathbb{R}^m$ where n_i are the number of components in the state vector of the agents and S are the number of equality constraints the optimization problem is subject to. The problem can be defined as follow:

$$\min_{z_1, \ldots, z_N} \sum_{i=1}^{N} c_i^T z_i \tag{19}$$

$$\text{subj.to} \sum_{i=1}^{N} H_i z_i = b \tag{20}$$

$$z_i \in P_i, \quad \forall i \in 1, \ldots, N \qquad c_i \in \mathbb{R}^{n_i} \tag{21}$$

$$H_i \in \mathbb{R}^{S \times n_i} \qquad b \in \mathbb{R}^S \tag{22}$$

With P_i being the compact polyhedron described by linear equality and inequality constraints; in practical sense, this polyhedron can be seen as a sensor range of the agent.

$$P_i \triangleq \{z_i \in \mathbb{R}_i^n | D_i z_i \leq d_i, G_i z_i = g_i\} \qquad \forall i \in \{1 \ldots N\} \tag{23}$$

Lagrangian equation of the task

$$L(z, \lambda) = \sum_{i=0}^{N} c_i^T z_i + \sum_{l=0}^{S} \lambda_l (\sum_{i=1}^{N} H_i z_i - b_l) \qquad b \in \mathbb{R}^S \tag{24}$$

The dual function then is:

$$q(\lambda) = \inf_{z \in P} L(z, \lambda) = \inf_{z \in P} \sum_{i=0}^{N} c_i^T z_i + \sum_{l=0}^{S} \lambda_l (\sum_{i=1}^{N} H_i z_i - b_l) \tag{25}$$

Since we are in a distributed scenario the lambda has the following vector form:

$$
\begin{bmatrix} \lambda_1^{t+1} \\ \vdots \\ \lambda_s^{t+1} \end{bmatrix} = \begin{bmatrix} v_1^{t+1} \\ \vdots \\ v_s^{t+1} \end{bmatrix} + \alpha^t \begin{bmatrix} \sum_{i=1}^{N} H_i z_i^{t+1} - b_1 \\ \vdots \\ \sum_{i=1}^{N} H_i z_i^{t+1} - b_s \end{bmatrix} \tag{26}
$$

Where v_i^{t+1} are the aggregation variables produced by weighting the current neighbours' lambdas as previously shown in (4). We can evaluate z_i^{t+1} by means of the following relation obtained by (6):

$$
z_i^{t+1} = \operatorname*{argmin}_{\substack{D_i z_i \leq d_i \\ 0 \leq z_i \leq 1;}} (c_i + v_i^{t+1} H_i) z_i \tag{27}
$$

therefore by applying the results of (16),(17)and(18) we are able to find the optimal value for the decisions variables

3.2   Assumptions for the execution

In order to solve the problem some assumption should be made in order to set the problem and then test it inside MATLAB environment:

- The values of c_i are randomly generated in order to give different weights to agents inside the optimization problem;

- The dimension of the state vector of the agents coincides with the dimension of the number of equality constraints active on the optimization problem (e.g $n_i \equiv S$), this assumption is made for the sake of simplicity of producing M equality constraints, but it can be easily generalized to $S \geq M$;

- $H_i = \mathbb{1}$ and and $b = [1 \dots 1]$ since for the sake of the implementation we want to have the following equality to be satisfied

$$
\sum_{i=1}^{N} z_{ij} = 1 \qquad \forall j \in \{1 \dots M\}
$$

this is made order to cap the sum of the k-th state component of all the vector in prevision of the requests inside Optimal task assignment chapter;

- D_i and d_i are default chosen in order to impose reasonable shape of the polyhedron,instead G_i and g_i are considered to be null, since for sake of simplicity only equality constraints are take in considerations, consequently the polyhedron becomes:

$$
P_i = \{z_i \in \mathbb{R}^{n_i} \mid D_i z_i \leq d_i\} \qquad \forall i \in \{1 \dots N\}
$$

- the UB and LB are respectively chosen to make the state components of the agents stay inside the interval $z_{ij} \in \{0, 1\}$;

In the deployed code we generate the above matrices by means of the function progen, which produces the said matrices by means of the number of agents N and the number of state components of the agents M. Therefore the problem is considered to be defined and can be solved by means of the linear program theory recalled at the beginning of the chapter.

Inside the MATLAB project we made use of the linprog function in order to solve the minimization problem exposed in equation (27) and then, applying the dual sub gradient results, we are able to finally solve our constrained coupled linear problem obtaining the optimal values of the decision variables for each agent.

4 Optimal task assignment

4.1 Problem formulation

In the task 2 there are N robots trying to self assigning N tasks scattered randomly in the environment each of them must be served by at most one robot, the objective is to minimize the total travelled distance of the agents. To do so we employ the the results achieved in the task 1 in order to use the distributed optimization procedure in order to make the agents self assign the task in a distributed optimization procedure.

Define: V_A as the node set, U_A as the task set E_A as the set of edges.

An edge $(i, k) \in E_A$ exists if and only if agent i can be assigned to task k. The cost c_{ik} (cost of agent i to perform task k) is a weight on the edge, calculated by considering the norm of the distance of agent i to the objective k. Objective: solve the linear optimization program:

$$\min_{z_1,\ldots,z_N} \sum_{i=1}^{N} c_i^T z_i \tag{28}$$

$$\text{subj.to} \sum_{k|(i,k)\in E_A} x_{ik} = 1 \qquad \forall i \in \{1 \ldots N\} \tag{29}$$

$$\sum_{i|(i,k)\in E_A} x_{ik} = 1 \qquad \forall i \in \{1 \ldots N\} \tag{30}$$

We can reformulate the problem by defining from (citation):(leggi la chat)

- z_i a vector stacking all $x_{ik} \forall$ k such that (i,k) $\in E_A$

- c_i a vector stacking all $c_{ik} \forall$ k such that (i,k) $\in E_A$

Define the local polyhedron P_i as

$$P_i \triangleq \{x_i | 0 \leq x_i \leq 1, \sum_{k|(i,k)\in E_A} x_{ik} = 1\} \qquad \forall i \in \{1 \ldots N\} \tag{31}$$

Also the matrix H_i is defined suitably by extracting from the $N \times N$ identity matrix the subset of columns corresponding to the tasks agent i can perform.

The b vector is defined as a vector of ones.
The problem becomes:

$$\min_{z_1,\ldots,z_N} \sum_{i=1}^{N} c_i^T z_i \tag{32}$$

$$\text{subj.to} \sum_{k|(i,k)\in E_A} H_i z_i = 1 \qquad \forall i \in \{1\ldots N\} \tag{33}$$

$$with : z_i \in P_i \tag{34}$$

In our project execution, the matrix H is either created by randomly assigning 0 or 1 to each of the entries or such that every agent can execute any task by activating a toggle inside the code that generates the problem.

4.2 Problem solution

The environment where task and robots are spawned is assumed to be a 1 $by1$ square with dimension 2 (default option) or possibly a circle with radius 1. Both tasks and robots are spawned randomly inside this space.

To solve the minimization problem, we make use of the linprog function in MATLAB. This function takes as inputs:
x = linprog(f,A,b,Aeq,beq,lb,ub,options)
with:

$$\min_{x} f^T \text{ such that } \begin{cases} Ax \leq b \\ A_{eq} = b_{eq} \\ lb \leq x \leq ub \end{cases} \tag{35}$$

In this scenario the following choices for the parameters were made:

$$f = c_i + (v(i,:)) * H_i \tag{36}$$
$$A_{eq} = G_i \tag{37}$$
$$b_{eq} = g_i \tag{38}$$
$$lb = LB_i \tag{39}$$
$$ub = UB_i \tag{40}$$

In the end the primal cost is computed as:

$$f_i = c_i * Z(i,:,t)^T \tag{41}$$
$$primCost(t) = primCost(t) + f_i \tag{42}$$
$$\tag{43}$$

The dual cost is:

$$q_i = c_i * Z(i, :, t)^T + \lambda(i, :, t) * (Z(i, :, t) - b_i^T)^T \tag{44}$$

$$dualCost(t) = dualCost(t) + q_i \tag{45}$$

4.3 Simulations and results

5 Conclusions and possible further developments

References

[1] Francesco Bullo. *Lectures on Network Systems*. Kindle Direct Publishing, 2020.