MECH 6V29: Multiagent Robotic Systems- HW 4

Jonas Wagner

2022, April 6^{th}

Contents

Preliminary Notes

a) Definitions

Definition 1. *Graph $G(V, E)$ is constructed with vertex set*

$$V = \{v_1, v_2, \dots, v_n\}$$

of n discrete vertices and edge set

$$E = \{e_1, \dots, e_m\} \subseteq V \times V$$

consisting of m edges $e_{k=(i,j)} = (v_i, v_j) \forall_{k=1,\dots,m}$ connecting vertices v_i and v_j.

Definition 2. *Let $V = \{v_1, v_2, \dots, v_n\}$ be vertices. Δ-Disk Graphs are constructed for a particular Δ such that*

$$(v_i, v_j) \in E \iff \|v_i, v_j\| \leq \Delta$$

Definition 3. *Let $V = \{v_1, v_2, \dots, v_n\}$ be vertices. A Gabriel Graph is defined as $G(V, E)$ in which*

$$\forall_{v_i, v_j \in V} (v_i, v_j) \in E \iff \forall_{v_k \in V} v_k \notin D(v_i, v_j)$$

where $D(a, b)$ is the closed disc with diameter between (a, b). In other words, a disk constructed from two adjacent vertices should not contain any other vertices.

Definition 4. *Let graph $G(V, E)$ with $V = \{v_1, \dots, v_n\}$ and $E \subseteq V \times V$.*

a. $G(V, E)$ is considered undirected if

$$(v_i, v_j) \in E \iff (v_j, v_i) \in E$$

otherwise, $G(V, E)$ is considered directed.

b. An undirected graph $G(V, E)$ is considered connected if there exists a path between any two vertices.

c. An undirected graph $G(V, E)$ is considered planer if the graph can be drawn on a plane without any edges crossing.

Definition 5. *Let graph $G(V, E)$ with $V = \{v_1, \dots, v_n\}$ and $E \subseteq V \times V$.*

a. The degree matrix $\Delta \in \mathbb{R}^{n \times n}$ is a diagonal matrix defined as

$$\Delta := \begin{bmatrix} \deg(v_1) & & & \\ & \deg(v_2) & & \\ & & \ddots & \\ & & & \deg(v_n) \end{bmatrix}$$

b. The adjacency matrix $A \in \mathbb{R}^{n \times n}$ is a symmetric matrix ($A = A^T$) defined s.t.

$$A = [a_{ij}] \ : \ a_{ij} \begin{cases} 1 & (v_i, v_j) \in V \\ 0 & (v_i, v_j) \notin V \end{cases}$$

c. *The incidence matrix* $D \in \mathbb{R}^{n \times m}$ *is defined as*

$$D = [d_{ij}] \; : \; d_{ij} \begin{cases} 1 & (v_i, -) \in e_j \\ -1 & (-, v_i) \in e_j \\ 0 & otherwise \end{cases}$$

d. *The Laplacian matrix* $L \in \mathbb{R}^{n \times n}$ *is a symmetric* ($L = L^T$) *and strictly semi-positive definite* ($L \succeq 0$) *is defined as*

$$L := \Delta - A = DD^T$$

and

$$L = \begin{bmatrix} \deg(v_1) & -a_{12} & -a_{13} & \cdots & -a_{1n} \\ -a_{21} & \deg(v_2) & -a_{23} & \cdots & -a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & -a_{n3} & \cdots & \deg(v_n) \end{bmatrix}$$

e. *For a weighted graph* $G(V, E, W)$, *the diagonal weighted matrix* $W \in \mathbb{R}^{m \times m}$ *is defined as*

$$W = [w_{ij}] \forall_{ij \in E}$$

were w_{ij} *are the corresponding weights for* $e_{ij} = (v_i, v_j)$.

Definition 6. *Consider a collection of* N *robots. Formation graph* $G(V, E_f, \omega)$ *consists of vertex set*

$$V = \{v_1, v_2, \ldots, v_N\}$$

of N *vertices* v_i *associated with robot* i, *edge set* E_f

$$E_f = \{e_1, \ldots, e_m\} \subseteq V \times V$$

of m *edges* $e_{k=(i,j)} = (v_i, v_j)$ *that indicate knowledge of the distance between robots* v_i *and* v_j, *and* $\omega : E_f \to \mathbb{R}_+$ *which associates a feasible desired inter-agent distance to each pair in* E_f.

Definition 7. *Consider a collection of* N *robots connected in formation graph* $G(V, E_f, \omega)$. *Let each robot be located at position* $P_i \in \mathbb{R}^d$ *within euclidean space* $(\mathbb{R}^p, \|\cdot\|_2)$.

a. *The formation position set is defined as the collection of associated robot positions*

$$P = \{P_1, \ldots, P_N\} \subseteq \mathbb{R}^{p \times p}$$

b. *The set of pair-wise inter-robot distances is defined by*

$$D = \{d_{ij} \geq 0 \; : \; d_{ij} = d_{ji}, \forall_{i,j \in i, \ldots, N}\}$$

where d_{ij} *is the distance* $\|P_i - P_j\|$.

c. D *is considered feasible if*

$$\exists_{P_1, \ldots, P_N \in \mathbb{R}^d} \; : \; \|P_i - P_j\| = d_{ij} \forall_{i,j \in \{1, \ldots, N\}}$$

d. *A framework* (G_f, P) *is a combination of a formation graph* G_f *and a set of feasible points* P.

e. *Framework (G_f, P) is considered generic if P is algebraically independent over $\mathbb{Q}$. (i.e. that the points are not collinear in 2-D or co-planer in 3-D)*

Definition 8. *Consider frameworks (G, P_0) and (G, P_1).*

a. *(G, P_0) and (G, P_1) are equivalent if*

$$\|P_0(i) - P_0(j)\| = \|P_1(i) - P_1(j)\| \forall_{(i,j)\in E_f}$$

meaning $d_{ij}^{(0)} = d_{ij}^{(1)}$ for the vertices that are neighbors.

b. *(G, P_0) and (G, P_1) are congruent if*

$$\|P_0(i) - P_0(j)\| = \|P_1(i) - P_1(j)\| \forall_{(i,j)\in V\times V}$$

meaning $d_{ij}^{(0)} = d_{ij}^{(1)}$ for every vertex.

c. *(G, P_0) is Globally Rigid if*

$$(G, P_1) \text{ equivalent } (G, P_0) \implies (G, P_1) \text{ congruent } (G, P_0)$$

d. *(G, P_0) is rigid if*

$$\exists_{\epsilon>0}\forall_{P_1} : (G, P_1) \text{ equivalent } (G, P_0) \wedge \forall_{i\in V}\|P_0(i) - P_1(i)\| < \epsilon \implies (G, P_1) \text{ congruent } (G, P_1)$$

Remark: *A framework being rigidity is equivalent to saying that every continuous motion maintaining distances where edges exist also maintains the distances between all other vertex pairs.*

Definition 9. *Let (G, P_0) be a d-dimensional generic framework.*

a. *The Rigidity Matrix is defined by the equations*

$$(x_i - x_j)^T(\dot{x}_i - \dot{x}_j) = 0 \ \forall_{i,j\in E}$$

resulting in a matrix $R(P_0)$ so that $R(P_0)\dot{P} = 0$.

b. ***Rigidity Test:*** *(G, P_0) is rigid if and only if*

$$\operatorname{rank}(R(P_0)) = \begin{cases} 2N-3 & d=2 \\ 3N-6 & d=3 \end{cases}$$

Additionally, the rank of the rigidity matrix remains the same for all generic realizations, thus G can be called Generically Rigid if any feasible generic realization is rigid.

Definition 10. *G_f is considered minimally rigid if it is rigid and the removal of any single edge renders it not rigid.*

Additionally, G_f is minimally rigid if and only if it is rigid and contains

$$\begin{cases} 2N-3 \textit{ edges} & d=2 \\ 3N-6 \textit{ edges} & d=3 \end{cases}$$

Problem 1

State a summary of **Notes 13 - 15**, (which Include the topics of persistence, combinatorial coverage and graph grammars) preferably by creating a concept map diagram (flow diagram). The whole purpose is to make sure that we are clear about the bigger picture, and reiterate why are we doing and discussing the specific topics in the class. Do not merely write the topics, instead create connections between topics to clarify the flow of information.

a) Big Picture Chart

TO DO: Update all the graph to this time...

Problem 2

Problem

We studied a way to assign edge directions in a minimally rigid graph to make them constraint consistent, and hence, minimally persistent. Assign orientations to edges in the following graphs to make them persistent. Show all the steps of the Henneberg construction.

Preliminaries

Definition 11.

Definition 12. *Consider directed graph G_F.*

a. G_F is Rigid if certain interagent distances are maintained then all interagent distances are maintained when the formation moves smoothly.

b. G_F is Constraint Consistent if the directed graph is able to maintain the specified interagent distances.

c. G_F is considered persistent if and only if G_F is Rigid and Constraint Consistent.

Solution

a) G_1

Problem 3

Preliminaries

Problem 4

See attached MATLAB file.

Problem 5

Preliminaries

Refering back to Definitions 4 and 3, we have that

a. An undirected graph $G(V, E)$ is considered connected if there exists a path between any two vertices.

b. An undirected graph $G(V, E)$ is considered planer if the graph can be drawn on a plane without any edges crossing.

c. A Gabriel Graph is defined as $G(V, E)$ in which

$$\forall_{v_i, v_j \in V} (v_i, v_j) \in E \iff \forall_{v_k \in V} v_k \notin D(v_i, v_j)$$

where $D(a, b)$ is the closed disc with diameter between (a, b). In other words, a disk constructed from two adjacent vertices should not contain any other vertices.

a) Show that Gabriel graphs are planar

Theorem 1. *All Gabriel graphs are planar.*

Proof. Let $G(V, E)$ be a Gabriel graph (in 2-dimensions). By definition of gabriel graphs, an edge only exists between two vertices if and only if no vertices are closer to the midpoint between those vertices. i.e.

$$\forall_{(v_i, v_j) \in E} \; d(m_{v_i, v_j}, v_i) = d(m_{v_i, v_j}, v_j) < d(m_{v_i, v_j}, v_k) \; \forall_{v_k \in V \neq v_i, v_j}$$

where m_{v_i, v_j} is the midpoint between v_i and v_j while $d(x, y)$ is the euclidean distance between x and y. Each of these edges can therefore be drawn on the plane without crossing as each edge will be drawn from vertices through the midpoint and since no other vertex is near the midpoint, no other edge could be constructed that would intersect it. $\square$

b) Show that Gabriel graphs are connected

Theorem 2. *All Gabriel graphs are connected.*

Proof. Let $G(V, E)$ be a Gabriel graph (in 2-dimensions). Consider all vertex pairs v_i and v_j. If there are no vertices within the disk induced by v_i and v_j, then v_i and v_j are adjacent and therefore a path exists between them. If this is not the case, then it must be true that $\exists_{v_{k_1} \neq v_j \in V}$ satisfying this condition for v_i. Additionally, it is also true that $\exists_{v_{k_n} \neq v_i \in V}$ satisfying the condition for v_j. If $v_k = v_{k_1} = v_{k_n}$, then there exists a path $[(v_i, v_k), (v_k, v_j)]$ connected v_i and v_j. However, if $v_{k_1} \neq v_{k_n}$, we can perform the same test again with v_{k_n} instead. Since we know that $\exists_{v_{k_{n+1}} \neq v_i \neq v_j \neq v_{k_1} \ldots \neq v_n \in V}$ that satisfies the adjacency condition. Eventually, for a finite (perhaps countable) set of vertices, where a path $[(v_i, v_{k_1}), (v_{k_1}, v_{n+1}), \ldots (v_{k_2}, v_j)]$ will be formed for $n < N - 2$. Since this is true $\forall_{v_i, v_j \in V}$ the graph is connected. $\square$

Problem 6

Graph grammars are a good tool for generating networks with some specific structure. Assume that we have a total of 2^n nodes for some positive integer n. Consider the following rules:

$$
\begin{aligned}
&R_0 : a \quad a \rightarrow l_1 \text{--} c \\
&R_1 : l_i \quad l_i \rightarrow l_{i+1} \text{--} c \quad 1 \leq i \leq n-1
\end{aligned}
$$

What structure do we get in the end?

Preliminaries

Solution

Problem 7

A MATLAB Code:

All code I write in this course can be found on my GitHub repository:
https://github.com/jonaswagner2826/MECH6V29_MultiagentRoboticSystems

MECH 6V29 - HW 4

Author: Jonas Wagner

Date: 2022-03-22

```
clear
clc
close all
```

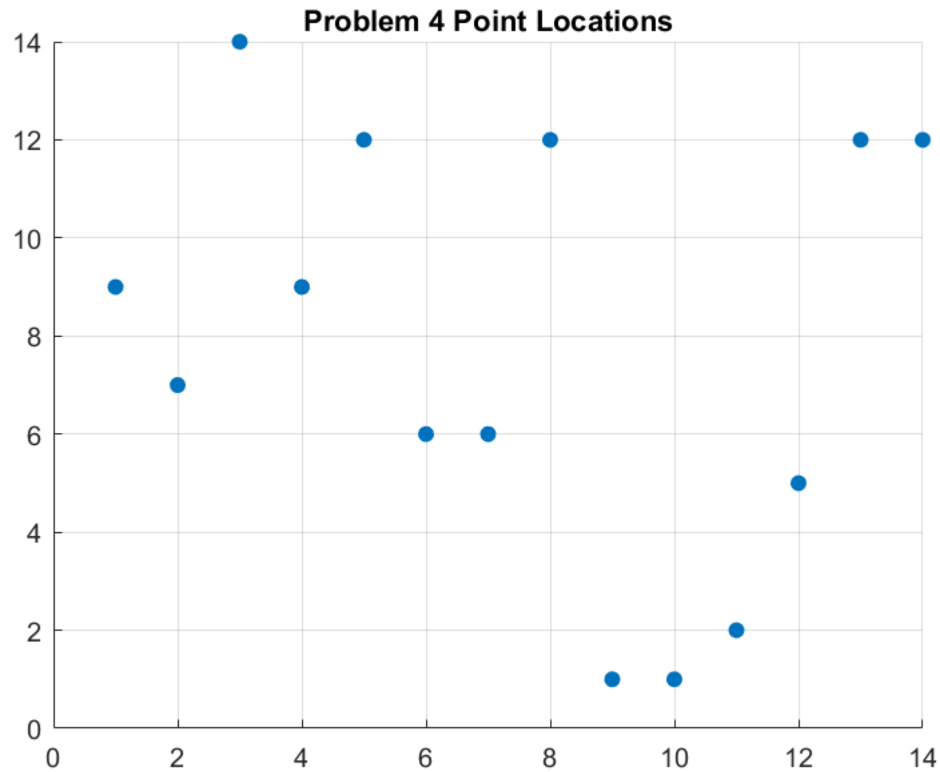
Problem 4

Draw a Gabriel graph induced by the following set of 14 points.

```
P = [
    1 2 3 4 5 6 7 8 9 10 11 12 13 14
    9 7 14 9 12 6 6 12 1 1 2 5 12 12
    ];
X = P(1,:)';
Y = P(2,:)';
```

Plot Point Locations

```
figure();
scatter(X, Y, 'filled')
title('Problem 4 Point Locations')
grid on
saveas(gcf, '.\fig\pblm4_points.png')
```




```
V = num2cell([X Y],2)
```

V = 14×1 cell

	1
1	[1,9]
2	[2,7]
3	[3,14]
4	[4,9]
5	[5,12]
6	[6,6]
7	[7,6]
8	[8,12]
9	[9,1]
10	[10,1]
11	[11,2]
12	[12,5]
13	[13,12]
14	[14,12]

Determine Gabriel Graph

```
idx_edge = 1;
tol_gabriel = 0.01;
for i = 1:size(V,1)
    for j = 1:size(V,1)
```

Generate Midpoints

```
        M{i,j}(1) = (X(i) + X(j))/2;
        M{i,j}(2) = (Y(i) + Y(j))/2;
        M_dist{i,j} = norm(M{i,j} - V{i});
```

Determine Edges

```
        if i ~= j
            for k = 1: size(V,1)
                if k == i || k == j
                    in_circle(i,j,k) = 0;
                else
                    in_circle(i,j,k) = norm(M{i,j} - V{k}) < M_dist{i,j} + tol_gabriel;
                end
            end
            if any(in_circle(i,j,:) == 1,'all')
                continue
            else
                E{idx_edge} = [i, j];
                idx_edge = idx_edge + 1;
                continue
            end
        end
    end
end
```

Gabriel Graph Edges

```
E = E'
```

E = 40×1 cell

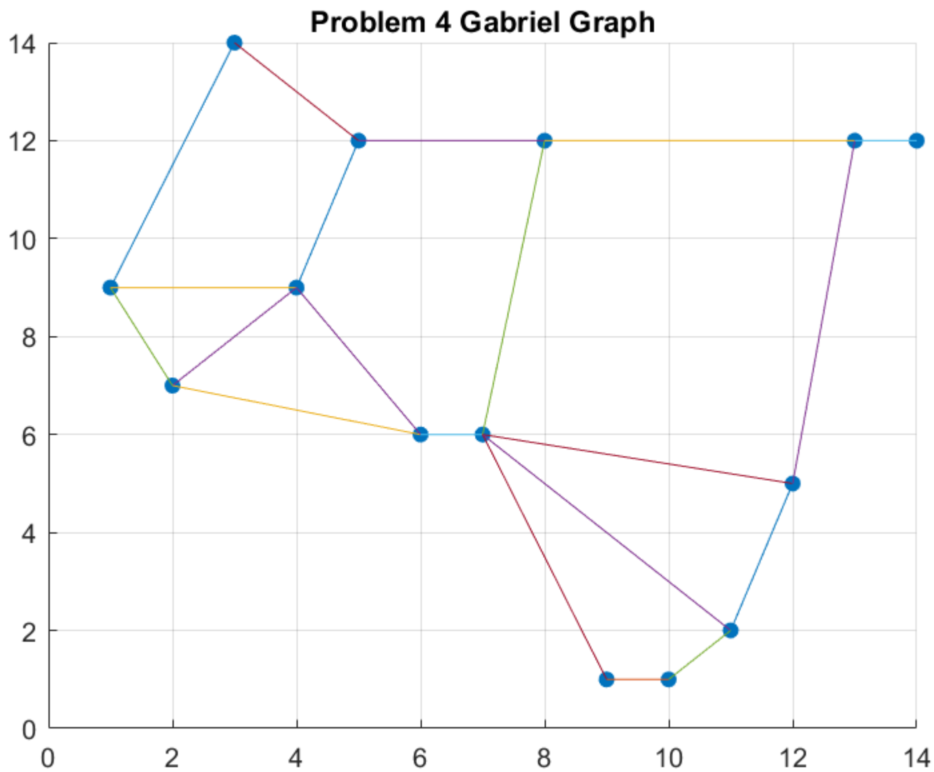
	1
1	[1,2]
2	[1,3]
3	[1,4]
4	[2,1]
5	[2,4]
6	[2,6]
7	[3,1]
8	[3,5]

	1
9	[4,1]
10	[4,2]
11	[4,5]
12	[4,6]
13	[5,3]
14	[5,4]

⋮

Plot Graph

```
fig_pblm4_resutls = plot_graph(V, E);
saveas(fig_pblm4_resutls, '.\fig\pblm4_results.png')
```



Export to PDF automatically

```
export("MECH6V29_HW04")
```