Lecture 6-2

Network Synthesis

Quote of the day

... Confusion between creativity & originality. Being original entails saying something that nobody has said before. Originality ... must be exhibited, or feigned, for academic advancement. Creativity, by contrast, reflects the inner experience of the individual overcoming a challenge. Creativity is not diminished when one achieves ... what has already been discovered

Shalom Carmy

http://
www.connectedacti
on.net/wp-content/
uploads/
2011/10/20111010-
NodeXL-Facebook-
Marc-Smith.png

Slime Mold Network

https://www.google.com/url?sa=t&source=web&cd=1&ved=0ahUKEwjIztC9ptbaAhUSwGMKHd9vA4MQo7QBCAkwAA&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DGwKuFREOgmo&usg=AOvVaw0JcNMdKecPZ98XdLGapeBu

So far we have examined certain classes of dynamics on networks, abstracted as graphs. The story line is that we are have nodes with "dynamic" states $x_i(t)$ that evolve in along the following lines:

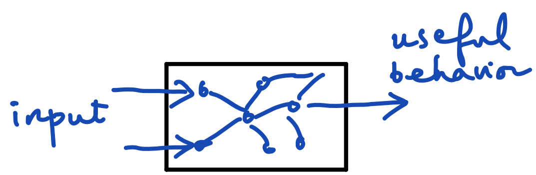
$$\dot{x}_i = f_i(x_i) + \text{coupling terms to neighboring nodes}$$

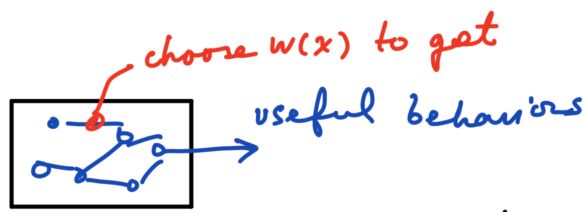
↳ depending on applications this can assume many different forms

e.g.,

$$\dot{x}_i = \sum_{j \sim i} (x_j - x_i)$$

Of course we can choose f_i's & the coupling terms in such a way that we end up with very complicated dynamics. We also looked at two canonical means of modifying that natural behavior of networked systems, namely through "inputs" & "state-dependent" couplings.





of course in practice the networks are subject to environment forces/disturbances, only some state of the network is of interest, there might be multiple decision makers, etc.

In the meantime, a natural question arise:

How did the network structure arise in the first place?

If we call the Network structure as "form"
The state evolution of nodes as "function"
then what we have done so far is to explore how

form $\rightsquigarrow$ function

or

form + input $\rightsquigarrow$ function

form + manipulation of form $\rightsquigarrow$ function

It might very well be that form itself is a dynamic process, something like

$$G(0),\ G(1),\ \dots,\ G(t),\ \dots$$

that somehow couple to

$$x(0),\ x(1),\ \dots,\ x(t),\ \dots$$

The evolution of the two can even be of a feedback form.

Today we want to explore some "models" for how network structure can arise or synthesized ... This is an active research topic & plenty of things still need to be worked out ... I will give you a few examples

Network design using Semi-definite programs: Suppose we are interested to maximize the algebraic connectivity of the graph, but we don't want to use too many edges. So we write the following optimization:

$$\max \quad \lambda_2(G)$$

$$|E(G)| \le m$$

we don't want to use more than m-edges

$$|V(G)| = n$$

this is okay for small n & m.

Note that if we want to max $\lambda_2(G)$ & $|E(G)| \le m$, we can just let $|E(G)| = m$ ∵ since as you add edges to G, $\lambda_2(G+e) \ge \lambda_2(G)$.

How many potential pairs in an n-node graph? $\binom{n}{2} = \frac{n!}{2!\,(n-2)!} = \underbrace{\frac{n(n-1)}{2}}_{O(n^2)}$

& then we have to choose m-edges among $\frac{n(n-1)}{2}$ possible choices:

$$\binom{\frac{n(n-1)}{2}}{m} = \frac{\frac{n(n-1)}{2}!}{m!\,(\underbrace{\frac{n(n-1)}{2}}_{\tilde{n}} - m)!}$$

to get an idea how large this is we can use the Stirling's formula.

$$\approx \frac{\tilde{n}^m}{m!}$$

Okay - so back to our set up:

$$\left.\begin{array}{c} \max \lambda_2(G) \\ |V(G)|=n \\ |E(G)|\leq m \end{array}\right\}$$

we need to introduce variables like

$$\max \lambda_2 \underbrace{\left\{ \sum_i x_i e_i e_i^T \right\}}_{f(x)}$$

$$\sum x_i \leq m \qquad \underbrace{x_i = 0 \text{ or } 1}$$

$$x_i(1-x_i)=0$$

$$i=1,2,\dots,\binom{n}{2}$$

So we can write this problem as

$$\max f(x)$$

$x \in \Omega$ = the set of feasible x's

feasible set

In this setting Ω is non-convex & complicated to work with.

The main issue is with the $x_i = 0$ or $x_i = 1$ constraint that characterizes the feasible set ... what if we relax the constraint to $0 \leq x_i \leq 1$? This turns out to be much nicer to work with ~ this constraint along with $\sum x_i = m$ or

$$\Omega_R = \{x \in \mathbb{R}^{\binom{n}{2}} \mid \mathbb{1}^T x = m,\ 0 \leq x \leq \mathbb{1}\}$$

defines a convex set.

So the relaxed version of the problem looks like

$$\max_{x \in \Omega_R} \lambda_2(G(x))$$

} convex set

But how can we deal will the objective ... we assume that we know that maximization of a concave function over a convex set is easy ... but how should we handle this? May be something like this

$$\begin{aligned} \max \quad & \alpha \\ & x \in \Omega_R \\ & \lambda_2(G(x)) \geq \alpha \end{aligned}$$

if $\lambda_2(G) \geq \alpha \Rightarrow \min\limits_{\substack{x \perp \mathbb{1} \\ x \neq 0}} \dfrac{x^T \mathcal{L}(G)\, x}{x^T x} \geq \alpha$

$\Rightarrow \min\limits_{\substack{x \perp \mathbb{1} \\ x \neq 0}} x^T \mathcal{L}(G)\, x \geq \alpha\, x^T x \Rightarrow \forall\ x \perp \mathbb{1} \quad x \neq 0$

$$x^T \{ \mathcal{L}(G) - \alpha I \} x \geq 0$$

if $x \perp \mathbb{1} \Rightarrow x = \sum \beta_i P_i$ where $P_1, P_2, \ldots, P_{n-1} \perp \mathbb{1}$

$= [P_1, \ldots, P_{n-1}]\, \beta$

So we write $\forall \beta, \beta \neq 0$

$$\beta^T \underbrace{[p_1, \dots, p_{n-1}]^T}_{P^T} \{ \mathcal{L}(G) - \alpha I \} [p_1, \dots, p_n] \beta \geq 0$$

$$\Rightarrow \beta^T P^T \{ \mathcal{L}(G) - \alpha I \} P \beta \geq 0$$

$$\Rightarrow P^T \mathcal{L}(G) P - \alpha P^T P$$ is positive semi-definite

we denote this by "curly" $\geq$ or $\succcurlyeq$

So our problem now looks like

$$\max_{x \in \Omega_R} \alpha$$

SDP inequality

$$\underbrace{P^T \mathcal{L}(G(x)) P}_{\hookrightarrow \text{ linear in } x} - \alpha P^T P \succcurlyeq 0$$

linear in α

where

$$\mathcal{L}(G(x)) = \sum x_i e_i e_i^T$$

This problem is called semi-definite programming & it is one of the most awesome optimization problems for the past few decades. Basically, although this optimization problem is over matrices, we can still solve it rather efficiently ...

in this case we have $\bar{n}+1$ variables & $\sim$ constraints

$\binom{n}{2}$ for "x" for α

solvable in time proportional to polynomial in $\bar{n}$!

Nash Networks

We now shift our attention to another aspect of network synthesis ...
local information for decision-making w/ a global objective.
This setup also allows us to discuss another "solution" concept -
Not of a maximizer or a minimizer, but that of a Nash equilibrium.
We will mainly focus on the solution concept in this part, as opposed
to algorithm for finding the solution - like what we did for the SDP
approach to maximizing $\lambda_2(G)$ ... A solution concept is essentially
a mathematical way of characterizing when we have satisfied
conditions where the decision making process can safely terminate.
For example in the so-called gradient descent for minimizing
a differentiable convex function, i.e.,

$$x_{k+1} = x_k - \gamma \nabla f(x_k)$$

once we reach $\nabla f(\bar{x}) = 0 \Rightarrow x_{k+1} = x_k$ & we ain't going nowhere
afterwards; having $\nabla f(x) = 0$ is a solution concept. Here
is another one:

Prisoner's Dilemma:

Prisoner 1 \ Prisoner 2	Q	C
Q	(-2, -2)	(-5, 0)
C	(0, -5)	(-4, -4)

if your prisoner 1 & you have decided to confess, there is no incentive for you to change your decision to Q unilaterally ...

More generally a pair of strategies (σ_1^*, σ_2^*) for the two decision makers is called a Nash equilibria if

$$\pi(\sigma_1^*, \sigma_2^*) \geqslant \pi(\sigma_1, \sigma_2^*) \quad \forall\ \sigma_1 \in \Sigma_1$$

$$\& \quad \pi(\sigma_1^*, \sigma_2^*) \geqslant \pi(\sigma_1^*, \sigma_2) \quad \forall\ \sigma_2 \in \Sigma_2$$

payoff

set of strategies available to player 2

This solution concept of course can be extended to multi-decision makers, i.e.,

$$\pi(\sigma_1^*, \sigma_2^*, \ldots, \sigma_n^*) \geqslant \pi(\sigma_1, \sigma_2^*, \ldots, \sigma_n^*) \quad \forall\ \sigma_1 \in \Sigma_1$$

& so on ...

So what this have to do anything with Network synthesis? let us consider the following scenario... each node in the network wants to be well connected to the rest of the network, in that he/she wants to minimize

$$\sum_{\substack{j \in V(G) \\ j \neq i}} dist(i,j)$$

local connection game.

but the node doesn't want to pay for it too much, in fact the cost for node i will be $\alpha\, d(i)$; so we let to total cost for node i as

$$\alpha\, d(i) + \sum_{j \neq i} dist(i,j)$$

& the social cost will be

$d(i,j) = \infty$ if $\nexists$ path from i to j

$$\alpha\, |E| + \sum_{i \neq j} dist(i,j)$$

We want to understand how "α" effects the solution concept & how suboptimal a Nash equilibria is.

Here is our first observation:

Prop: If $\alpha \geq 2$ then star is socially optimal. On the other hand, if $\alpha < 2$, the complete graph is social optimal.

Proof: Suppose we have m-edges in the socially optimal network connecting 2m vertices.

social cost $\hookrightarrow \alpha m + 2m + \underset{\text{at least}}{2}\left(n(n-1) - 2m\right)$

lower bound $= \alpha m + 2m + 2n(n-1) - 4m = (\alpha - 2)m + 2n(n-1)$

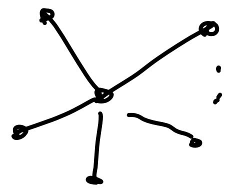
if $\alpha \geq 2 \rightsquigarrow$ min has to be a tree

$\alpha < 2 \rightarrow$ complete graph

Prop: If $\alpha \geq 1$ then any star graph is a Nash equilibrium.

If $\alpha \leq 1$ then the complete graph is a Nash equilibrium.

Proof: Let $\alpha \geq 1$ &

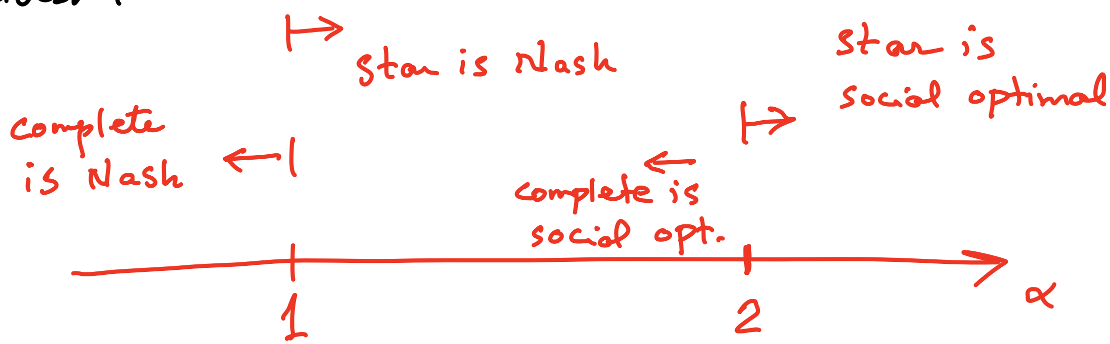


think of yourself as the center node.
you have no incentive to disconnect!

If you are a leaf node, you can add k-edges at a price of αk & distances drop by k. if $\alpha \geq 1$ this doesn't make sense!

Let $\alpha \leq 1$ & complete graph. An agent can save αk by removing k edges, but total distance goes up by k ... again if $\alpha \leq 1$ this doesn't make sense.

Summary:



Let us consider "costs" in the context of decision-making. Two definitions:

$$\underbrace{\frac{\text{cost of the best NE}}{\text{social optimal cost}}}_{\text{price of stability}}$$

Nash equilibria

$$\underbrace{\frac{\text{cost of any NE}}{\text{social optimal cost}}}_{\text{price of anarchy}}$$

if $PA = 1$ then Nash is also socially optimal;

in general $PA \geqslant 1$.

Observation: when $1<\alpha<2$, the price of anarchy is at most $4/3$.

$$c(G) \geq \alpha m + 2m + 2\left(n(n-1) - 2m\right) = 2n(n-1) + m(\alpha-2)$$

social cost of a graph w/ m edges

$\rightsquigarrow$ when $1<\alpha<2$ $\rightsquigarrow$ m should be maximum!

$$\rightsquigarrow \text{price of anarchy} = \frac{c(\text{star})}{c(\text{complete})} = \frac{(n-1)(\alpha-2+2n)}{n(n-1)\left(\frac{\alpha-2}{2}+2\right)}$$

$$= \frac{4}{2+\alpha} - \frac{4-2\alpha}{n(2+\alpha)} < \frac{4}{2+\alpha} < 4/3.$$

In fact one can show that price of anarchy is at most $O(\sqrt{\alpha})$

$$\text{Price of anarchy} = \frac{\text{Nash cost}}{\text{Social cost}}.$$

If you want more on this topic...

Let me know in class!