15-382 Collective Intelligence

Salman Hajizada

First Edition

Disclaimer

This document aims to summarize the content of the slides for 15-382, including what the author considers important. As always, the definition of important is highly subjective so the author might have omitted something that was important to another person, or included something that is trivial to another.

Good luck,
SH

Dynamical Systems

Fingerprints of Complex Systems

- Multi-agent / multi-component
- Decentralized
- Local interactions
- Dynamic

Types of Abstract Models:

- **Agent-based:** mechanistic implemtation of the multi-compenent interactions
- **Mathematical (white-box)**: formally descibe the relations among the relevant components.
- **Black-box:**: Input-output pairs from the system are used to predict the output for a given input, or to tweak the internal parameters to get the desired output, no attempt made to understand how the system works on the inside
- **Statistical:** describing patterns and correlations betwen variables

Systems of ODEs

Here is a system of $n \geq 1$ Ordinary Differential Equations

$$\begin{aligned} \frac{dx_1}{dt} &= f_1(\mathbf{x}(t)) \\ \frac{dx_2}{dt} &= f_2(\mathbf{x}(t)) \\ &\vdots \\ \frac{dx_n}{dt} &= f_n(\mathbf{x}(t)) \end{aligned}$$

where $\mathbf{x}(t)$ is an n-dim vector.
A continuous-time Dynamical System is defined by a system of differential equations:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}, t; \theta)$$

where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a vector field. If $\mathbf{f}$ does not explicitly depend on t, it is an **Autonomous System**: $\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x})$.

Linear Systems ($\mathbf{f}(\mathbf{x}) = A\mathbf{x}$)

The simplest case, where $\mathbf{f}$ is linear:

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x}(t), \quad \mathbf{x} \in \mathbb{R}^n, \quad A \text{ is an } n \times n \text{ coefficient matrix}$$

General Solution of Linear ODEs

The general solution is a linear combination of exponentials:

$$\mathbf{x}(t) = c_1 e^{\lambda_1 t}\mathbf{u}_1 + c_2 e^{\lambda_2 t}\mathbf{u}_2 + \cdots + c_n e^{\lambda_n t}\mathbf{u}_n$$

where λ_i is an **eigenvalue** and $\mathbf{u}_i$ is the corresponding **eigenvector** of matrix A.
Alternatively, using the Matrix Exponential:

$$\mathbf{x}(t) = e^{At}\mathbf{x}(0)$$

Critical Points (Equilibrium Points)

A critical point $\mathbf{x}_{\text{eq}}$ is a constant solution where the system's velocity is zero:

$$\mathbf{f}(\mathbf{x}_{\text{eq}}) = \mathbf{0} \implies \left.\frac{d\mathbf{x}}{dt}\right|_{\mathbf{x}=\mathbf{x}_{\text{eq}}} = \mathbf{0}$$

For a linear system $A\mathbf{x} = \mathbf{0}$, the origin $\mathbf{x}_{\text{eq}} = \mathbf{0}$ is always a critical point if $\det(A) \neq 0$ (non-singular matrix).

Stability Analysis for 2D Linear Systems

The stability and type of critical point at the origin are determined entirely by the **eigenvalues** of A.

1. Real Eigenvalues ($r_1 \neq r_2$)

- **Node** if r_1, r_2 have the same sign.
  - **Stable Node** if $r_1, r_2 < 0$. $\mathbf{x}(t) \to \mathbf{0}$ as $t \to \infty$.
  - **Unstable Node** if $r_1, r_2 > 0$. $\mathbf{x}(t) \to \infty$ as $t \to \infty$.
- **Saddle Point** if r_1, r_2 have opposite signs ($r_1 < 0 < r_2$ or $r_2 < 0 < r_1$).
  - **Unstable**. Trajectories approach $\mathbf{0}$ along the stable eigenvector and diverge along the unstable eigenvector.

2. Repeated Real Eigenvalues ($r_1 = r_2 = r$)

- **Stable Proper/Improper Node or Star** if $r < 0$.
- **Unstable Proper/Improper Node or Star** if $r > 0$.

3. Complex Eigenvalues with Nonzero Real Part (Spiral / Focus)

$$\lambda_{1,2} = a \pm bi$$

Behavior:

- **Stable Spiral** if $a < 0$. Trajectories spiral inward, approaching $\mathbf{0}$.
- **Unstable Spiral** if $a > 0$. Trajectories spiral outward, diverging from $\mathbf{0}$.

4. Pure Imaginary Eigenvalues (Center)

$$\lambda_{1,2} = \pm i\omega$$

Behavior:

- Trajectories are closed orbits (circles or ellipses).
- Neither converge nor diverge — **neutrally stable**.

Non-Linear Systems $\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x})$

Challenge: A closed-form solution for non-linear systems is often very difficult or unfeasible to derive.

Local Stability Analysis via Linearization

To analyze a non-linear system around an equilibrium point $\mathbf{x}_{\text{eq}}$, we use **Linearization**:

1. **Find Equilibrium Points** ($\mathbf{x}_{\text{eq}}$) by solving $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.
2. **Compute the Jacobian Matrix** $J(\mathbf{x})$:
$$J_{ij}(\mathbf{x}) = \frac{\partial f_i}{\partial x_j}$$
3. **Evaluate the Jacobian** at the equilibrium point, $J(\mathbf{x}_{\text{eq}})$.
4. **Find the Eigenvalues** of $J(\mathbf{x}_{\text{eq}})$ and use the linear stability rules to determine the local behavior (Node, Saddle, Spiral, Center).

- **Center Manifold Theorem**: If the linearization results in a **Center** (pure imaginary eigenvalues), the linearization is **inconclusive** because the stability depends on higher-order, non-linear terms.

Bifurcations

A **bifurcation** is a qualitative change in the topological structure of a system's flow as a parameter is varied.

- **Saddle-Node (Fold) Bifurcation:** Two equilibrium points collide (one stable, one unstable) and annihilate, or are created from nothing.
- **Transcritical Bifurcation:** Two equilibrium points exist, and they swap their stability as they pass through each other.
- **Pitchfork Bifurcation:** A single equilibrium point splits into three (two stable/unstable, one saddle), or vice versa.
- **Hopf Bifurcation:** An equilibrium point changes from a stable to an unstable spiral, leading to the creation of a stable, self-sustaining oscillation called a **limit cycle**.

Chaos and Strange Attractors

- **Chaos:** Bounded, non-periodic behavior in a deterministic system that exhibits **Sensitive Dependence on Initial Conditions (SDIC)**.
- **SDIC:** Means that arbitrarily small differences in initial conditions lead to exponentially diverging trajectories over time.
- **Strange Attractor:** An attractor that is **fractal** (has a non-integer dimension) and exhibits chaotic dynamics.

Fractal Dimension (e.g., Box-counting dimension):

$$D = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)}$$

where $N(\epsilon)$ is the minimum number of boxes of side ϵ needed to cover the object.

Discrete Dynamical Systems (Iterated Maps) & CAs

Discrete Dynamical Systems (Iterated Maps)

The general form of a discrete dynamical system (iterated map) is:

$$\mathbf{x}_n = \mathbf{f}(\mathbf{x}_{n-1})$$

For a 1D map: $x_n = f(x_{n-1})$.

Fixed Points and Stability

- **Fixed Point** ($\mathbf{x}^*$): $\mathbf{x}^* = \mathbf{f}(\mathbf{x}^*)$
- **1D Stability:** Determined by the derivative:
  - $|f'(x^*)| < 1$: **Stable** (attracting).
  - $|f'(x^*)| > 1$: **Unstable** (repelling).
- **Higher-D Stability:** Determined by **Jacobian eigenvalues** (λ_i):
  - **Stable** if $|\lambda_i| < 1$ for **all** eigenvalues.

The Logistic Map and Chaos

The Logistic Map: $x_{n+1} = rx_n(1 - x_n)$.

- Exhibits **period-doubling bifurcations** leading to **chaos** as parameter r increases.

Lyapunov Exponent (λ): Long-term average exponential rate of separation between nearby trajectories.

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln |f'(x_i)|$$

- $\lambda < 0$: Stable.
- $\lambda > 0$: **Chaotic** (SDIC is present).

Cellular Automata (CAs)

A discrete dynamical system that evolves in **discrete space** and **discrete time**.

- **Local Transition Rule (f):** $s_i(t+1) = f(\{s_j(t) : j \in N_i\})$, where N_i is the neighborhood of cell i.
- **Elementary CAs:** 1D, binary state, radius $r = 1$ ($2^8 = 256$ rules).

Wolfram's Classification

- **Class I (Fixed Point):** Converge quickly to a uniform, stable state.
- **Class II (Periodic):** Converge to simple, separated periodic structures.
- **Class III (Chaotic):** Generate pseudo-random, chaotic patterns (e.g., Rule 30).
- **Class IV (Complex):** Produce complex, localized structures (particles) capable of **universal computation** (e.g., Rule 110, Game of Life).

Conway's Game of Life (2D CA)

- **Survival (Live cell):** 2 or 3 live neighbors.
- **Birth (Dead cell):** Exactly 3 live neighbors.

Networks

Key Network Properties

- **Degree (k):** Number of connections a node has.
- **Degree Distribution ($P(k)$):** Probability that a randomly selected node has degree k.
- **Path Length (h):** Number of edges in the shortest path between two nodes. **Average Path Length** ($\bar{h}$).
- **Clustering Coefficient (C):** Measures the likelihood that a node's neighbors are also neighbors of each other. $C_i = \frac{\text{Number of edges between neighbors of } i}{\text{Max possible edges between neighbors of } i}$.

Network Models

- **Erdős–Rényi (ER) Random Graph:**
  - **Properties:** Poisson $P(k)$ (homogeneous), very low C (≈ 0), **small world** (short $\bar{h}$).
  - **Message:** Too random to explain real-world networks.
- **Watts–Strogatz (WS) Small World:**
  - **Mechanism:** Starts as a regular lattice, rewires edges with probability p.
  - **Properties:** High C + short $\bar{h}$ (**small-world effect**).
  - **Weakness:** Still has a homogeneous degree distribution (no hubs).
- **Barabási–Albert (BA) Scale-Free:**
  - **Mechanism: Growth** (new nodes added) + **Preferential Attachment** (new nodes connect preferentially to existing nodes with high degree).
  - **Properties: Power Law** degree distribution ($P(k) \sim k^{-\gamma}$), formation of **hubs**.
  - **Message:** Growth + Preferential Attachment are key drivers of real networks.

Network Centrality Measures

Measures for quantifying a node's importance/power.

- **Degree Centrality (C_D):** The number of direct links a node has. $C_D(i) = k_i$.
- **Closeness Centrality (C_C):** How close a node is to all other nodes.

$$C_C(i) = \frac{N-1}{\sum_{j=1}^{N} d(i,j)} \quad \text{(Inverse of average distance)}$$

- **Betweenness Centrality (C_B):** How often a node lies on the shortest path between other pairs of nodes (broker/gatekeeper role).
- **Eigenvector Centrality (C_E):** Importance proportional to the importance of its neighbors (a node is important if its neighbors are also important).

$$\lambda \mathbf{c} = A\mathbf{c} \quad (\mathbf{c} \text{ is the eigenvector of } A \text{ corresponding to the largest eigenvalue } \lambda)$$

- **PageRank:** The probability that a random web-surfer will land on a page (steady-state probability).

$$PR(A) = \frac{1-d}{N} + d \sum_{i \in B_A} \frac{PR(i)}{L(i)}$$

d: damping factor (probability of following a link), B_A: set of pages linking to A, $L(i)$: number of out-links from page i.

Swarm Intelligence and Optimization

Stigmergy

- **Stigmergy:** Any form of **indirect communication** among distributed agents through acts of local modification of the environment and local sensing of the outcomes (e.g., pheromones).
- Best analogy: Blackboard/Post-it style asynchronous communication.

Particle Swarm Optimization (PSO)

Bio-inspired, collective intelligence meta-heuristic for continuous, multimodal optimization.

PSO Algorithm Core

Each particle i has: **Position** ($\mathbf{x}_i$), **Velocity** ($\mathbf{v}_i$), **Personal Best** ($\mathbf{pBest}_i$), and **Global Best** ($\mathbf{gBest}$).
Velocity Update Equation:

$$\mathbf{v}_i(t+1) = \underbrace{\omega \mathbf{v}_i(t)}_{\text{Inertia}} + \underbrace{c_1 r_1 (\mathbf{pBest}_i - \mathbf{x}_i(t))}_{\text{Cognitive (Own Experience)}} + \underbrace{c_2 r_2 (\mathbf{gBest} - \mathbf{x}_i(t))}_{\text{Social (Swarm Experience)}}$$

- ω: Inertia weight (influences balance between exploration/exploitation).
- c_1, c_2: Acceleration coefficients.
- r_1, r_2: Uniform random numbers in $[0, 1]$.

Position Update Equation:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

Hybrid PSO (HPSO)

- **Concept:** Combines cooperative PSO exploration with a highly specialized **Local Search (LS)** intensification.
- **Process:** Run PSO for global exploration, then use LS (e.g., gradient descent) on the **gBest** and a few top **pBest** particles to exploit local minima, then share the improved results back to the swarm.

Ant Colony Optimization (ACO)

Stigmergy-based meta-heuristic for combinatorial optimization (e.g., Traveling Salesperson Problem, TSP).

ACO Core

- **Ants (Agents):** Probabilistically construct solutions by moving from node to node.
- **Pheromone (τ_{ij}):** Stigmergic variable on edges (i, j). High τ means high attraction.
- **Heuristic Information (η_{ij}):** Problem-specific local information, often inverse of distance ($1/d_{ij}$).

Probabilistic Transition Rule: The probability of an ant moving from i to j is proportional to:

$$P_{ij} \propto (\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}$$

- α: controls the weight of pheromone influence.
- β: controls the weight of heuristic influence.

Pheromone Update:

1. **Evaporation (Decay):** $\tau_{ij} \leftarrow (1-\rho)\tau_{ij}$ (ρ is the evaporation rate). Prevents convergence to sub-optimal paths.
2. **Deposition (Reward):** Pheromone is added by ants that have completed a path, proportional to the quality of the solution found (e.g., inverse of path length).

Robot Swarms and Task Allocation

Cyber-Physical Multi-Agent Systems (CP MAS)

Systems of multiple interacting cyber-physical agents (robots, sensors) combining mechatronic structures, communication, and processing.

Design Objectives of Robot Swarms

- **Scalability:** Performance should degrade gracefully as swarm size increases.
- **Robustness:** Tolerance to individual robot failures or environmental changes.
- **Flexibility / Adaptability:** Ability to handle different tasks or environments.
- **Distributed Control:** No single point of failure; decision-making is local.

Multi-Robot Task Allocation (MRTA)

The problem of deciding **who** (which robot) does **what** (which task) and **where/when/how**.

MRTA Taxonomy (ST-SR-IA)

This taxonomy classifies task allocation problems based on three binary dimensions:

- **Single-Task (ST) vs. Multi-Task (MT):** Can a robot execute one task type or multiple task types?
- **Single-Robot (SR) vs. Multi-Robot (MR):** Does a task require one robot or a team/coalition?
- **Instantaneous Assignment (IA) vs. Time-Extended Assignment (TA):** Static, one-time assignment vs. sequential, time-dependent planning/scheduling.

The **ST-SR-IA** problem is the simplest, involving a one-to-one assignment, optimally solved by the **Hungarian Algorithm** (a form of Linear Assignment Problem).

Complex MRTA Cases and Modeling

The complexity of MRTA problems increases significantly when moving from instantaneous, single-robot assignments to multi-robot, time-extended scenarios. The table below maps these complex cases to the canonical optimization models they are equivalent to.

- **MT-SR-TA (Multi-Task, Single-Robot, Time-Extended):** Requires a single robot to determine the optimal sequence of tasks to execute over time. This is a classic scheduling problem, directly modeled as a **Traveling Salesperson Problem (TSP)** or its variations like the **Vehicle Routing Problem (VRP)**.
- **ST-MR-IA (Single-Task, Multi-Robot, Instantaneous):** The task requires a group/coalition of robots. The problem is to cover all required tasks (coalitions) with the available robots at minimum cost/maximum utility. This maps to the **Set Covering Problem (SCP)**.
- **MT-MR-TA (Multi-Task, Multi-Robot, Time-Extended):** The most general and complex case, involving simultaneous scheduling, routing, and coalition formation for multiple tasks and robots.

Table 1: MRTA Taxonomy: Complex Cases and Equivalent Optimization Models

Robots	Tasks	Assignment Type	Equivalent Optimization Model
ST-SR-TA	Single → Single	Time-Extended	**Traveling Salesperson Problem (TSP)**
MT-SR-IA	Multi → Single	Instantaneous	**Generalized Assignment Problem (GAP)**
ST-MR-IA	Single → Multi	Instantaneous	**Set Covering Problem (SCP)** or **Set Partitioning**
MT-SR-TA	Multi → Single	Time-Extended	**TSP** or **Vehicle Routing Problem (VRP)**
ST-MR-TA	Single → Multi	Time-Extended	**Equivalent to MT-SR-TA** (Scheduling + Coalition)
MT-MR-TA	Multi → Multi	Time-Extended	**Job Shop Scheduling Problem** (Multi-Proc. Tasks)

Solution Approaches in MRTA

MRTA problems are generally **NP-hard**, meaning that heuristic and meta-heuristic approaches are often required, especially for large, dynamic systems.

Table 2: Comparison of Task Allocation Solution Approaches

Approach	Advantage	Limitation	Best Case Scenario
Emergent/Swarm	Self-organizing; scalable; reduces central control complexity.	Lacks explicit optimality guarantees; sensitive to parameter tuning.	Systems requiring only local coordination (e.g., foraging, aggregation).
Optimization	Provides optimal or near-optimal solutions; well-suited for static scenarios.	Computationally intensive (NP-hard problems); less adaptable to changes.	Well-defined, static problems (e.g., initial ST-SR-IA).
Market-Based	Flexible, scalable, decentralized via bidding (e.g., MURDOCH/auction).	May not yield global optima; requires effective bidding and communication.	Dynamic environments with varying tasks; heterogeneous robot teams.
Learning-Based	Adaptable; learns and improves over time in uncertain environments (e.g., RL).	Requires extensive training time; initially suboptimal; high computational cost for training.	Complex and uncertain environments where performance must be learned through experience.

Optimization Solution Methods (for Centralized Models):

- **Exact Methods:** Solving Integer Programming (IP) formulations to optimality using commercial solvers (e.g., Branch-and-Bound with MIP gap) or polynomial-time algorithms for special cases (e.g., Hungarian Algorithm for ST-SR-IA).
- **Heuristics/Meta-Heuristics:** Using approximation algorithms like ACO, PSO, or Genetic Algorithms (GA) to find high-quality, near-optimal solutions in a reasonable amount of time.