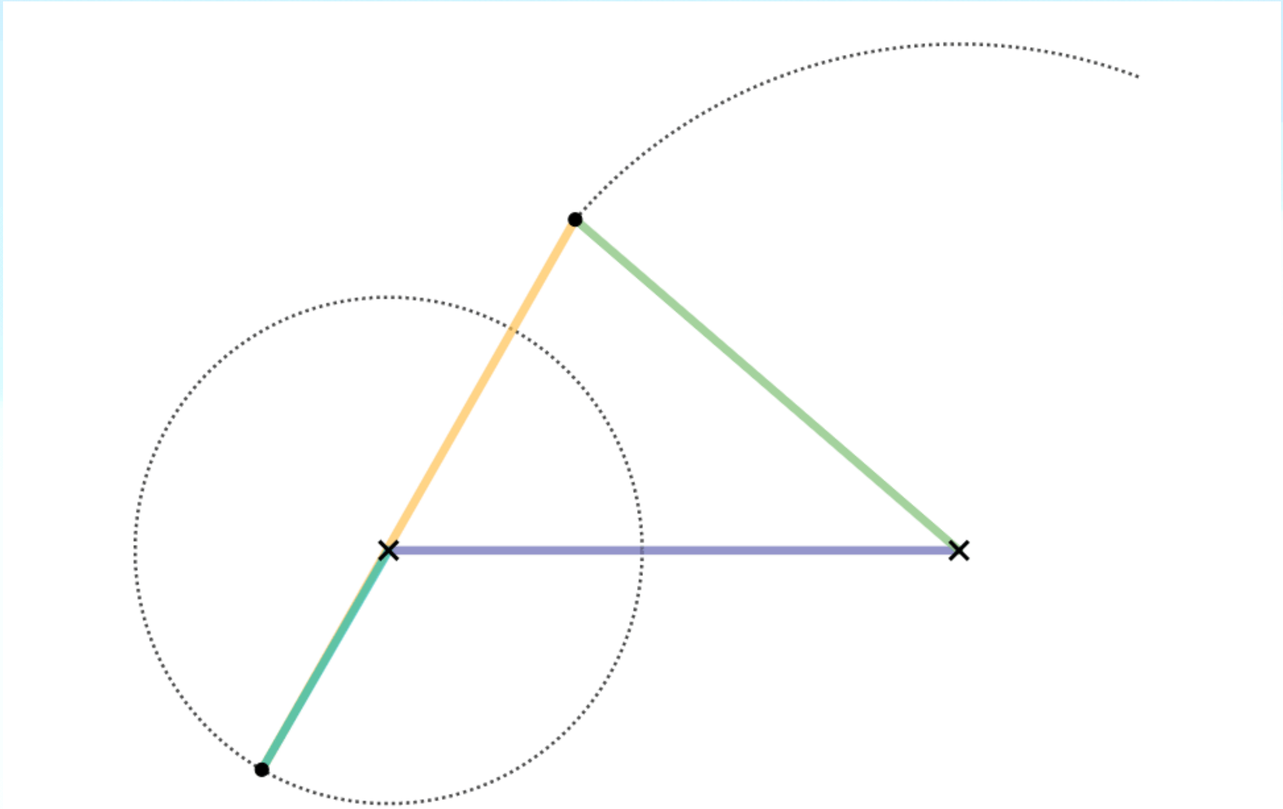
Kinematics of Linkages and Algebraic Constraints

Tamar Neter, 07.07.2025

Kinematics of Linkages and Algebraic Constraints

A simulation of a Crank-Rocker 4-Bar Linkage



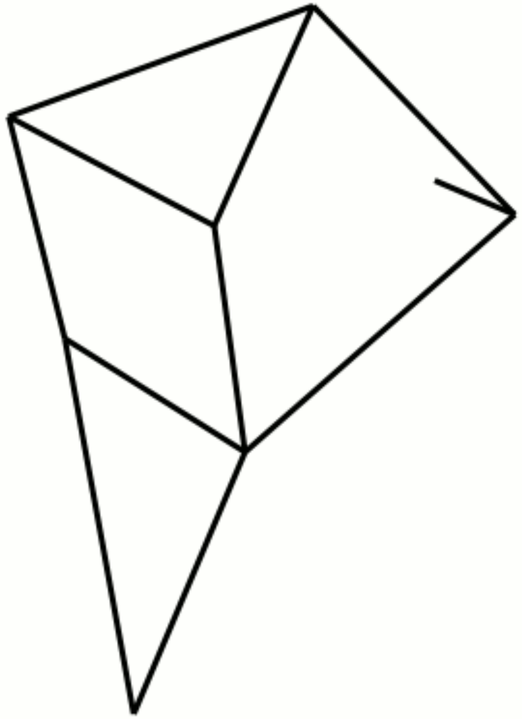
What are Mechanical Linkages?

- Assemblies of rigid bodies (links) connected by joints.
- Purpose: **Transforming motion** and forces

Oil wel pump-jack

Why are Linkages Important?

- Everywhere in Engineering & Daily life:
  - Automotive (wipers, engines)
  - Robotics (grippers, walking robots)
  - Industrial Machinery (packaging)
  - Consumer Products (sewing machines, toys)

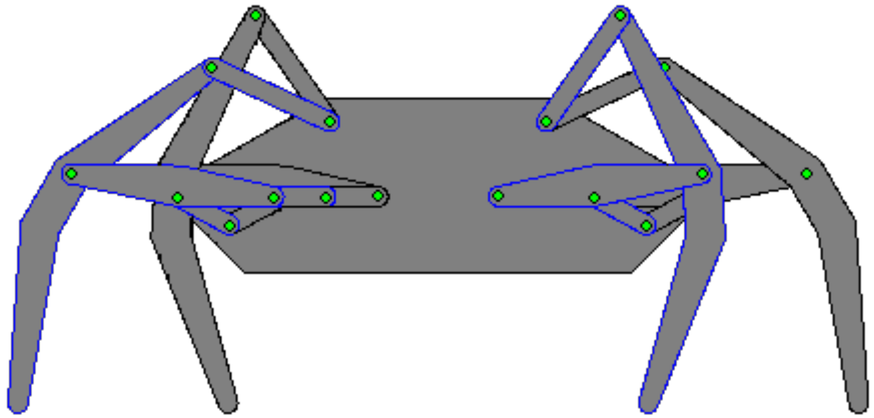


Why are Linkages Important?

- Fundamental to Motion Design:
  - Motion Generation: Controlling the movement of all links.
  - Path Generation: Tracing specific desired curves (straight lines, complex coupler curves).
  - Function Generation: Mapping an input motion to a desired output motion (a specific speed change).
- Legacy & Modern Relevance: A centuries-old field, but still active research due to increasing demands for precision and complexity.

Describing Linkage Motion

- Linkage motion is inherently non-linear and complex
- Traditional methods often struggle with:
  - Multiple possible configurations
  - Singularities
  - Tracing continuous motion paths robustly



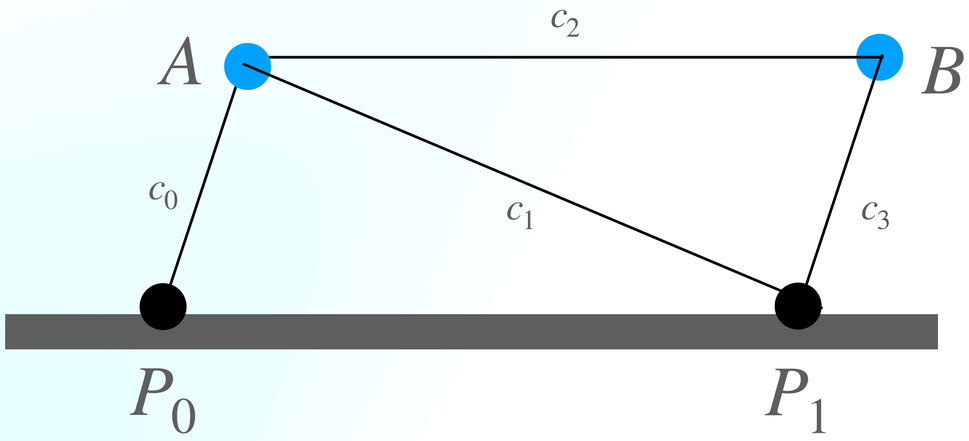
Example of Klann linkage- which is a planar mechanism designed to simulate the walk of a legged animal and function as a wheel replacement, a leg mechanism

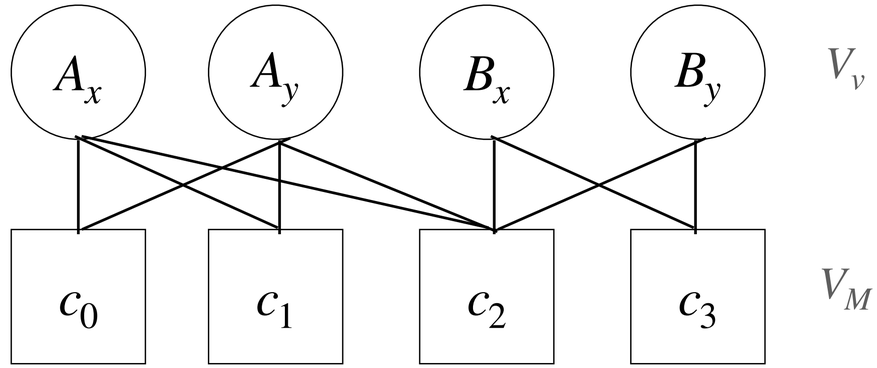
The Presented Approach

- Formulating motion as Algebraic Constraints
  - Transforming geometry into a system of polynomial equations
  - Leveraging computational geometry and specialized solvers (e.g., spline based) for robust analysis and design

Mapping Motion to Algebraic Constraints

- Goal: Translate physical geometry into mathematical equations
- (c_0, c_1, c_2, c_3) Links: Rigid bodies with fixed lengths
- P_0, P_1 fixed ground joints
- $A = (A_x, A_y), B = (B_x, B_y)$ moving joints
- Motion is described by changes in joint coordinates





Subdivision-Based Solvers

- van-Sosin & Elber 2016: Systematically explore the entire solution space
- Recursive subdivision:
  - Divide the unknown’s domain into smaller sub-regions
  - Test each sub-region for the potential presence of a solution
  - Discard regions guaranteed not to contain solution
  - Recursively subdivide promising regions until desired accuracy is met
- **Guaranteed Global Search:** ensures all real solutions are found within the specified domain
- **Robustness:** less sensitive to singularities and complex solution landscapes

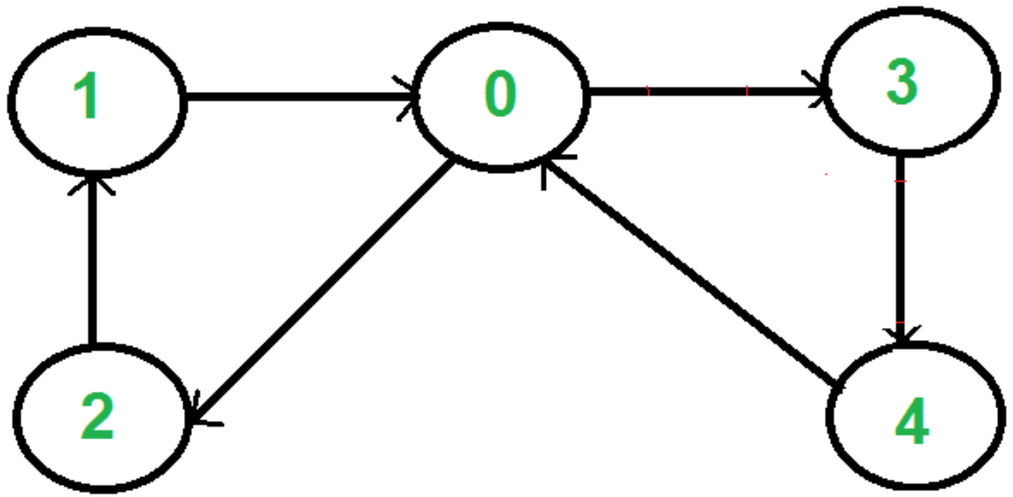
Complexity- some motivation

- Subdivision solvers scale exponentially with number of variables - k
- The complexity without decomposition $\approx c^k$
- The complexity with decomposition into d subsystems $\approx d \cdot c^{\frac{k}{d}}$

Where c is the average subdivisions per variable

Directed Graphs

- A **directed graph** is a set of vertices connected by **oriented** edges
- Represented as $G = (V, E)$, where $E \subseteq V \times V$
- Allowed cycles, paths and direction-based dependencies

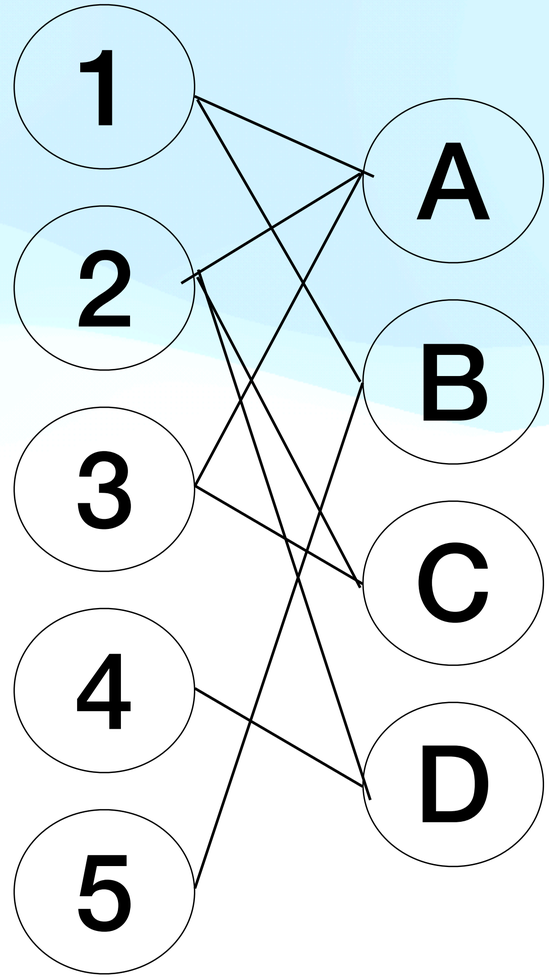


Bipartite Graph

- A graph $G = (V, E)$ is bipartite if its vertex set V can be divided into two disjointed sets U and W such that:

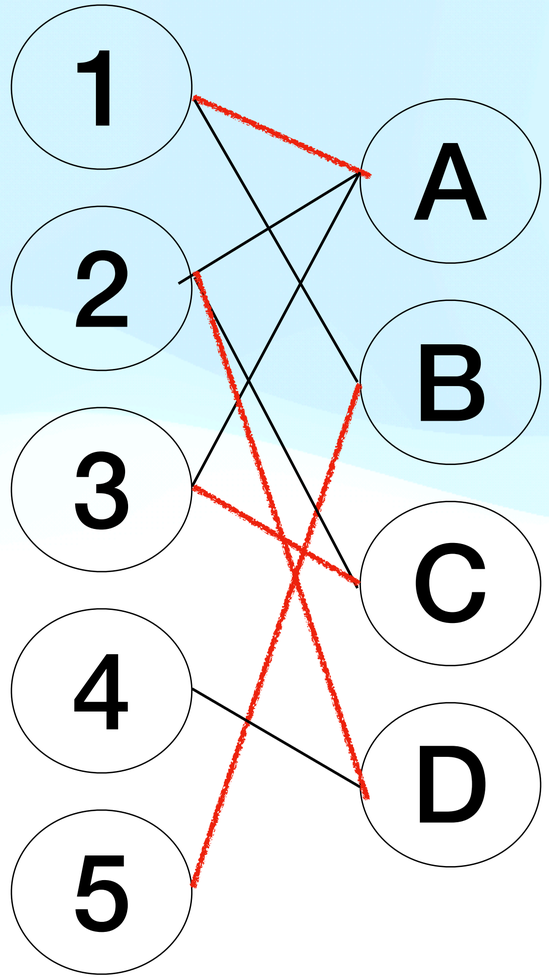
$$E \subseteq \{(u, w) \mid u \in U, w \in W\}$$

- No edges within U or within W
- Edges are undirected
- Often used to model constraint systems



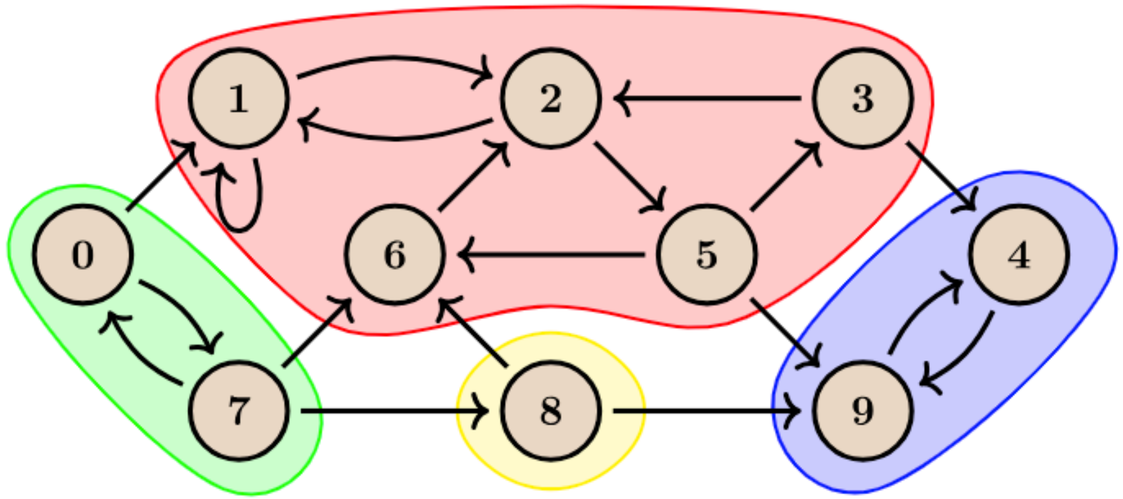
Maximum Matching

- Given a graph $G = (V, E)$:
  - A **matching** is a set of edges without common vertices
  - A **maximum matching** is a matching with the largest possible number of edges
  - An algorithm that finds maximum matching in a bipartite graph is Hopcroft-Karp - with time complexity of $O(E\sqrt{V})$



Strongly Connected Component

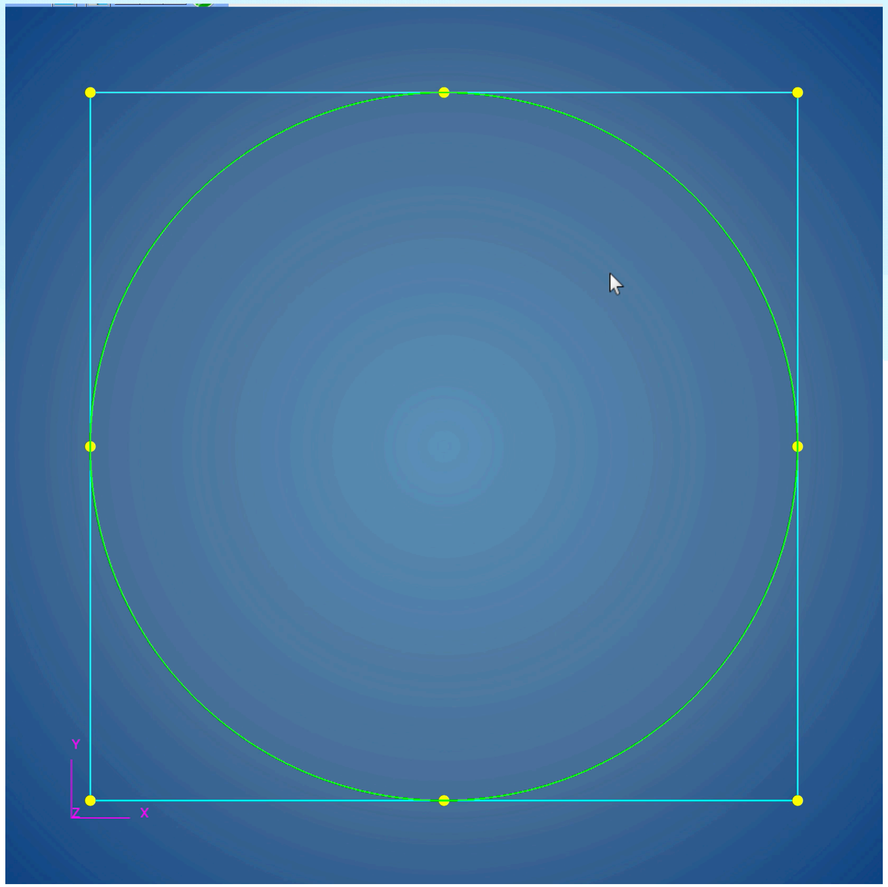
- An SCC is a **maximal subgraph** in which every vertex is reachable from every other vertex
- SCCs partition the directed graph
- Useful for solving decomposed systems independently



B-Spline Property: The Convex Hull

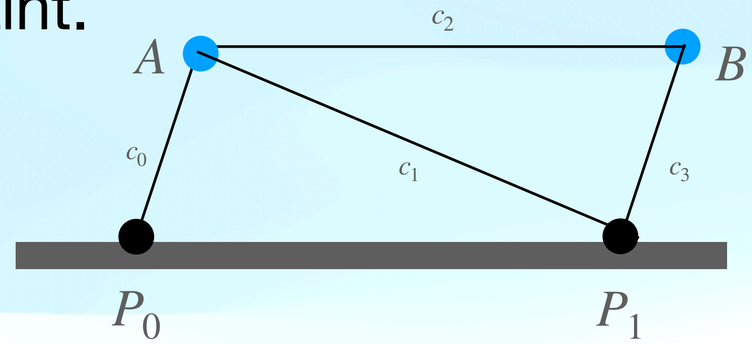
- A B-Spline curve is **always** contained entirely within the convex hull of its control points.
- The control points form a ‘control polygon’
- The convex hull is the shape enclosing all control points
- The resulting curve will never leave this bounding shape

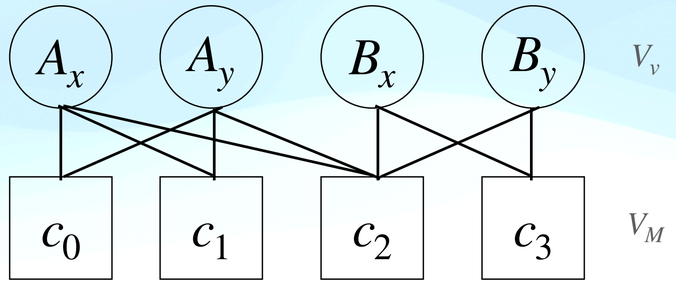
This property makes the curve’s behavior predictable and is fundamental for many algorithms



The Algorithm

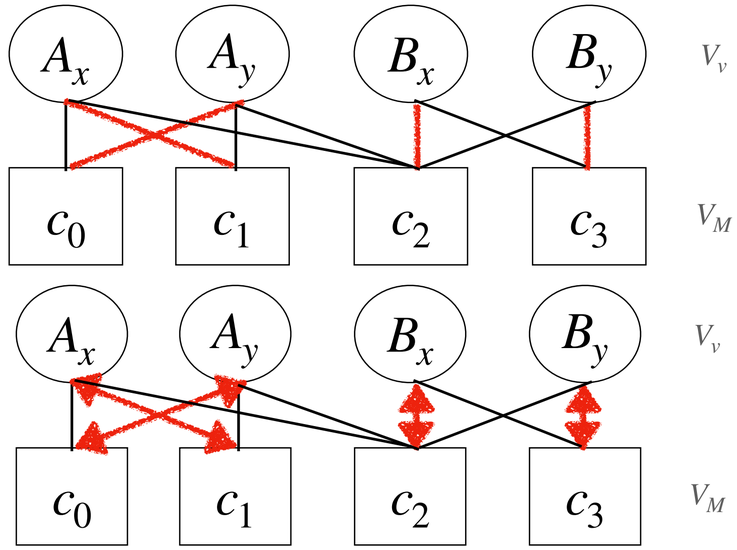
- Constructing a variable constraint graph (G) that represents the zero constraints of the problem - V_v vertex for each variable, V_M vertex for each constraint.





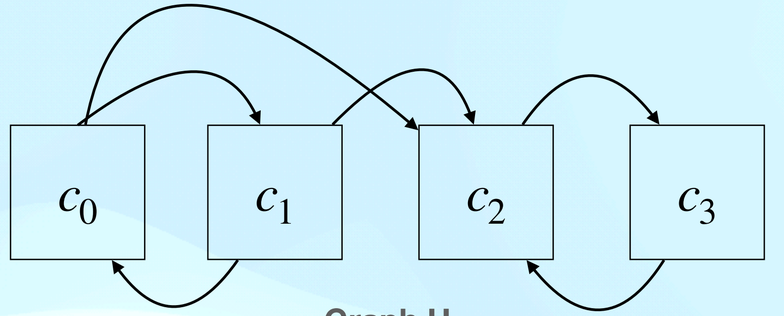
Graph G

- Decomposition into subsystems
  - Finding maximum matching in the graph G
  - Building a new **directed** graph G'

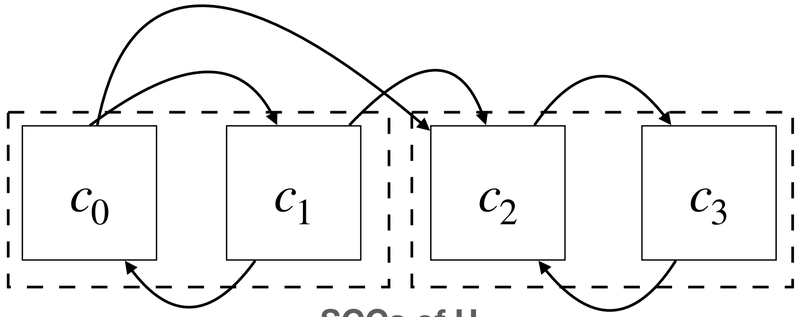


The Algorithm cont.

- Decomposition into subsystems
  - Finding maximum matching in the graph G
  - Building a new **directed** graph G'
  - Condensing the dependency graph (G') into a new graph H constructed by the set of all constraint vertices of G'
  - Construct SCC of graph H
  - Adding inequality constraints to each sub-system



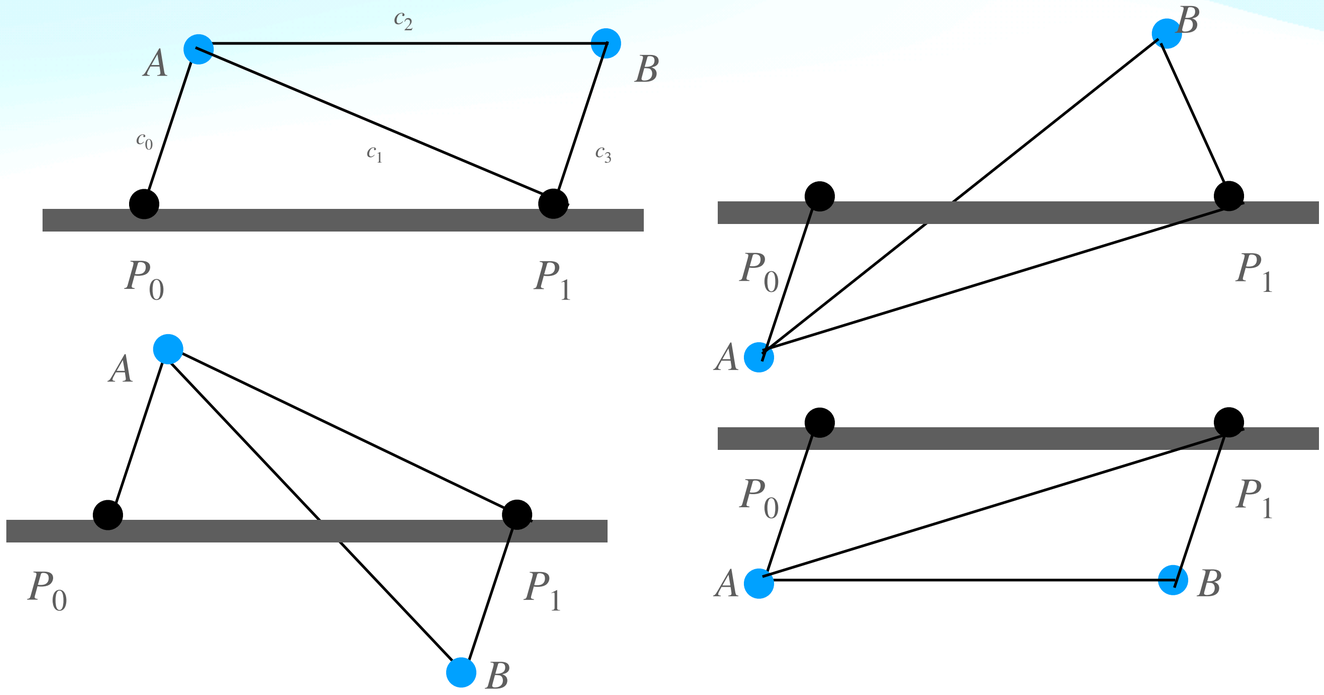
Graph H



SCCs of H

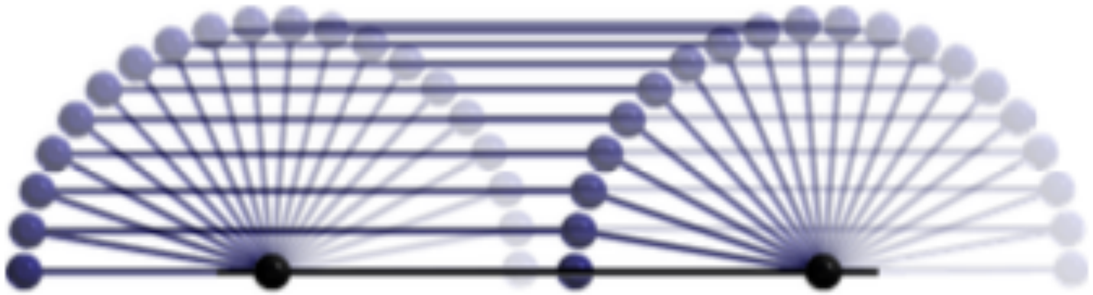
Algebraic Representation of the example

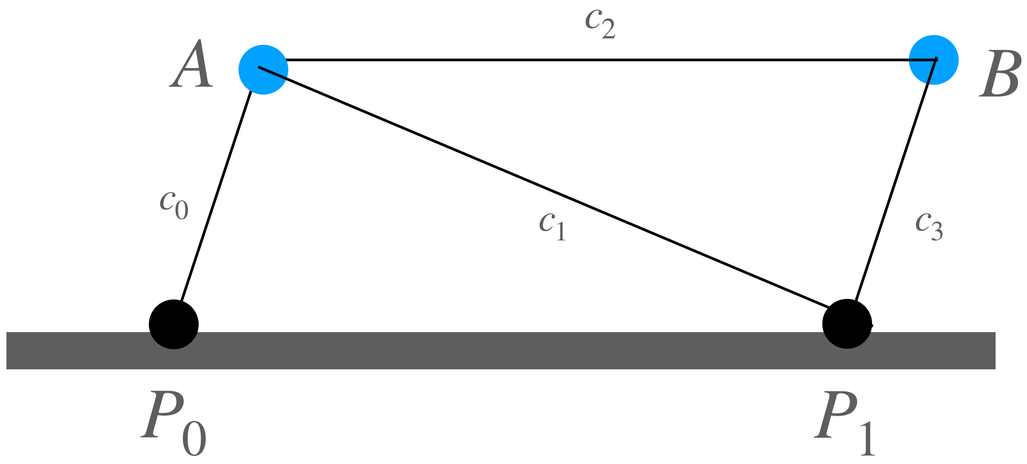
- For $P_0 = (-1,0), P_1 = (1,0)$ and $c_0 = 0.9, c_1 = 2.5, c_2 = 2, c_3 = 0.9$
- $$\begin{cases} |A - P_0|^2 - c_0^2 = 0 \\ |A - P_1|^2 - c_1^2 = 0 \\ |A - B|^2 - c_2^2 = 0 \\ |B - P_1|^2 - c_3^2 = 0 \end{cases}$$



Example cont.

- Equality constraints: $\begin{cases} |A - P_0|^2 - c_0^2 = 0 \\ |A - P_1|^2 - c_1^2 = 0 \\ |A - B|^2 - c_2^2 = 0 \\ |B - P_1|^2 - c_3^2 = 0 \end{cases}$
- Inequality constraints: $\begin{cases} A_y \geq 0 \\ z((B - P_1) \times (A - P_1)) \geq 0 \end{cases}$

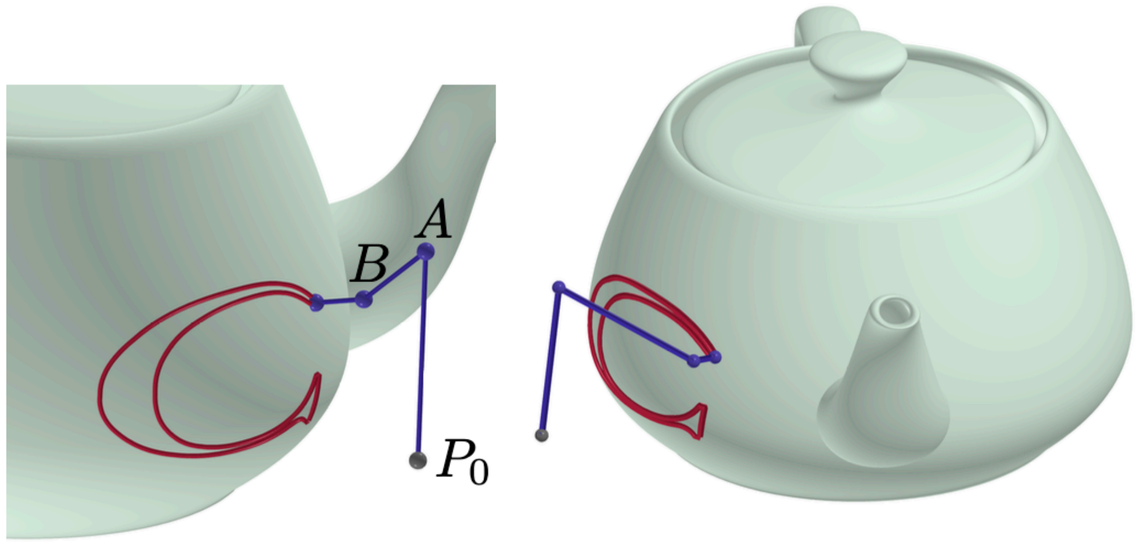




Example 2

The goal: Embed a 2D engraving curve onto the surface of a teapot.

- The teapot is the Utah teapot- represented as a surface $S(u, v)$
- The engraving is a planar curve $c(t)$
- The embedding of the engraving onto the surface is done by functional composition: $c_S(t) = S(c(t))$

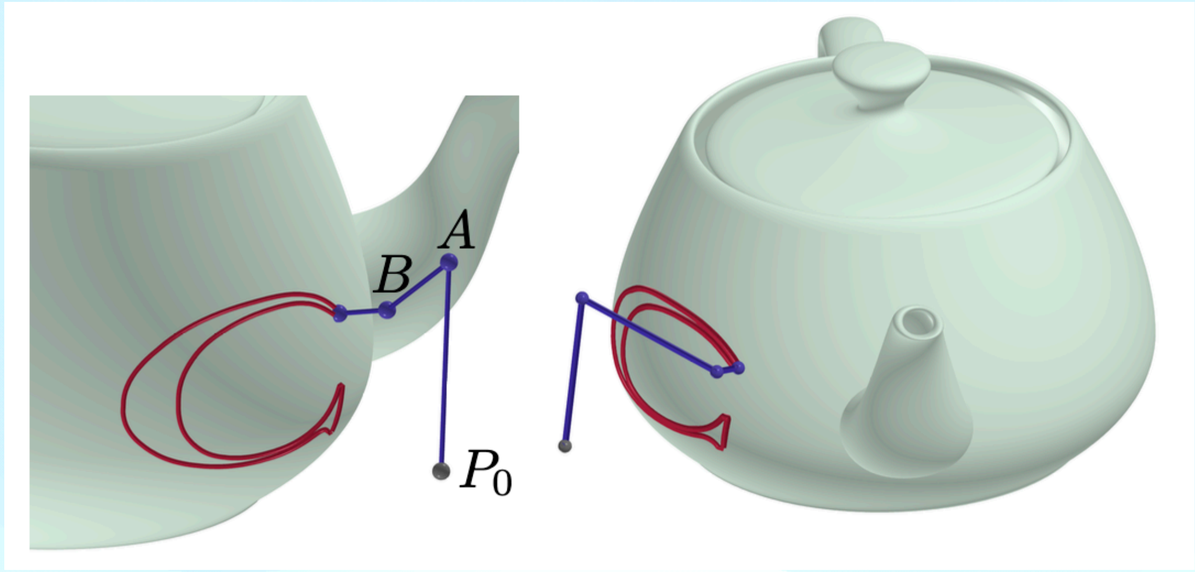


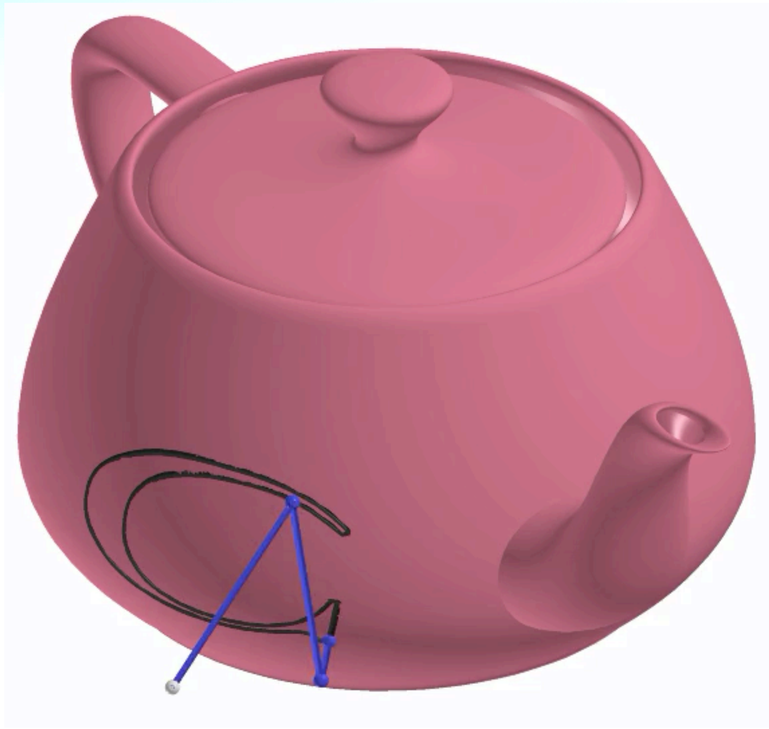
Example 2 cont.

- The constraints:

$$\begin{cases} |c_S(t) - B|^2 - L_3^2 = 0 \\ \left\langle S_u(c(t)), B - c_S(t) \right\rangle = 0 \\ \left\langle S_v(c(t)), B - c_S(t) \right\rangle = 0 \\ |B - A|^2 - L_2^2 = 0 \\ |A - P_0|^2 - L_1^2 = 0 \\ z((B - A) \times (P_0 - A)) = 0 \end{cases}$$

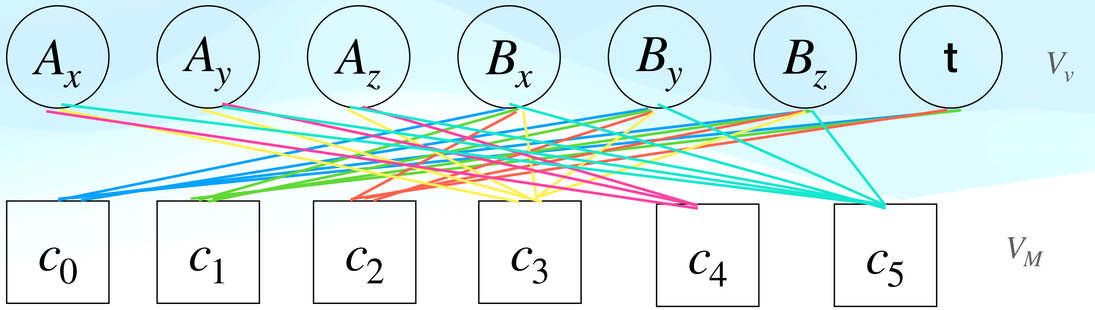
- Lets denote the constraints $c_0 \dots c_5$





The Algorithm in Context

- This is the bipartite graph of the teapot example- continuing this will lead us to the solution path



How it all connects…

- Any polynomial can be exactly represented as a multivariate B-spline
- The solver performs this conversion automatically behind the scenes - we don’t input control points or knots!
- The method combines smart, graph-based decomposition with a guaranteed subdivision solver
- This approach makes it possible to solve previously intractable geometric problems
- The solver’s efficiency comes from the convex hull property of B-splines