Synthesizing Robot Designs
(Sanjeev Khanna and Tarik Tosun)

Objective:

Design an algorithm that, given two robot designs, can synthesize an optimal new design that implements both.

Motivation:

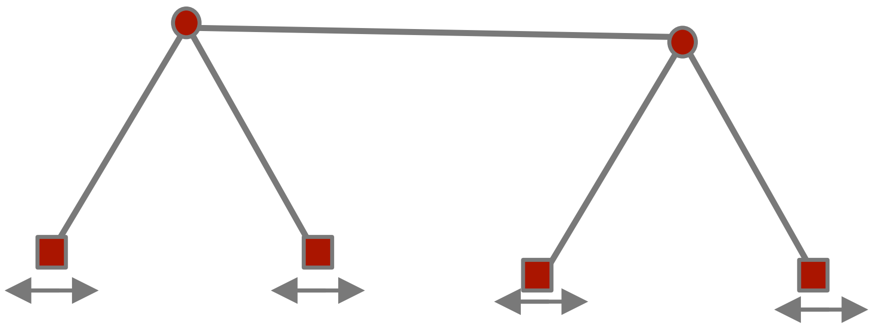
Enabling a modular robot design system allowing novice users to create new designs using a collection of base designs.

Methodology

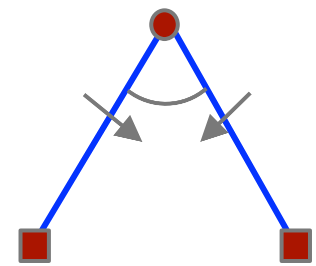
- Proceeding from a primarily topological perspective
  - Eventual goal of incorporating many more real-world constraints

- Robot designs represented as configuration graphs
  - Capturing two designs in a single design requires a suitable notion of configuration graph embedding

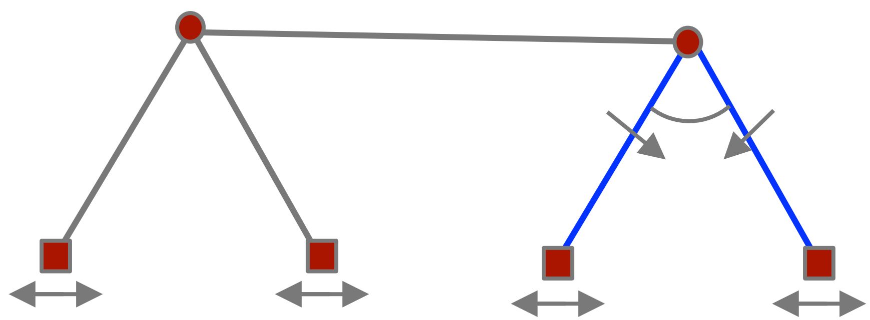
Example



(A) “Walker” design



(B) “Grasper” design



(C) Design embedding both “walker” and “grasper”

Embedding Definition

- Our abstraction of a design:
  - Designs are represented as labeled graphs – $G(V,E)$
  - Some additional information is captured
    - Functionality of each joint – $f(v)$ *for each vertex* v *in* V
    - Set of end effectors - X
    - Lengths of edges – $\ell(u, v)$
  - Resulting design specification is a 4-tuple: $\langle G(V,E), X, f, \ell \rangle$
- We say that a design G_1 embeds in a design G_2 if there exists a 1-1 mapping Φ from vertices in G_1 to vertices in G_2 such that the adjacency structure of G_1 is preserved in G_2 when edge lengths of G_1 are scaled by a factor of λ.

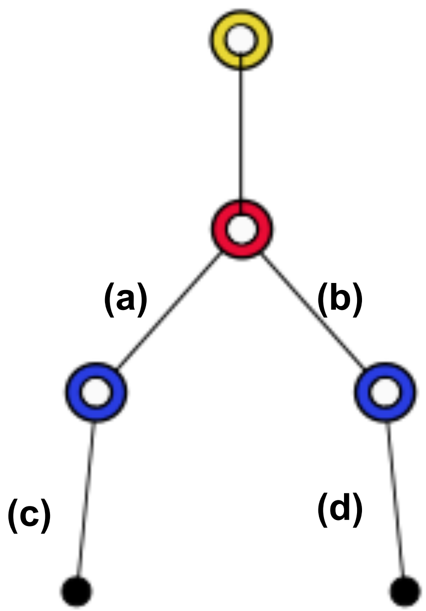
Conditions to embed G_1 in G_2

- Vertex-to-Vertex Correspondence
  - For each vertex $u \in V_1$, there must be a corresponding vertex $v \in V_2$ and this vertex must subsume the functionality of u: $f_2(\varphi(u)) \succeq f_1(u)$
- Edge-to-Path Correspondence
  - For each edge $(u, v) \in E_1$, there must be a corresponding path $P_{u,v} \in G_2$

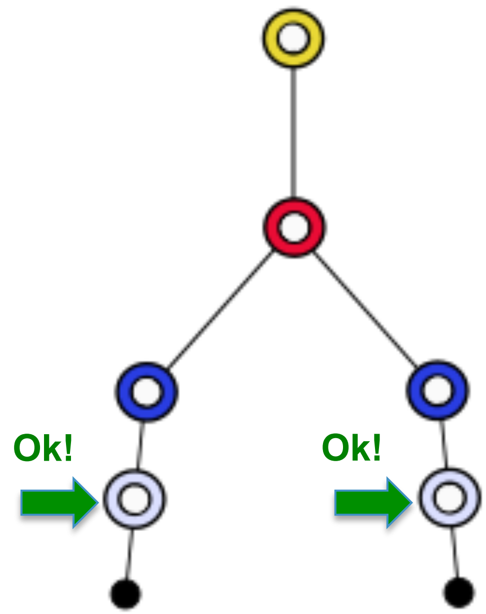
Conditions to Embed G_1 in G_2

- Vertex-Disjointness of Paths
  - For each pair of edges $(u_1, v_1), (u_2, v_2) \in G_1$, the corresponding paths $P_{u_1 v_1}, P_{u_2 v_2} \in G_2$ must be vertex-disjoint
- Uniform Length Scaling
  - The length of each path in G_2 must be proportional to the length of its corresponding edge in G_1.
  - $\ell_2(P_{uv}) = \lambda \ell_1(u, v)$

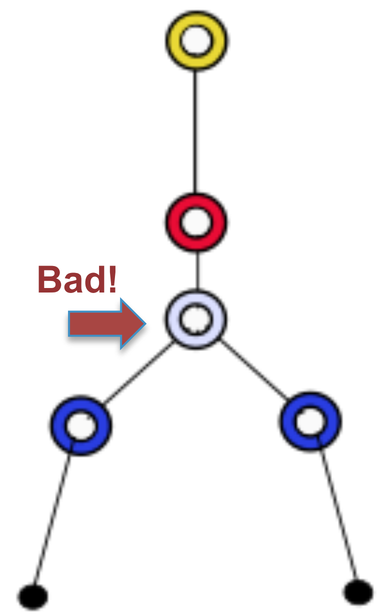
Example



Design 1



Design 2 embeds design 1



Design 3 does not embed design 1; violates path vertex-disjointness

Algorithm to Test Embeddability

- Assume that design graphs are trees rooted at a specific vertex
- Dynamic programming algorithm progresses upwards through two trees, beginning at the end effectors
  - We define subproblem as *subtree embedding*
- Create table $T(\boldsymbol{a},\boldsymbol{b})$, where $T(\boldsymbol{a},\boldsymbol{b}) = 1$ iff the subtree of node $\boldsymbol{a}$ in G_1 embeds in the subtree of node $\boldsymbol{b}$ in G_2 such that $\Phi(\boldsymbol{a}) = \boldsymbol{b}$.

Embedding Detection Algorithm

To embed a subtree rooted at a node ***a*** in G_1 in a subtree rooted at a node ***b*** in G_2, two conditions must be satisfied:

1. The functionality of node ***b*** subsumes the functionality of node ***a***, $f_2(b) \succeq f_1(a)$

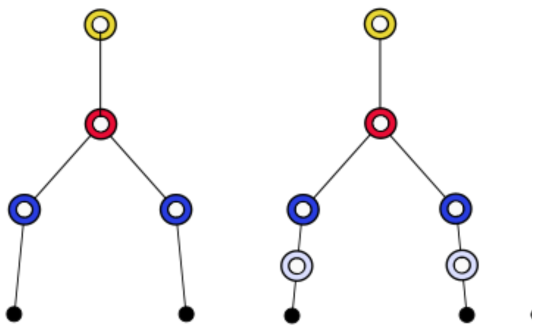
2. For each child $\boldsymbol{a_i}$ of ***a*** , there must be some descendant $\boldsymbol{b'_j}$ of ***b*** that embeds $\boldsymbol{a_i}$ without violating either the length condition or the vertex-disjointness condition

How do we check condition 2?

- Form a bipartite graph $H(P, Q)$
  - P are the children of a, $\{a_1, a_2, \ldots, a_p\}$
  - Q are the children of b, $\{b_1, b_2, \ldots, b_q\}$
  - Form edge (a_i, b_j) if for some descendent b'_j of b_j, both:
    - $\ell_1(a, a_i) = \lambda \ell_2(b, b'_j)$, and
    - $T[a_i, b'_j] = 1$
- We seek a matching that assigns each node in P to a distinct node in Q using edges in $H(P,Q)$.

Additional Constraints

- We are incorporating more real-world constraints.
- Edge Rigidity:
  - Joints aren't always able to "lock" in place and form rigid paths
  - A rigidness characteristic may be associated with edges and a locking characteristic may be associated with joints
  - Easy to verify that rigid edges map to paths with only locking joints while checking the length condition.



Additional Constraints

- Non-Uniform Length Scaling:
  - Uniform length scaling requirement can be relaxed
  - Define a scaling tolerance $\tau_i((u, v)) \in [0, 1]$ for each edge $(u, v) \in E_i$.
  - Length constraint becomes an inequality
    - Lower bound of $(1 - \tau)\lambda\ell$
    - Upper bound of $\frac{\lambda}{(1-\tau)}\ell$
  - $\tau((u, v)) = 0$ corresponds to an equality constraint
  - $\tau((u, v)) = 1$ corresponds to complete relaxation of the length condition on paths embedding (u, v)

Ongoing Work

- Algorithm for design merging
- More detailed specification of joint functionality
- Higher-order reasoning in embedding detection
  - Kinematic redundancy and equivalence involving more than one joint

An Illustration of Design Merging