

# **Evolutionary Design for Robot Configuration Synthesis**

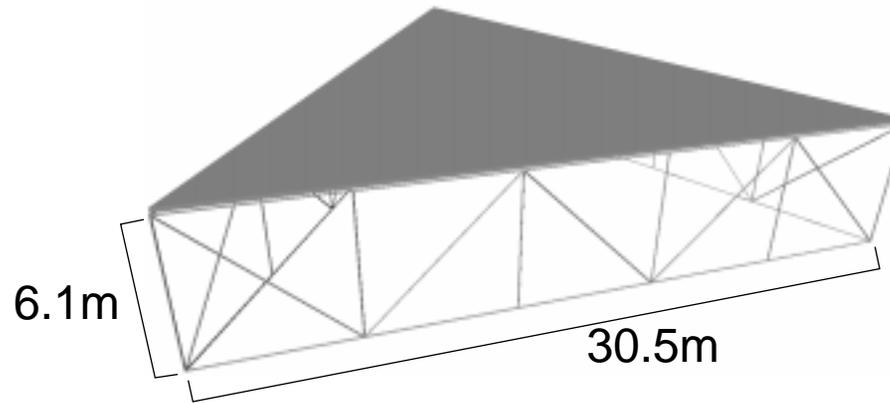
**Chris Leger**

**NASA Jet Propulsion Laboratory  
July 12, 2001**

# Design example

---

**We want to design a robot for inspecting a Space Solar Power satellite section.**



- **Walkers are preferred over free-flyers**
- **Mobility requirements:**
  - Move along long truss sections (longerons)
  - Move between longerons of different orientations
  - Out-of-plane motions and sensor positioning

# Design Challenges for Robotics

---

- What kinematic configuration should be used?
- How big should the robot be? ~1m? 2m? 3m?
- What actuators (motors and gearheads)?
- How should the robot move? (Controller design)
- **All of these factors are interdependent.**
  - Hard to predict all impacts of a single design variable
- **Significant effort required to specify designs to the point that they can be evaluated.**
- **Critical design decisions are made early, on basis of least information**
  - Many limitations aren't known until robot is undergoing testing
  - Most robot designs are one-offs; little or no time/money for iteration

# Overview

---

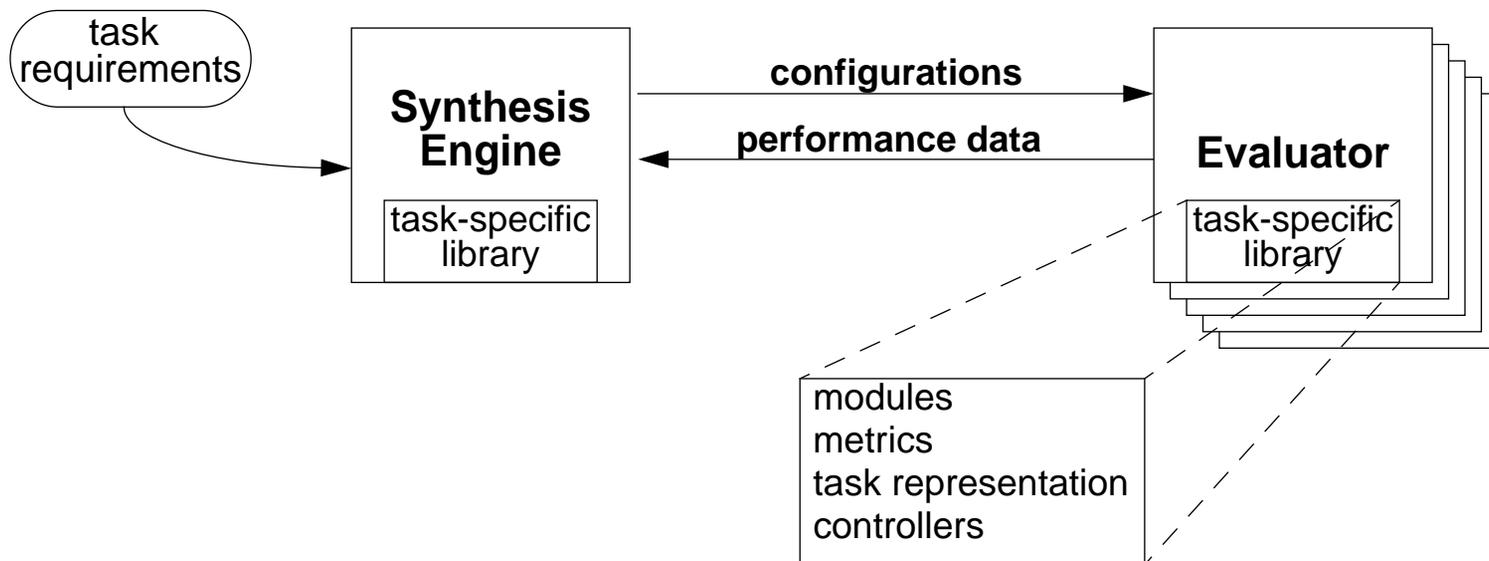
***Darwin2K* is an extensible system for robot configuration synthesis**

- **Distributed evolutionary algorithm**
- **Robot representation uses graph of parameterized modules**
- **Performance evaluation through task-specific simulation**
- **Multiple performance constraints and objectives**
- **Extensible software architecture**

# System architecture

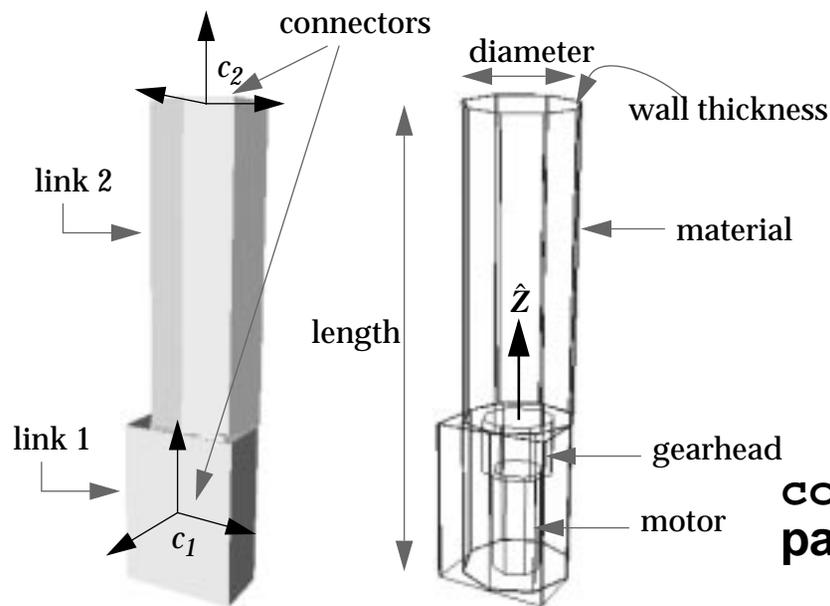
---

- **Central *synthesis engine*; multiple *evaluators***
- **Dynamic libraries for task-specific modules, metrics, simulation algorithms**
- **Synthesis engine is independent of task and module details**



# Parameterized Modules

Modules for bases (including mobile), joints, links, tools.  
A `rightAngleJoint`:



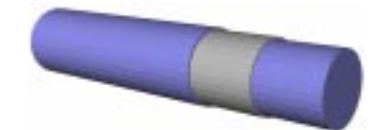
Six parameters:

- motor selection
- gearhead selection
- material selection
- tube outer diameter
- tube wall thickness
- overall length

`const-flag` allows designer to fix parameter values

Other modules:

`irtYoke`  
and  
`balloonWheel`  
(includes  
motor + gearhead)



`prismaticTube`  
(2 telescoping joints)

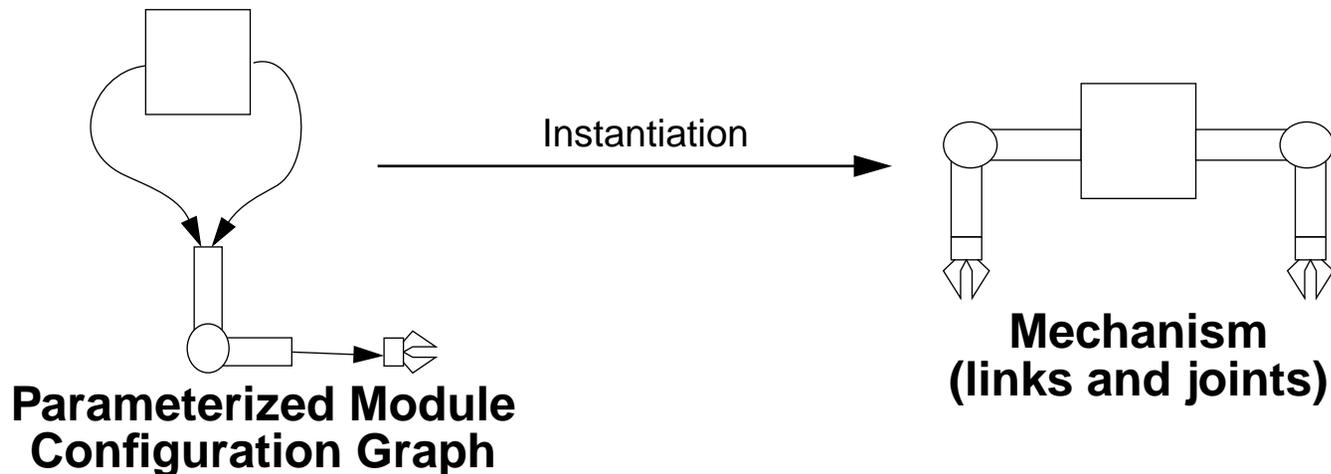


`scaraElbow`  
(3 joints)

# Parameterized Module Configuration Graphs

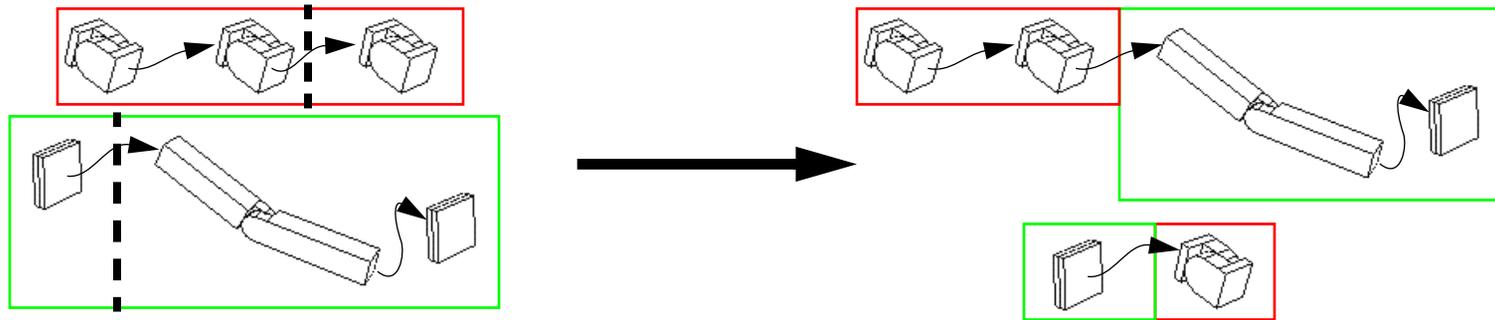
---

- Robots represented as a directed, acyclic graph
- Connections include twist parameter and `const-flag`
- Symmetry can be preserved through multiple connections

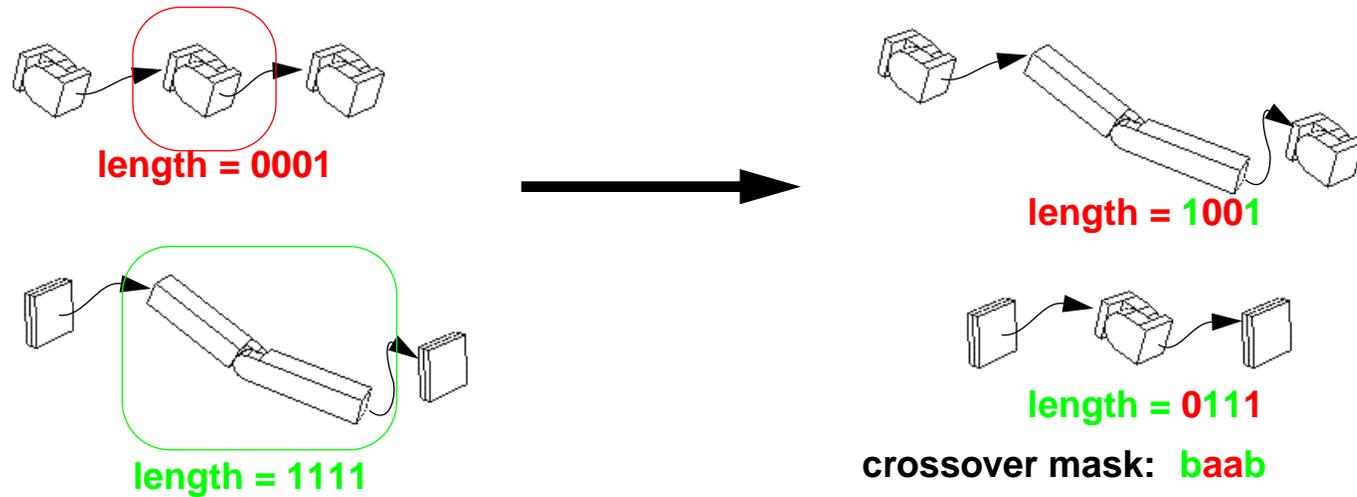


# Crossover Operators

## Module Crossover

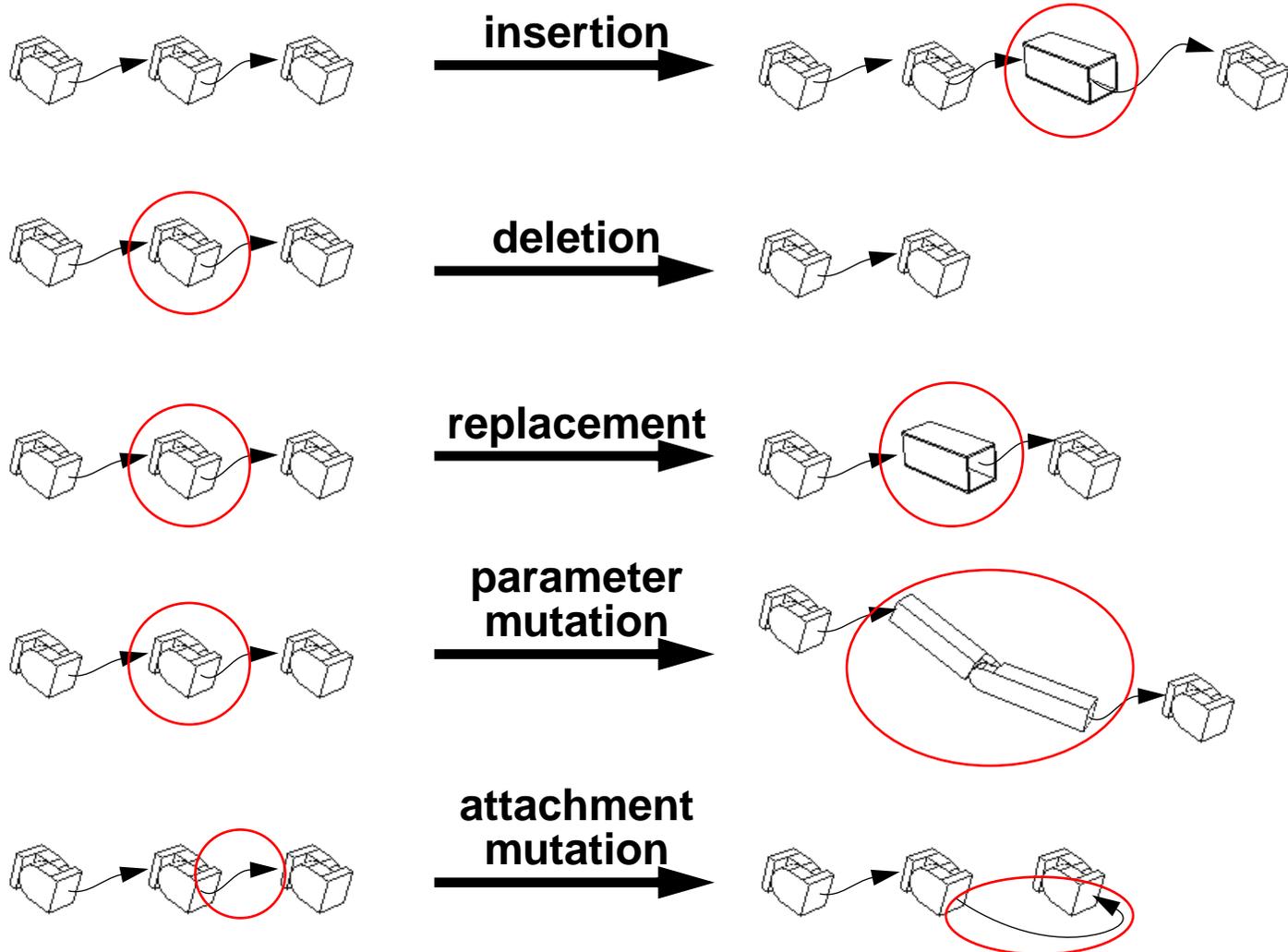


## Parameter Crossover



# Mutation operators

---



# Multiple requirements and objectives

---

**Robot performance objectives and constraints are highly multidimensional.**

- Kinematics: reachability
- Dynamics: response time
- Actuators: adequate torque and velocity
- Structural: low deflection, adequate safety factor
- Control: accuracy
- Feasibility: no collisions, no tip-over, task completion
- Mass, power, time, complexity, cost

**Need to capture requirements and designer's understanding of their relative priorities.**

- **e.g. If the robot only completes 10% of a task, power usage is irrelevant**

# Requirement Prioritization

---

**RP tells the synthesizer which metrics to use for selecting configurations. Rationale:**

- If no configurations satisfy a requirement, then frequently use the corresponding metric for selection
  - Don't optimize low-priority requirements (power) until high priority requirements (task completion) are satisfied by many robots
  - Adaptively set frequencies for each metric based on fraction of population satisfying that metric.
- **Designer gives an acceptance threshold and priority to each requirement:**

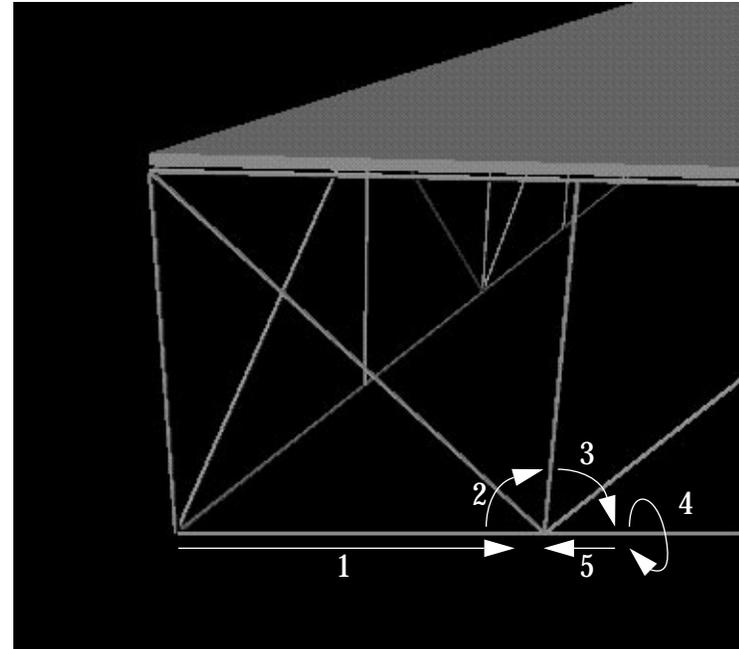
```
priority = 2   metric 0   threshold: == 100% // task completion
               metric 1   threshold: == 0     // # collisions

priority = 1   metric 2   threshold: < 3cm   // error
               metric 3   threshold: < 1mm   // deflection
               metric 4   threshold: < 50%   // actuator saturation

priority = 0   metric 5   threshold: none    // time
```

# Truss Walker

- **Restrict to 7 DOF, symmetric designs**
- **Four requirements:**
  - task completion = 100%
  - collisions = 0
  - link deflection < 1mm
  - continuous saturation < 80%
- **Three objective functions:**
  - minimize mass
  - minimize power
  - minimize time

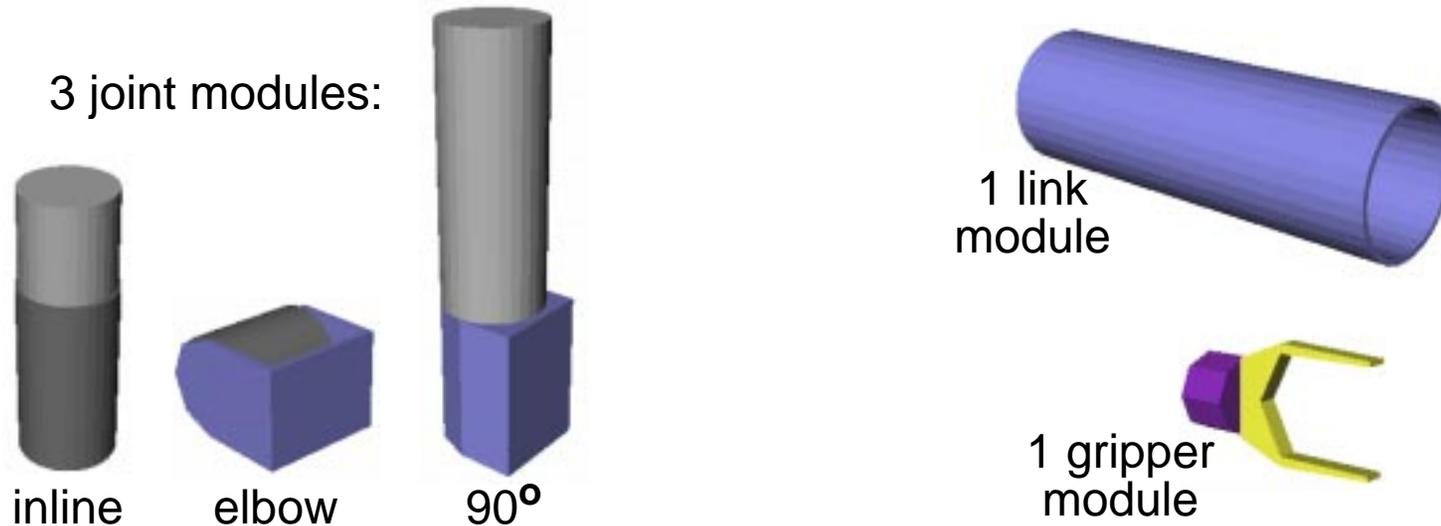


- 1 - walk along truss
- 2 - move to perpendicular rod
- 3 - move to another perpendicular rod
- 4 - move to back of truss
- 5 - move end effector to inspection position

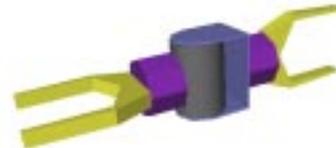
# Walker modules

---

- **Five parameterized modules:**



- **Include gait parameters with each design:**
  - stride length, grasp approach vector, via point location and tolerance
- **Simple *kernel* configuration**



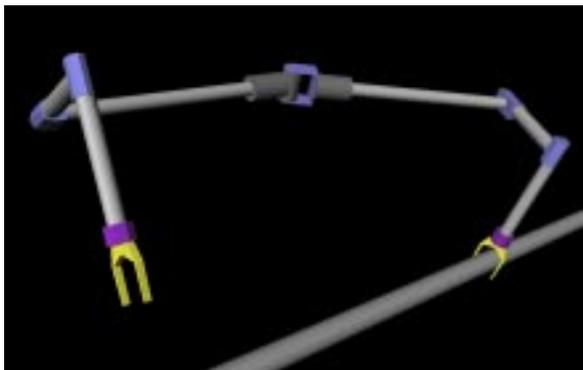
# Synthesized walkers

---

**Synthesizer produced a range of Pareto-optimal configurations of the feasible set**

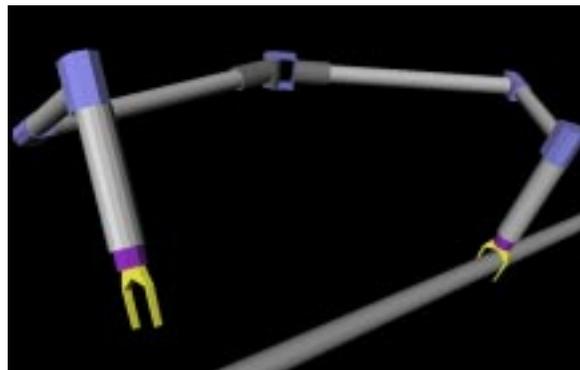
- All Pareto-optimal configurations in the feasible set are considered 'best' and are never deleted
- **Extreme (best in one metric) configurations shown below**

**Lightest - 12.2kg**



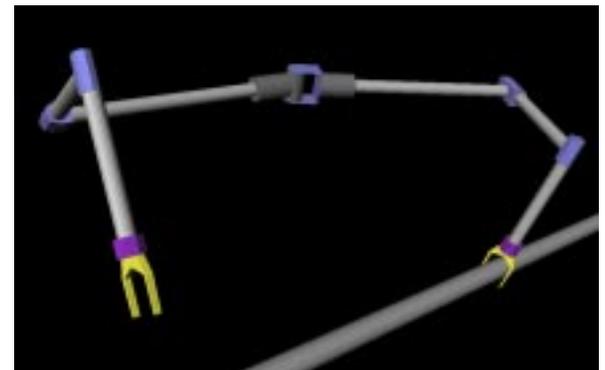
energy 255J time 77.2s  
thin, lightweight links

**Fastest - 55.2s**



mass 18.6kg energy 1112J  
powerful actuators and stiff links  
high acceleration and velocity

**Least Power - 221 J**

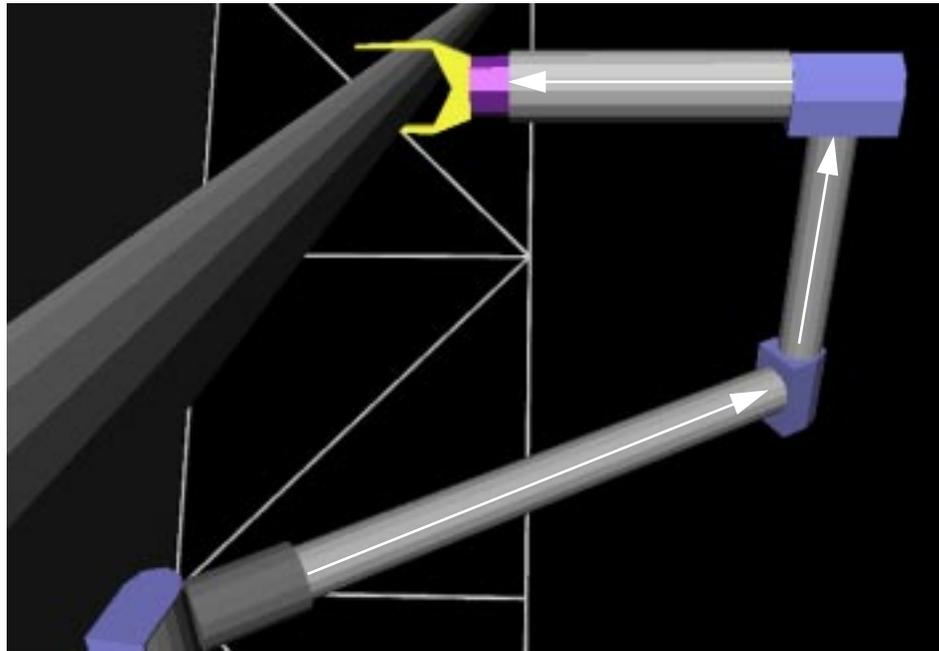


mass 12.3kg time 84.7s  
low acceleration and velocity

# Walker wrist kinematics

---

Evolved wrist structure ***completely eliminates self-collisions*** within each half of the arm



Useful, since controller doesn't try to avoid self-collision  
(*controller bias*)

# Task Impact

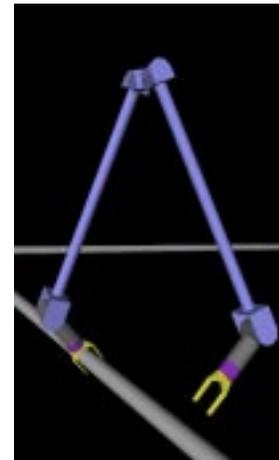
What if the task is **only the walking stage** during synthesis?

**Hybrid lander/  
walker**



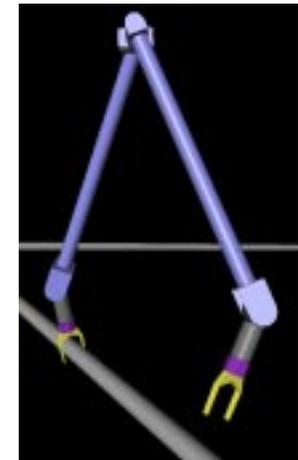
6.3kg (vs. 12.2kg)

**Lightest**



38J (vs. 82J)

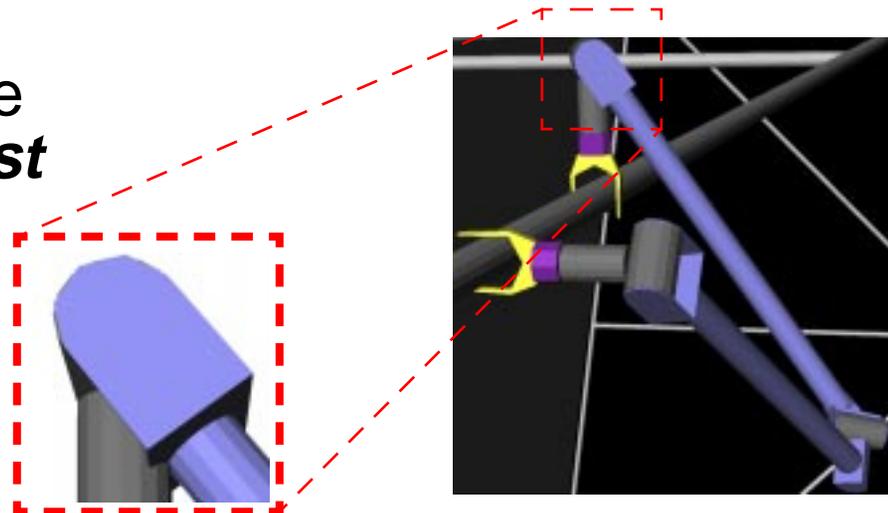
**Least Energy**



19.6s (vs. 19.8)  
14 kg (vs. 18.6)

**Fastest**

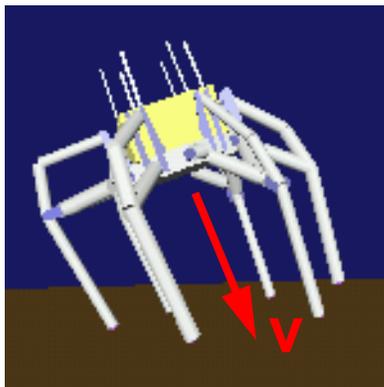
Robots are **lighter** and have different wrist, but have **wrist self collisions** when evaluated on full task.



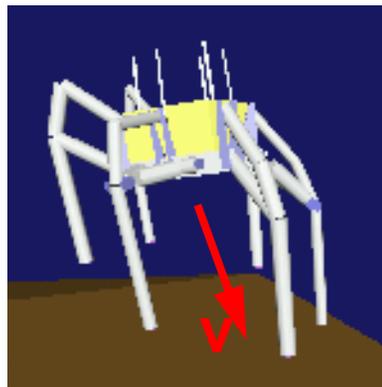
# Hybrid Lander/Walker

- **Determine feasibility of using legs for both walking and landing**
  - Requirements: max accel., structural safety, actuator saturation, stability
  - Objectives: total mass, payload mass, stowed volume
- **New design regime; no existing designs or rules**
- **Very expensive evaluation (90 sec. /design)**
- **Synthesized designs showed generality**

Training evaluations (2 of 4)



Level terrain

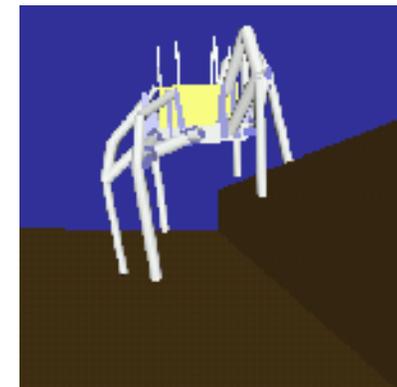


30° slope

Test (post-synthesis) evaluations



Rough terrain



1 leg on  
75cm step

# Design Rules via Synthesis

**Use synthesis to create optimized designs in new application area (inflatable-wheeled Mars rovers)**

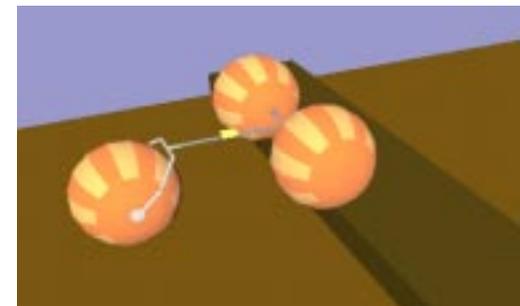
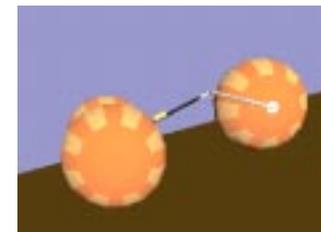
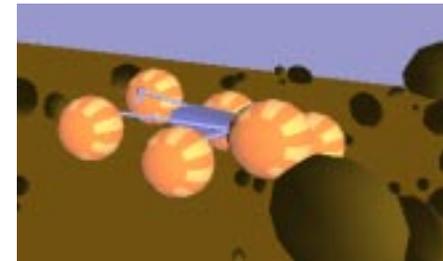
- No existing design rules or scaling laws in domain
- Performance limits unknown

**Extract design rules from space of Pareto-optimal configurations**

- e.g. Structural mass for a 6-wheeled rover is

$$K(\text{wheelbase})(\text{maxObstacleSize})(\text{payload mass})^{2.2}$$

**Also perform configuration synthesis and optimization for rover to be built in 2002**



# Challenges in Automated Design for Robotics

---

- **Significant domain expertise required to use system**
  - Task-specific control and evaluation code
  - Task-specific modules
    - Structural analysis, actuator models, geometry for each new model
- **Can be hard to ensure adequate coverage in evaluation**
  - Robots are usually general-purpose or reprogrammable
  - Must carefully choose representative task used in evaluation
- **Evaluation through simulation requires a means of controlling the robot**
  - Controller must be able to deal with 10K-100K different robot designs
    - May not be any closed-form solutions
  - Often trying to get very specific, highly constrained behavior (e.g. follow trajectory within 1mm)
  - Optimal control is not generally solved problem, and deliberative planning is extremely expensive
  - Significant controller co-evolution is extremely expensive; should be feasible w/ more CPU power

# Conclusions and Observations

---

- **Automated synthesis & optimization of robots is feasible**
  - Most appropriate for configuration design (rather than conceptual or detailed)
- **Significant effort often required for new applications**
- **High dimensionality of robot performance leads to simulation complexity**
  - 95%+ of code and coding time is related to simulation, not synthesis/optimization
    - Indicates lack of maturity in existing robot simulation tools
- **Capturing all task requirements is crucial**
- **Planning/control during synthesis is still not generally solved**