AUTOMATIC DESIGN

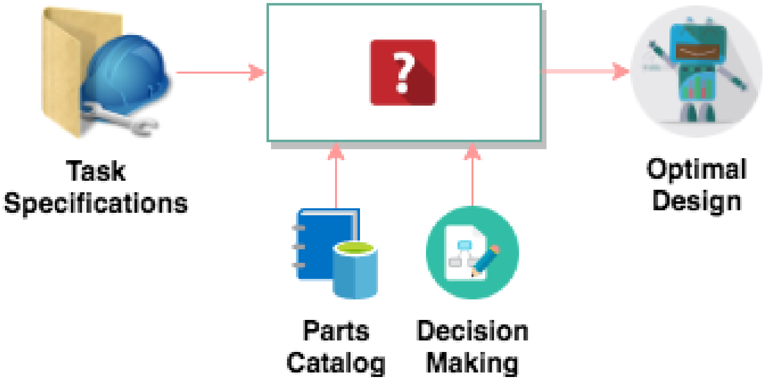
CSE 668 FINAL PRESENTATION

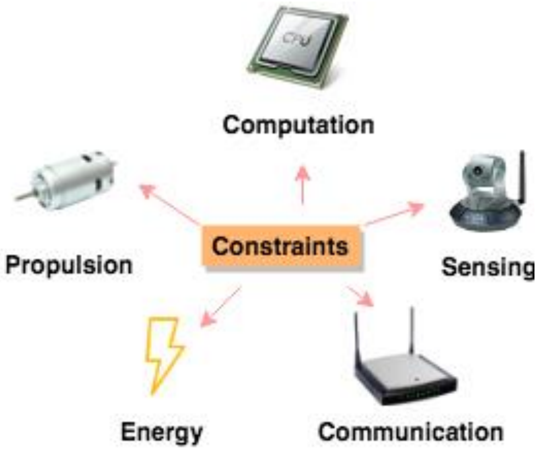
Aditya Singh Rathore

Harish Ganesan

INTRODUCTION

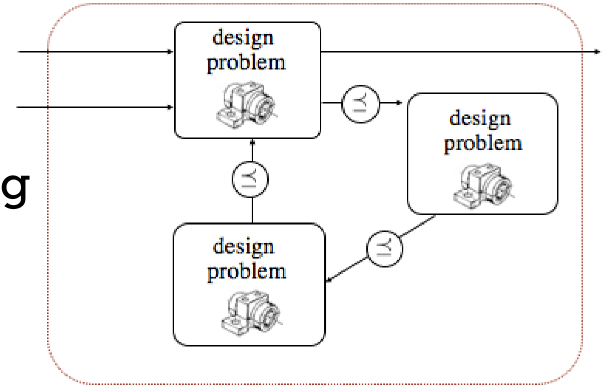
- The best robot for every task -> one that performs the task using minimal resources.





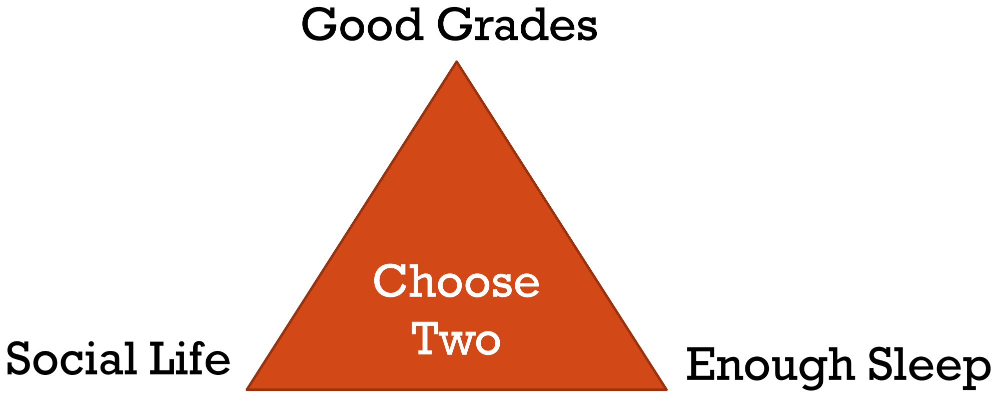
Huge Design Space!

The design problem is interconnection between the design problems according to graph structure.



BACKGROUND: PARTIALLY-ORDERED SETS

- High level: A poset is a set with a reflexive, anti-symmetric and transitive relation. ≼

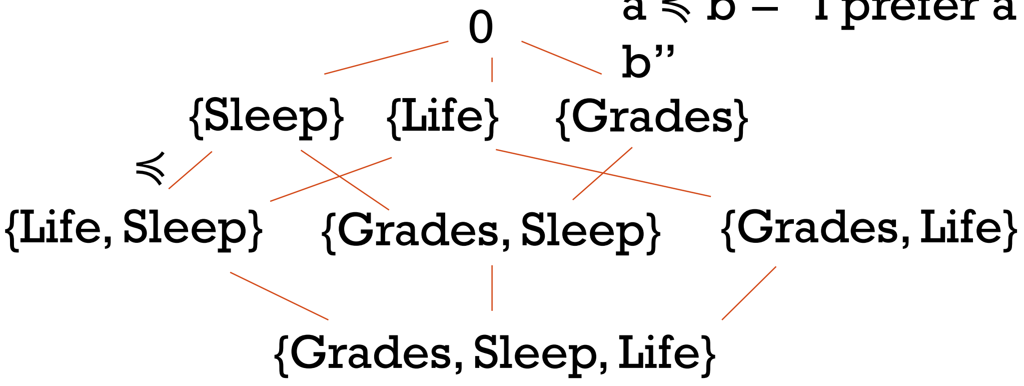


Good Grades

Choose Two

Social Life

Enough Sleep



0

{Sleep} {Life} {Grades}

≼

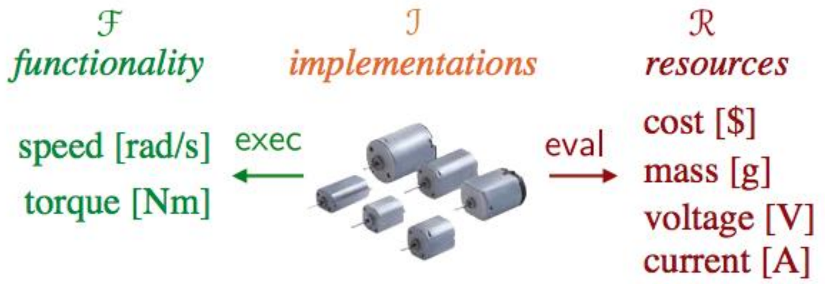
{Life, Sleep} {Grades, Sleep} {Grades, Life}

{Grades, Sleep, Life}

a ≼ b = “I prefer a to b”

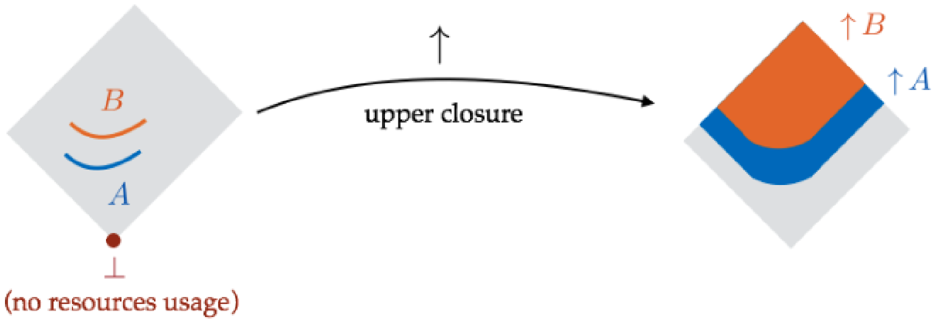
- A design problem is a relation between the required inputs (functionalities) and outputs (resources).

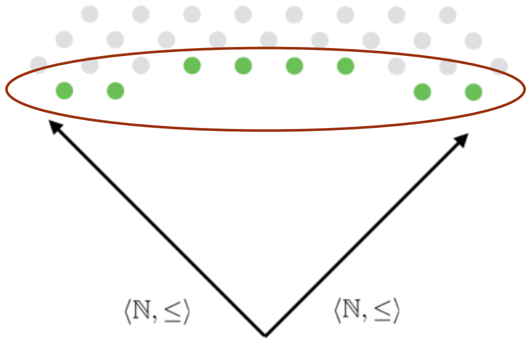
  - dp = {F, R, I, exec, eval}, where
  - F is a poset of functionalities,
  - R is poset of resources,
  - I is a set of implementations.



BACKGROUND: MINIMAL SOLUTION

- Antichains is subset of a partially ordered sets such that any two distinct elements in the subset are incomparable.





$\langle \mathbb{N}, \leq \rangle$
$\langle \mathbb{N}, \leq \rangle$

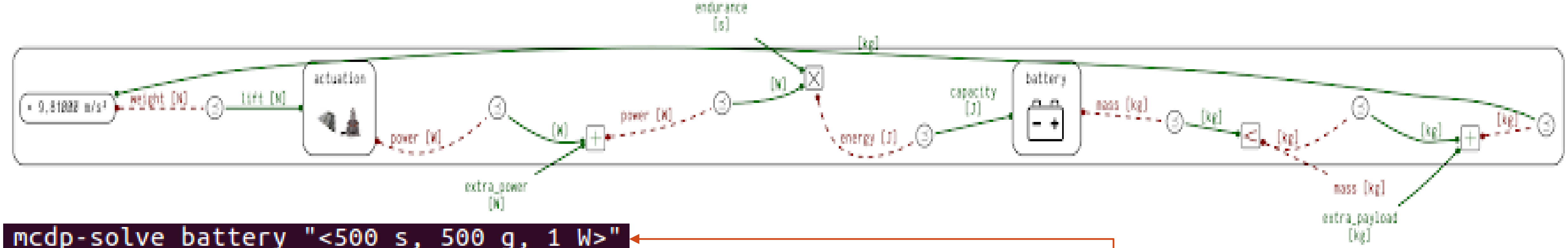
Minimal elements

- A design problem is “monotone” if increasing the recourses available or decreasing the functionality required, will never decrease the number of feasible solutions.

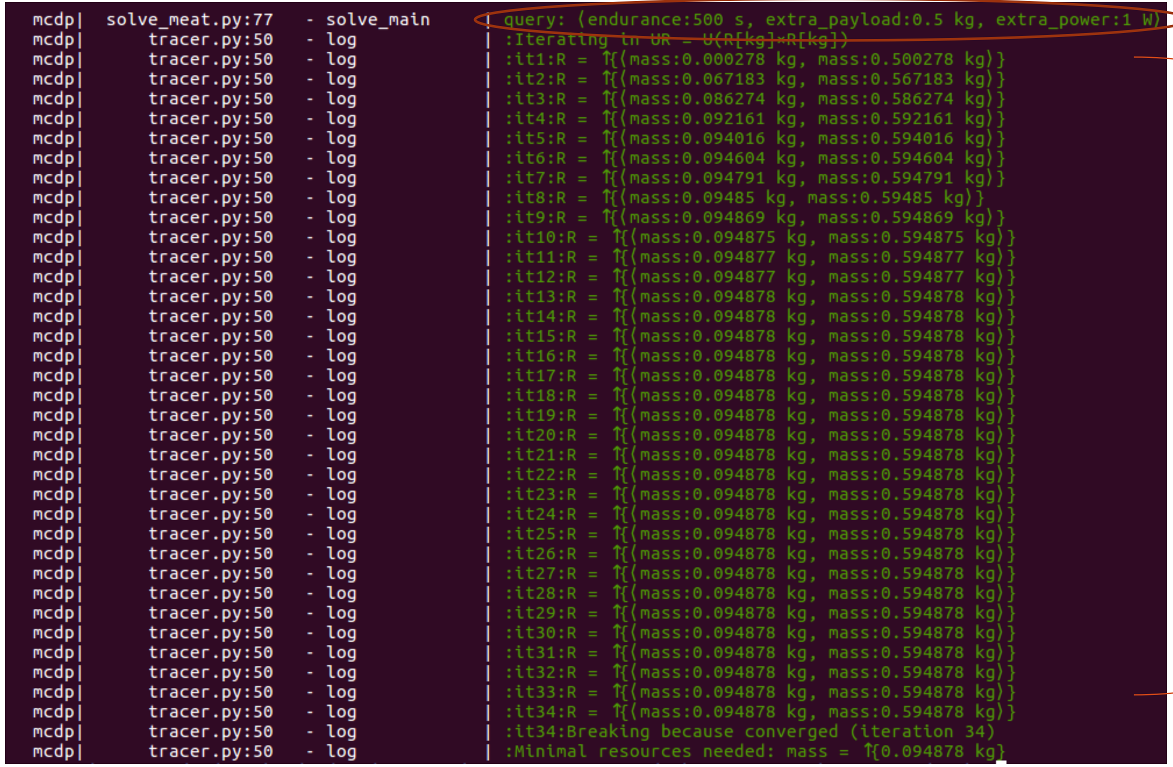
GOALS FROM LAST TIME

- The goals were:
- Running the code supplied in the MCDPL software to fully understand the inputs/outputs and working of the software.
- Take a subset of parts from Pololu and build a small library for ground-based robots.

OVERVIEW OF MCDPL SOFTWARE



```
mcdp-solve battery "<500 s, 500 g, 1 W>"
```

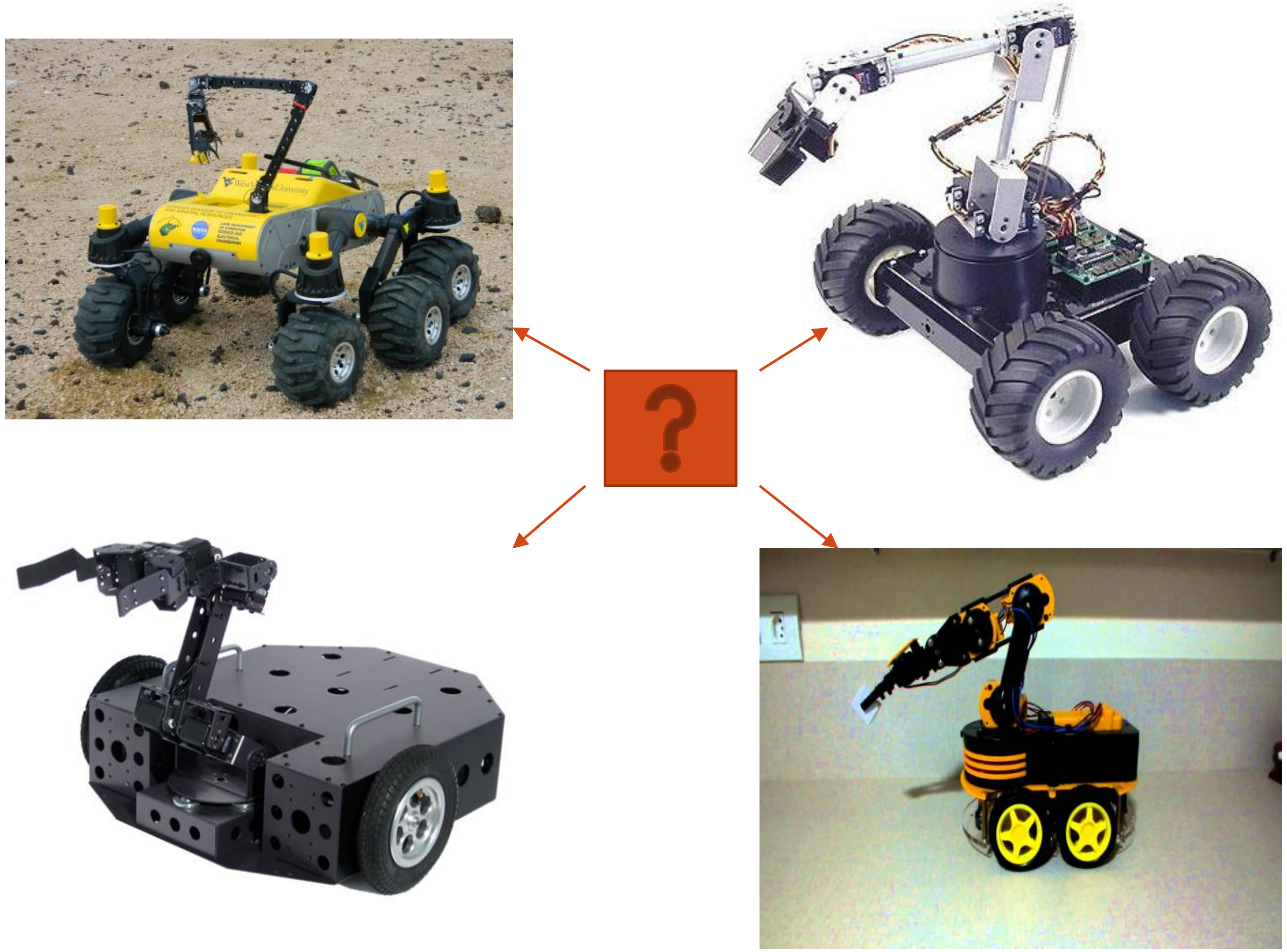


```
mcdp|   solve_meat.py:77   - solve_main          | query: ⟨endurance:500 s, extra_payload:0.5 kg, extra_power:1 W⟩
mcdp|       tracer.py:50   - log                 | :Iterating in UR = U(R[kg]×R[kg])
mcdp|       tracer.py:50   - log                 | :it1:R =  ↑{⟨mass:0.000278 kg, mass:0.500278 kg⟩}
mcdp|       tracer.py:50   - log                 | :it2:R =  ↑{⟨mass:0.067183 kg, mass:0.567183 kg⟩}
mcdp|       tracer.py:50   - log                 | :it3:R =  ↑{⟨mass:0.086274 kg, mass:0.586274 kg⟩}
mcdp|       tracer.py:50   - log                 | :it4:R =  ↑{⟨mass:0.092161 kg, mass:0.592161 kg⟩}
mcdp|       tracer.py:50   - log                 | :it5:R =  ↑{⟨mass:0.094016 kg, mass:0.594016 kg⟩}
mcdp|       tracer.py:50   - log                 | :it6:R =  ↑{⟨mass:0.094604 kg, mass:0.594604 kg⟩}
mcdp|       tracer.py:50   - log                 | :it7:R =  ↑{⟨mass:0.094791 kg, mass:0.594791 kg⟩}
mcdp|       tracer.py:50   - log                 | :it8:R =  ↑{⟨mass:0.09485 kg, mass:0.59485 kg⟩}
mcdp|       tracer.py:50   - log                 | :it9:R =  ↑{⟨mass:0.094869 kg, mass:0.594869 kg⟩}
mcdp|       tracer.py:50   - log                 | :it10:R =  ↑{⟨mass:0.094875 kg, mass:0.594875 kg⟩}
mcdp|       tracer.py:50   - log                 | :it11:R =  ↑{⟨mass:0.094877 kg, mass:0.594877 kg⟩}
mcdp|       tracer.py:50   - log                 | :it12:R =  ↑{⟨mass:0.094877 kg, mass:0.594877 kg⟩}
mcdp|       tracer.py:50   - log                 | :it13:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it14:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it15:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it16:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it17:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it18:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it19:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it20:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it21:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it22:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it23:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it24:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it25:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it26:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it27:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it28:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it29:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it30:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it31:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it32:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it33:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it34:R =  ↑{⟨mass:0.094878 kg, mass:0.594878 kg⟩}
mcdp|       tracer.py:50   - log                 | :it34:Breaking because converged (iteration 34)
mcdp|       tracer.py:50   - log                 | :Minimal resources needed: mass =  ↑{0.094878 kg}
```

Iterations

Minimum mass of resources = 0.094 kg

GROUND BASED ROBOTS

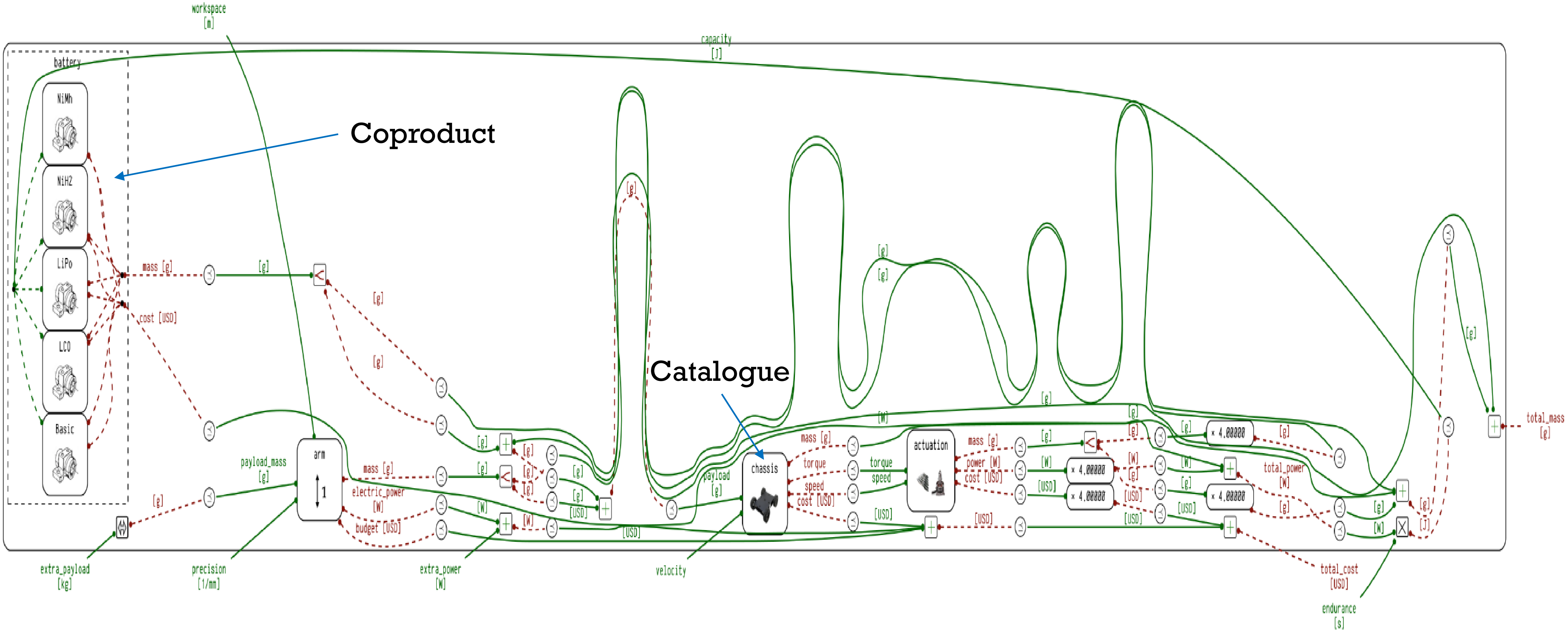


These robots are increasingly used for disaster prone area, exploration, lab environments etc.

SMALL-SCALE LIBRARY

Parts	Type1	Type2	Type3	Type4	Type5	Function-ality provided	Resources required
Battery	NiMh	NiH2	LiPo	LCO	NiCad	Capacity	Mass, Cost
Motor		<5 Motors with varying parameters>				Torque, Speed	Power, Mass, Cost
Chassis	4 Wheel Drive Basic	4 Wheel Drive ATV	Nomad 4 Wheel Drive off-road chassis			Payload, Velocity	Torque, Speed, Mass, Cost
Robotic Arm	UArm	UArmPro	Dobot	PhantomX	PhantomX Reactor	Workspace, Payload, Precision	Power, Mass, Cost

SYSTEM DESIGN

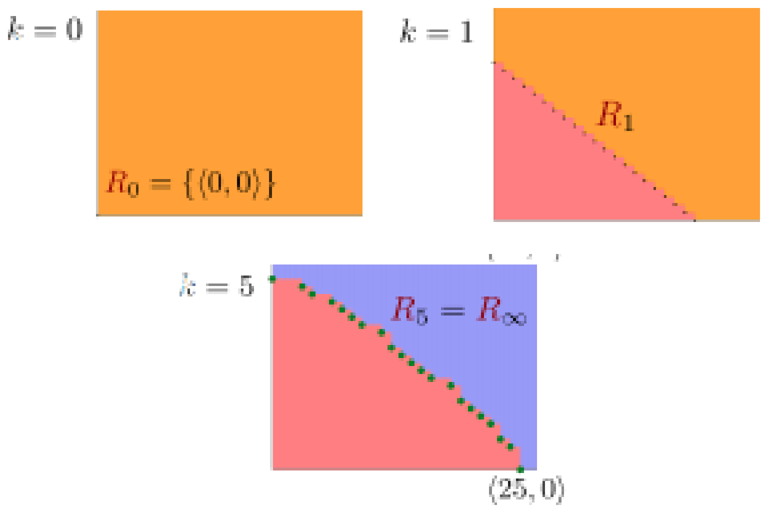


Coproduct

Catalogue

ITERATIONS

- Kleene Ascent: Is used to start from the bottom and compute till the least fixed point.



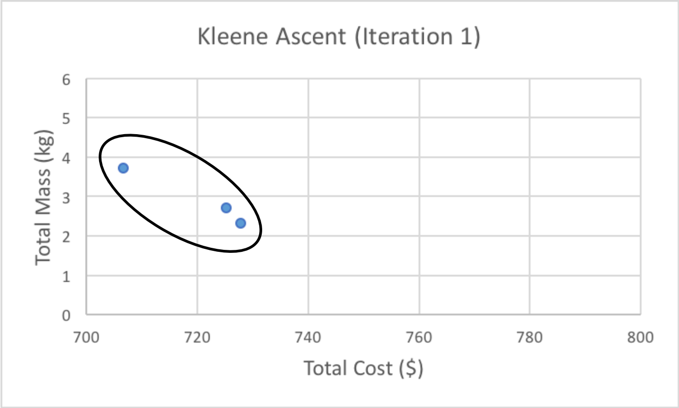
● ● ● The current antichain

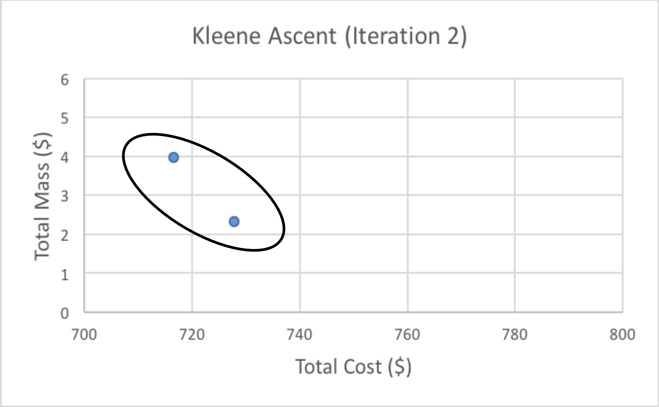
🟥 We know this area is unfeasible.

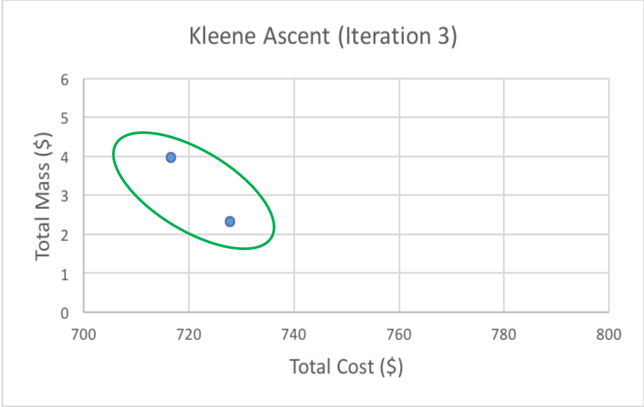
🟧 "Maybe" - there *might* be minimal solutions here.

🟢 🟢 These points are minimal solutions.

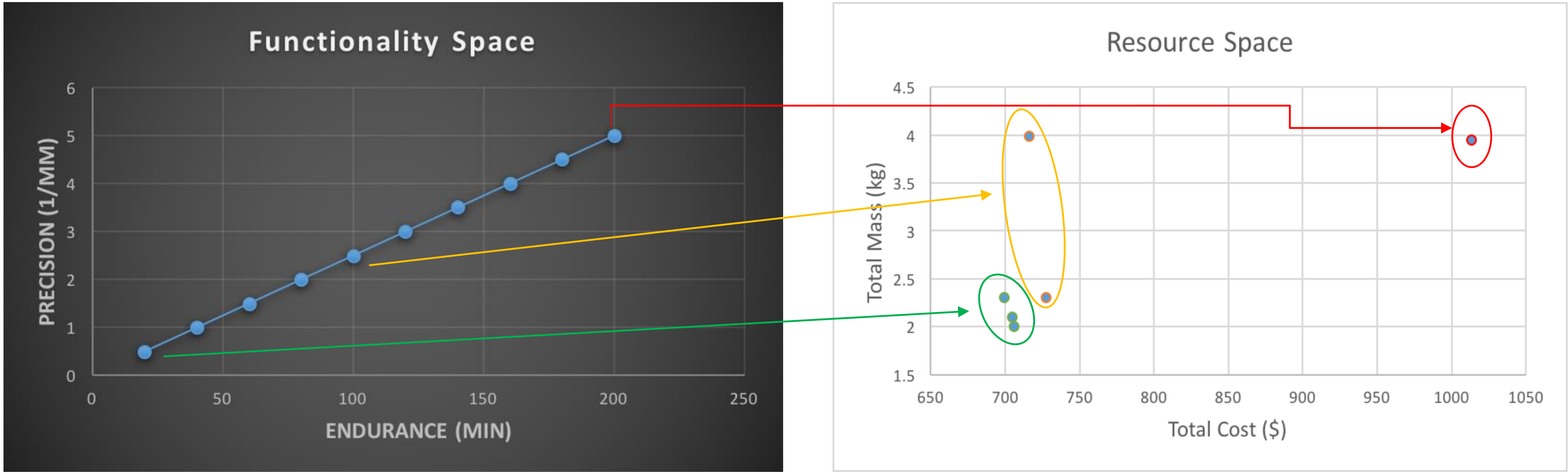
🟦 "Don't care" - Not necessarily feasible, but we know that there are no feasible points that are not dominated by the minimal points already found.



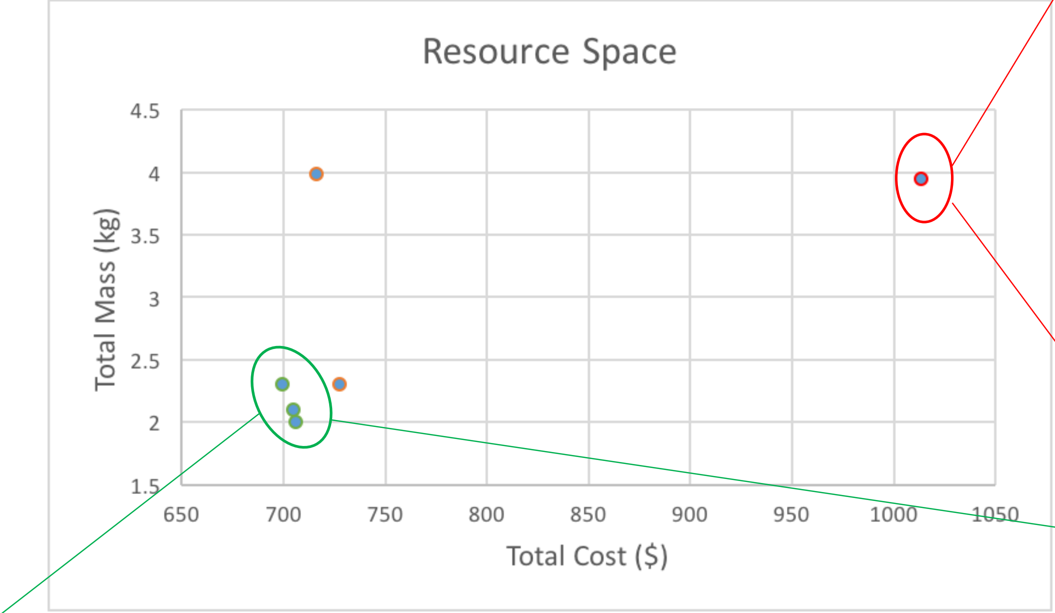


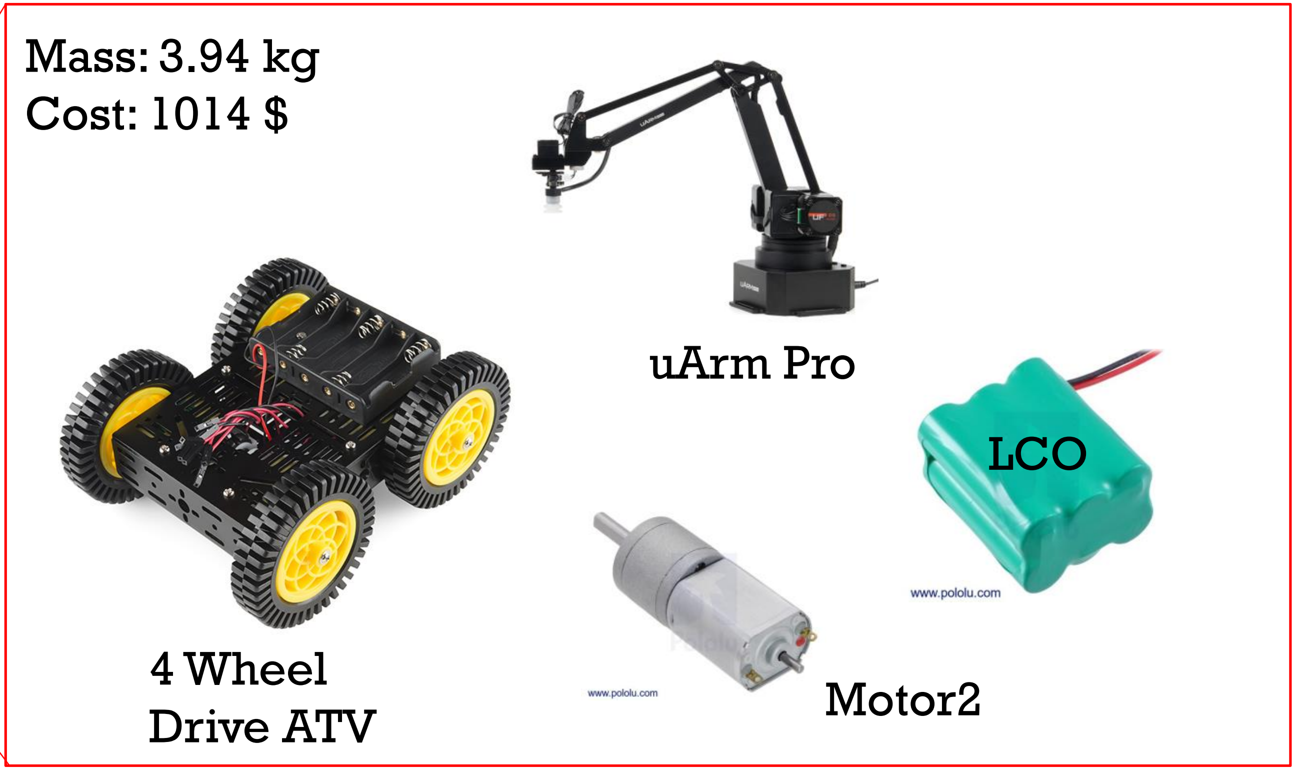


TRADE-OFF CURVE



RESULTS





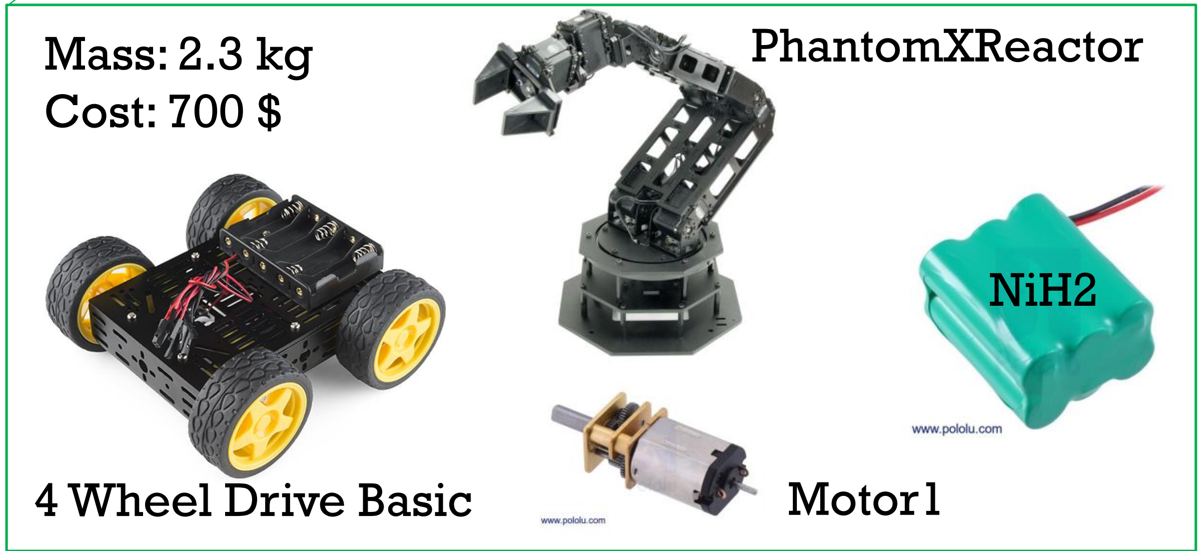
Mass: 3.94 kg
Cost: 1014 $

uArm Pro

LCO

4 Wheel
Drive ATV

Motor2



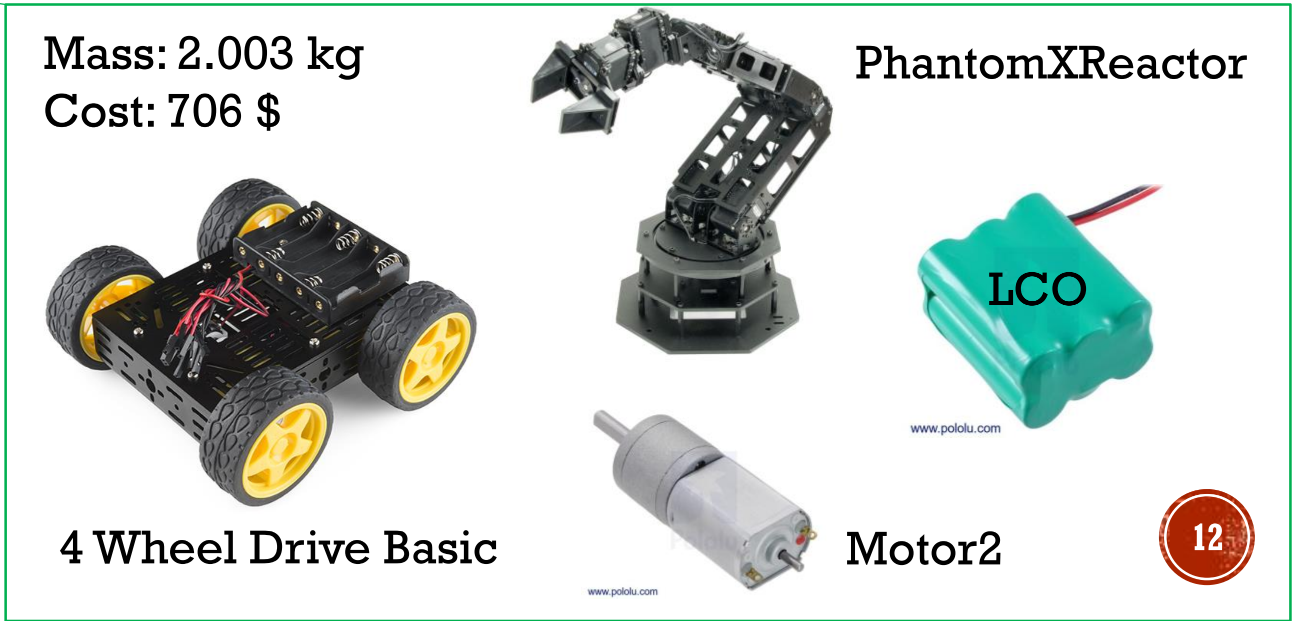
Mass: 2.3 kg
Cost: 700 $

PhantomXReactor

NiH2

4 Wheel Drive Basic

Motor1



Mass: 2.003 kg
Cost: 706 $

PhantomXReactor

LCO

4 Wheel Drive Basic

Motor2

CONCLUSION

- We have created a small library with parts to build Ground Based Robots
- Now, we can obtain the optimal solution given a set of constraints on the "provides" of the robot
- Future enhancements could be to add new models for robots.
- The size of the catalogue of parts could also be increased
- Overall, this method can be used not only to design robots, but any design problem could eventually be tackled in this manner.
- The code will be posted on Github!

THANK YOU!