Evolutionary Algorithms for Efficient and Generalizable Tuning of Machine Control

Victor Parque

Hiroshima University

2025

Towards Living Organisms - Machines

“Build”

“Compute”

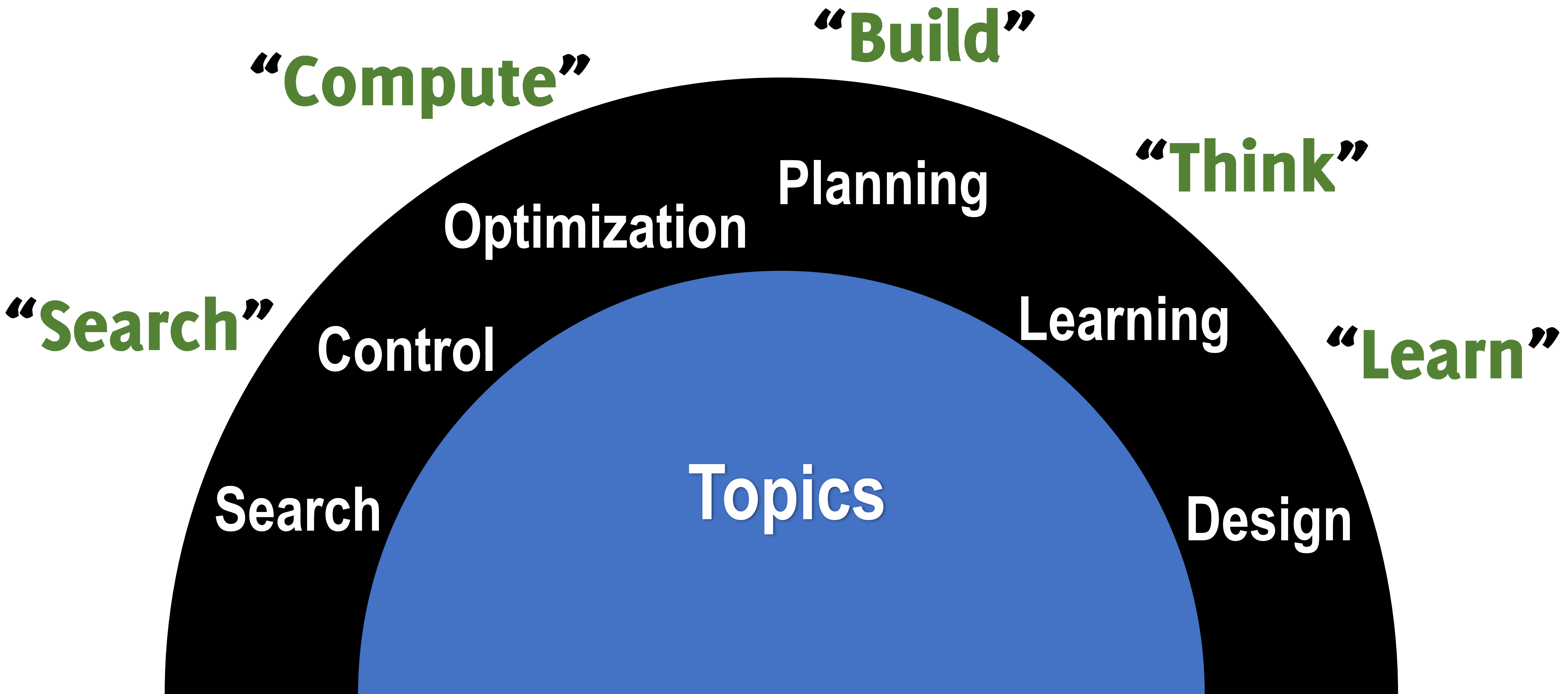
“Think”

Organism

“Search”

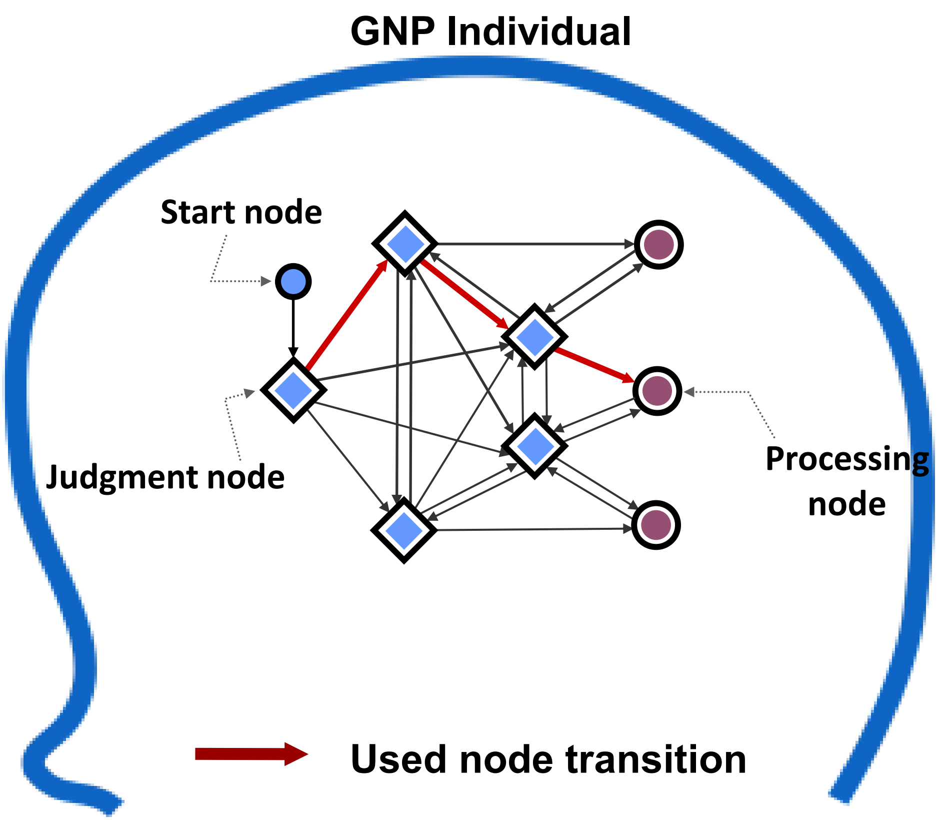
“Learn”

Towards Living Organisms - Machines



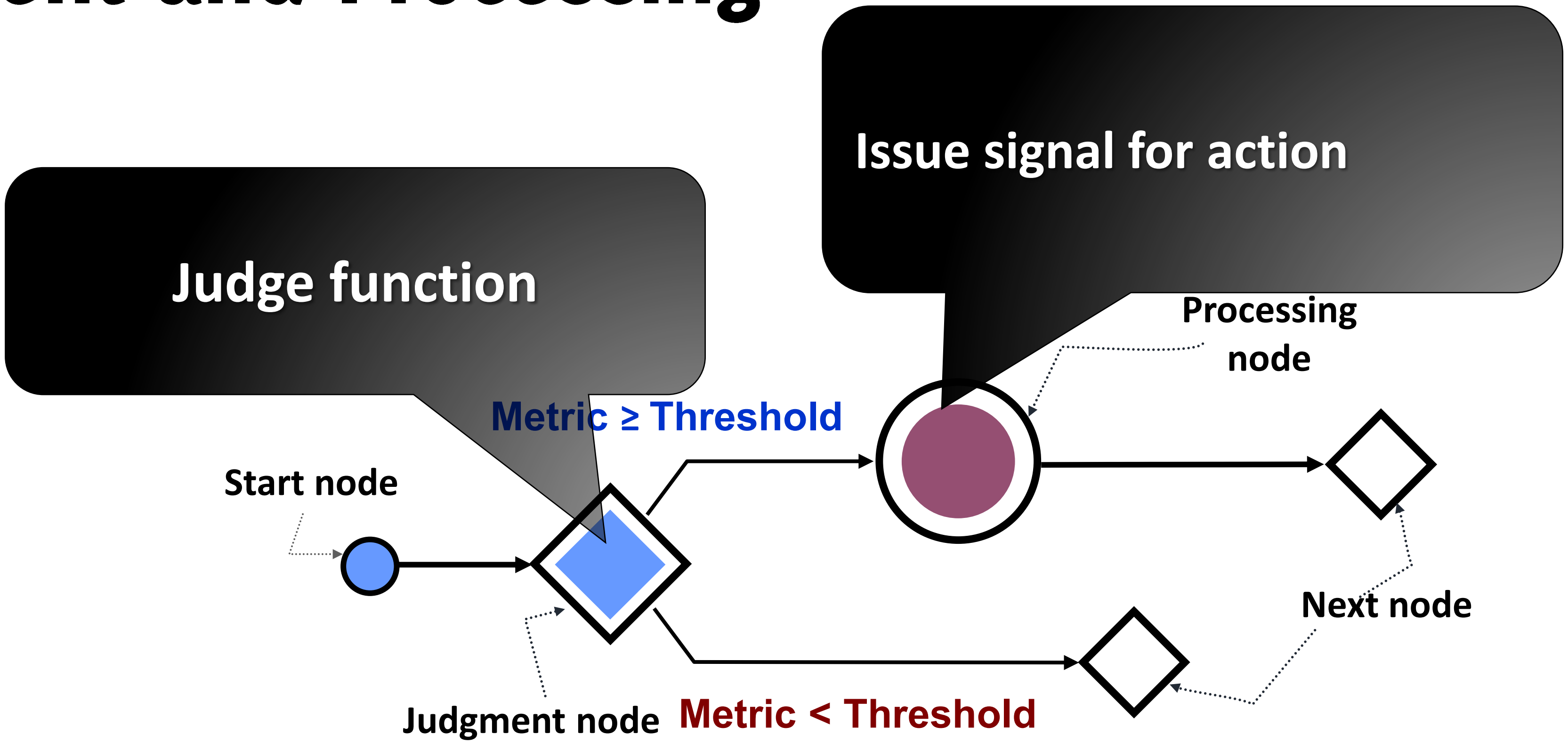
Genetic Network Programming (GNP)

- **Evolutionary** optimization algorithm
  - *Population based*
  - *Selection, crossover, mutation*
- Phenotype is a directed **graph**
  - *GP handles trees.*
  - *GA handles strings.*
- Uses **judgement** and **processing** nodes.
  - *Effective for decission making.*

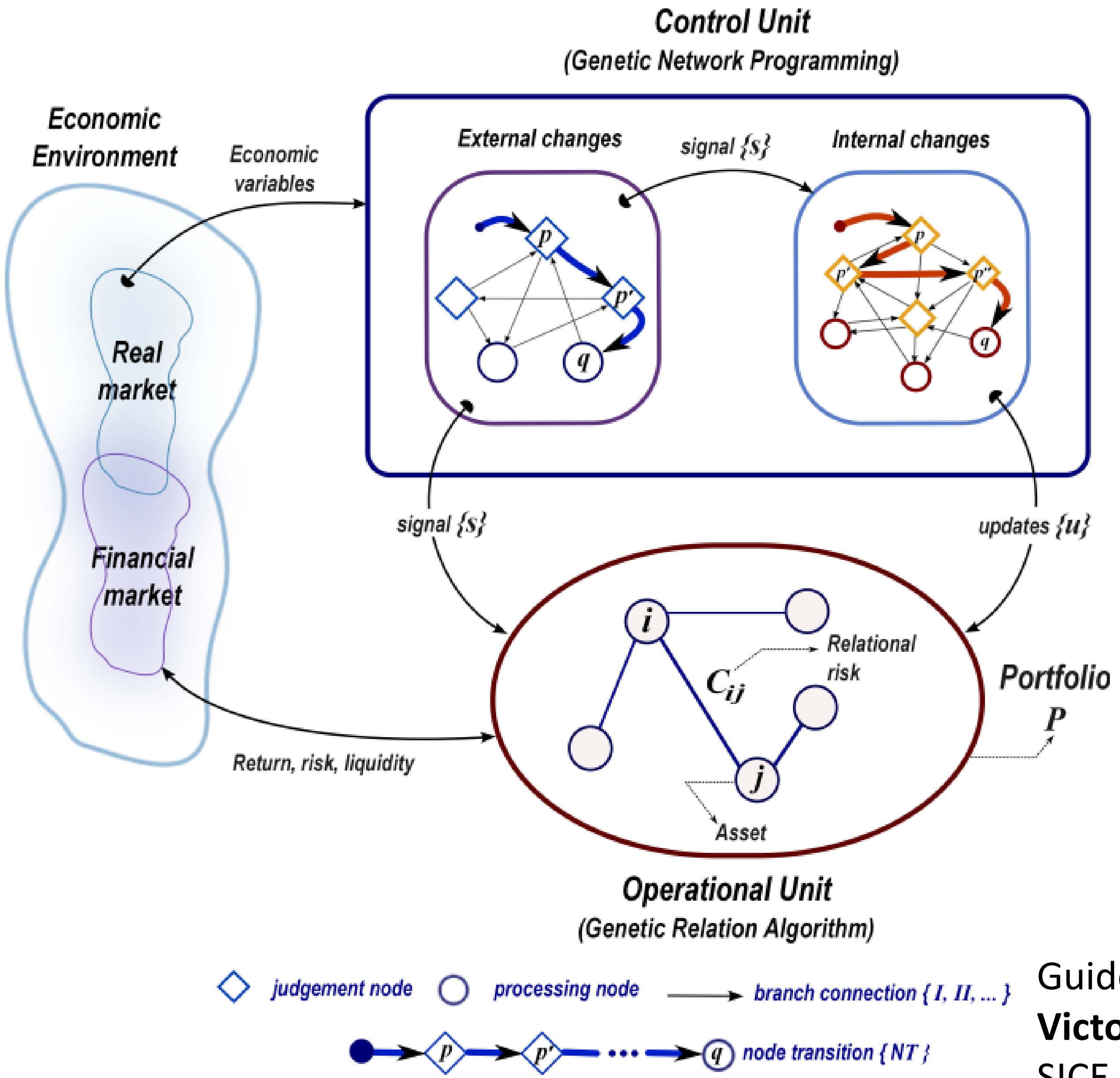


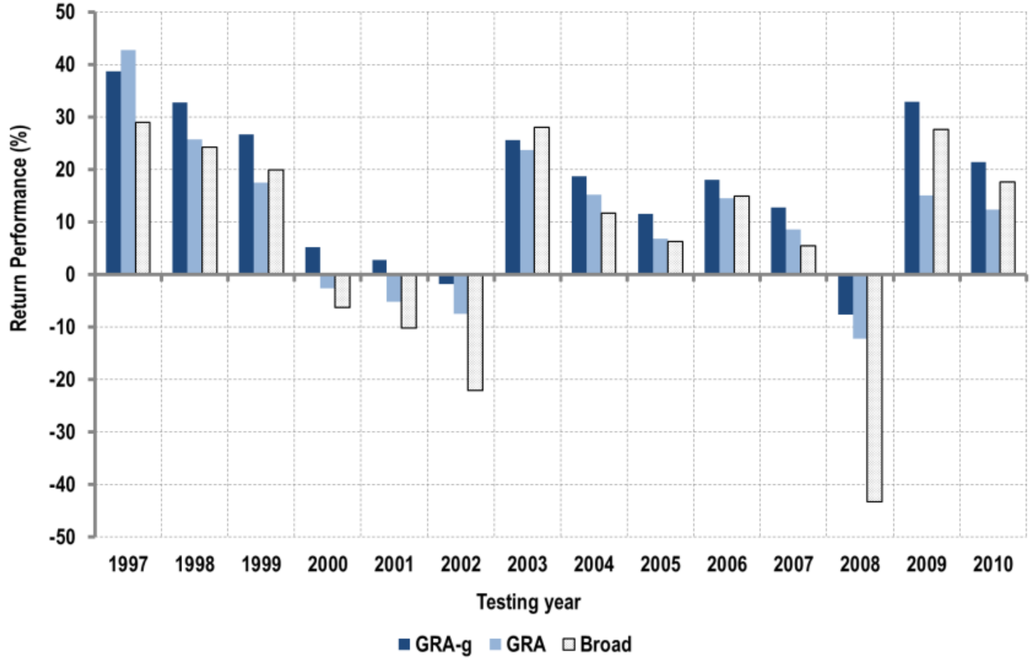
Shingo Mabu, **Kotaro Hirasawa**, Jinglu Hu
A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning
Evolutionary Computation (2007) 15 (3): 369–398, MIT Press.

Judgment and Processing



- GNP based (or other forecasting models)
  - Judgment nodes evaluate drivers that determine the states.
  - Processing nodes determine the state.
- Fitness

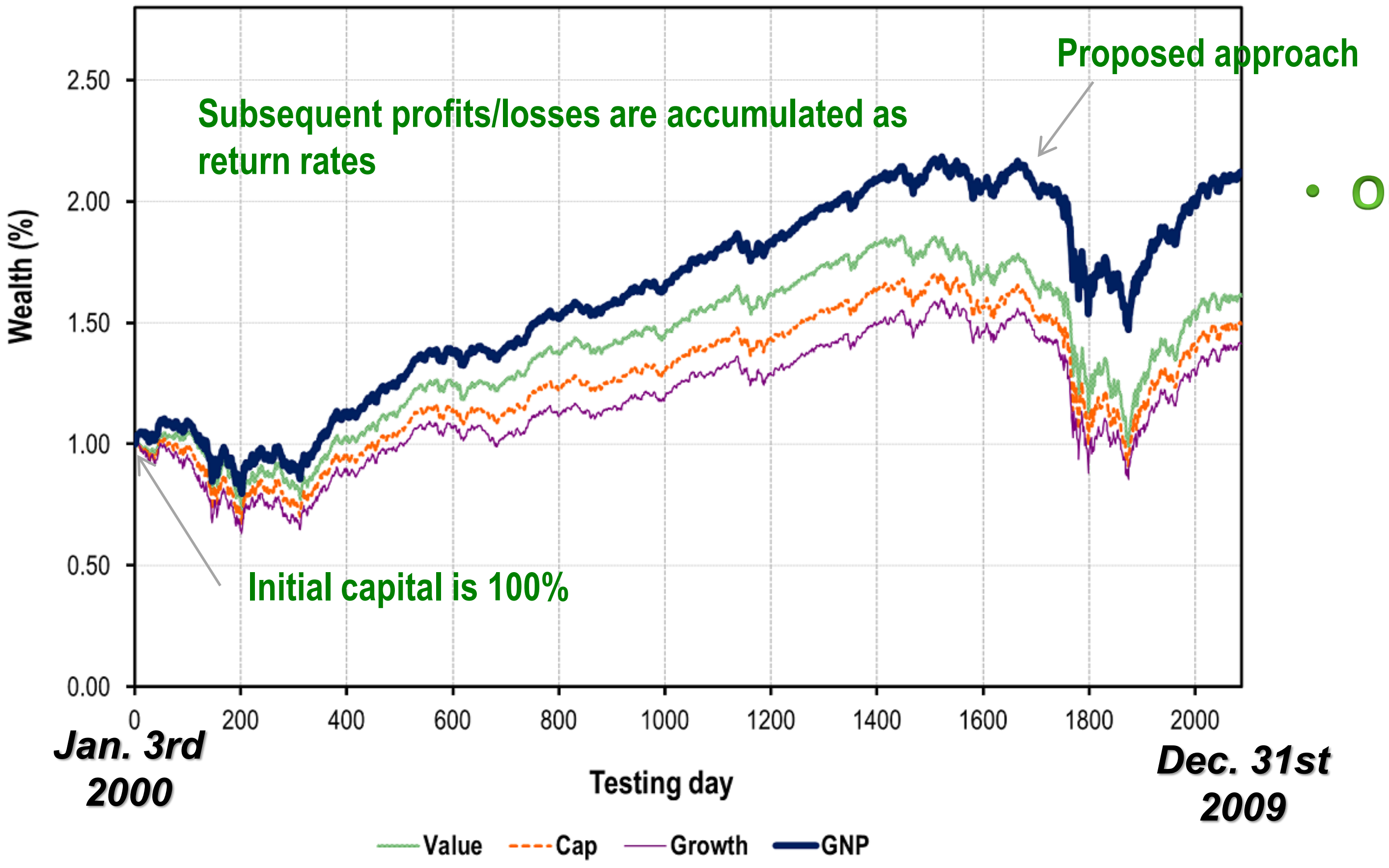




Guided Genetic Relation Algorithm on the Adaptive Asset Allocation
Victor Parque, Shingo Mabu and Kotaro Hirasawa
SICE Annual Conference, Tokyo, Japan, 2011

Comparison in the testing period

Accumulative wealth

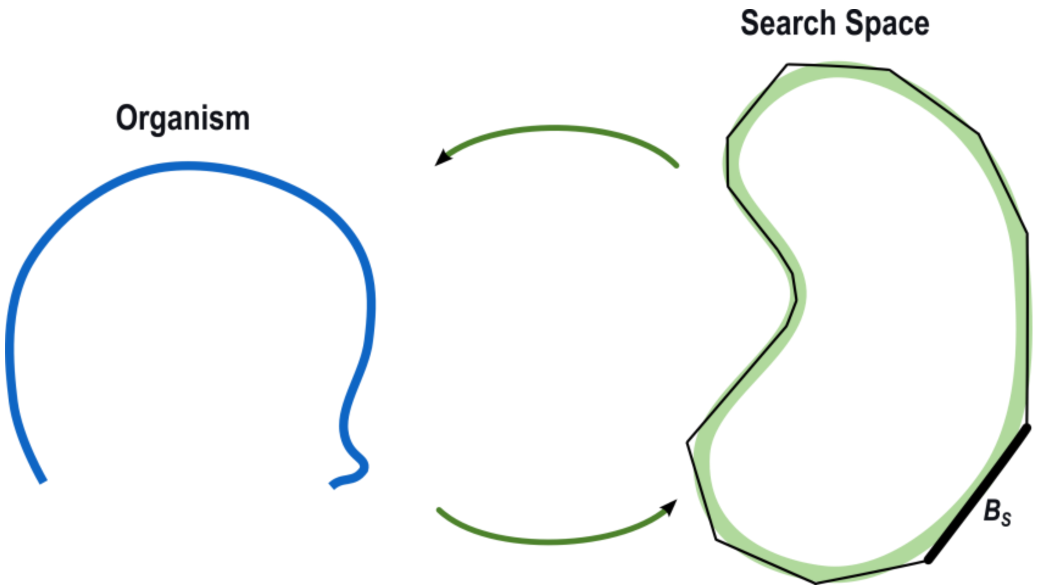


- **Outperforms the benchmarks**
  - **1.38 times better in return accumulation over the long term.**
  - **1.42 times better in return accumulation during the latest financial distress period.**

Guided Genetic Relation Algorithm on the Adaptive Asset Allocation
Victor Parque, Shingo Mabu and Kotaro Hirasawa
SICE Annual Conference, Tokyo, Japan, 2011

Search

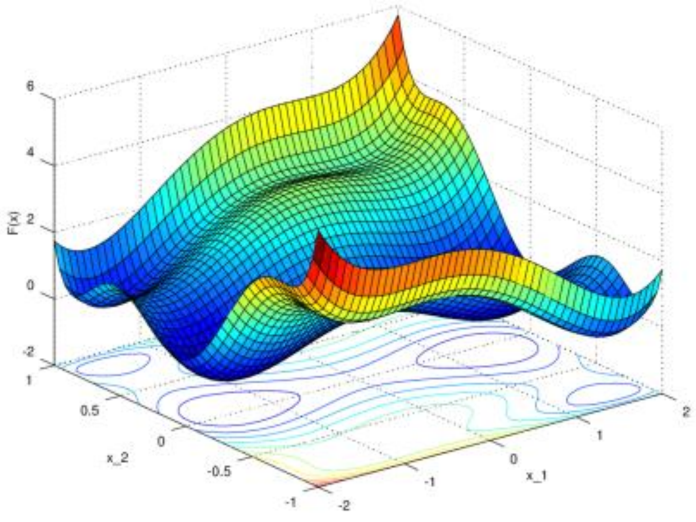
How to search for the global optimum?



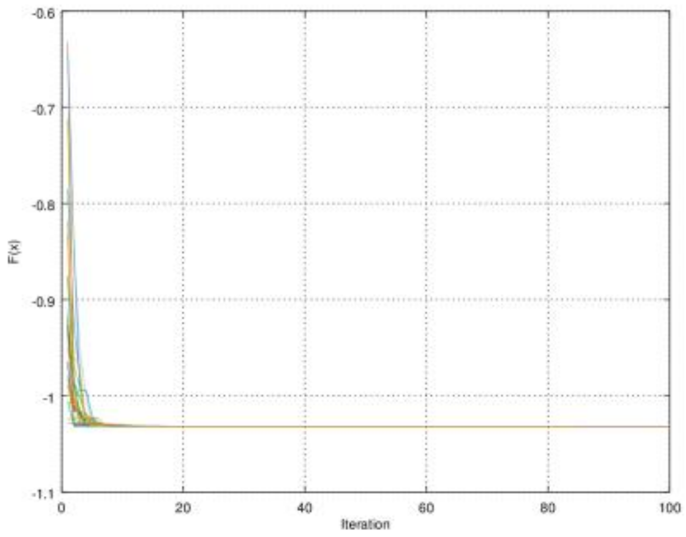
Enables to build almost anything

products, machines from scratch

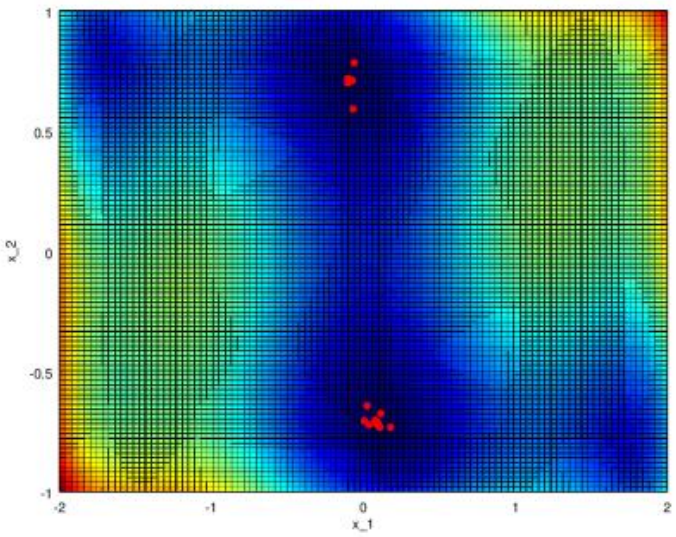
Global Optimization



Landscape

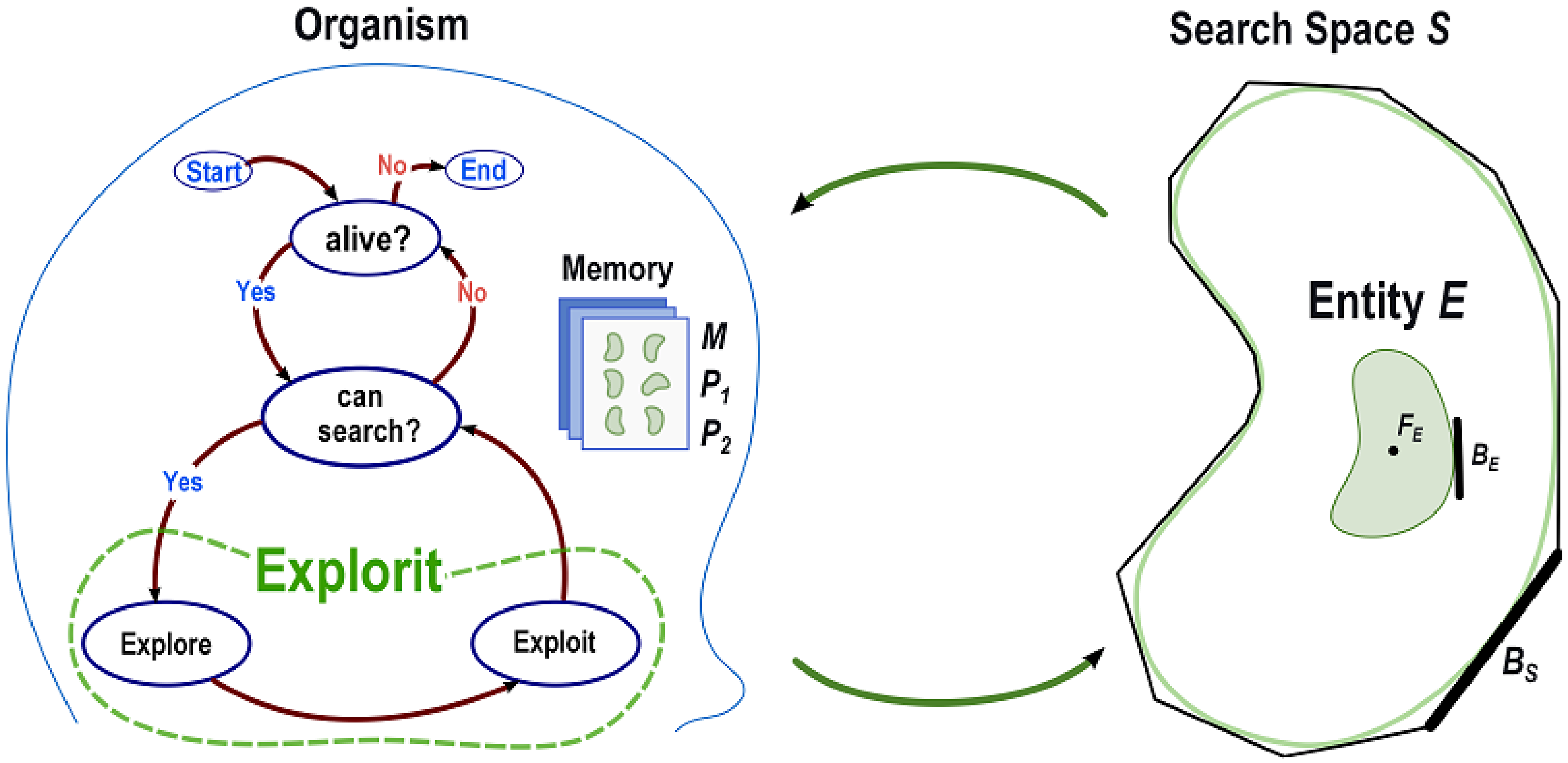


Convergence



Minimal Points

Exploration and Exploitation



INSTANCE	EXPLORIT		BENCHMARK	
	Evaluations	Performance	Evaluations	Performance
Synthetic	48±15	2.12E-6±1.41E-6	87±18	1.29E-4±1.71E-4
Multi-dimensional	3E6	1.62E9±1.5E8	3E6	1.15E11±5.12E11
Vehicle Powertrain	386±71	(5.18, 0.58, 0.24, 0.27)	1020±192	(5.59, 2.14, 0.25, 0.24)

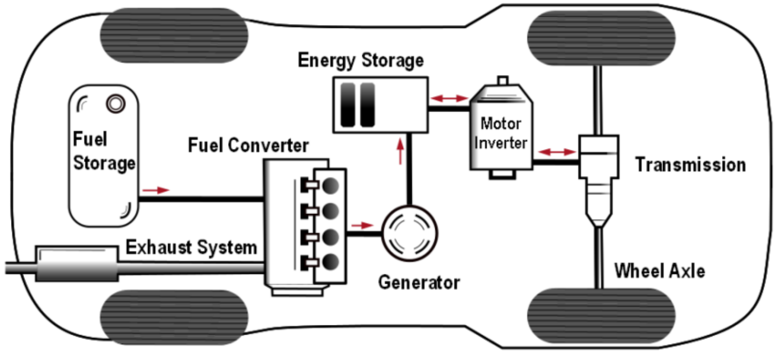
Victor Parque, Masakazu Kobayashi and Masatake Higashi. Explorit for Global Optimization
5th NIPS Workshop on Optimization for Machine Learning, Lake Tahoe, Sierra Nevada, US, 2012

Case: Vehicle Powertrains

Simulator: **Advisor** (based on real world tests)

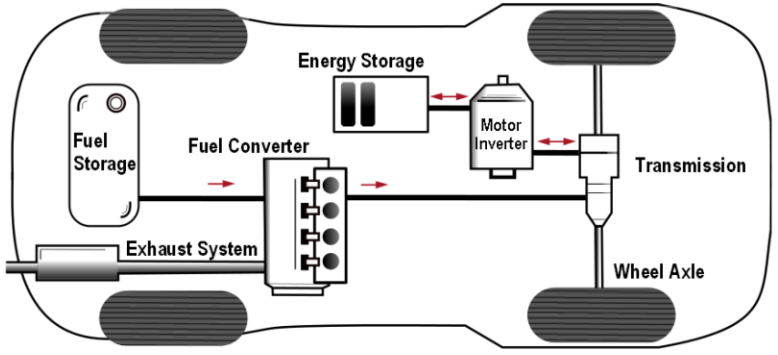
Goal: Maximize mileage, and minimize emissions

SHEV



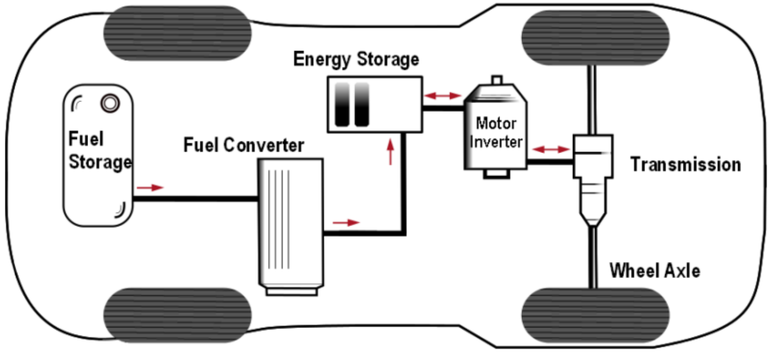
a) Series Hybrid Electric Vehicle (SHEV)

PHEV



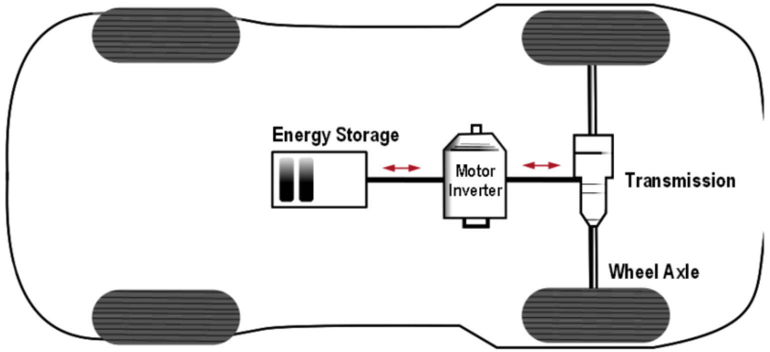
b) Parallel Hybrid Electric Vehicle (PHEV)

FCV



c) Fuel Cell Vehicle (FCV)

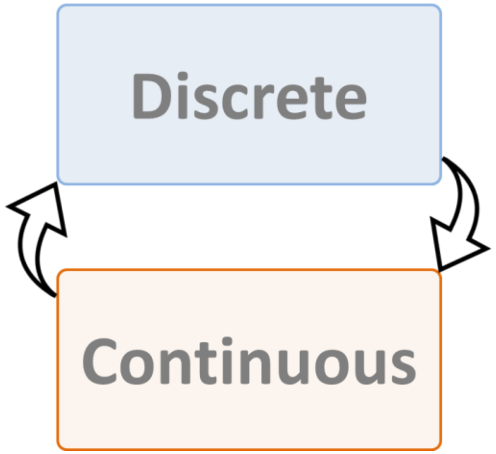
EV

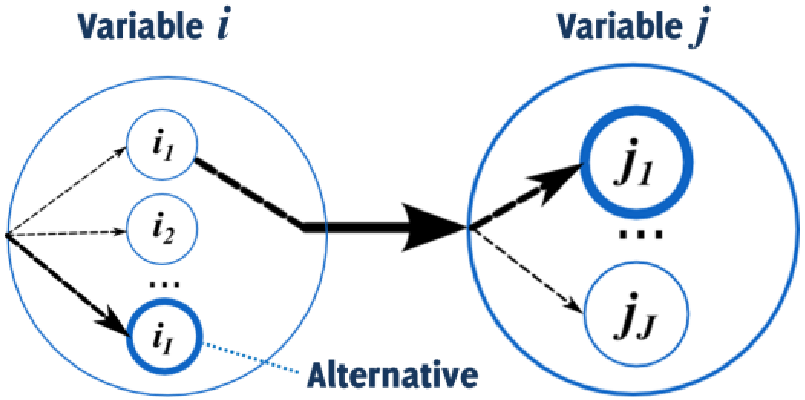


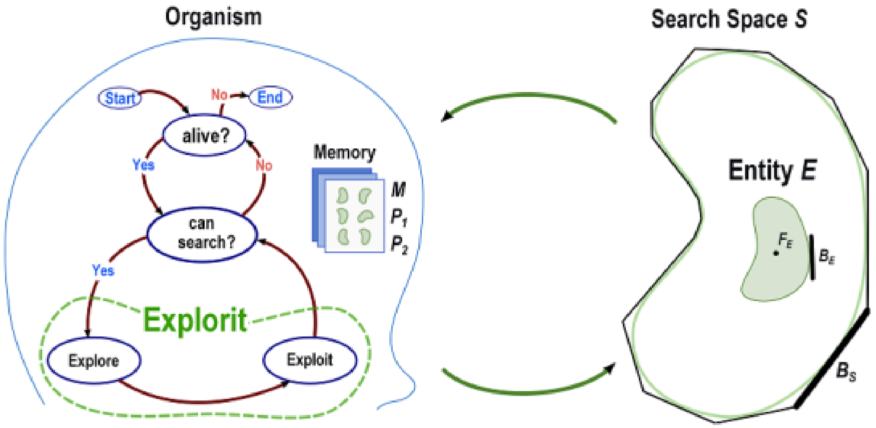
d) Electric Vehicle (EV)

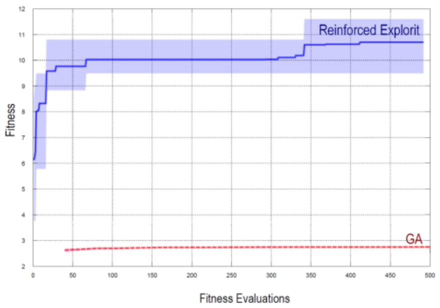
Victor Parque, Tomoyuki Miyashita: Learning the Optimal Product Design Through History. ICONIP (1) 2015: 382-389

Reinforced Explorit

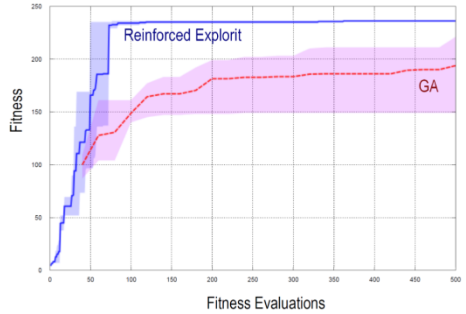




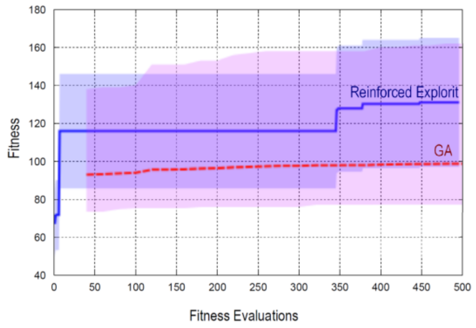




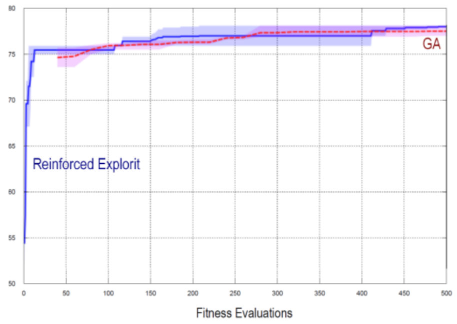
(a) Series Hybrid Electric Vehicle



(b) Parallel Hybrid Electric Vehicle



(d) Electric Vehicle

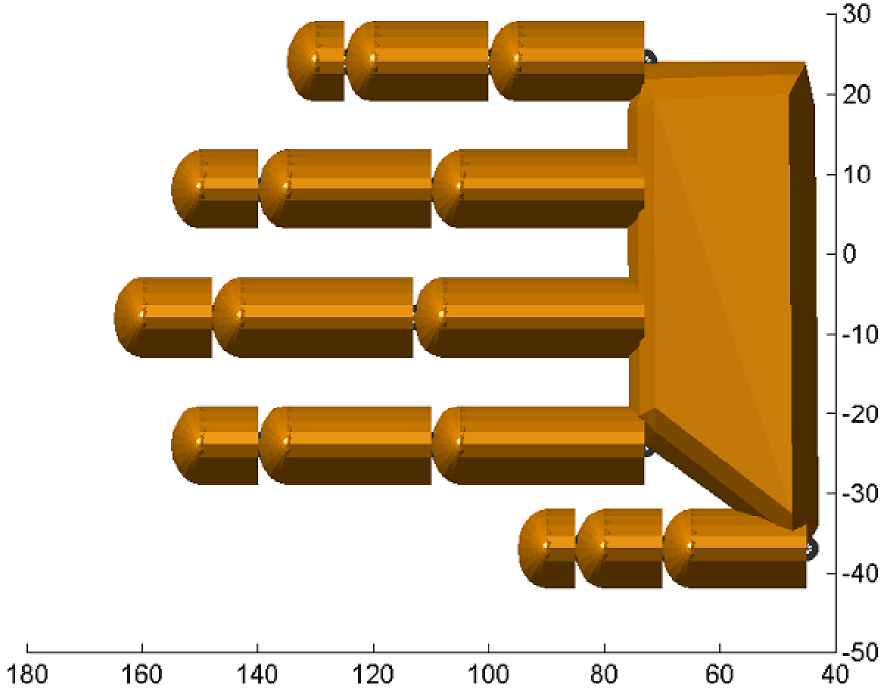


(c) Fuel Cell Vehicle

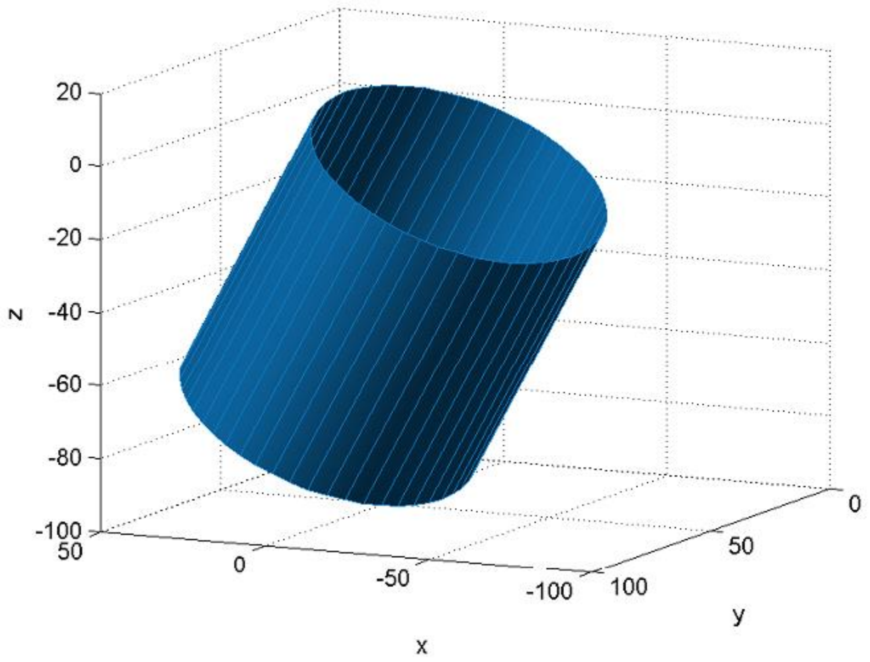
Serial HEV **Parallel HEV** **Electric** **Fuel Cell**

Neural Computing with Concurrent Synchrony

Robotic Hand

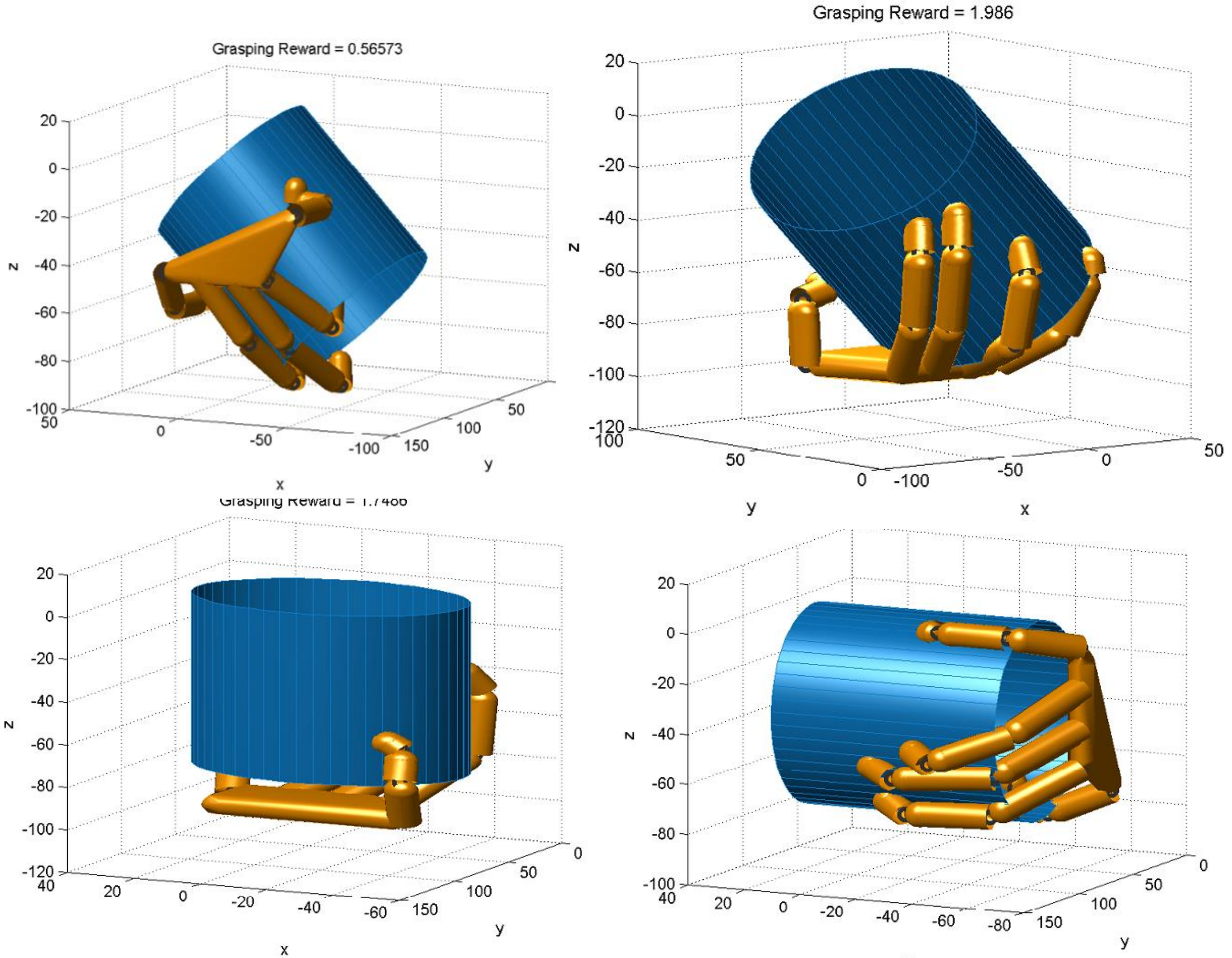


Object to Grasp



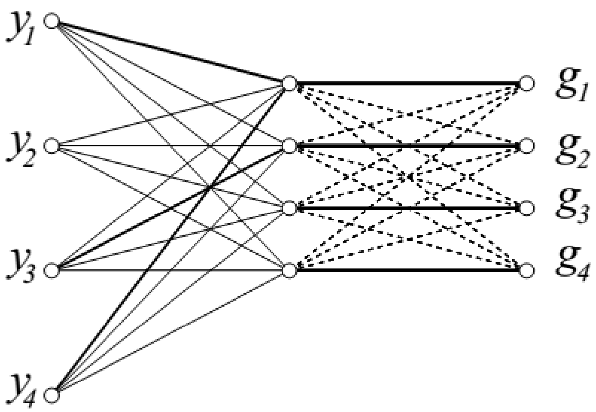
Victor Parque, Masakazu Kobayashi, Masatake Higashi:
Neural Computing with Concurrent Synchrony. ICONIP (1) 2014: 304-311

Grasping Results



Object

Action



Excitatory

Direct Transition

Inhibitory

Indirect Transition

A Study of Fairness Functionals for Smooth Path Planning in Mobile Robots

Victor Parque

2021

How to compute smooth curves effectively?

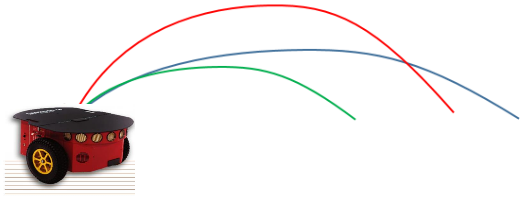
e.g. safe path planning in robots

Torque Inputs

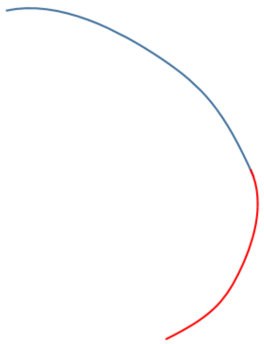
Pulse Width Modulation

2 Servo Motors

Path Primitives



Path Compounding



Dynamic Models

- ✓ Principled
- ⚠ Unrealistic (noise)

Data-driven Models

- ✓ Practical
- ⚠ Computationally intensive

Curvature

$$\boldsymbol{r}(u) = \sum_{i=0}^{n} B_i^n(u)\boldsymbol{p}_i, \quad u \in [0,1]$$

Bézier curve

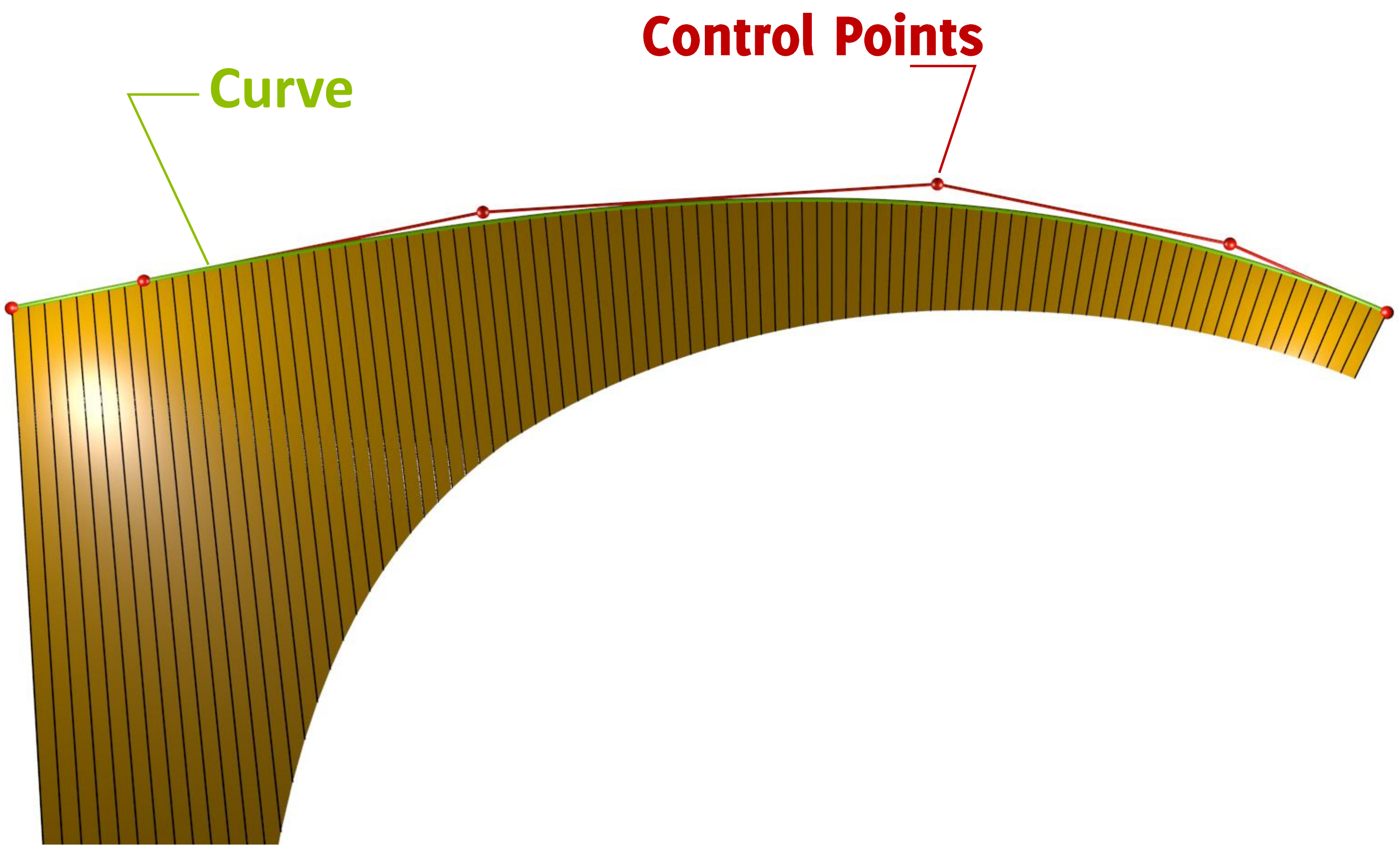
$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

Bernstein polynomial

$$\kappa = \frac{\|\boldsymbol{r}' \times \boldsymbol{r}''\|}{\|\boldsymbol{r}'\|^3}$$

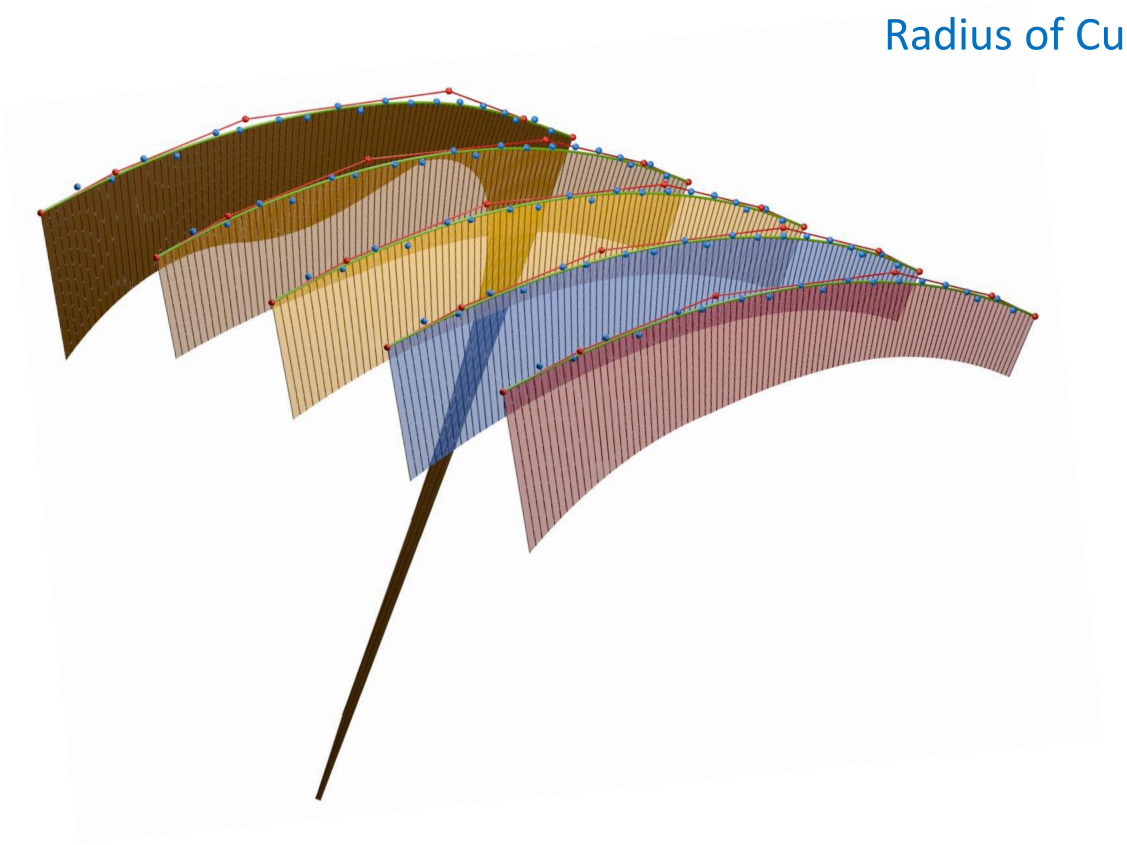
Curvature

Curvature Profile

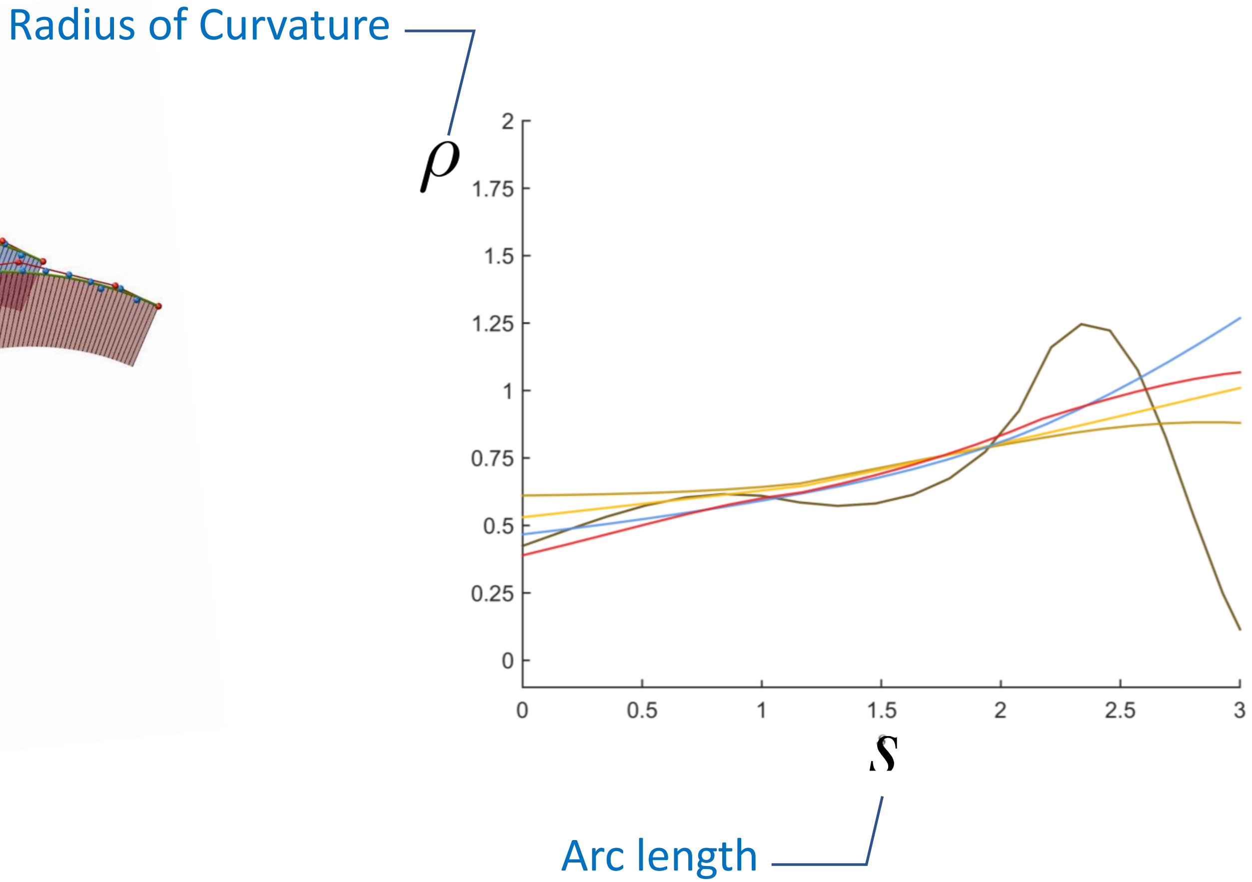


Curvature Profile
Radius of curvature

Curve profiles



Curvature Profiles



Computing Smooth Curves

$$\min_{p_i} \; F = E + \lambda H$$

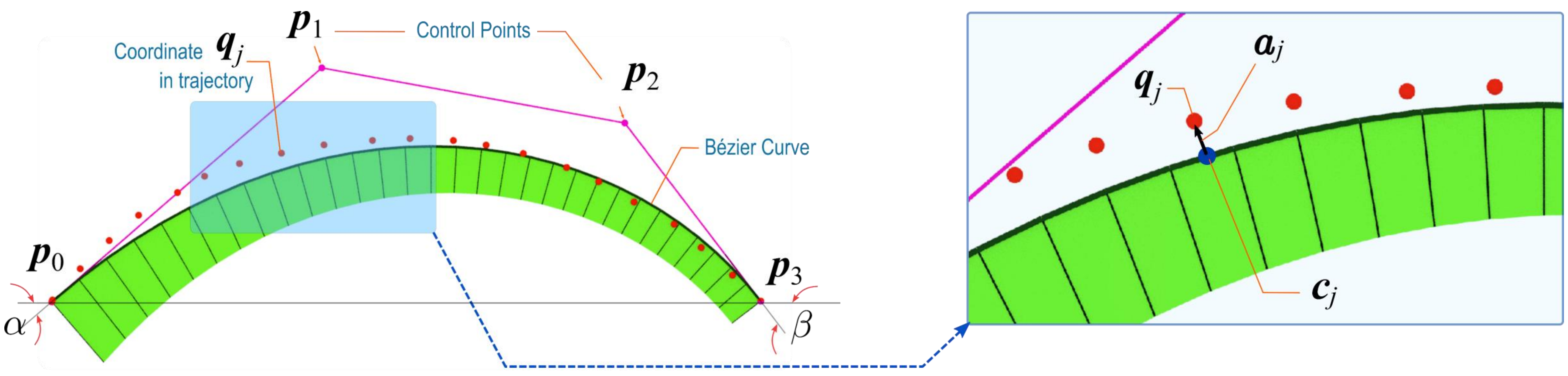
Fit to anchor points

$$E = \sum_{j=1}^{m} \|a_j\|^2$$

Fairness functional

$$H(\kappa) = \int \left(h(\kappa)\right)^2 ds$$

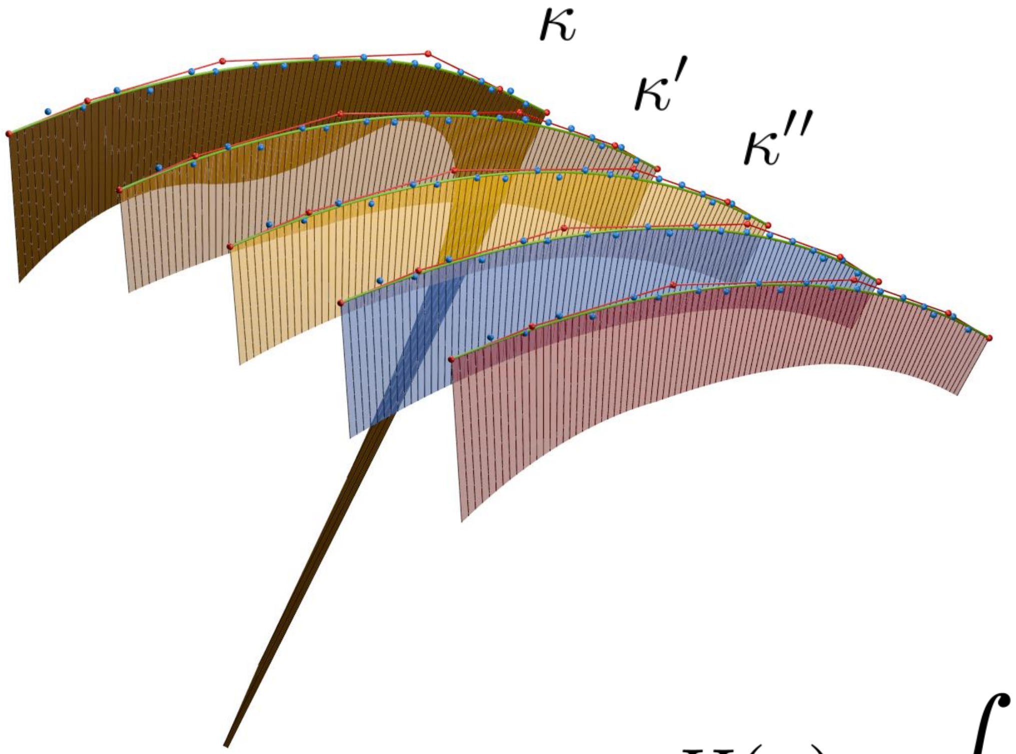
curvature **arc length**



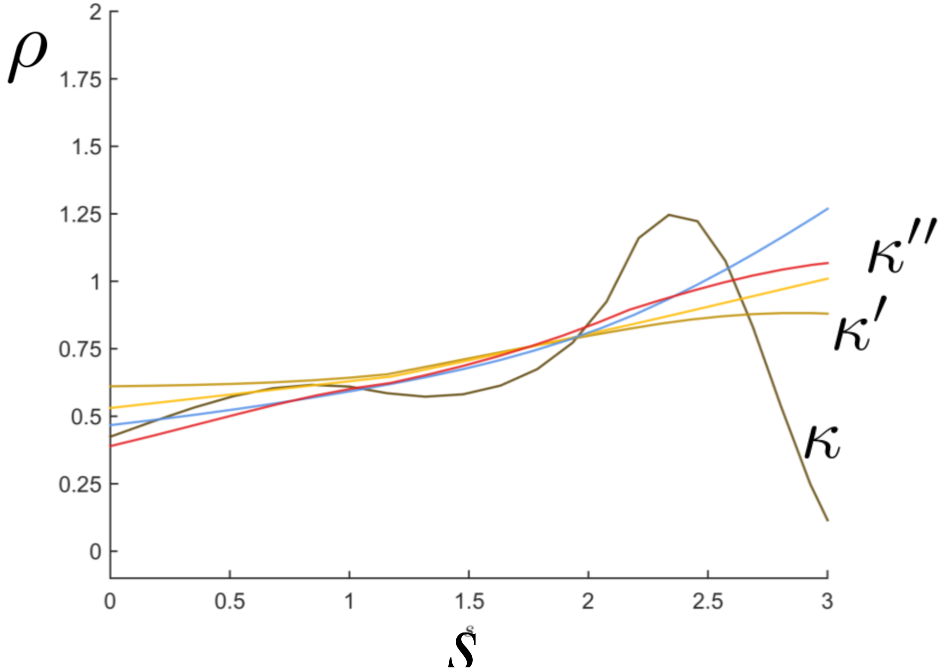
(a) Curvature Profile (b) Vector $\boldsymbol{a}_j$

Smoothness: Fairness Functionals

Curve profiles



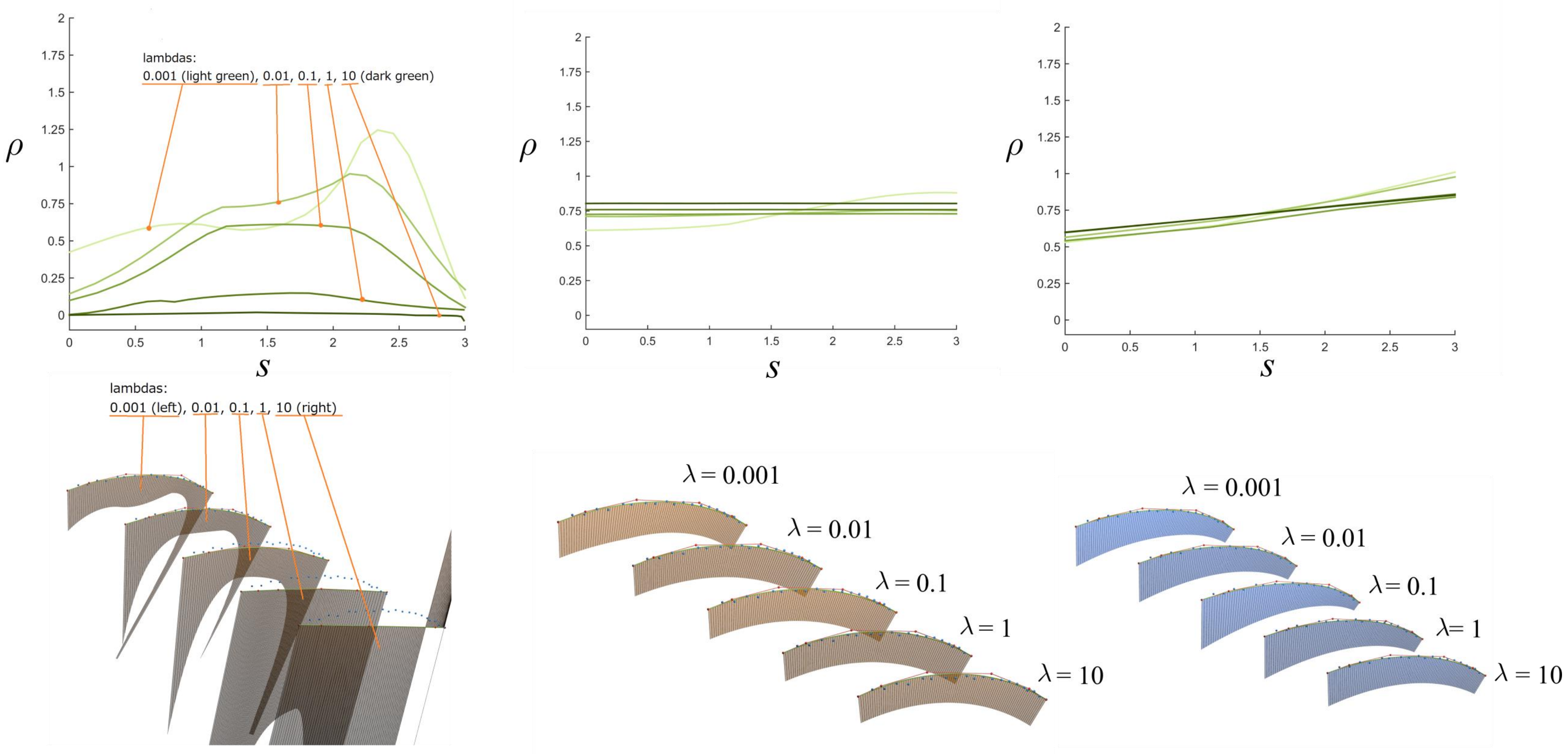
Curvature Profiles



$$H(\kappa) = \int \left(h(\kappa)\right)^2 ds$$

Computing Smooth Curves

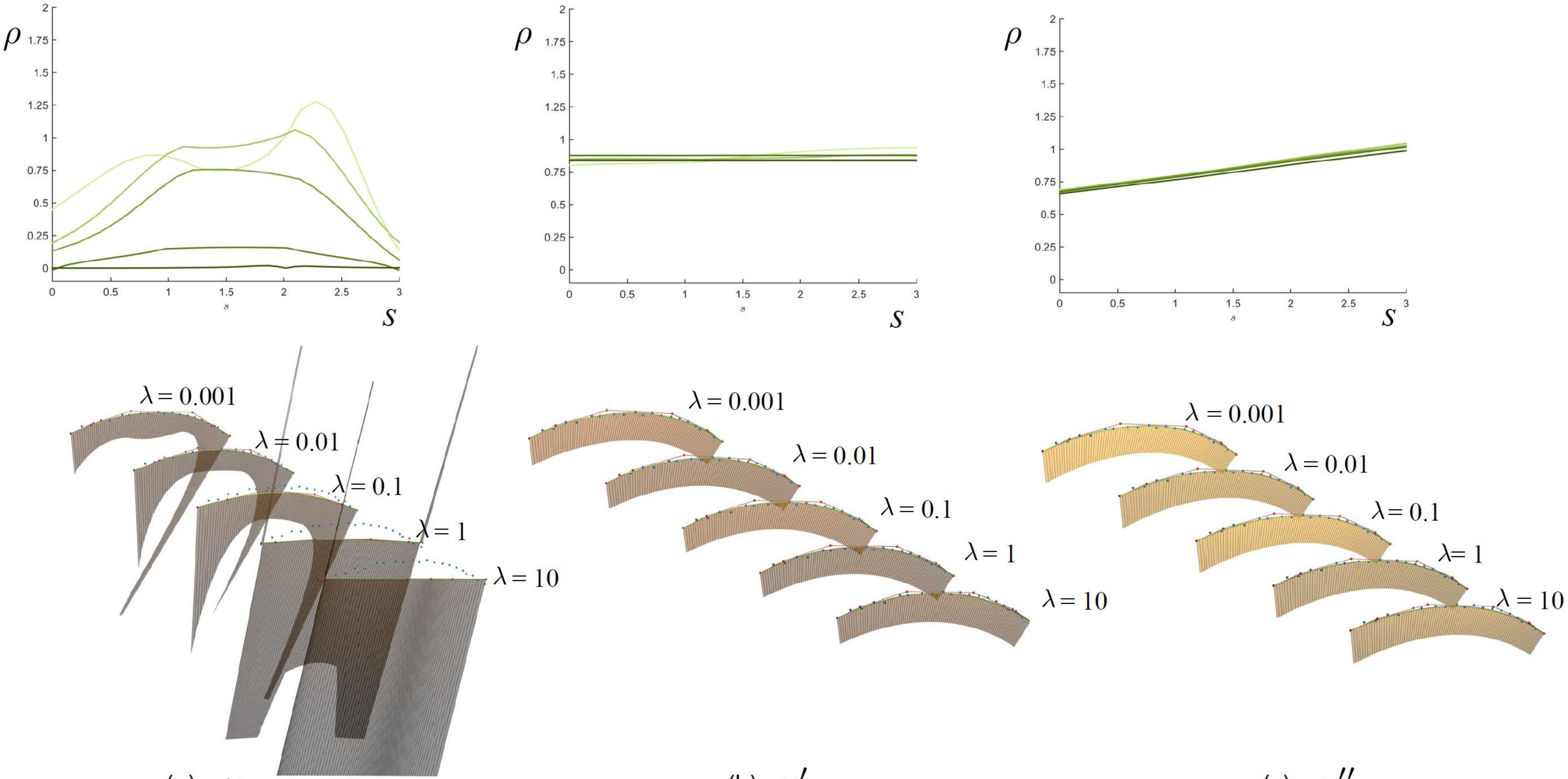
The functional $h(\kappa) = \kappa''$ is robust



(a) κ (b) κ' (c) κ''

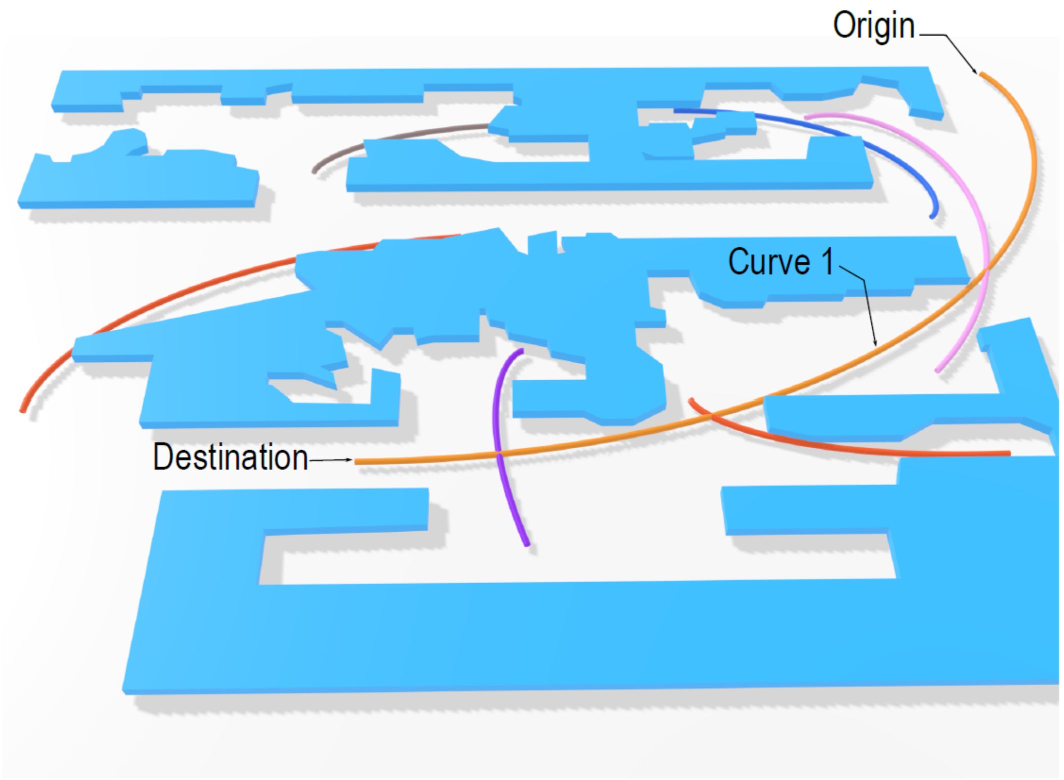
Computing Smooth Curves

The functional $h(\kappa) = \kappa''$ is robust

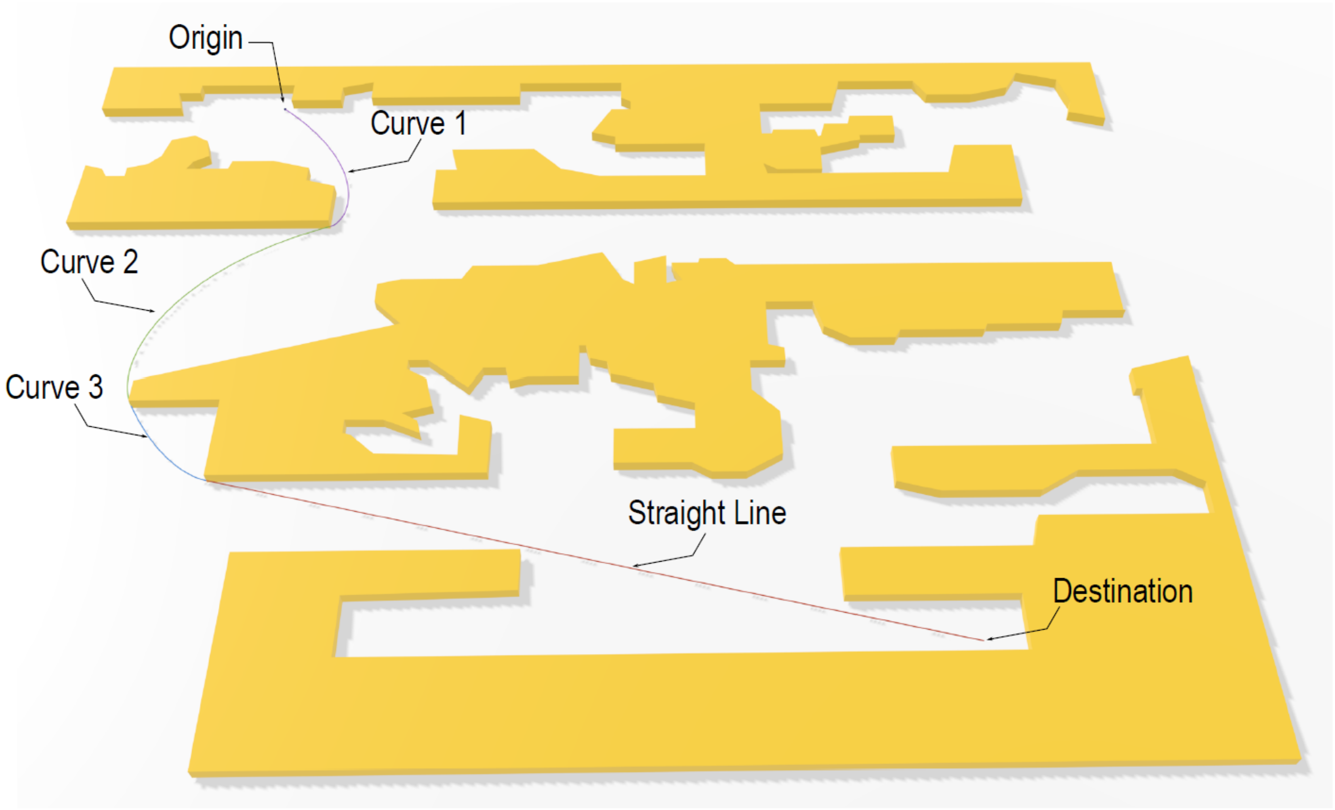


(a) κ (b) κ' (c) κ''

Examples in Path Planning



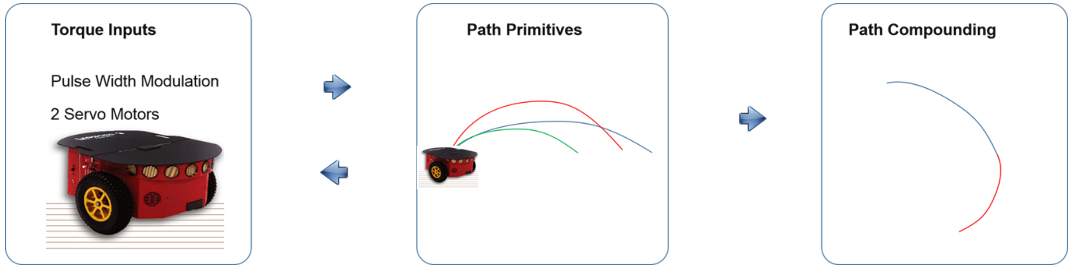
(a) Single Curves



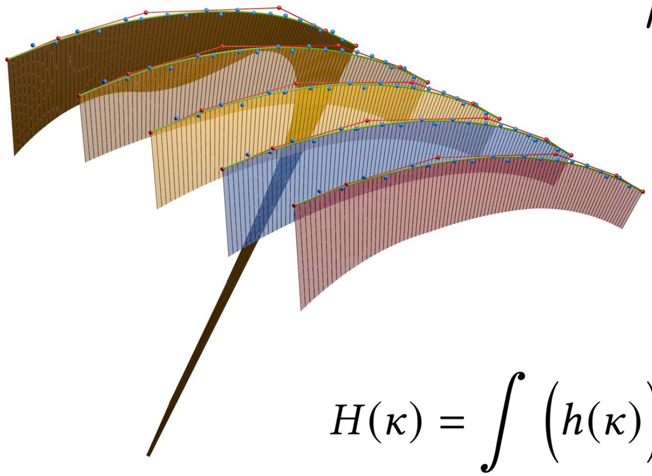
(b) Compounded curves

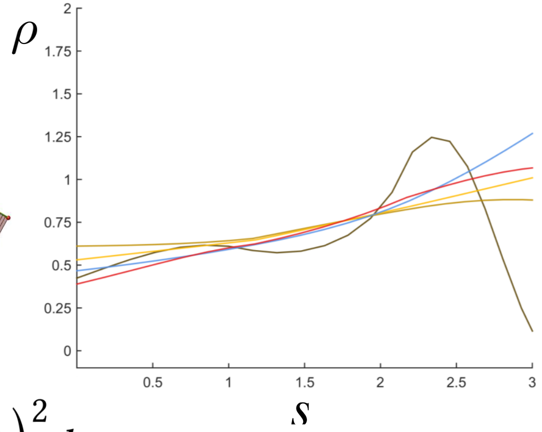
How to compute smooth curves effectively?

e.g. safe path planning in robots



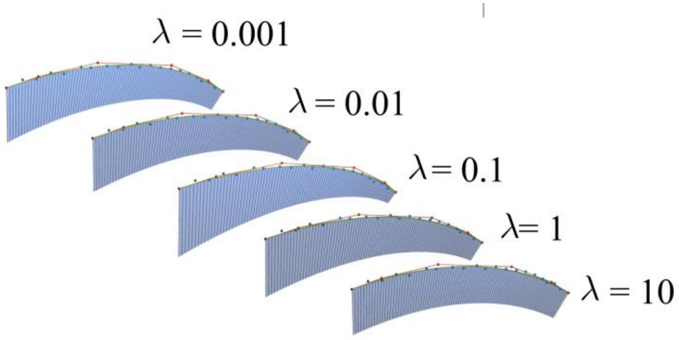
Lesson 1. The fairness functional is key!

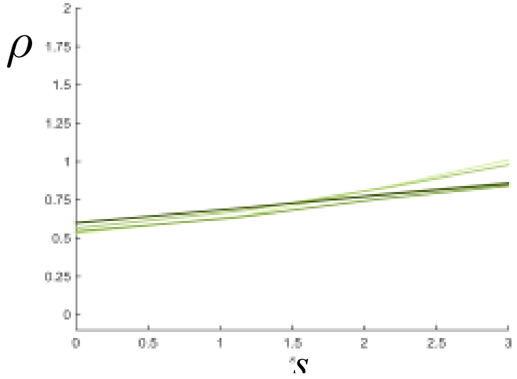




$$H(\kappa) = \int \left(h(\kappa)\right)^2 ds$$

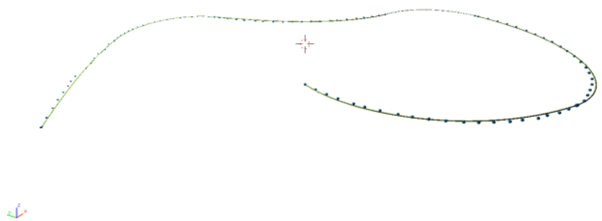
Lesson 2. The functional $h(\kappa) = \kappa''$ is robust…

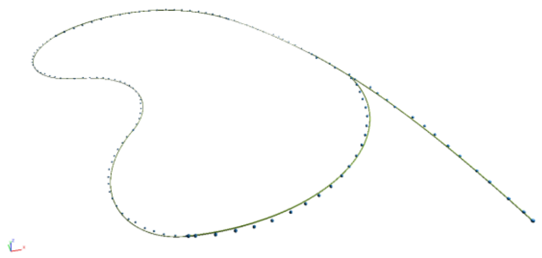




$$\min_{p_i} \; F = E + \lambda H$$

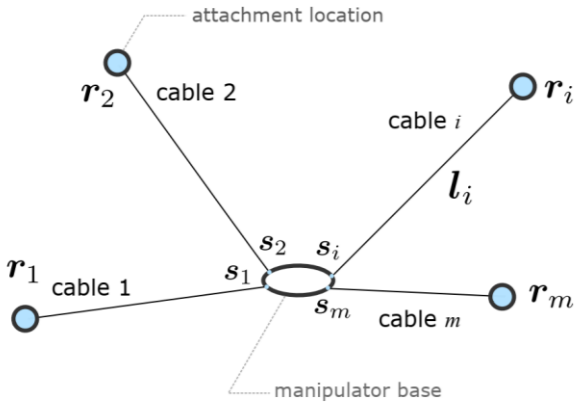
… to compute smooth curves efficiently!

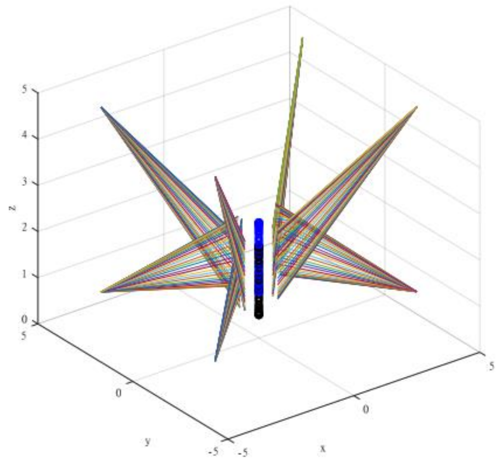


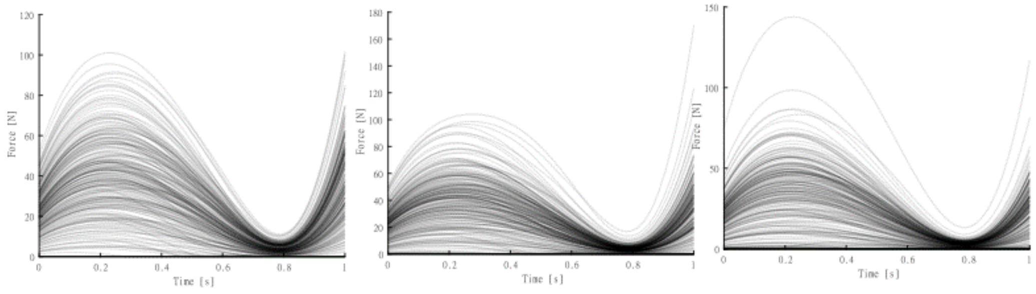


A Study of Fairness Functionals for Smooth Path Planning in Mobile Robots

Victor Parque **Waseda University**



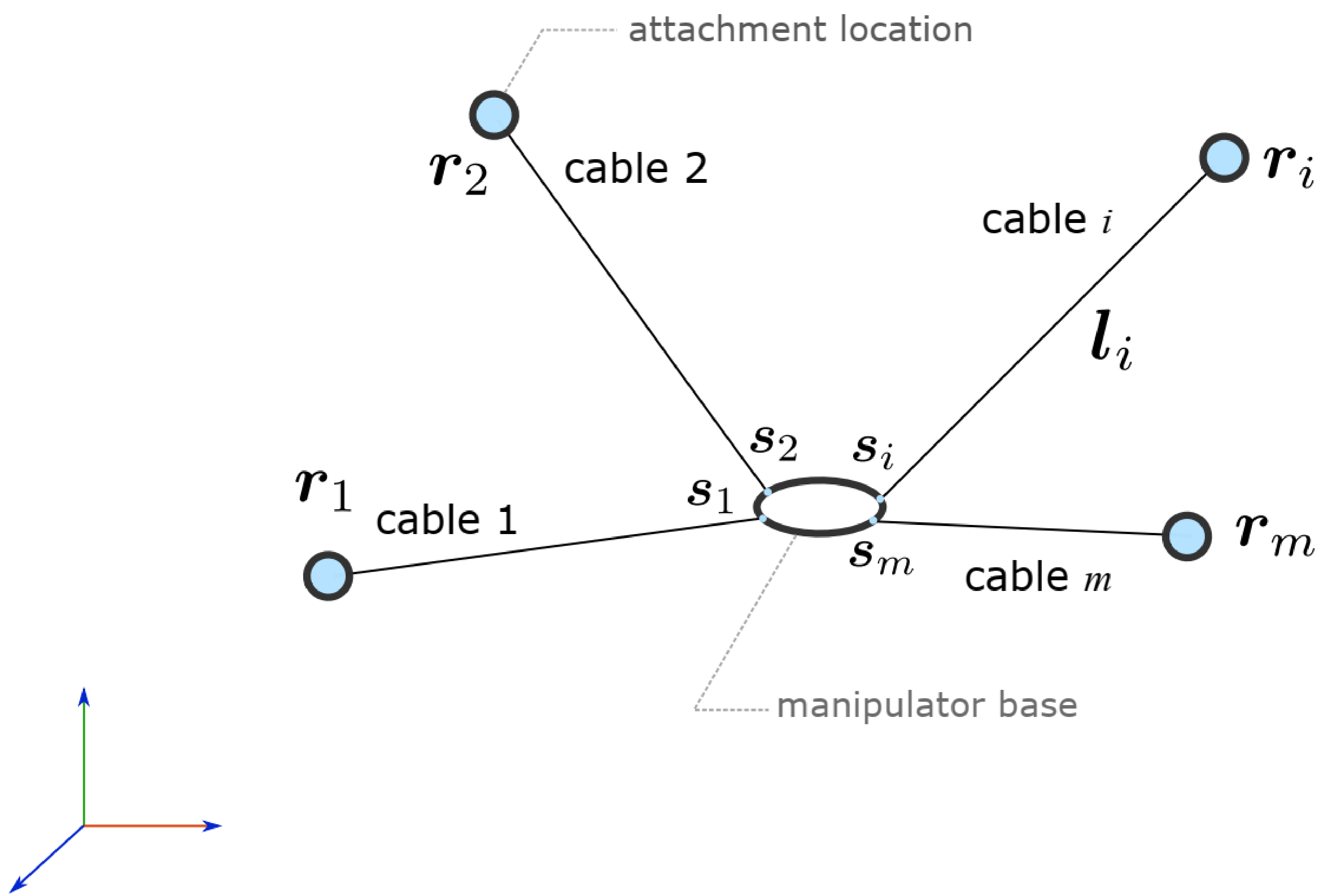




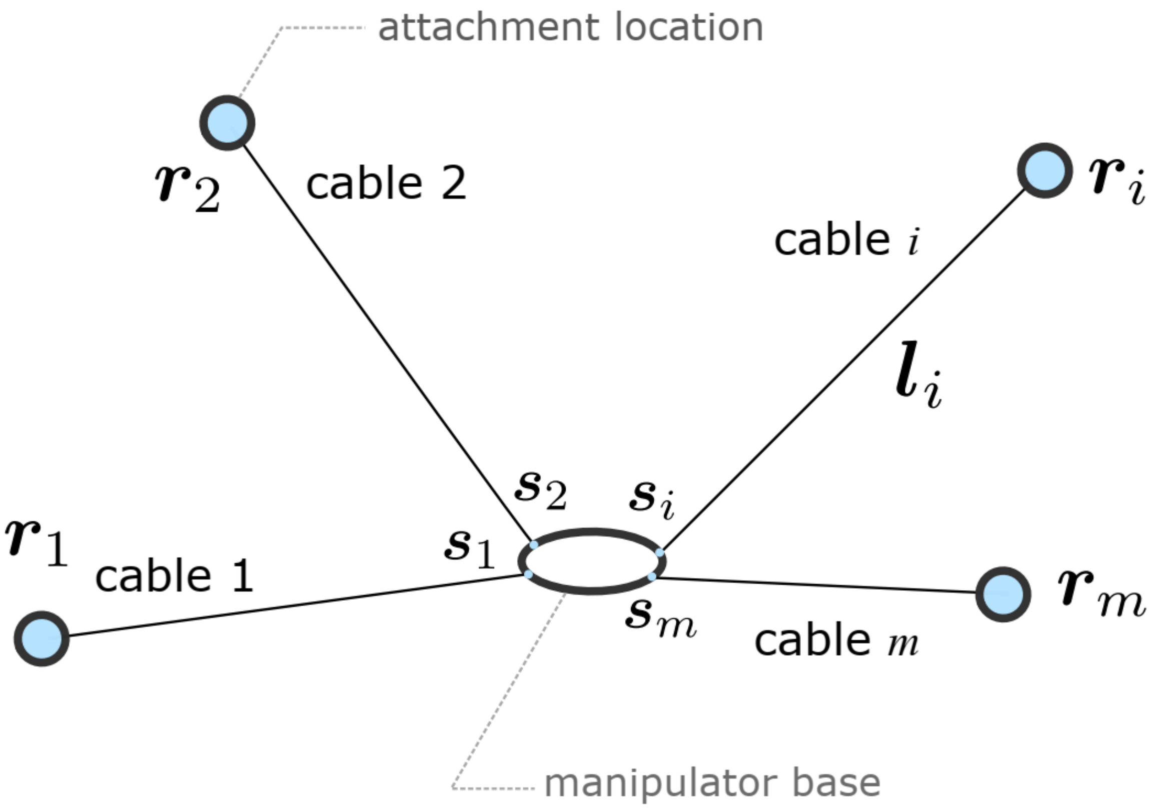
Optimal Design of Cable-Driven Parallel Robots by Particle Schemes

Victor Parque, Tomoyuki Miyashita

Cable-driven Parallel Robot



Cable-driven Parallel Robot



Cable lengths

Jacobian Matrix

$$\dot{\boldsymbol{l}} = L(\boldsymbol{q})\dot{\boldsymbol{q}},$$

Manipulator pose

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\dot{\boldsymbol{q}}, \boldsymbol{q}) + G(\boldsymbol{q}) + \boldsymbol{w}_e = -L^{\mathsf{T}}(\boldsymbol{q})\boldsymbol{f}$$

Inertia Matrix

Centrifugal vector

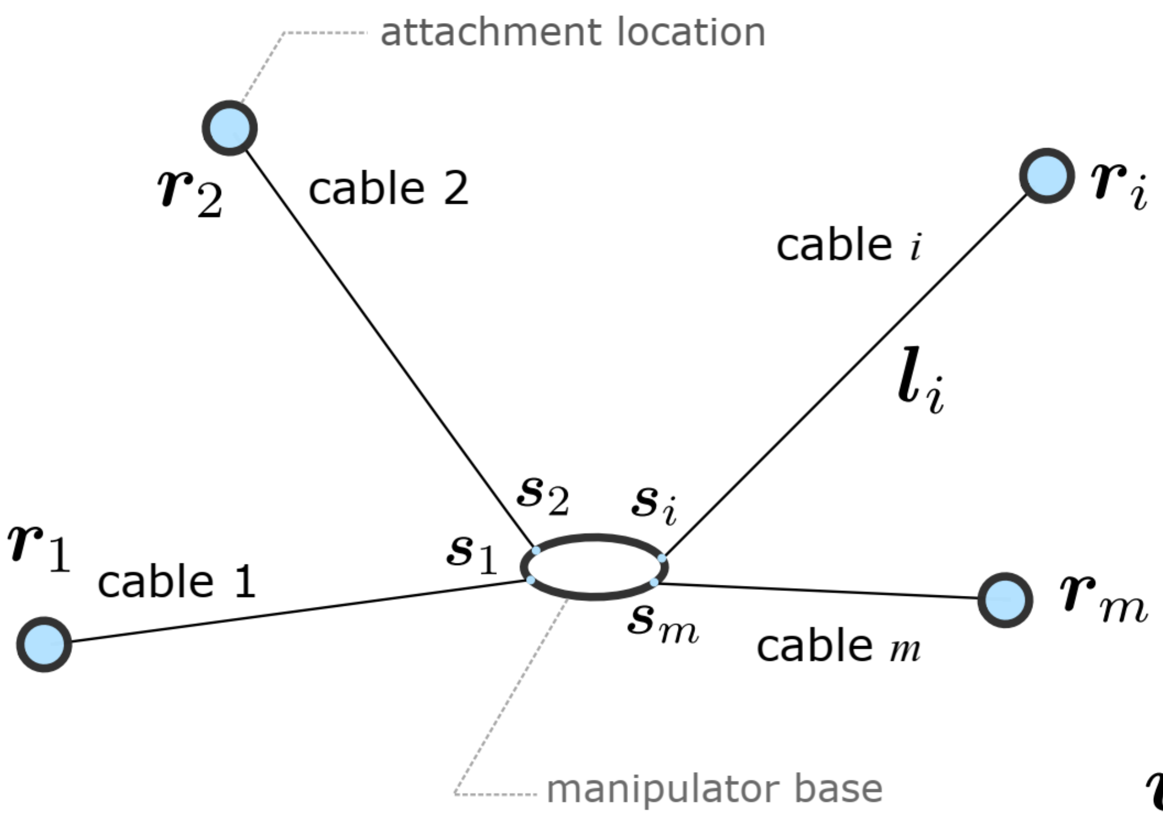
Gravity vector

External wrench

Cable Forces

$$0 \leq \boldsymbol{f}_{\min} \leq \boldsymbol{f} \leq \boldsymbol{f}_{\max}$$

Tension Distribution Problem



Find cable forces by solving:

$$\underset{\boldsymbol{f}}{\textbf{Minimize}} \ \frac{1}{2}\boldsymbol{f}^{\mathsf{T}} H \boldsymbol{f}$$

$$\text{s.t. } \boldsymbol{f} \in [\boldsymbol{f}_{\min}, \boldsymbol{f}_{\max}]$$

$$-L^{\mathsf{T}}(\boldsymbol{q})\boldsymbol{f} = \boldsymbol{w}$$

Inertia Matrix

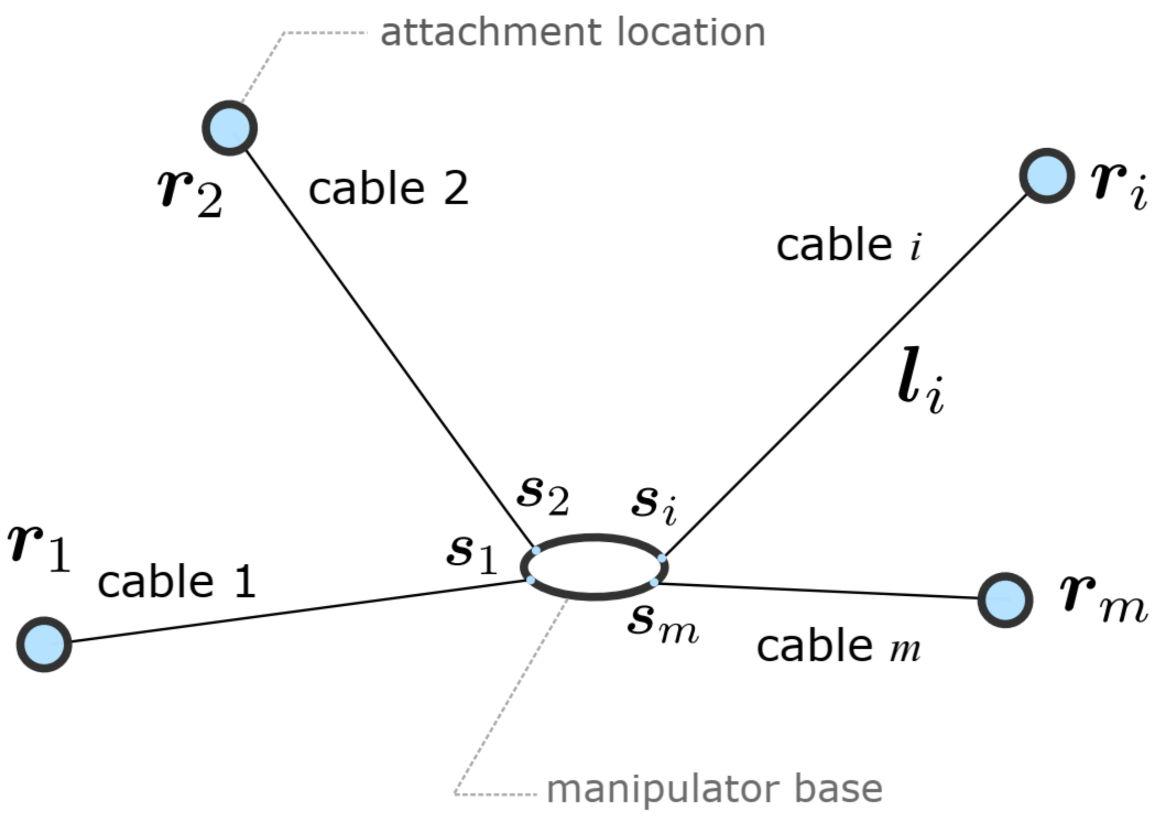
Centrifugal vector

Gravity vector

$$\boldsymbol{w} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\dot{\boldsymbol{q}}, \boldsymbol{q}) + G(\boldsymbol{q})$$

Trajectory tasks

Cable Configuration Problem



Find cable attachment locations by

Number of cables

Penalty for constraint violation

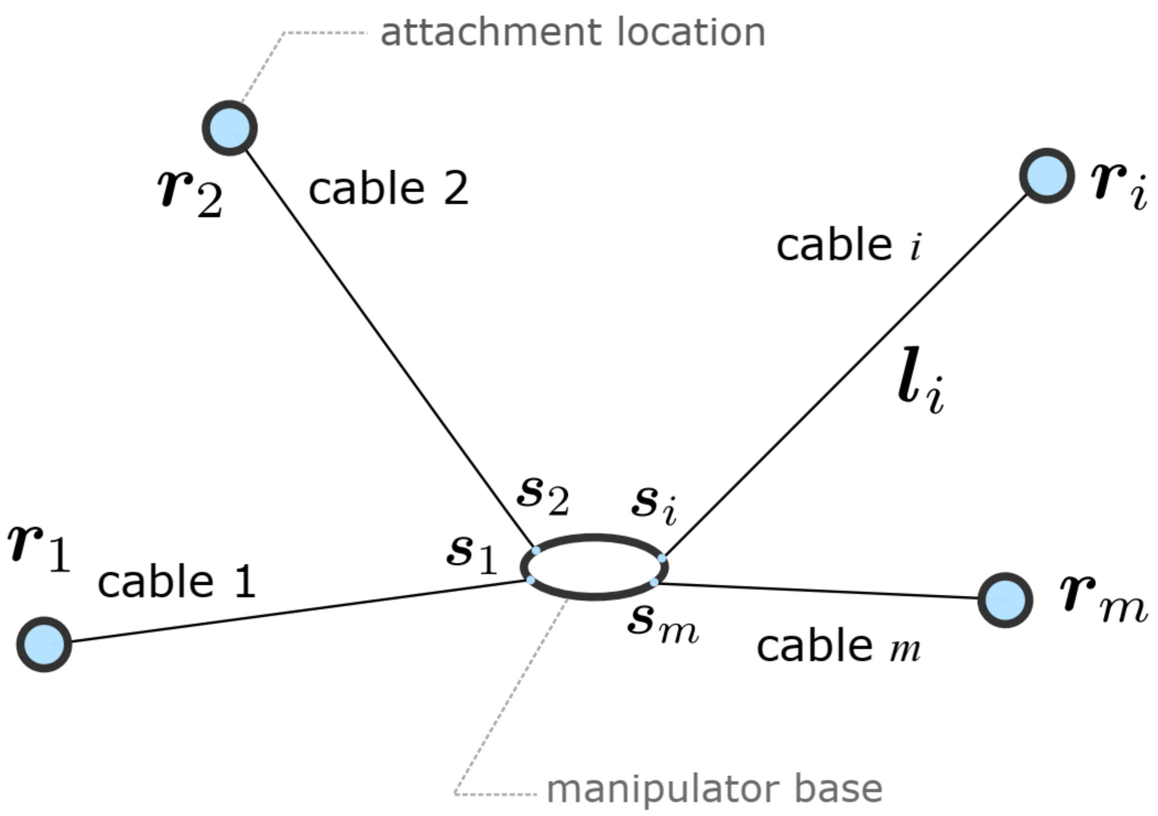
$$\underset{\boldsymbol{r},\boldsymbol{s}}{\text{Minimize}} \sum_{t\in[0,t_{\max}]} \sum_{i=1}^{m} (f_i^{*2} + \lambda)$$

$$\text{s.t. } \boldsymbol{r} \in \mathcal{R}, \boldsymbol{s} \in \mathcal{S}$$

Optimal force in tension distribution problem

Time in user-defined trajectory

Cable Configuration Problem: Challenges



Geometry Methods

✓ **Intuitive, Principled**

⚠ **Specific to the robot architecture**

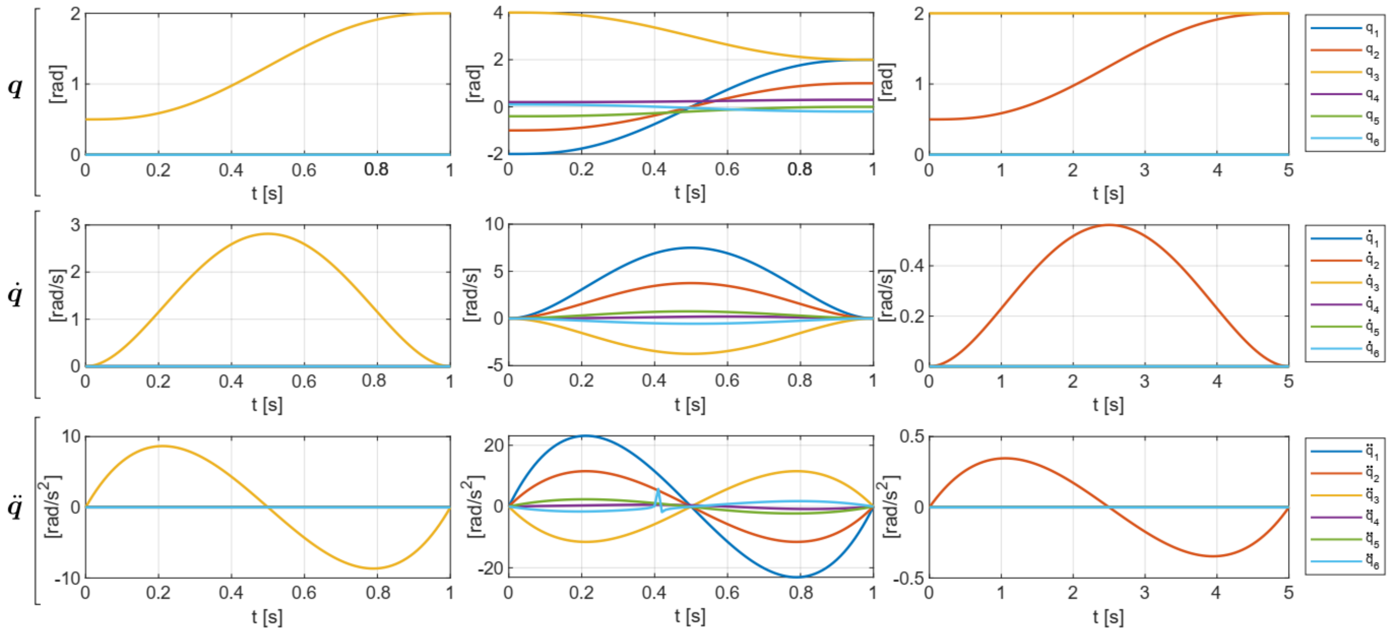
Optimization Methods

✓ **Applicable to various architectures**

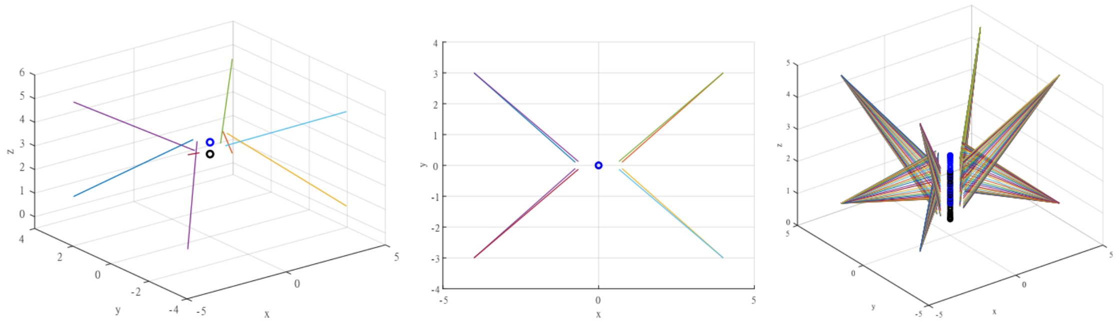
⚠ **No design principles**

Cable Configuration Problem: 8 cables, 6DOF

Trajectory tasks



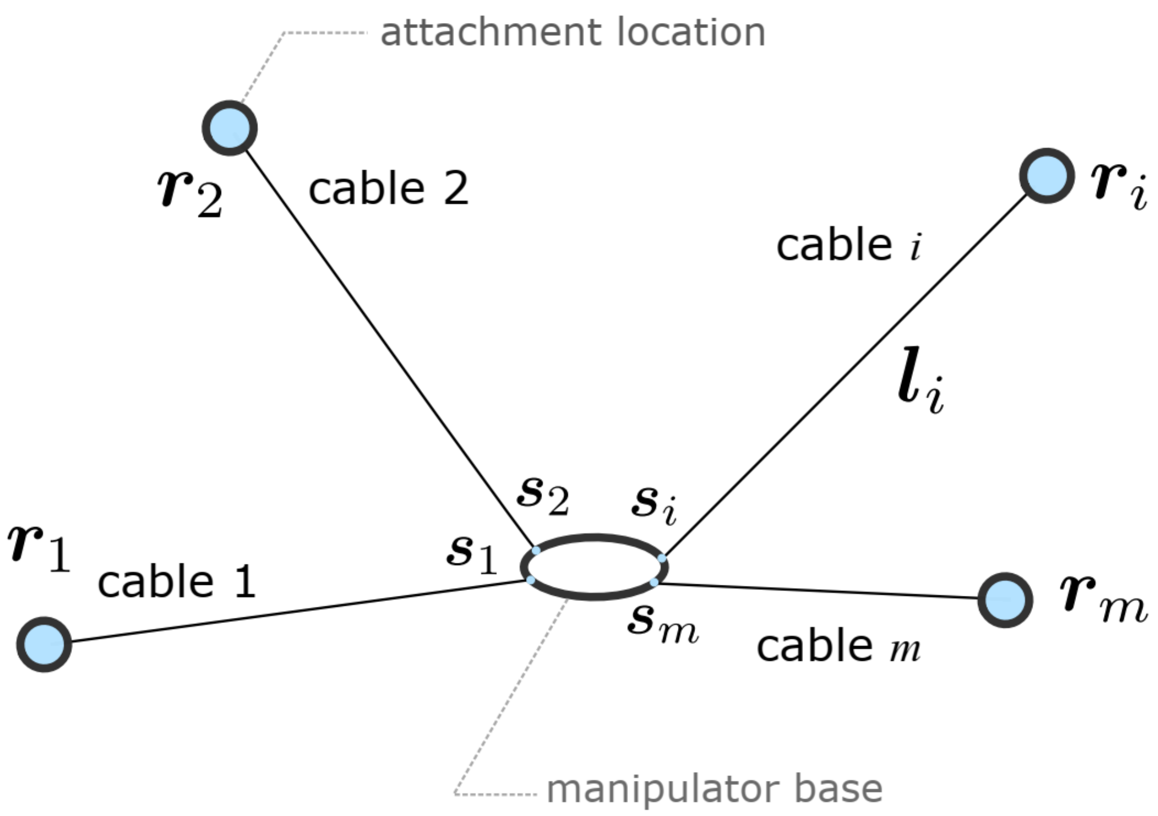
(a) Trajectory 1 (b) Trajectory 2 (c) Trajectory 3



(a) Perspective (b) Top View (c) Cable States

Pott et al, **IPAnema**: A family of Cable-Driven Parallel Robots for Industrial Applications, 2013

Cable Configuration Problem



Find cable attachment locations by

Number of cables

Penalty for constraint violation

$$\underset{\boldsymbol{r},\boldsymbol{s}}{\text{Minimize}} \sum_{t \in [0, t_{\max}]} \sum_{i=1}^{m} ({f_i^*}^2 + \lambda)$$

$$\text{s.t. } \boldsymbol{r} \in \mathcal{R}, \boldsymbol{s} \in \mathcal{S}$$

Optimal force in tension distribution problem

Time in user-defined trajectory

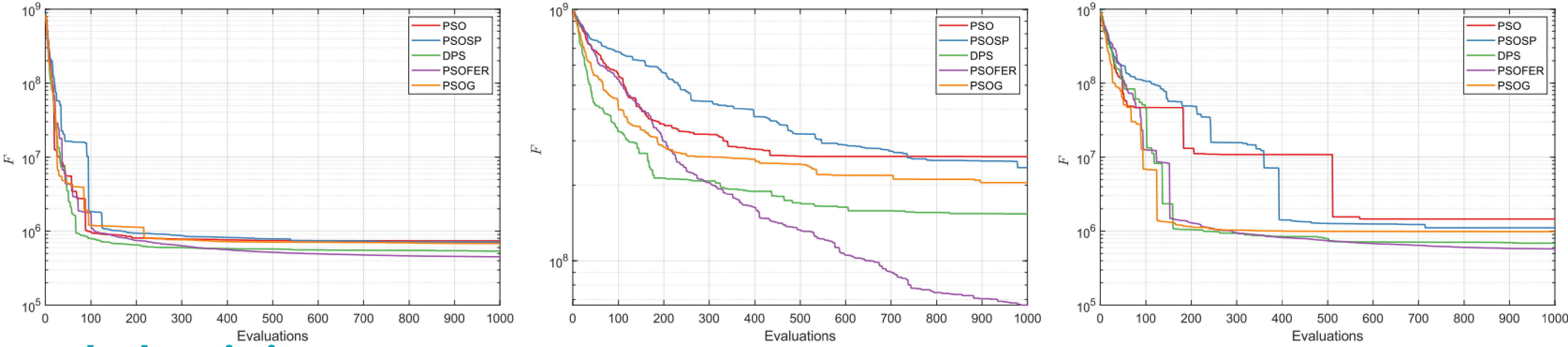
Non-linear Problem: We compare several Particle Swarm Heuristics

Cable Configuration Problem: Particle Schemes

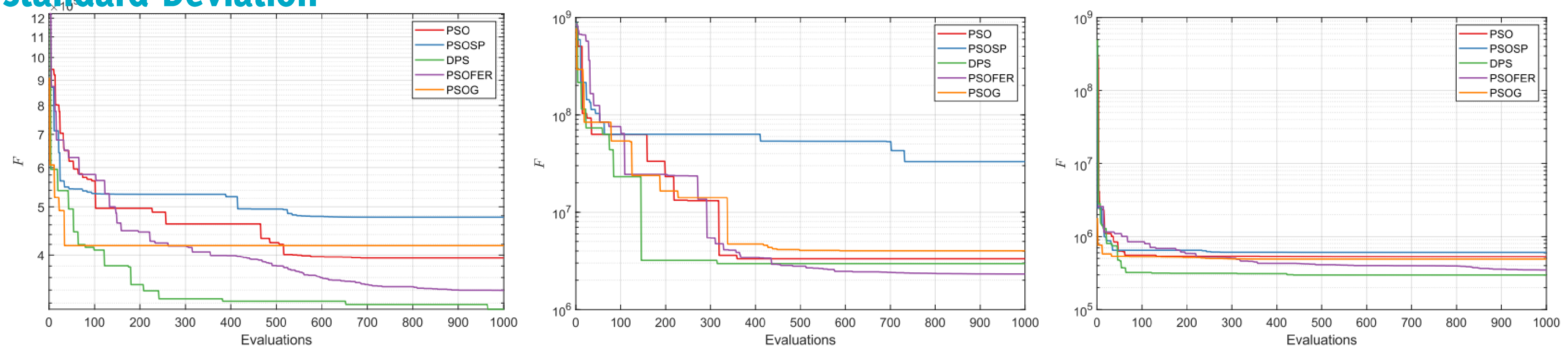
- Particle Swarm Optimization (PSO)
- Particle Swarm with **Speciation** (PSOSP)
- **Differential Particle** Scheme (DPS)
- Particle Swarm **with Fitness Euclidean Ratio**(PSOFER)
- Particle Swarm Optimization with **Global Explorative Strategy**(PSOG)

1000 function evaluations, 30 independent runs

Mean Convergence



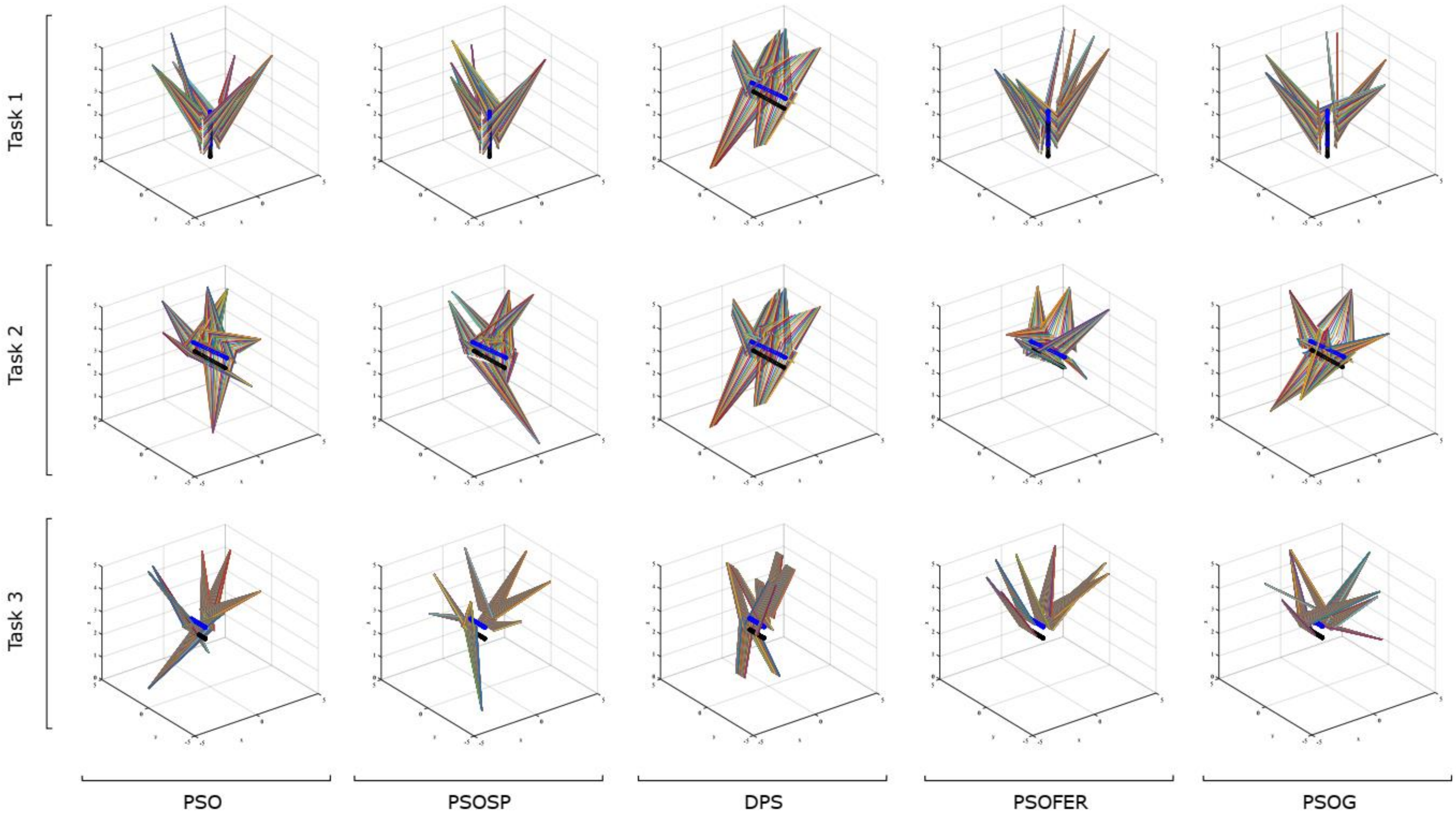
Standard Deviation



(a) Task 1 (b) Task 2 (c) Task 3

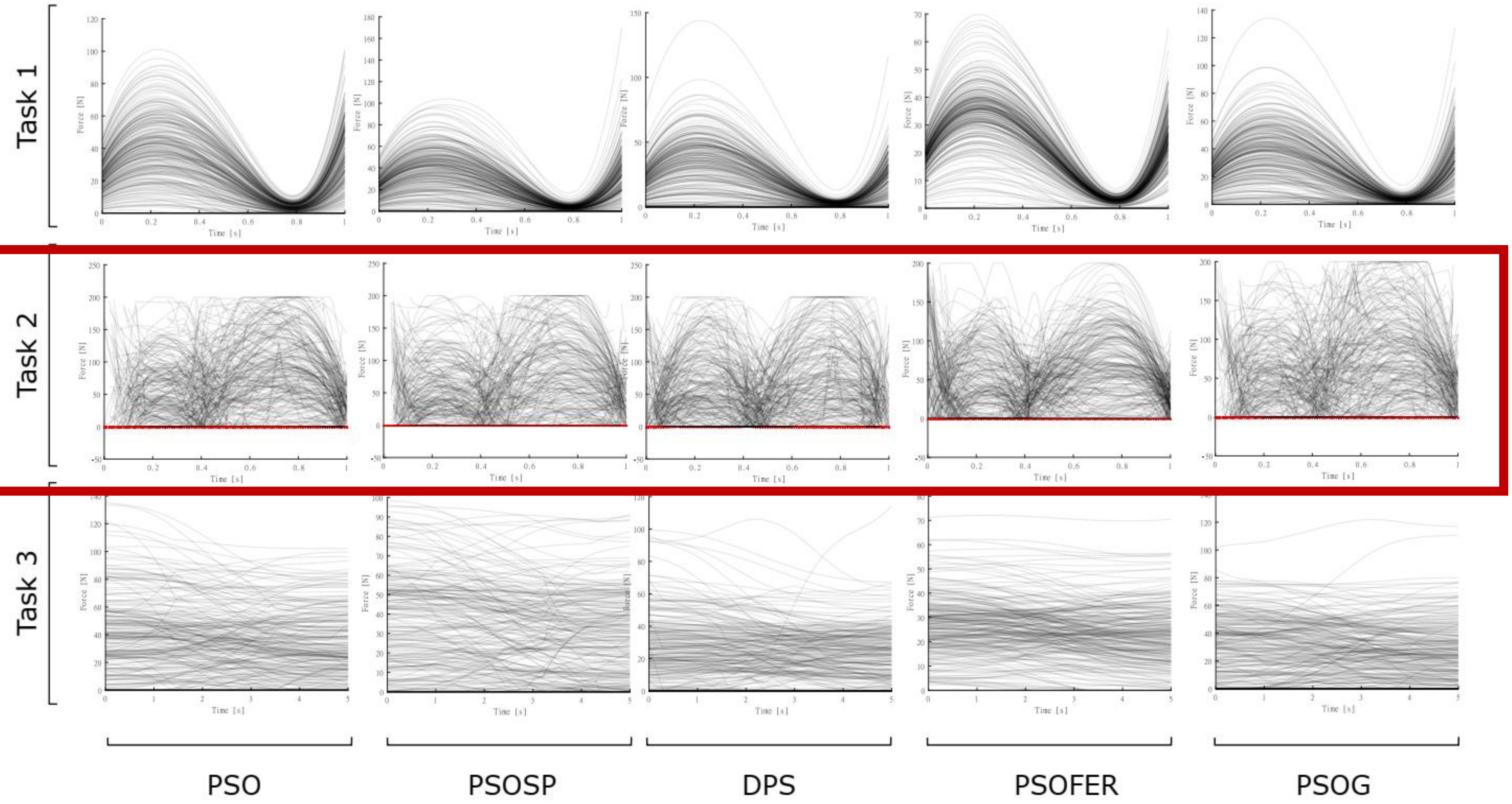
PSOFER and DPS converge faster in 200 - 800 evaluations

Cable Configuration Problem – Best Configurations (out of 30 runs)



Each algorithm converges to distinct configurations

Force Distribution Problem – Solutions over 30 runs



Solutions to the tension problem are not always smooth

Statistical Comparison – Solutions over 30 runs

	PSO	PSOSP	DPS	PSOFER	PSOG
PSO		[=] 1.12e-01	[-] 2.15e-06	[-] 6.52e-09	[=] 7.98e-02
PSOSP	[=] 1.12e-01		[-] 1.04e-04	[-] 2.92e-09	[=] 6.41e-01
DPS	[+] 2.15e-06	[+] 1.04e-04		[-] 2.92e-02	[+] 8.12e-04
PSOFER	[+] 6.52e-09	[+] 2.92e-09	[+] 2.92e-02		[+] 2.60e-08
PSOG	[=] 7.98e-02	[=] 6.41e-01	[-] 8.12e-04	[-] 2.60e-08	

(a) Task 1

	PSO	PSOSP	DPS	PSOFER	PSOG
PSO		[=] 5.01e-01	[-] 7.96e-03	[-] 2.78e-07	[=] 1.62e-01
PSOSP	[=] 5.01e-01		[-] 1.99e-02	[-] 3.01e-07	[=] 4.46e-01
DPS	[+] 7.96e-03	[+] 1.99e-02		[-] 1.24e-03	[=] 1.54e-01
PSOFER	[+] 2.78e-07	[+] 3.01e-07	[+] 1.24e-03		[+] 5.27e-06
PSOG	[=] 1.62e-01	[=] 4.46e-01	[=] 1.54e-01	[-] 5.27e-06	

(b) Task 2

	PSO	PSOSP	DPS	PSOFER	PSOG
PSO		[-] 2.24e-02	[-] 4.11e-07	[-] 3.82e-09	[-] 6.55e-04
PSOSP	[+] 2.24e-02		[-] 1.09e-05	[-] 2.19e-08	[=] 2.12e-01
DPS	[+] 4.11e-07	[+] 1.09e-05		[=] 5.01e-01	[+] 2.39e-04
PSOFER	[+] 3.82e-09	[+] 2.19e-08	[=] 5.01e-01		[+] 6.05e-07
PSOG	[+] 6.55e-04	[=] 2.12e-01	[-] 2.39e-04	[-] 6.05e-07	

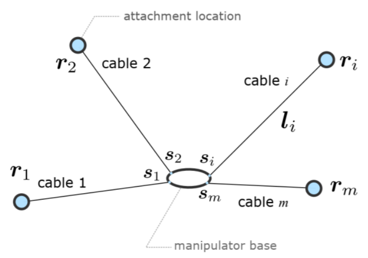
(c) Task 3

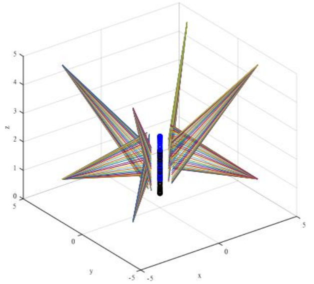
PSOFER (niching) and DPS (stagnation avoidance) outperform other optimization algorithms

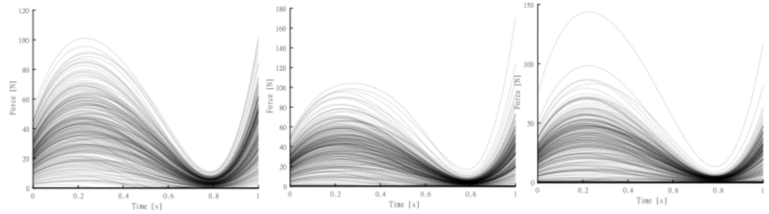
Conclusion

Final Notes

Results







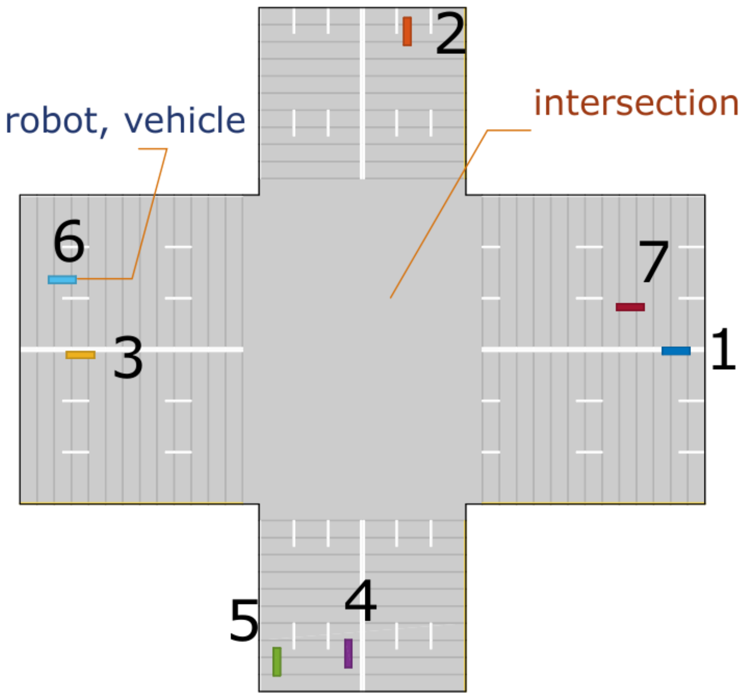
- We tackle the **configuration problem** in cable-driven parallel robots.
- Cable configuration problems are solvable quite **efficiently (10^3 evaluations)**, but force solutions are not always smooth.
- Strategies for **niching** and **stagnation** avoidance outperform other heuristics.

Future Work

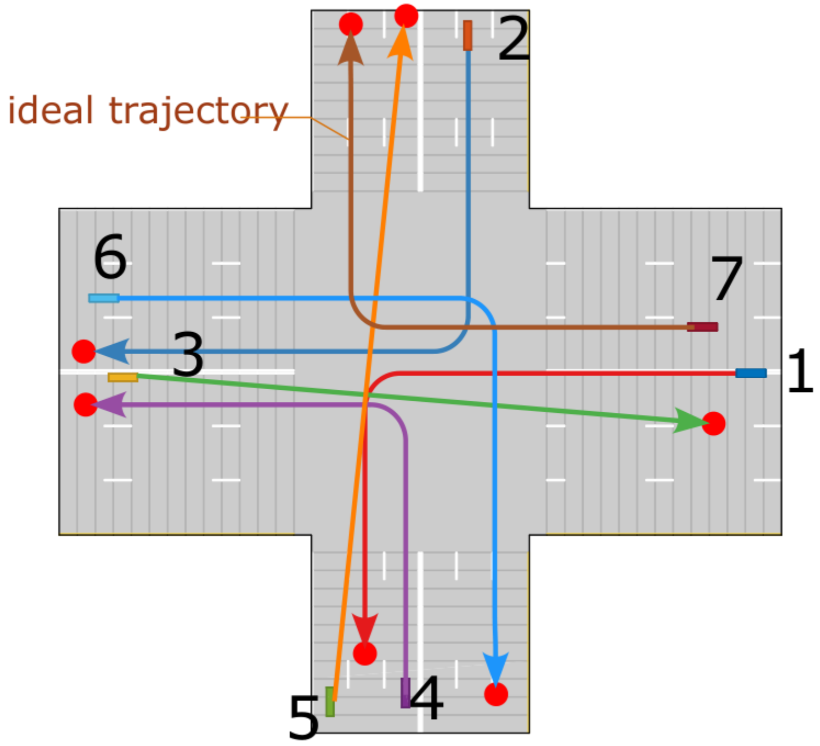
- Applications: Robot Leg (Cable-driven), Arm Exoskeleton, Manipulation

A Hybrid Evolutionary Approach for Multi Robot Coordinated Planning at Intersections

Background



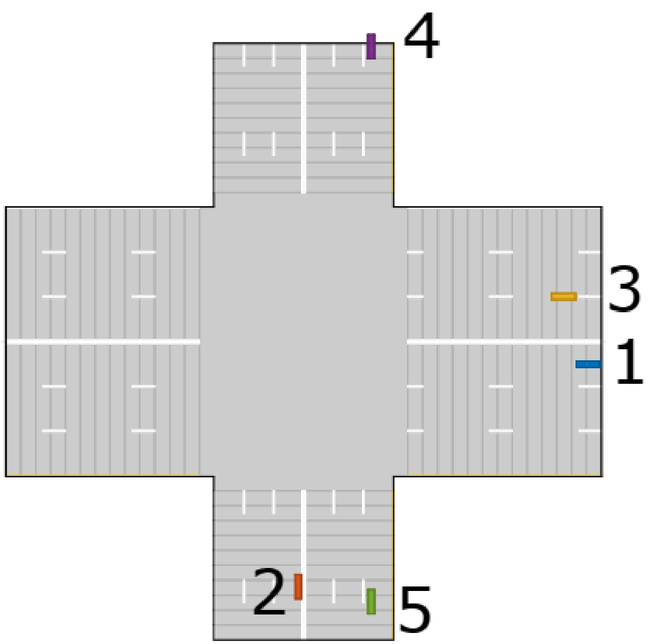
Vehicles at Intersections



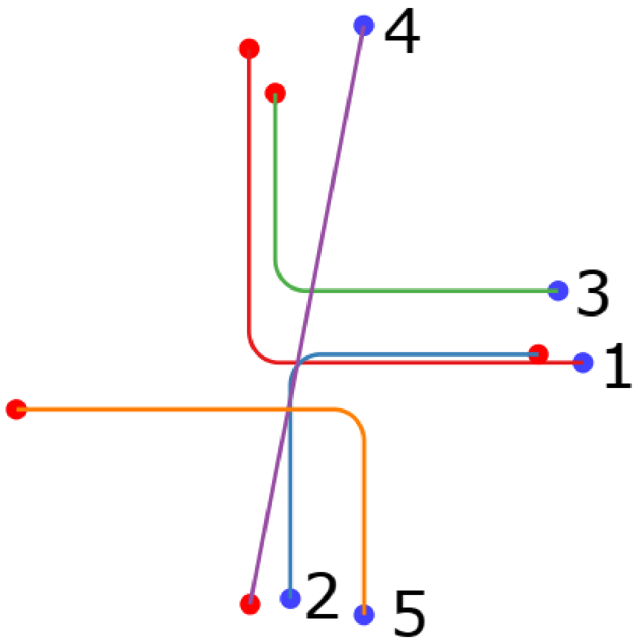
Desirable Trajectories, and Kinematic Constraints

How to plan at intersections?

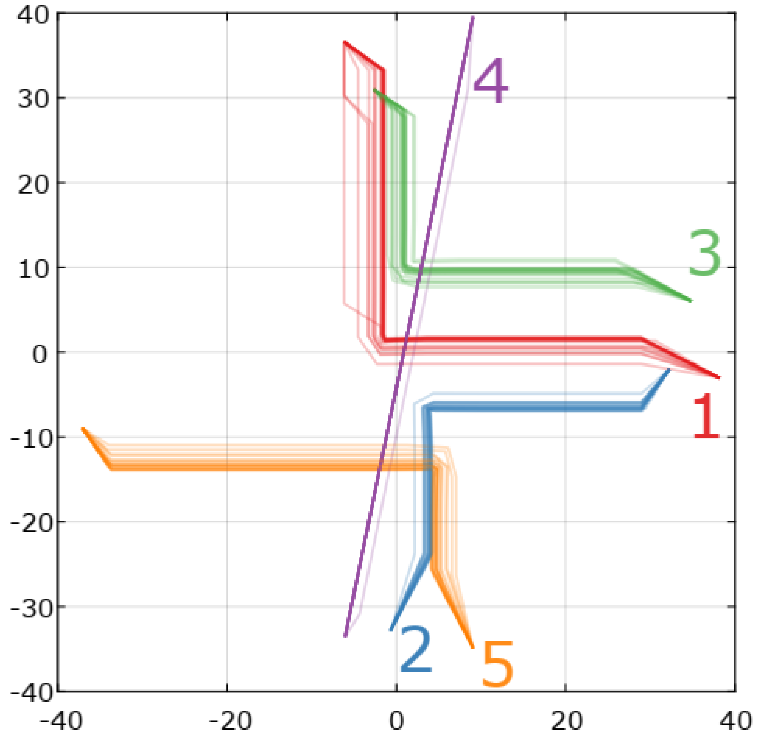
Background



Robots at intersections

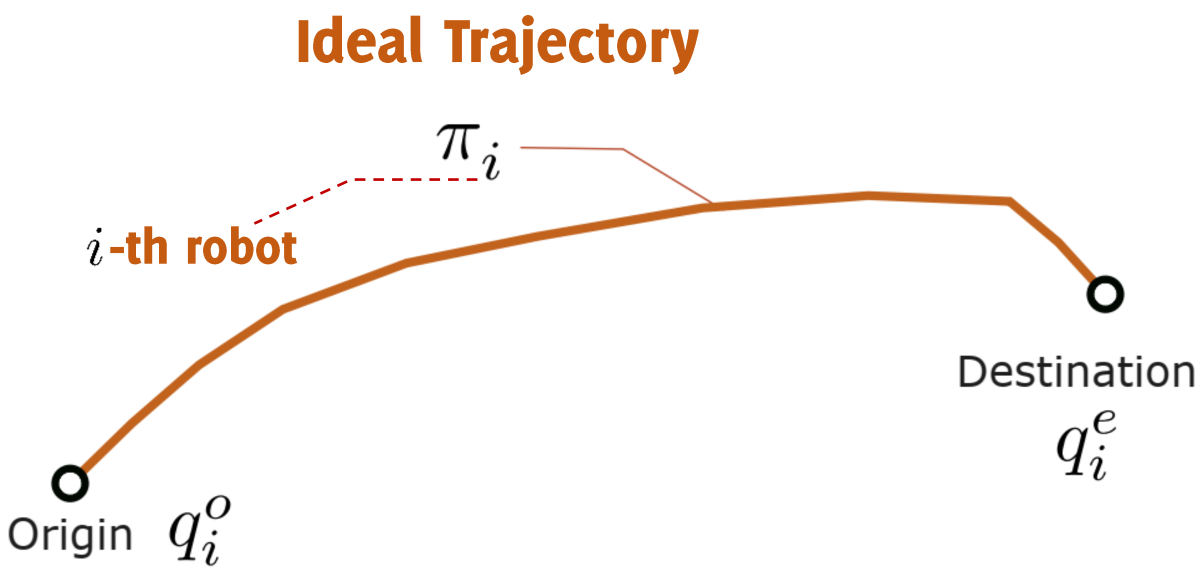


Ideal trajectories
Origin - Destination

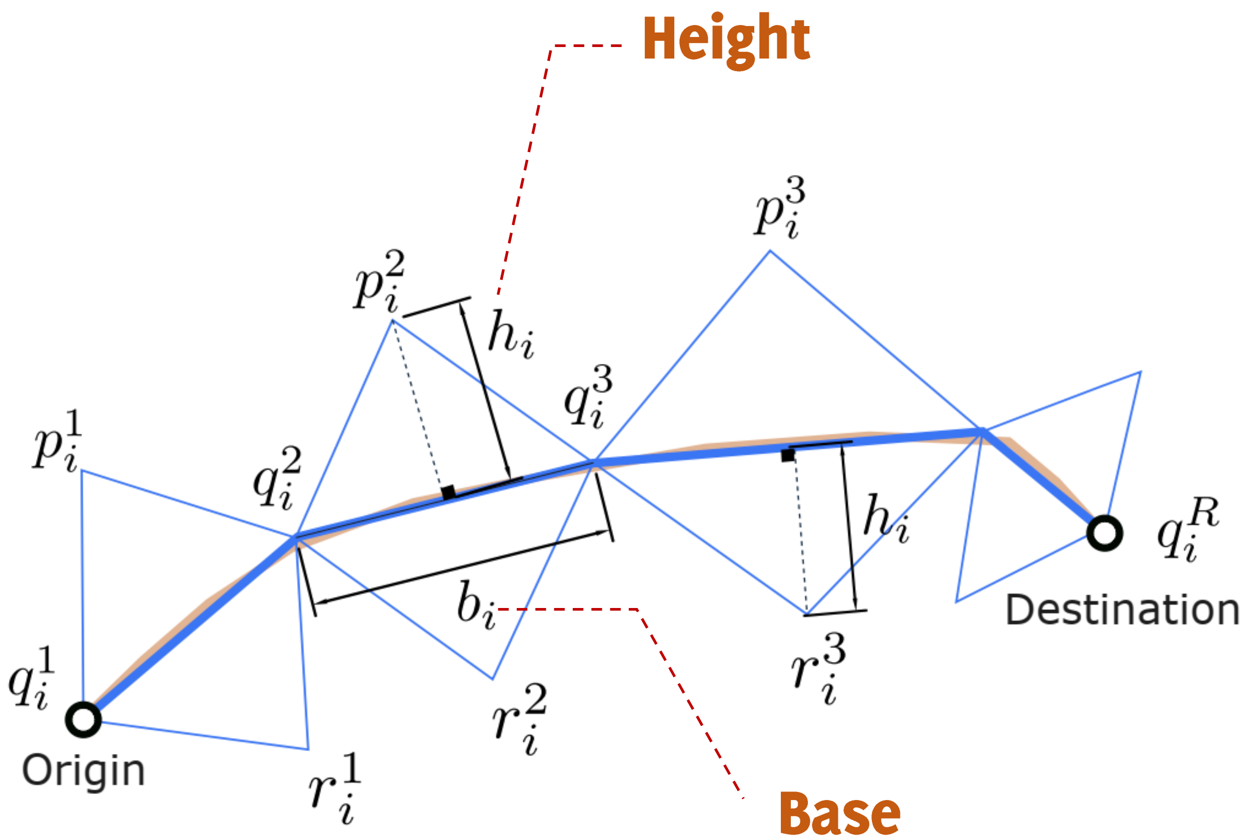


Coordinated trajectory planning

Lattice Configuration

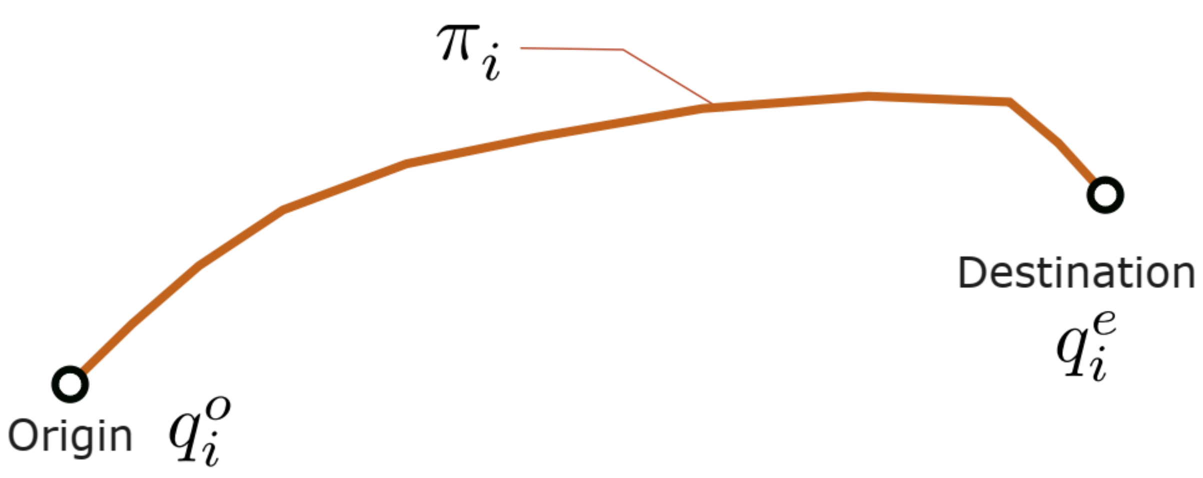


Origin - Destination



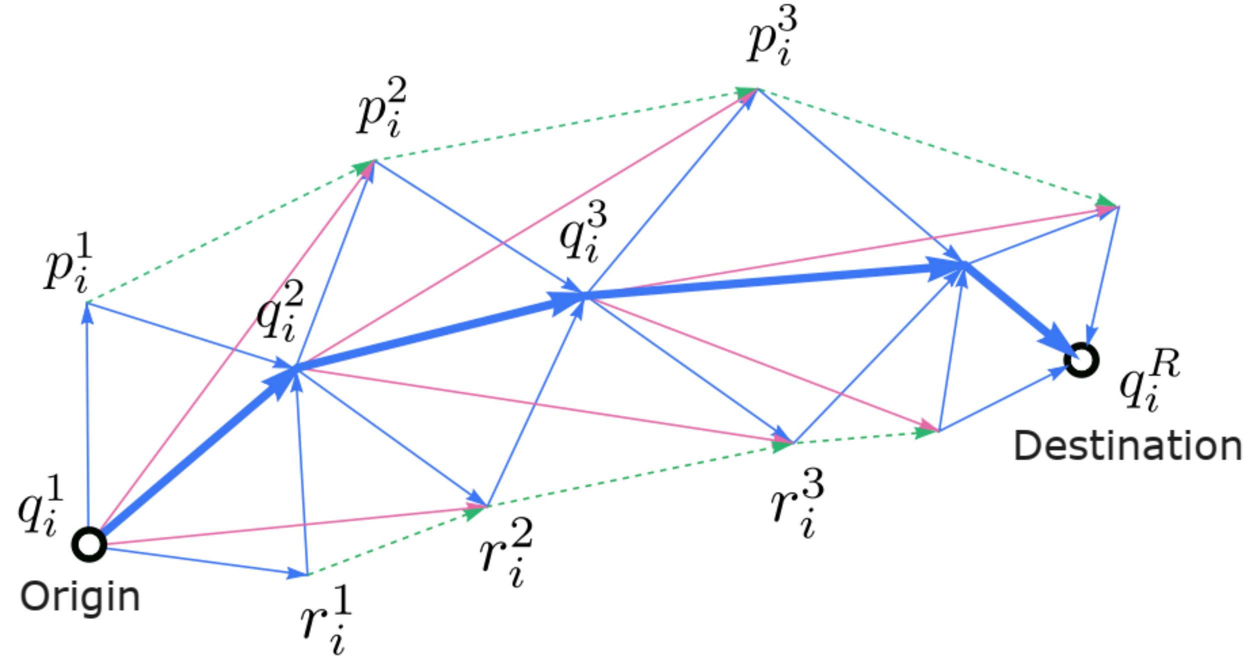
Triangulation

Lattice Configuration



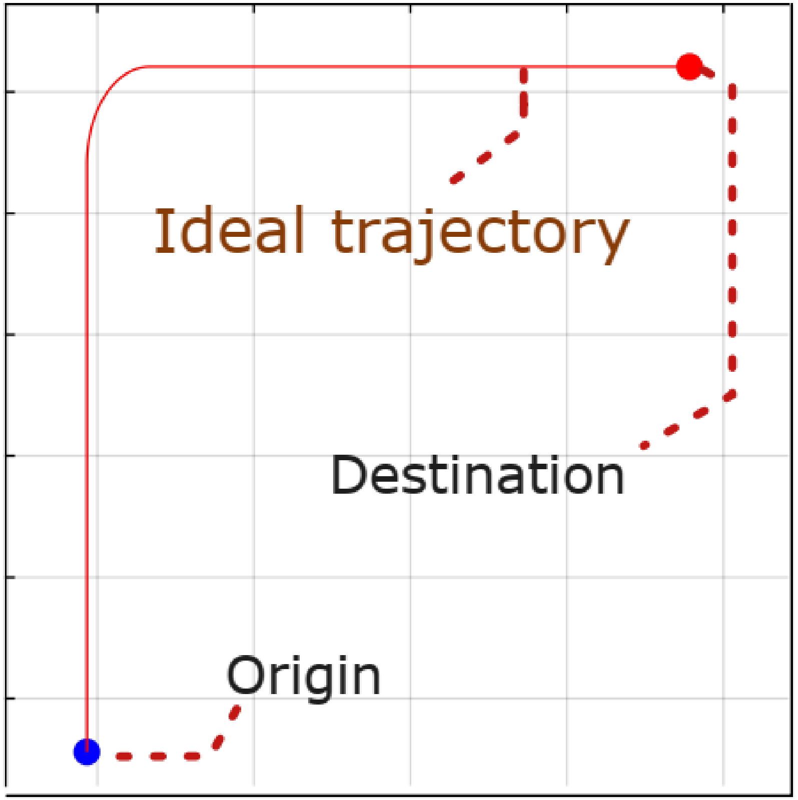
Origin - Destination

Directed Graph $G_i = (V_i,\ E_i)$

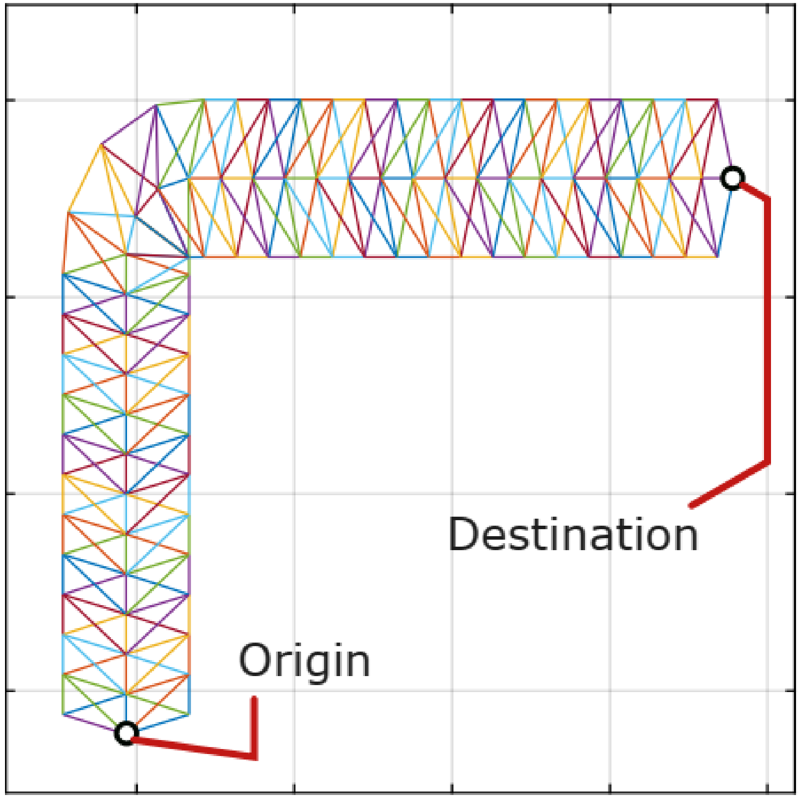


Triangulation

Lattice Configuration - Example

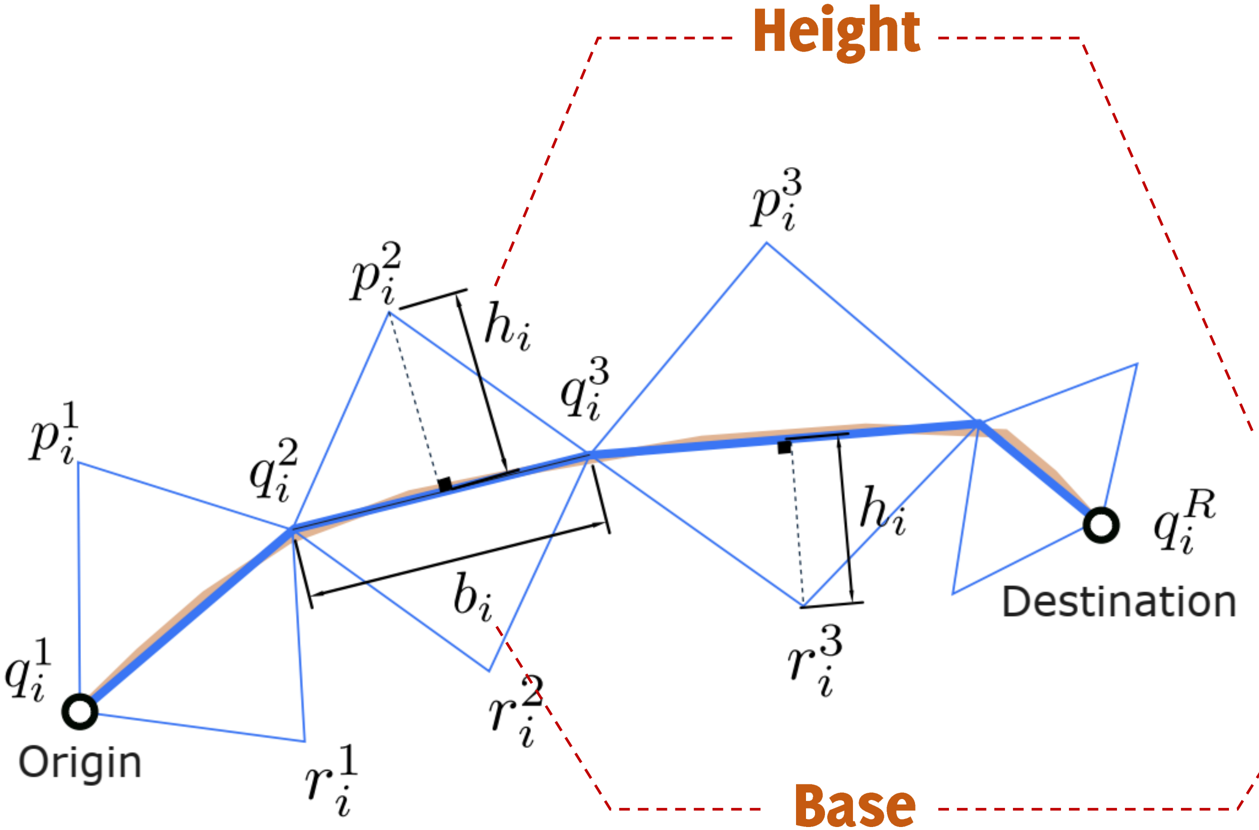


Origin - Destination



Triangulation

Multi-Robot Coordinated Path Planning



#Robots

Length of path

dRRT* shortest path

$$\underset{x}{\text{Minimize}} \quad \sum_{i=1}^{N} L(P_i)$$

$$\text{subject to} \quad P_i \subset E_i, \; i = 1, \ldots, N.$$

Roadmap (triangulation)

$$\boldsymbol{x} = (b_i, h_i)_{i \in \{1, \ldots, N\}}$$

Lattice-based Roadmap **Gradient-Free Optimization**

RADES: Rank-Based Differential Evolution with Successful Archive

Objective function

Solution vector

$$\boldsymbol{x}_{i,t+1} = \begin{cases} \boldsymbol{u}_{i,t}, & \text{for } f(\boldsymbol{u}_{i,t}) < f(\boldsymbol{x}_{i,t}) \\ \boldsymbol{x}_{i,t}, & \text{otherwise} \end{cases}$$

Reference vector

Crossover vector (binary)

$$\boldsymbol{u}_{i,t} = \boldsymbol{x}^r_{i,t} + \boldsymbol{b}_{i,t} \circ (\boldsymbol{v}_{i,t} - \boldsymbol{x}^r_{i,t})$$

Stagnation counter

$$\boldsymbol{x}^r_{i,t} = \begin{cases} \boldsymbol{x}_{i,t}, & q_i \leq Q \\ \mathcal{A}_{i,t}, & \text{otherwise} \end{cases}$$

Archive

Scalar

$$\boldsymbol{v}_{i,t} = \begin{cases} \boldsymbol{x}_{\text{best}} + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}), & q_i \leq Q \\ \mathcal{A}_{\text{best}} + F(\mathcal{A}_{r_1} - \mathcal{A}_{r_2}), & \text{otherwise} \end{cases}$$

Whitley Distribution

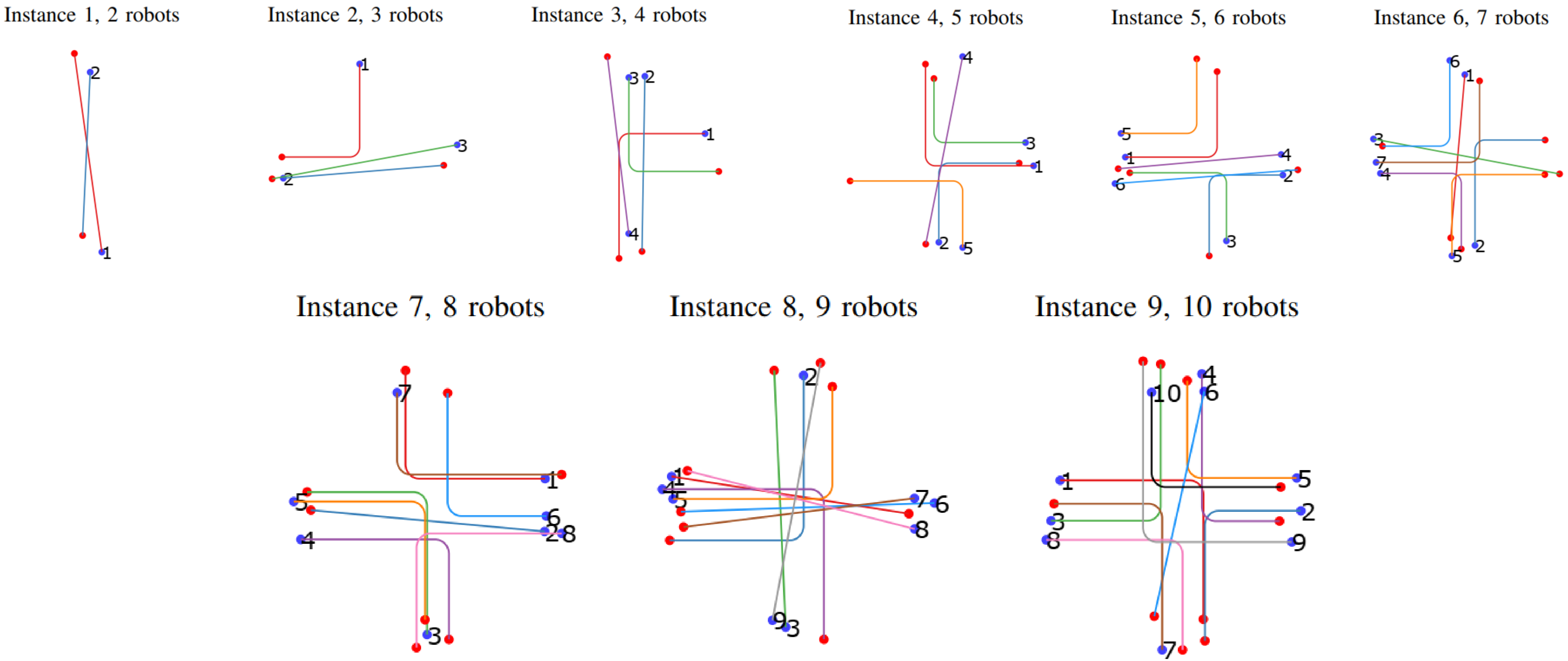
$$r_{1,2} = \left\lfloor \frac{|\mathcal{P}|}{2\beta - 1}\left(\beta - \sqrt{\beta^2 - 4(\beta - 1)r}\right) \right\rfloor$$

$Q = 128 \quad F = 0.7 \quad \beta = 2$

Algorithm 1: RADES

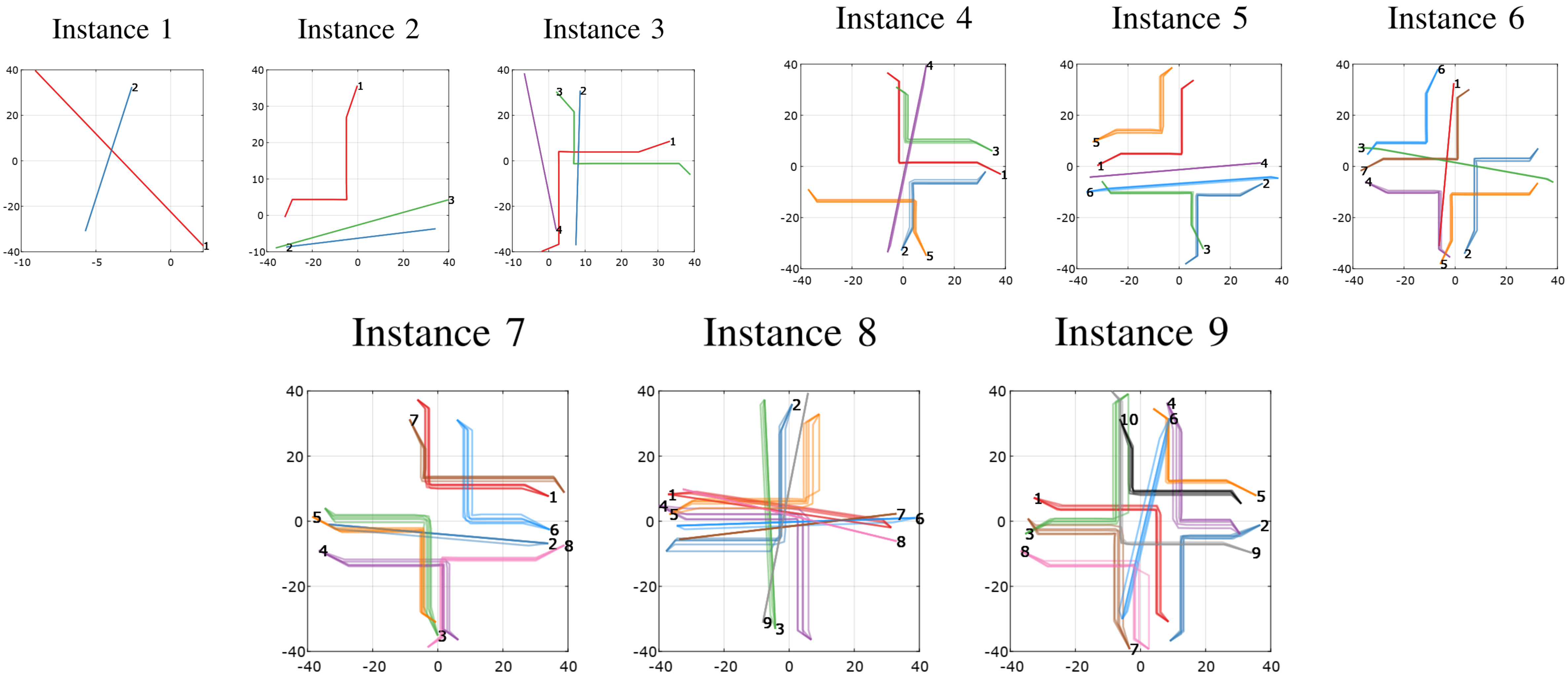
```
FEs = 0, t = 0, q_i = 0;
Generate a set of N individuals randomly as initial
  population set P;
Initialize the archive A from the population set P;
Initialize the F and CR;
FEs = FEs + N;
while FEs ≤ MaxFEs do
    t = t + 1;
    Find out the best individual x_best and A_best from
      the population and the archive A respectively;
    for i = 1 to NP do
        Generate r_1 and r_2 using (10);
        Generate the mutant vector v_{i,t}
        Generate the trial vector u_{i,t}
        if f(u_{i,t}) < f(x_{i,t}) then
            x_{i,t+1} = u_{i,t};
            u_{i,t} → A;
            Delete an element from A if |A| > |P|;
            q_i = 0;
        else
            x_{i,t+1} = x_{i,t};
            q_i = q_i + 1;
        end
        FEs = FEs + 1;
    end
end
```

Computational Experiments



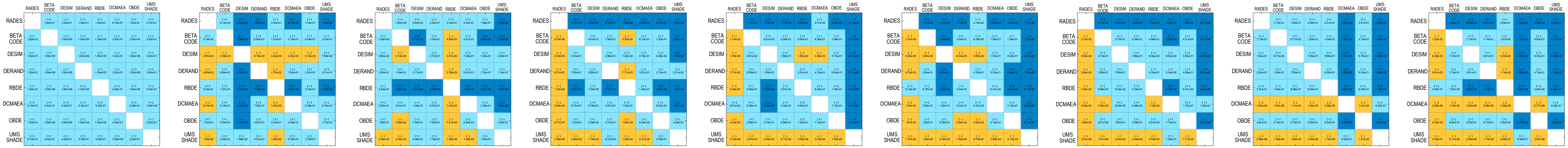
10 scenarios, diverse origin-destination and configurations

Computational Experiments



It is possible to compute feasible collision-free multi-robot trajectories

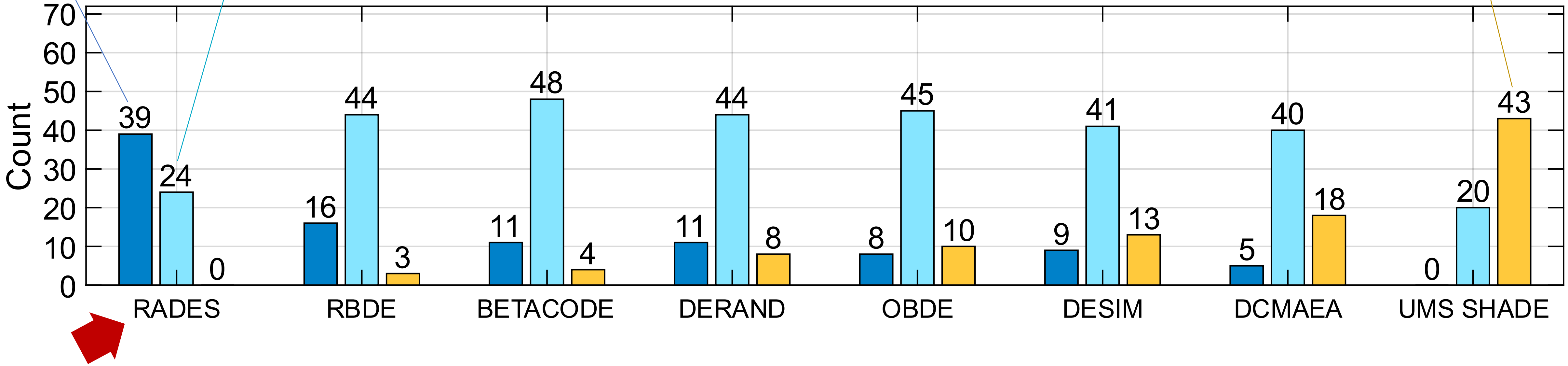
Statistical Comparisons



Number of cases of outperformance

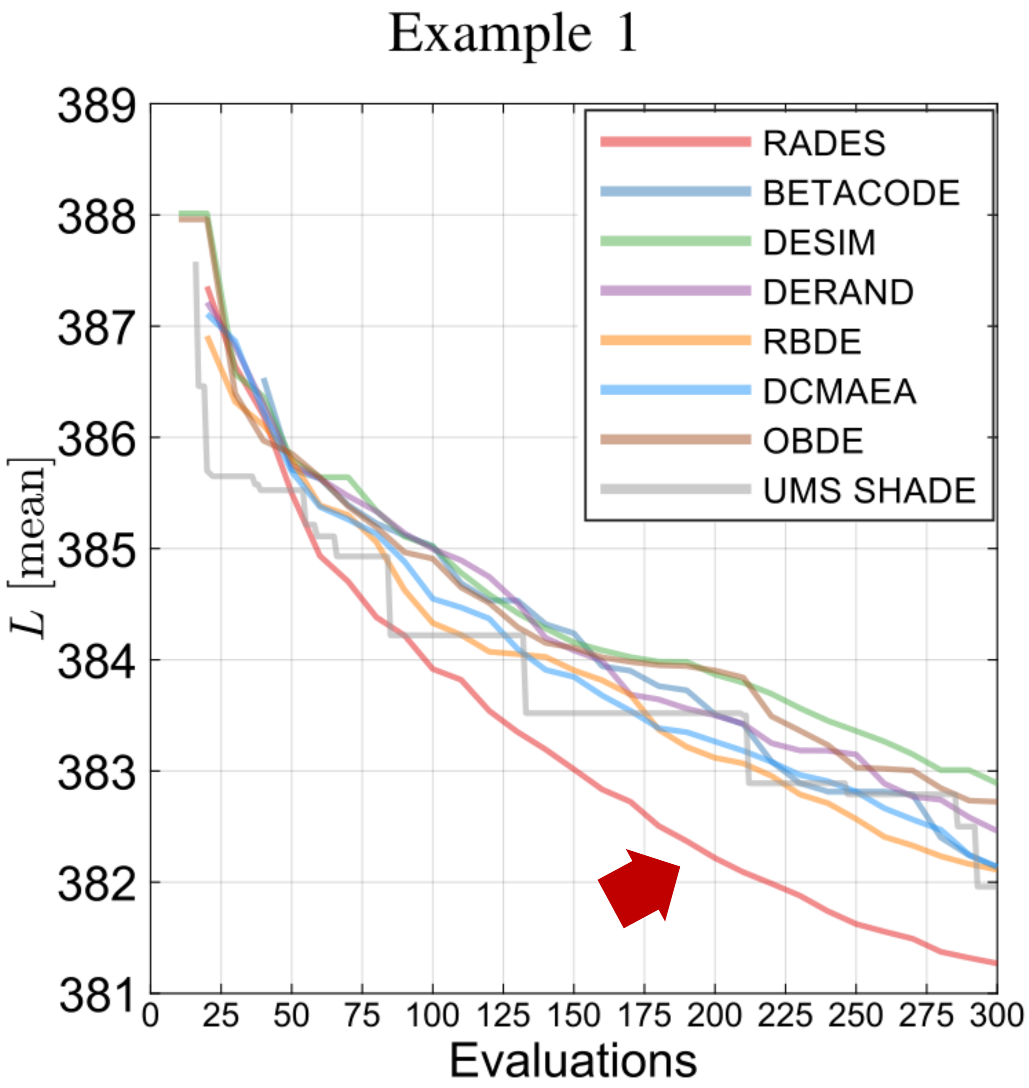
Number of cases of equal outperformance

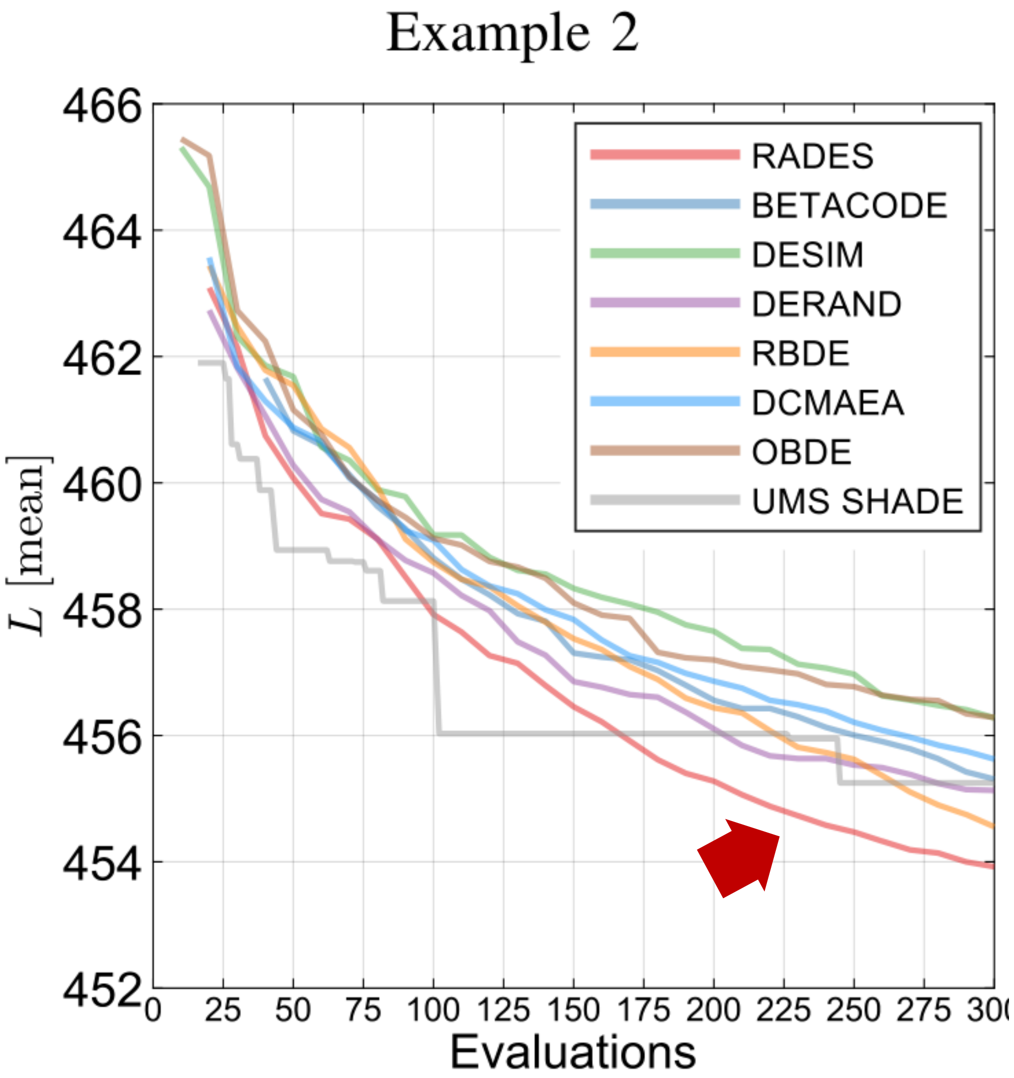
Number of cases of underperformance



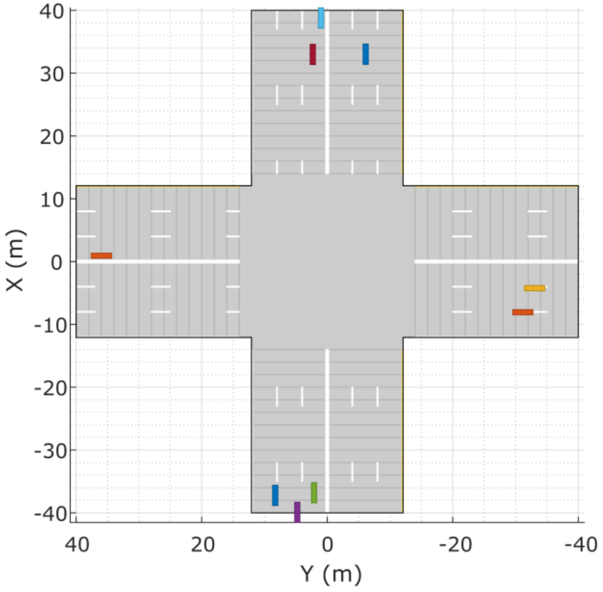
RADES outperforms or performs equally to other baselines

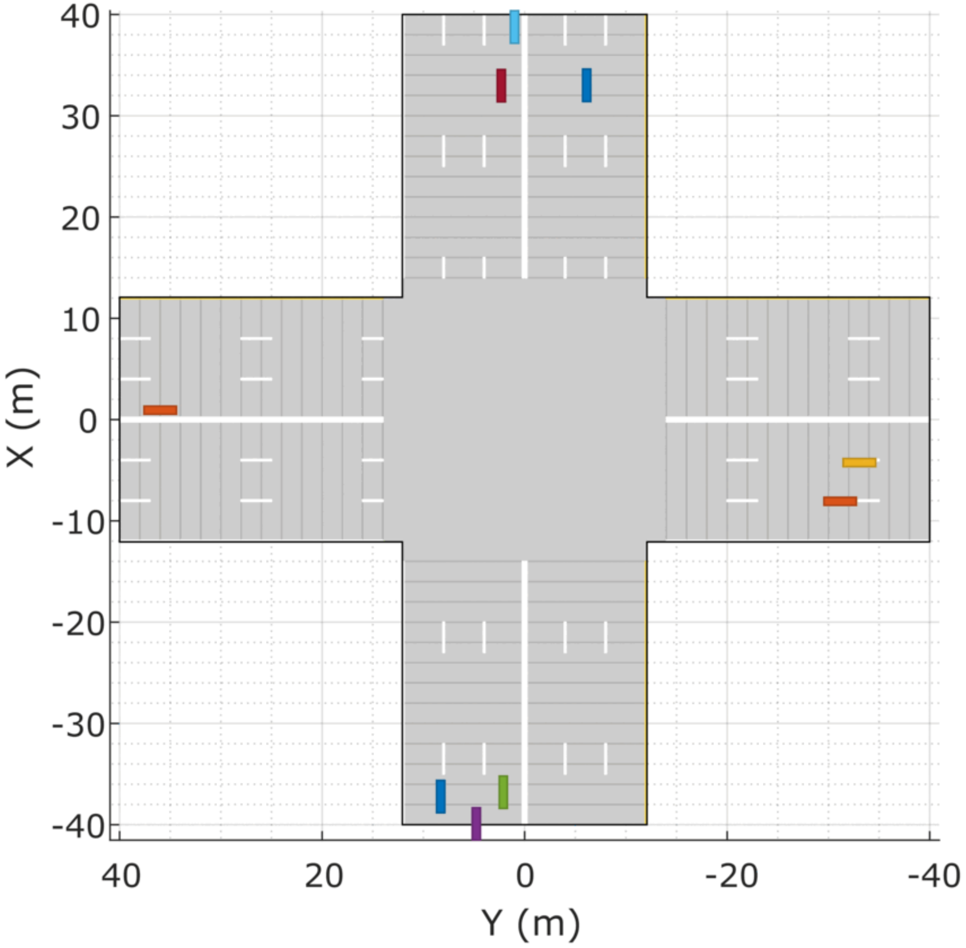
Optimization Convergence

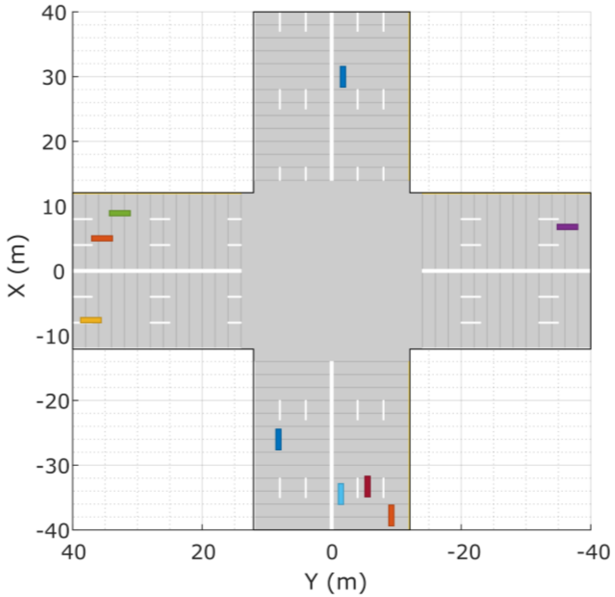




Example







Origin

Destination

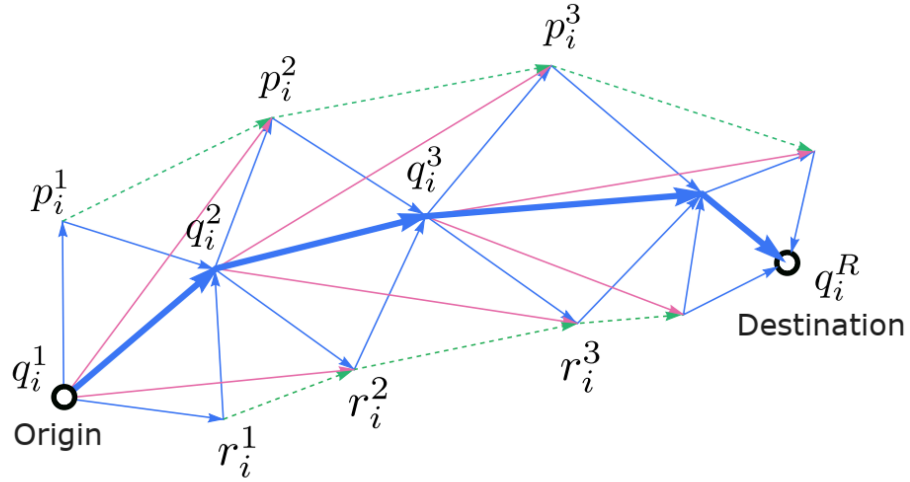
Conclusion
Final Notes

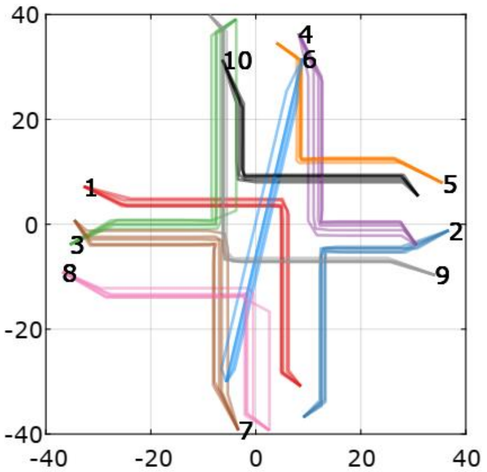
Results

- Coordinated trajectory planning for multi-robots
- Using lattice roadmaps and gradient-free optimization
- Efficient convergence by proposed RADES to composite lattice paths

Future Work

- Curvature
- Arbitrary Environments



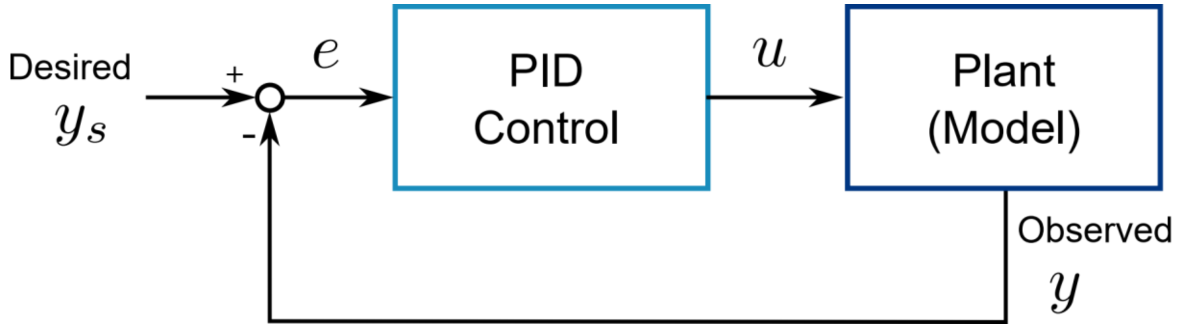


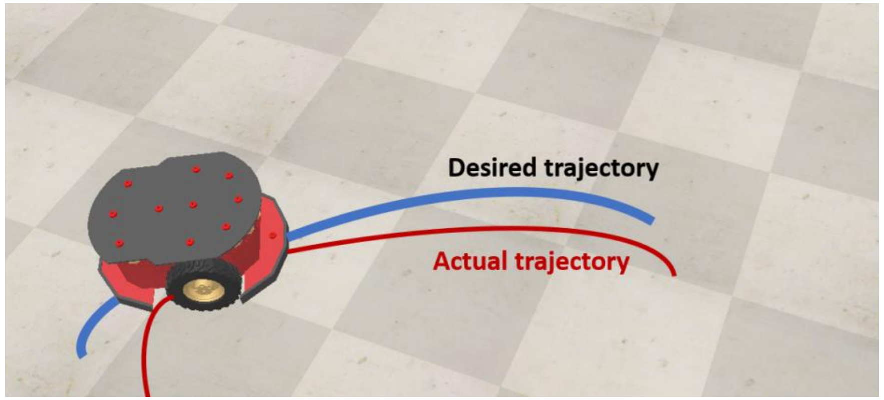
PID Tuning using Differential Evolution with Success-based Particle Adaptations

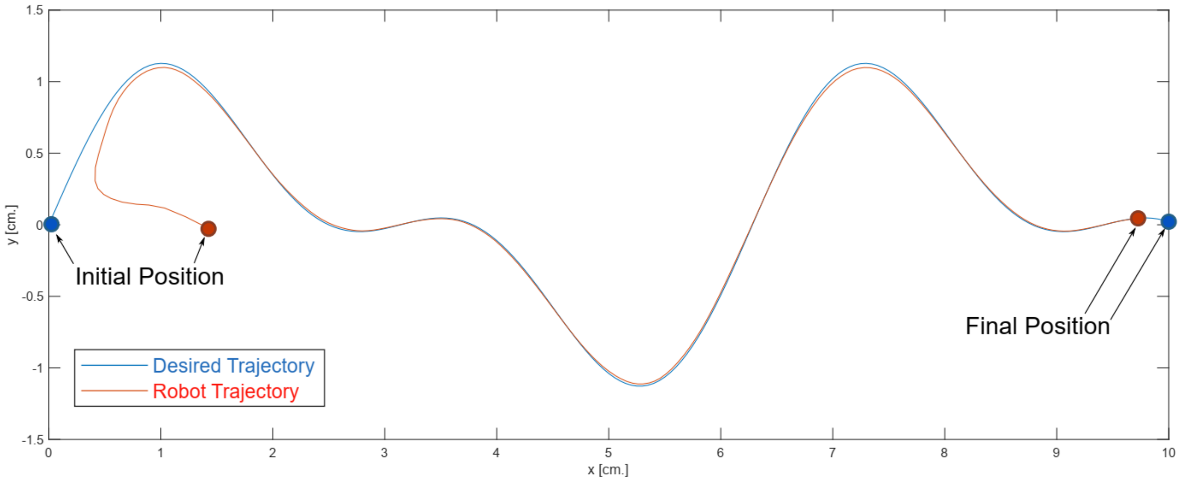
Victor Parque[1], Alaa Khalifa[2]

[1]Hiroshima University, [2]Menoufia University

PID Control
Tracking desired targets







$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt}$$

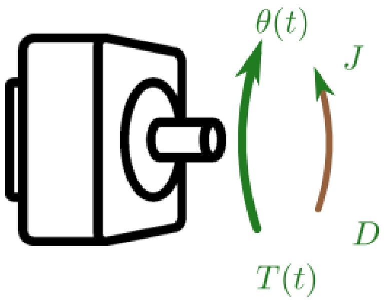
$$e(t) = y_s(t) - y(t) \qquad \boldsymbol{x} = [k_p \; k_i \; k_d]$$

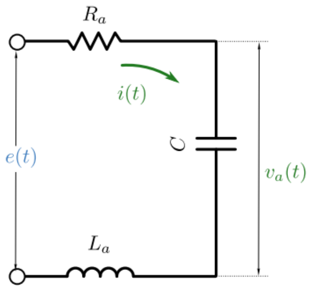
$$\mathrm{IAE}(\boldsymbol{x}) = \int_0^{T_s} |e(t)| dt \qquad \mathrm{ITAE}(\boldsymbol{x}) = \int_0^{T_s} t|e(t)| dt \qquad \mathrm{ITSE}(\boldsymbol{x}) = \int_0^{T_s} t e(t)^2 dt \qquad \mathrm{ISE}(\boldsymbol{x}) = \int_0^{T_s} e(t)^2 dt$$

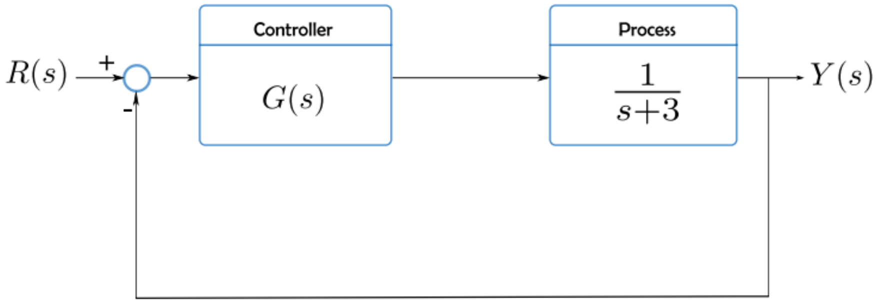
PID Control

Tracking desired targets

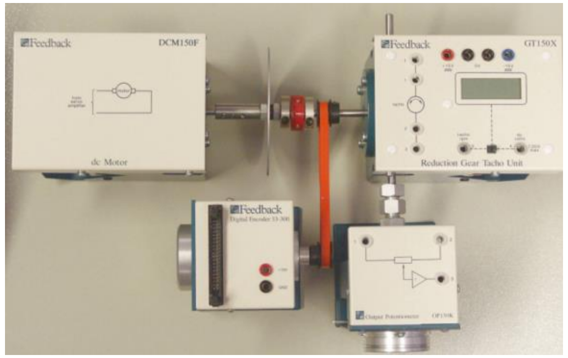
Dynamic Models

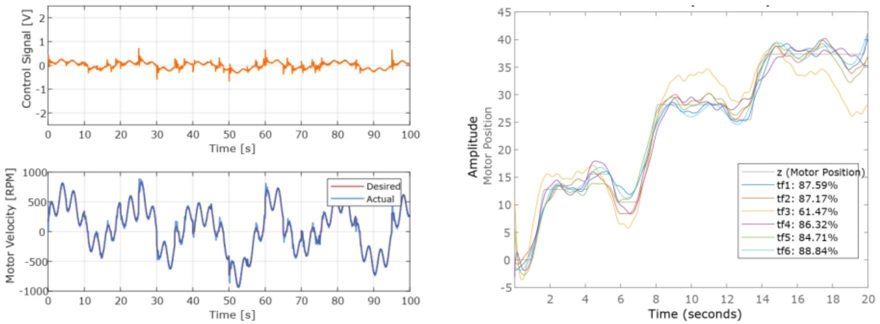






Data-driven Models





Principled

Sim2Real

Practical

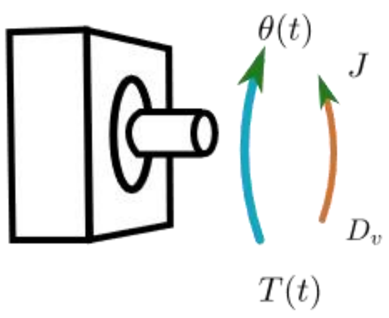
Generalizability

How to design algorithms for generalizable PID tunning

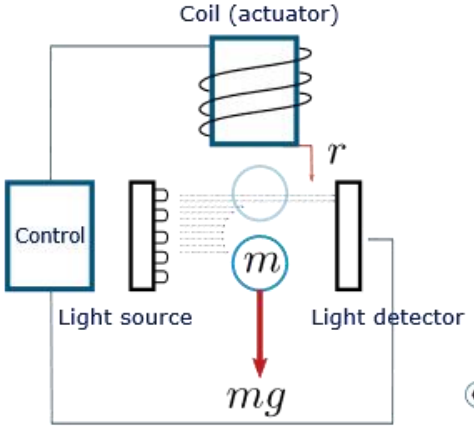
PID Control

Towards generalizable frameworks

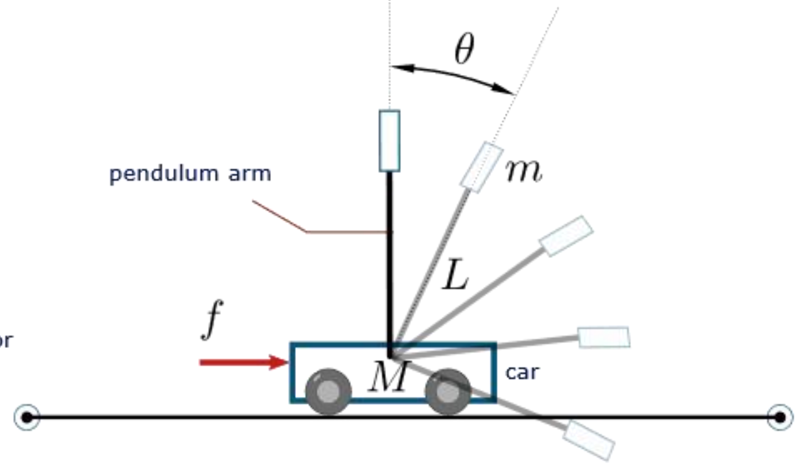
DC Motor Position Control
DC Motor Velocity Control



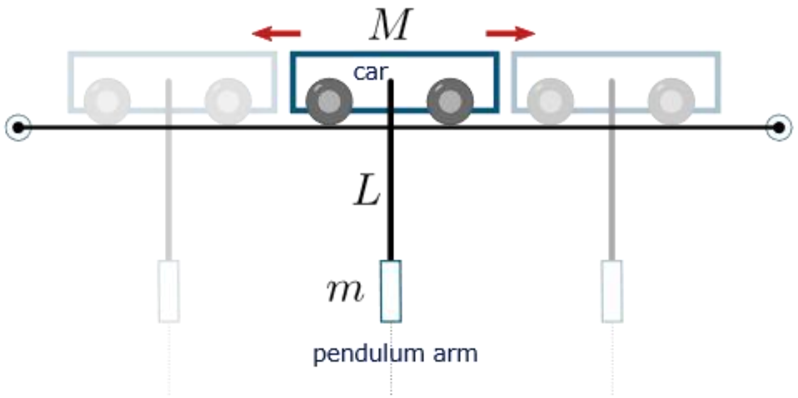
Magnetic Levitation Control

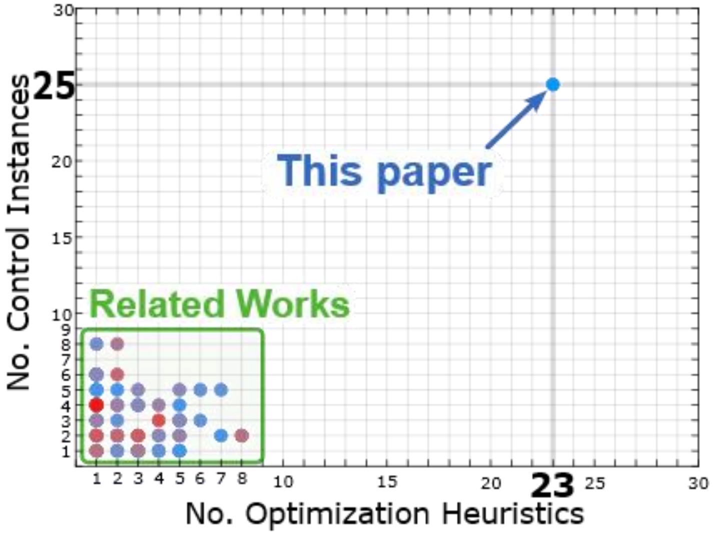


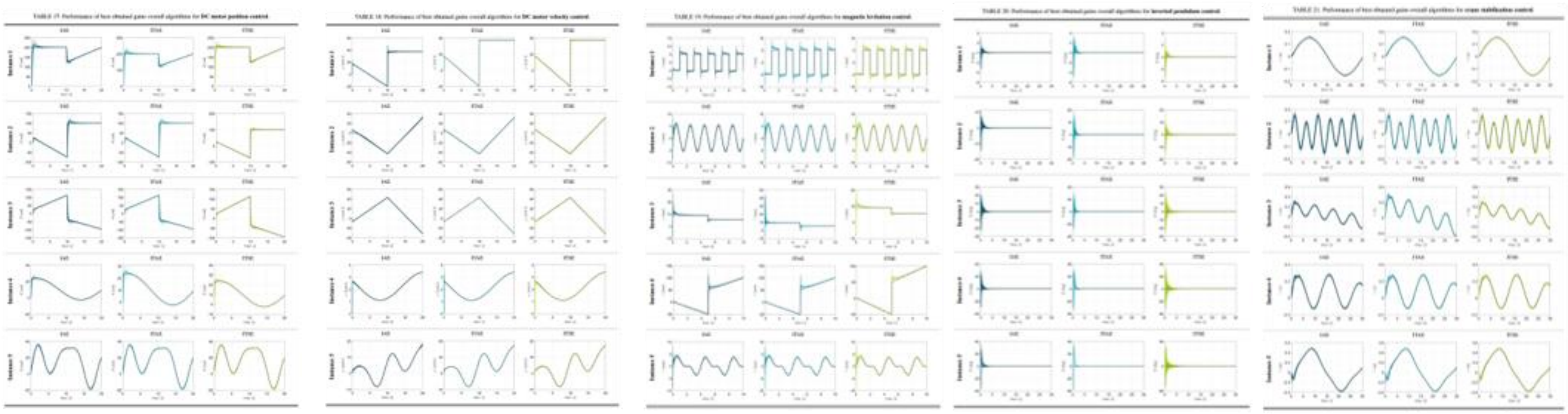
Inverted Pendulum Control



Crane Stabilization Control







PID Control

DEPA: Differential Evolution with Particle Adaptations

$$\boldsymbol{x}_{i,g+1} = \begin{cases} \boldsymbol{u}_{i,g}, & \text{if } f(\boldsymbol{u}_{i,g}) < f(\boldsymbol{x}_{i,g}) \\ \boldsymbol{x}_{i,g}, & \text{otherwise} \end{cases}$$

$$\boldsymbol{u}_{i,g} = \boldsymbol{x}^{r}_{i,g} + \boldsymbol{v}^{*}_{i,g}$$

$$\boldsymbol{v}^{*}_{i,g} = \omega \boldsymbol{v}_{i,g} + F_i(\boldsymbol{x}_{\text{pbest},g} - \boldsymbol{x}^{r}_{i,g})r_1 + G_i(\boldsymbol{x}_{\text{gbest},g} - \boldsymbol{x}^{r}_{i,g})r_2$$

$$\boldsymbol{v}_{i,g+1} = \begin{cases} \boldsymbol{v}^{*}_{i,g}, & \text{if } f(\boldsymbol{u}_{i,g}) < f(\boldsymbol{x}_{i,g}) \\ \boldsymbol{v}_{i,g}, & \text{otherwise} \end{cases}$$

$$\boldsymbol{x}^{r}_{i,g} \in_R \begin{cases} \mathcal{P}, \text{ if } q_i < Q \\ \mathcal{A}, \text{ otherwise} \end{cases}$$

$$\mathcal{A}_g = \begin{cases} \mathcal{P}, \text{ if } g = 0 \\ \mathcal{A}_g \cup \{\boldsymbol{u}_{i,g}\}, \text{ if } g > 0 \text{ and } f(\boldsymbol{u}_{i,g}) < f(\boldsymbol{x}_{i,g}) \end{cases}$$

Algorithm 1: DEPA

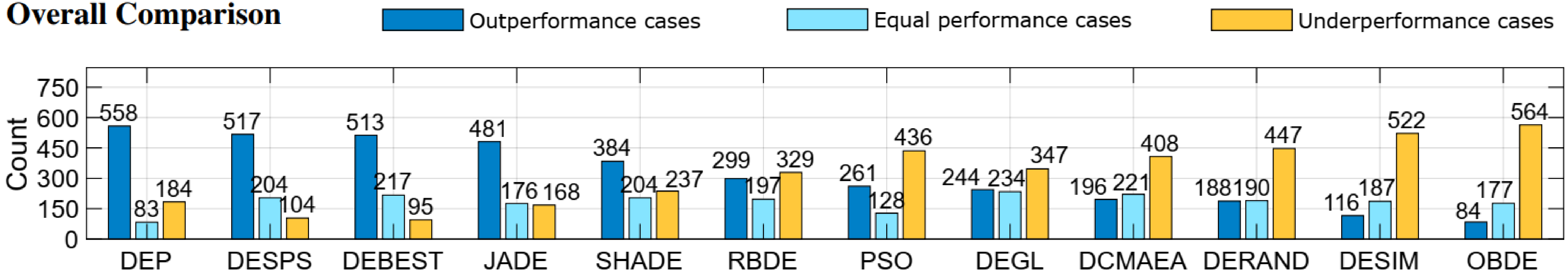
$FEs = 0$, $g = 0$, $q_i = 0$, $k = 0$;
Generate a set of N individuals randomly as initial population set $\mathcal{P}$;
Initialize the archive $\mathcal{A}$ from the population set $\mathcal{P}$;
Initialize the tuples $\mathcal{M}^k_F$ and $\mathcal{M}^k_G$;
$FEs = FEs + N$;
while $FEs \leq MaxFEs$ **do**
    $g = g + 1$;
    $S_F = \{\}$, $S_G = \{\}$;
    for $i = 1$ **to** N **do**
        Sample $a_i \sim U[1,h]$, $b_i \sim U[1,h]$;
        Generate F_i and G_i using (14)-(15);
    end
    Find out the best individual $\boldsymbol{x}_{\text{gbest},g}$ overall $\mathcal{P}$;
    for $i = 1$ **to** N **do**
        Update the i-th best individual $\boldsymbol{x}_{\text{pbest},g}$;
        Generate the trial vector $\boldsymbol{u}_{i,g}$ using (9)-(12);
        if $f(\boldsymbol{u}_{i,g}) < f(\boldsymbol{x}_{i,g})$ **then**
            $\boldsymbol{x}_{i,g+1} = \boldsymbol{u}_{i,g}$, $\boldsymbol{v}_{i,g+1} = \boldsymbol{v}^{*}_{i,g}$;
            $\boldsymbol{u}_{i,g} \to \mathcal{A}$;
            Delete an element from $\mathcal{A}$ if $|\mathcal{A}| > \lambda|\mathcal{P}|$;
            $F_i \to S_F$, $G_i \to S_G$;
            $q_i = 0$;
        else
            $\boldsymbol{x}_{i,g+1} = \boldsymbol{x}_{i,g}$;
            $\boldsymbol{v}_{i,g+1} = \boldsymbol{v}_{i,g}$;
            $q_i = q_i + 1$;
        end
        $FEs = FEs + 1$;
    end
    if $S_F \neq \{\} \wedge S_G \neq \{\}$ **then**
        $k = k + 1$;
        Update $\mathcal{M}^k_F$ and $\mathcal{M}^k_G$ using (16) and (17);
        Set $k = 1$ if $k > h$;
    end
end

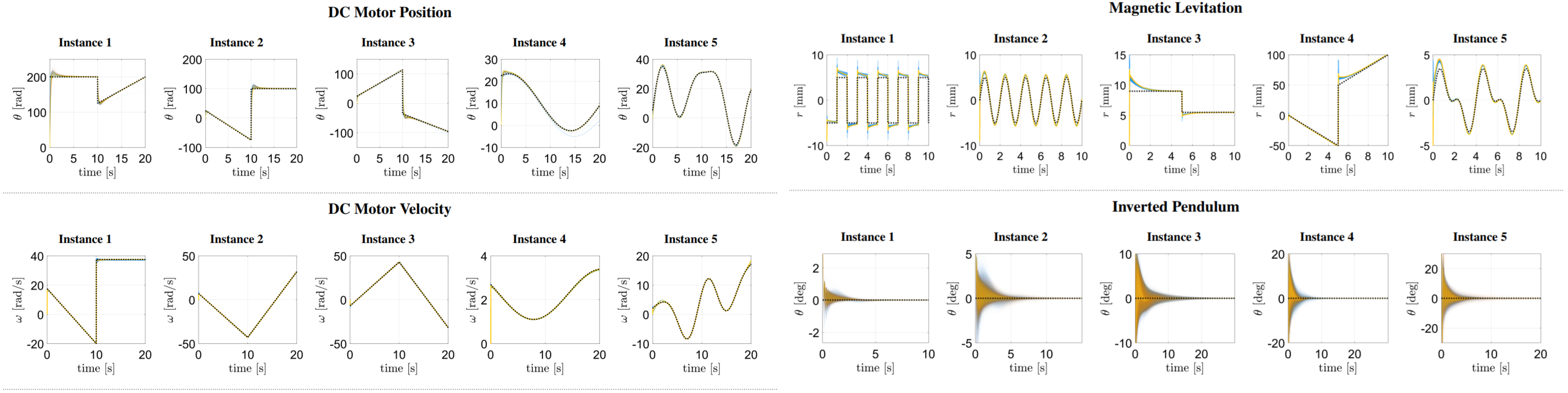
Victor Parque, Alaa Khalifa, "PID Tuning using Differential Evolution with Success-based Particle Adaptations", IEEE Access, 2024

PID Control

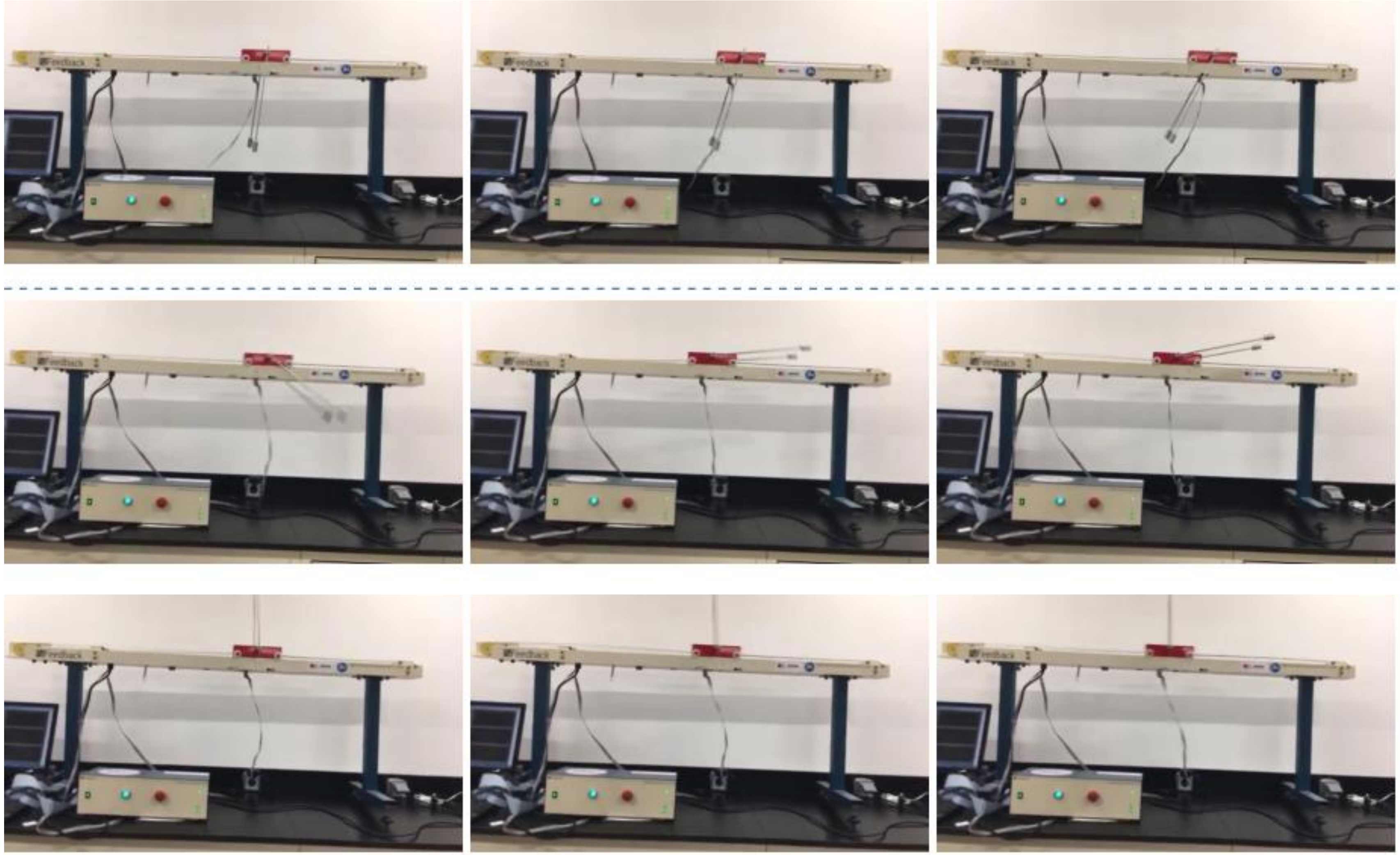
DEPA: Differential Evolution with Particle Adaptations

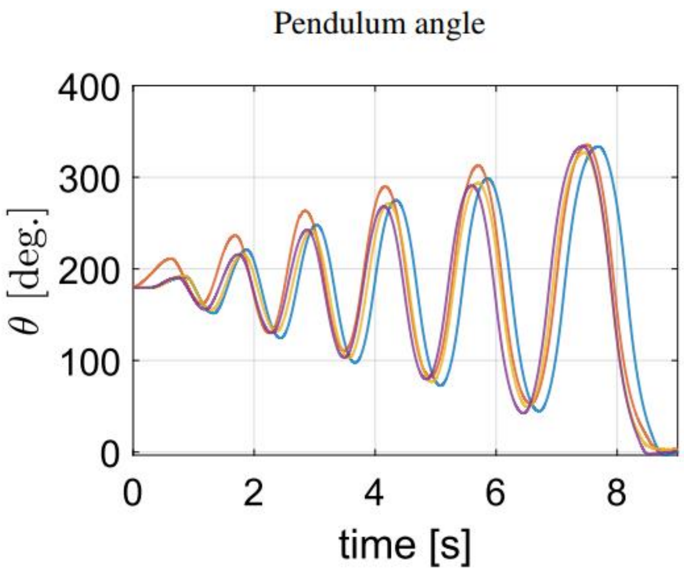
Overall Comparison

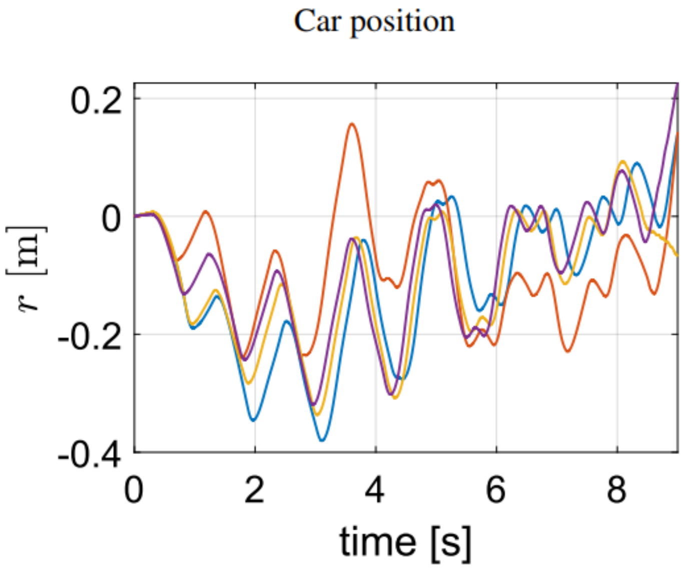




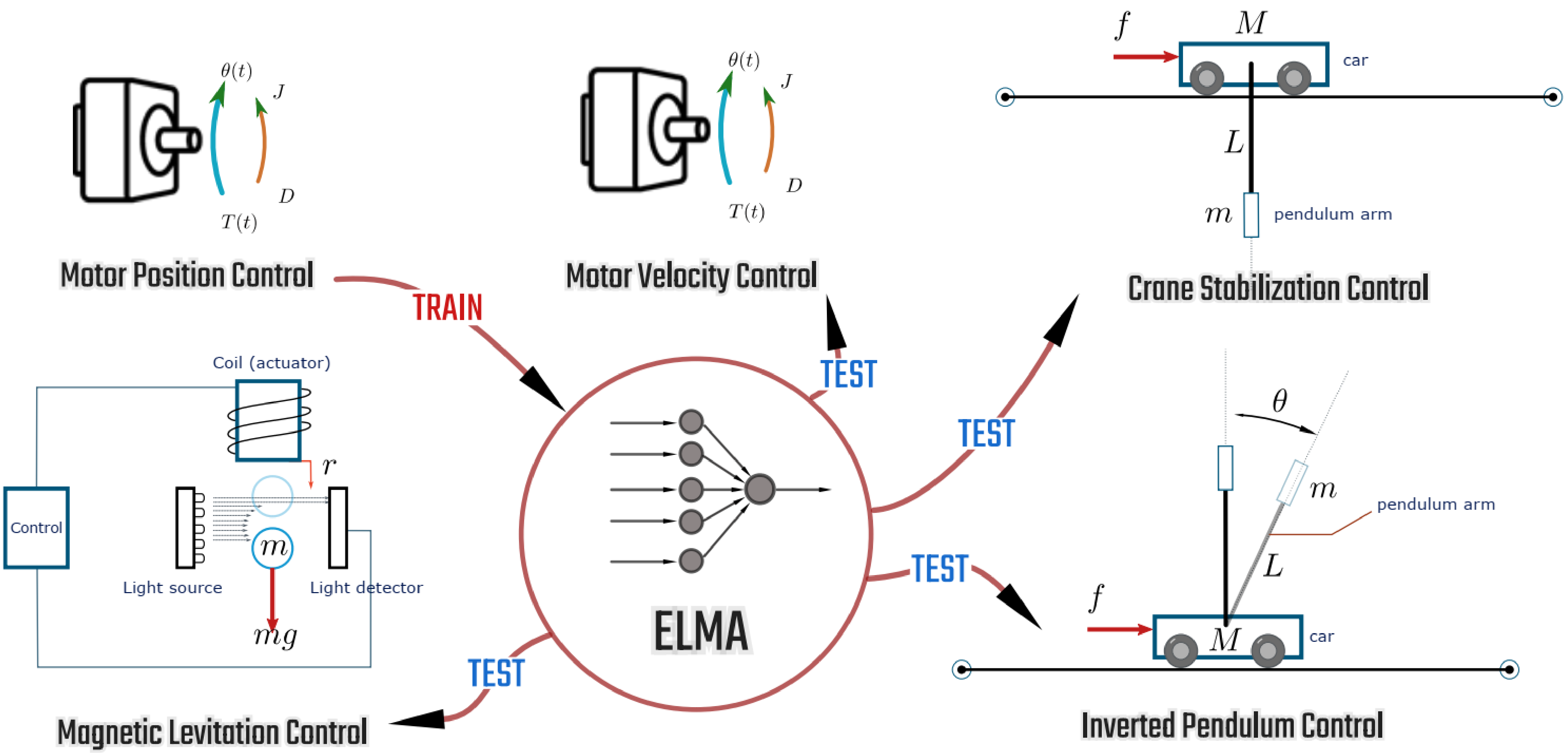
PID Control
Inverted Pendulum





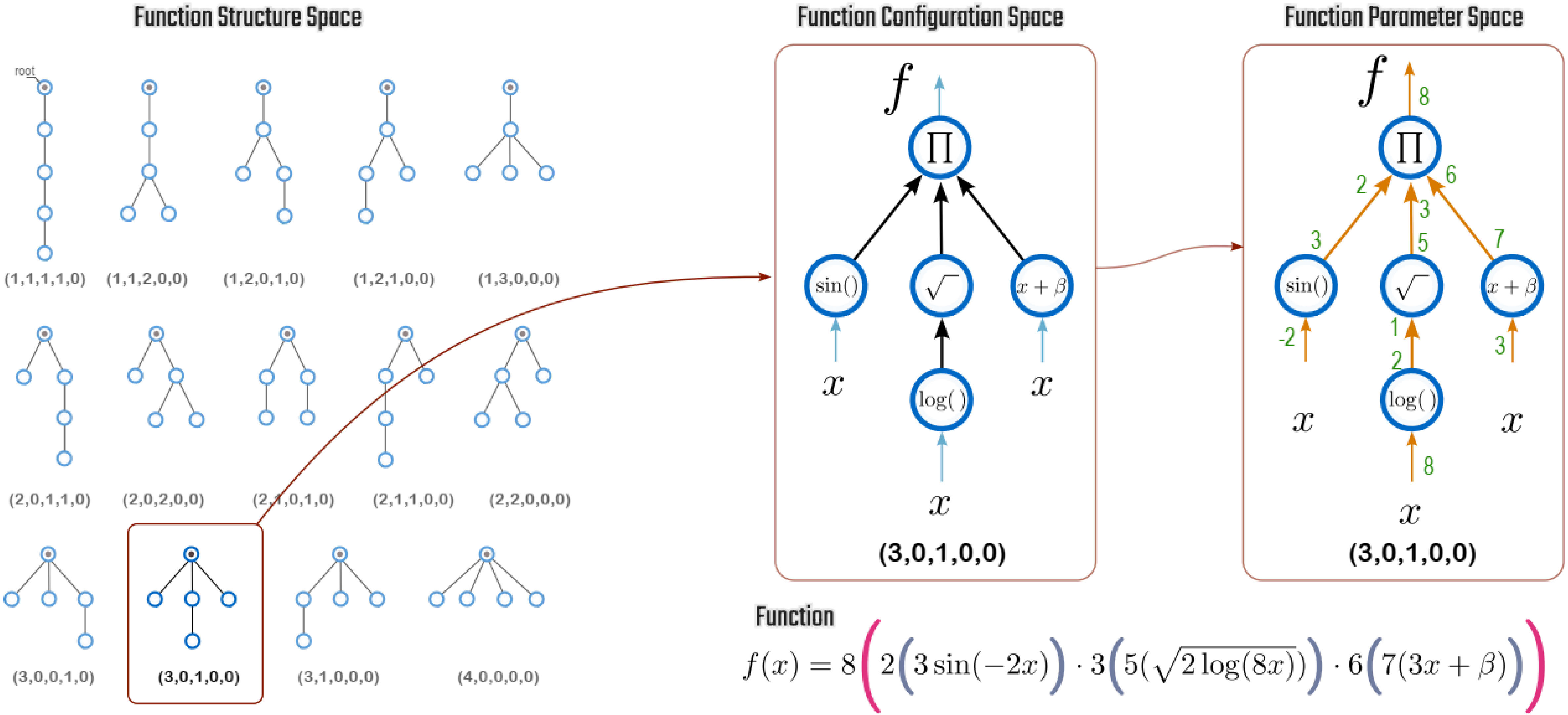


Learning Generalizable Controllers
ELMA: Extreme Learning Machine with Learnable Activation Functions



Learning Generalizable Controllers

ELMA: Extreme Learning Machine with Learnable Activation Functions

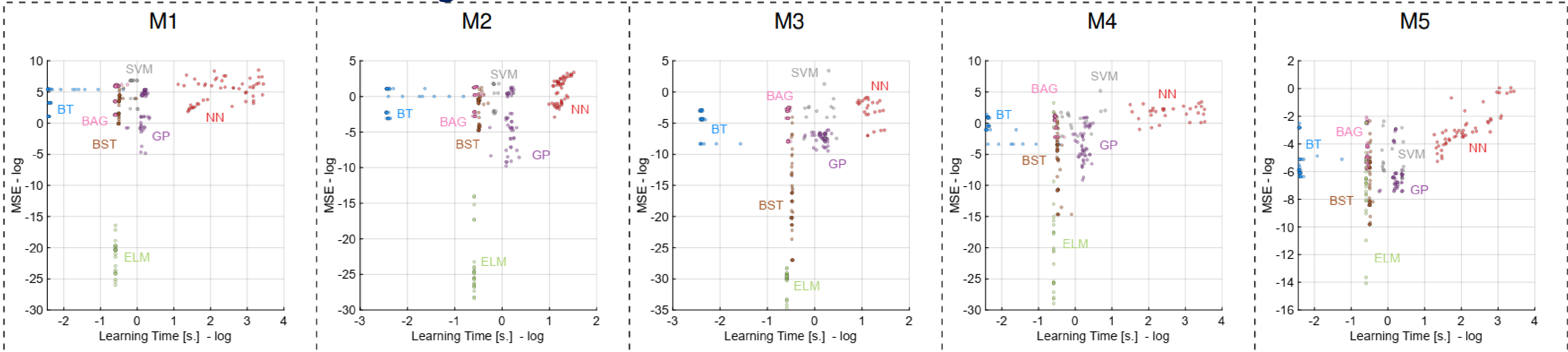


Function

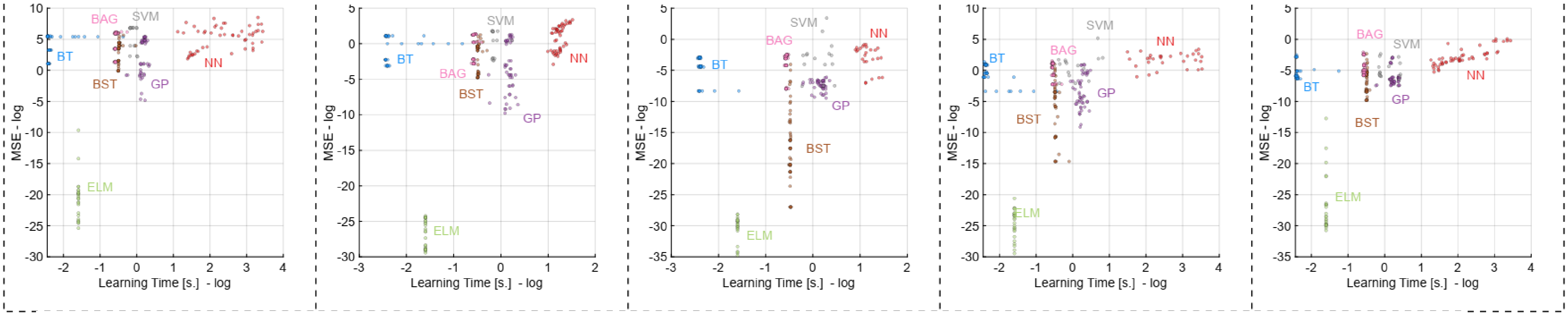
$$f(x) = 8\Big(2\big(3\sin(-2x)\big)\cdot 3\big(5(\sqrt{2\log(8x)})\big)\cdot 6\big(7(3x+\beta)\big)\Big)$$

Learning Generalizable Controllers

ELMA: Extreme Learning Machine with Learnable Activation Functions



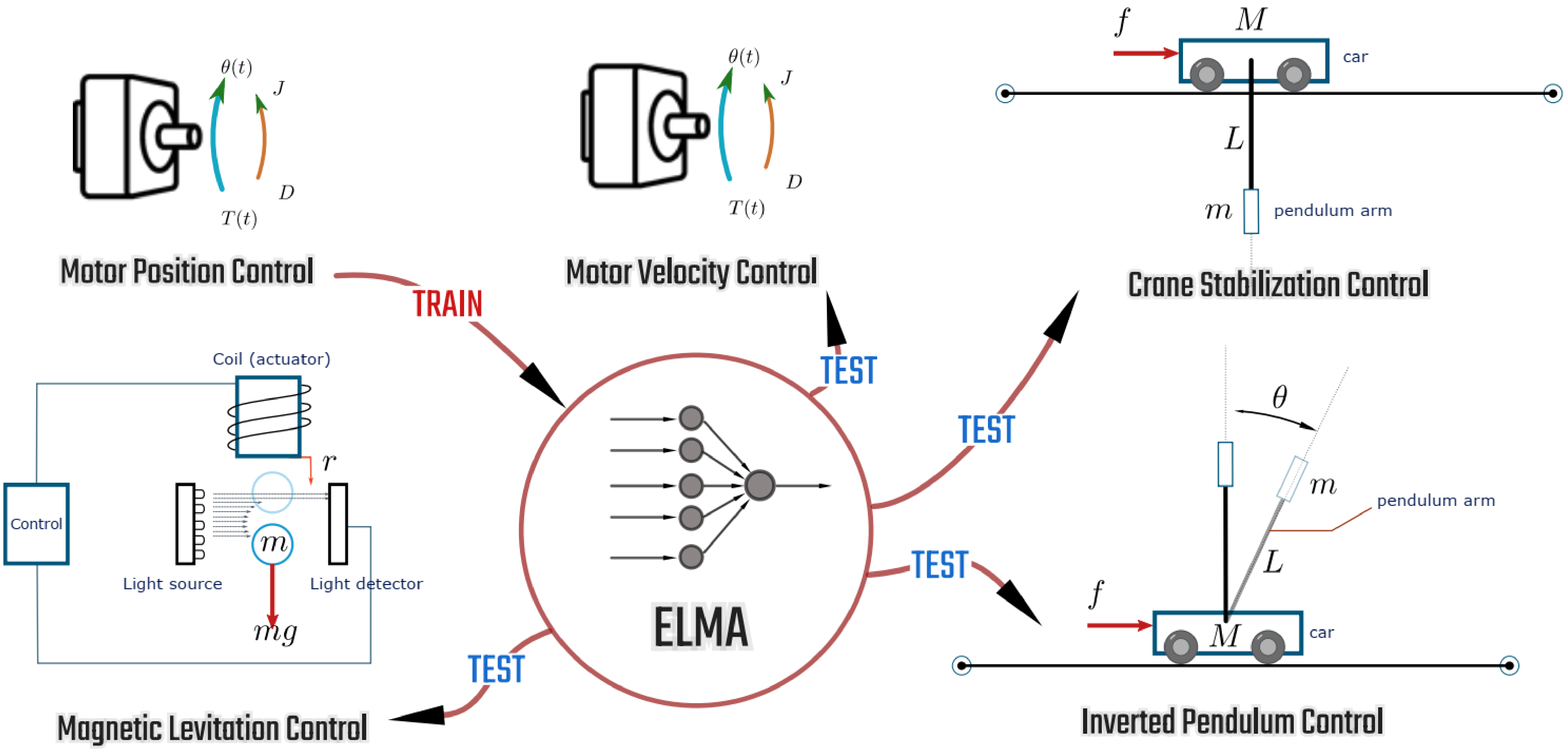
Performance of ELMA when activation functions are learned in M1: **Motor Position Control**



Performance of ELMA when activation functions are learned in M5: **Magnetic Levitation Control**.

Learning Generalizable Controllers

ELMA: Extreme Learning Machine with Learnable Activation Functions



Thank you

Victor Parque

Associate Professor

Informatics and Data Science Program

Graduate School of Advanced Science and Engineering

Hiroshima University, Japan

parque@hiroshima-u.ac.jp