10th UCAAT User Conference on Advanced Automated Testing

ETSI The Standards People

Metaheuristic Algorithms and their Applications to Fuzzy Control, Fuzzy Modeling and Mobile Robot Navigation

Presented by: Radu-Emil Precup

Universitatea Politehnica Timișoara 2020

PRIMUL VEAC DE ÎNVĂȚĂMÂNT UNIVERSITAR ÎN TIMIȘOARA

16/11/2023

Outline

- **The Process Control research group of the Politehnica University of Timisoara**
  - **Nature-inspired optimization algorithms (NIOAs)**
  - **NIOAs in the optimal tuning of fuzzy controllers**
  - **NIOAs in the optimal tuning of fuzzy models**
  - **NIOAs in optimal path planning algorithms for mobile robots**

Our university

- Politehnica University of Timisoara (UPT), a **symbol of higher education** in Central and Eastern Europe, has contributed since its early stages to the **development of Timisoara**, a city of constant economic growth, an example for social and cultural effervescence and a university city of incredible energy
- **In 1920** the city obtained the title of **Civitas Academica** due to the creation of our university. This has contributed to the evolution of the city into a first class industrial center in Romania, by attracting to the capital of Banat many large foreign companies that provide Timisoara's citizens with tens of thousands of jobs. Due to its academics and professors, to the impressive scientific discoveries and inventions, to its over 135.000 graduates, UPT has had a great influence on the entire scientific, economic and socio-cultural landscape in Western Romania

The Process Control research group

Prof. Radu-Emil Precup, Prof. Stefan Preitl, Assoc. Prof. Florin Dragan, Assoc. Prof. Adriana Albu, Assoc. Prof. Loredana Stanciu, Assoc. Prof. Claudia-Adina Bojan-Dragos, Lect. Alexandra-Iulia Szedlak-Stinean, Lect. Daniel Iercan, Lect. Raul-Cristian Roman, Assist. Lect. Elena-Lorena Hedrea, Assist. Lect. Emil-Ioan Voisan, PhD student Teodor-Adrian Teban, PhD student Lucian-Ovidiu Fedorovici, PhD student Iuliu Alexandru Zamfirache, PhD student Miruna-Maria Damian, PhD student Monica-Lavinia Nedelcea + PhD students …

The small team

Assoc. Prof. Claudia-Adina
BOJAN-DRAGOȘ

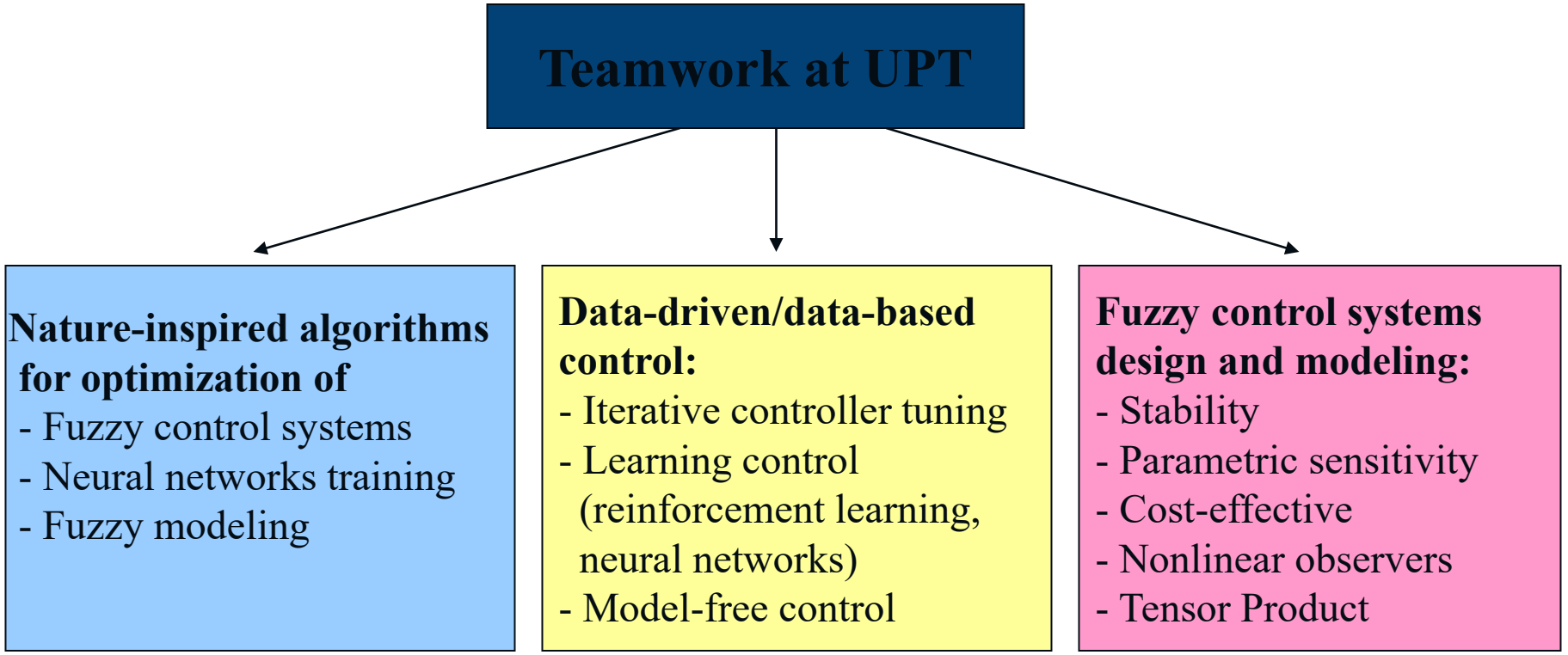
Prof. Radu-Emil **PRECUP**

Lect. Alexandra-Iulia
SZEDLAK-STÎNEAN

Assist. Lect.
Elena-Lorena **HEDREA**

Lect. Raul-Cristian
ROMAN

Teamwork at UPT

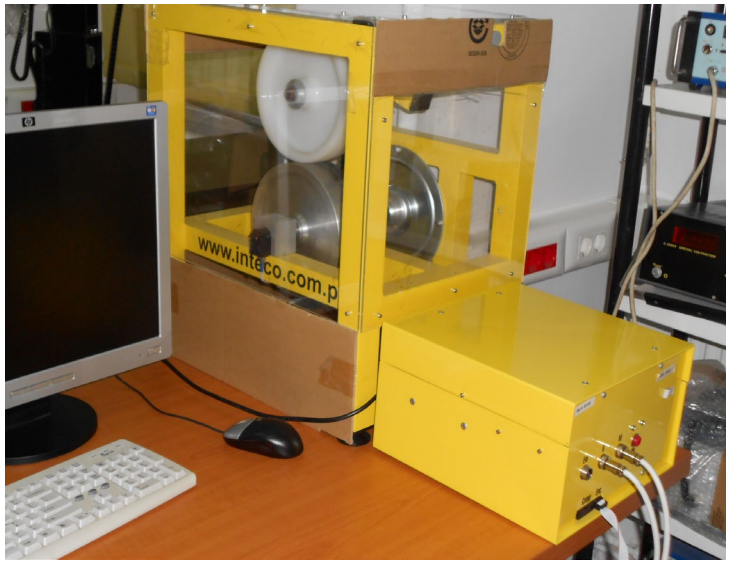


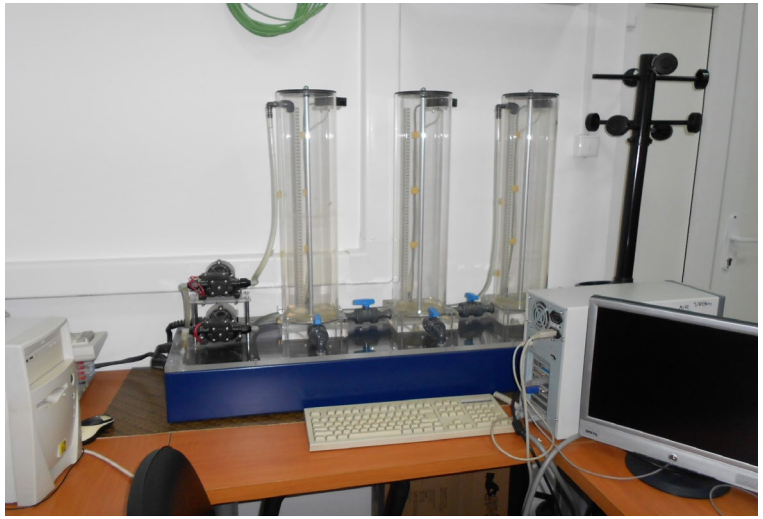
Applications: control and modeling of mechatronics applications, automotive processes, biomedical processes, mobile robots

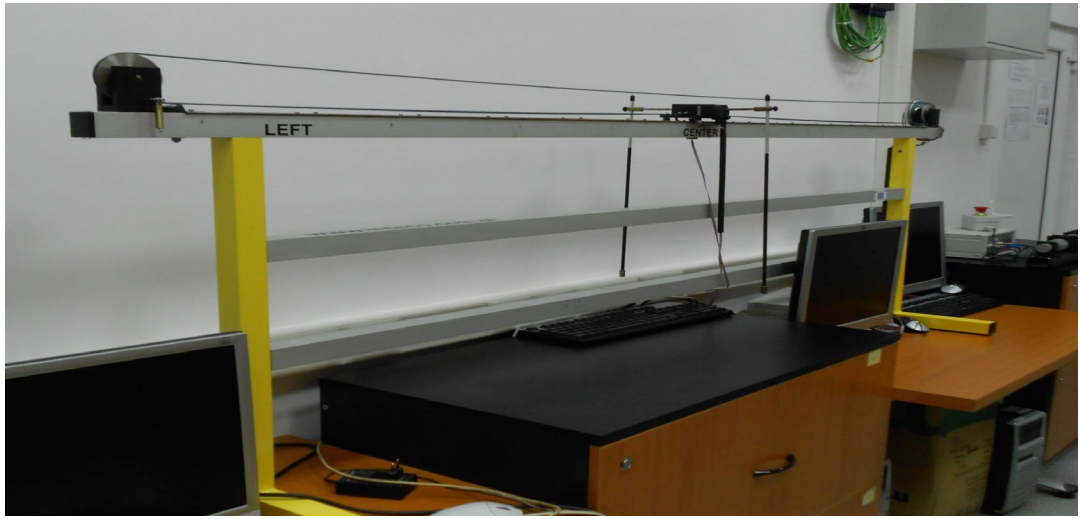
Cooperation

- **Budapest University of Technology and Economics**, Hungary (2002 →), Acad. István Nagy, Prof. Péter Korondi (now with **University of Debrecen**, Hungary) and their teams: control algorithms for mobile robots operating in Intelligent Space (Hashimoto Lab, University of Tokyo)
- **Óbuda University**, Budapest, Hungary (2003 →), Prof. Imre J. Rudas, Prof. János Fodor, Prof. Levente Kovács and their teams: fuzzy (control) systems, telerobotics control problems in space medicine
- **Bremen University**, Germany (2005 →), Prof. Axel Gräser and his team: Iterative Feedback Tuning and automotive control
- **University of Ljubljana**, Slovenia (2006 →), Prof. Igor Škrjanc, Prof. Sašo Blažič and their teams: fuzzy control systems
- **University of Ottawa**, Canada (2007 →), Prof. Emil M. Petriu and his team: soft computing and signal processing
- **Delft University of Technology**, The Netherlands (2007 →), Prof. Hans Hellendoorn and his team: industrial applications of fuzzy control
- **Hungarian Academy of Sciences** (2008 →), Prof. Péter Baranyi (now with **Széchenyi István University**, Győr, Hungary) and his team: tensor product-based model transformation
- **Coventry University**, UK (2009 →), Prof. Keith J. Burnham (now with **University of Wolverhampton**, UK) and his team: control systems and system identification
- **Széchenyi István University**, Győr, Hungary (2009 →), Prof. László T. Kóczy, Prof. Claudiu Pozna and their team: fuzzy logic, artificial intelligence and robotics
- **Lancaster University**, UK (2014 →), Prof. Plamen Angelov and his team: evolving fuzzy systems
- **China University of Petroleum**, East China (2022→), Prof. Jian Wang and his team: computational intelligence
- **Chalmers University of Technology**, Sweden, **Swedish National Road and Transport Research Institute** (VTI), **Zhejiang University**, **Hong Kong Polytechnic University**, **Chongqing University**, China (2023→), Dr. Kun Gao, Prof. Sheng Jin, Prof. Shuaian Wang, Prof. Xiaosong Hu, Dr. David Daniels and their teams: electric multimodal transport systems for enhancing urban accessibility and connectivity
- **Université Polytechnique Hauts-de-France**, **INSA Hauts-de-France**, France (2023 →), Prof. Anh-Tu Nguyen and his team: fuzzy control in mechatronics applications
- Universities, companies and local authorities in Romania, Sweden and China

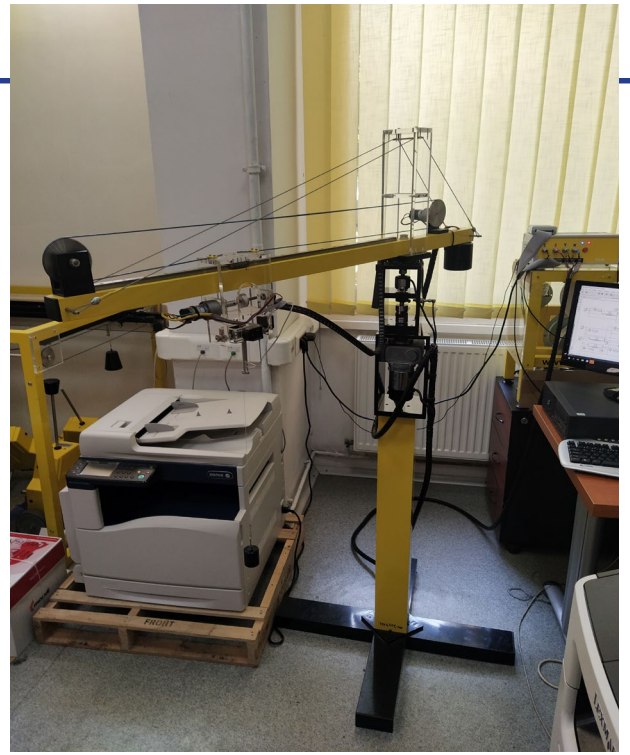
Our labs / 1

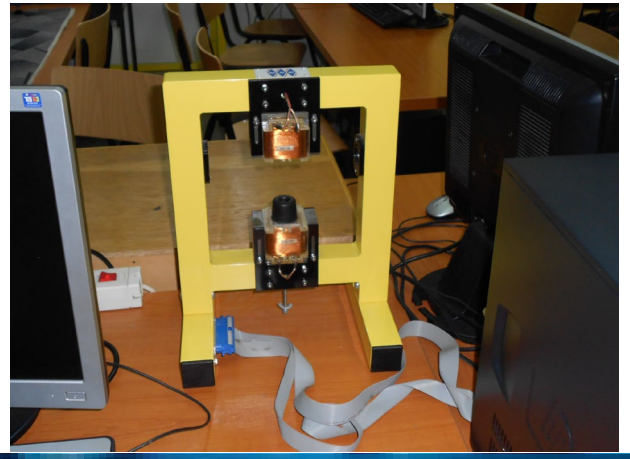




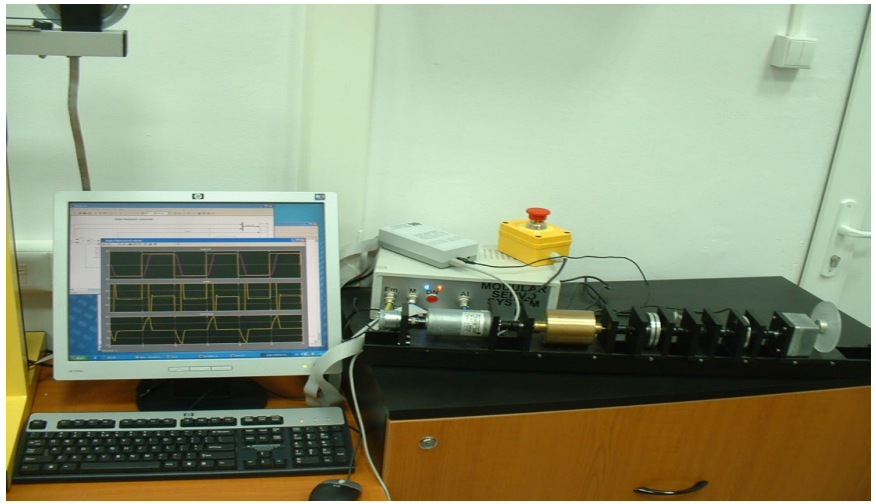


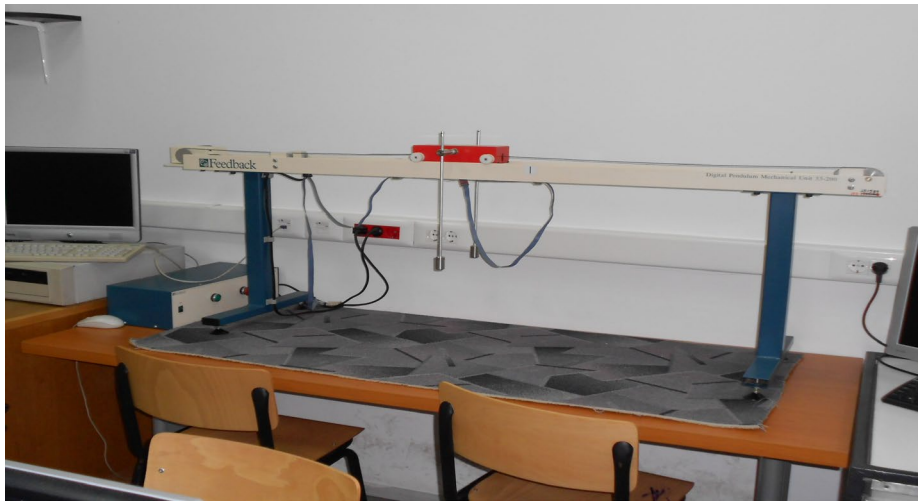
Our labs / 2

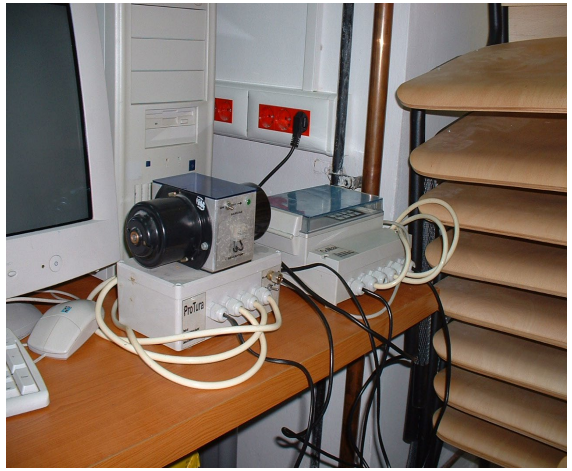




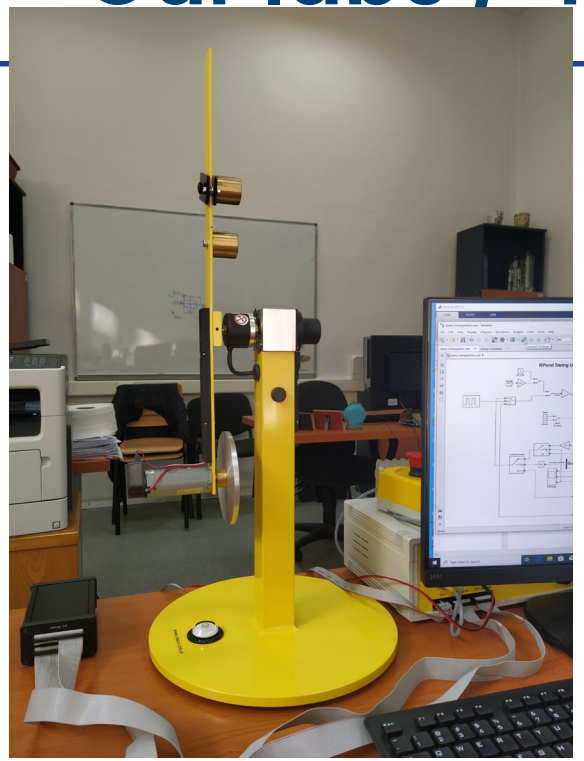
Our labs / 3

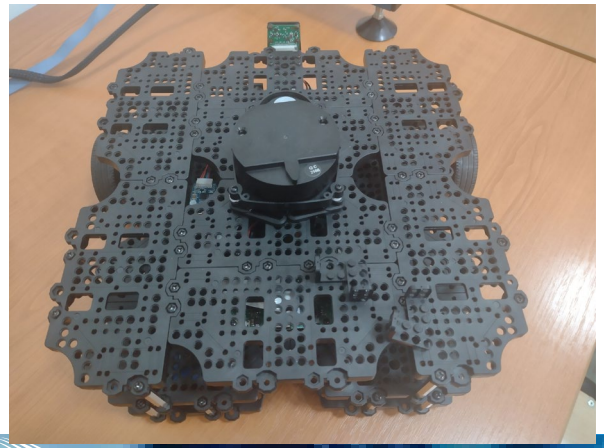




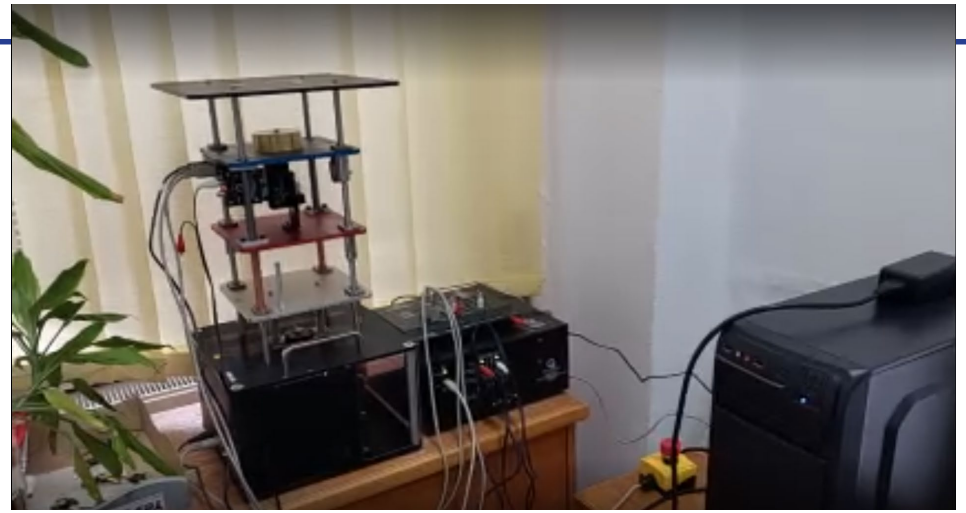


Our labs / 4





Our labs / 5



> 450000 EUR ...

Outline

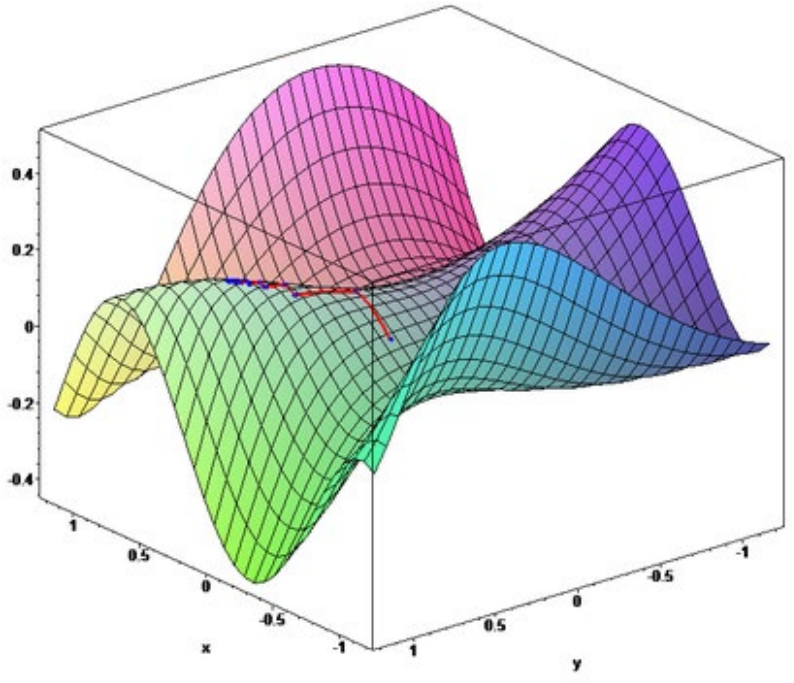
- The Process Control research group of the Politehnica University of Timisoara
- **Nature-inspired optimization algorithms (NIOAs)**
- NIOAs in the optimal tuning of fuzzy controllers
- NIOAs in the optimal tuning of fuzzy models
- NIOAs in optimal path planning algorithms for mobile robots

Nature-inspired optimization algorithms (NIOAs) / 1

- An optimization problem is the problem of finding the best solution (i.e., the optimal solution) from all feasible solutions
- There are two key components in an optimization problem: the objective function and the constraints (optional)
- The objective function (or the cost function) assesses and compares the solutions in the context of all feasible solutions by computing the desired quantity to be minimized or maximized.
- The constraints can be added to limit the possible values for the variables of the objective function

NIOAs / 2

- The optimization algorithms find the optimal solutions by trying variations on the initial solution using the information gained to improve the solution (learning)
- The complexity of the classical algorithms is very high which requires enormous amount of computational work. Therefore, alternative algorithms with lower complexity are appreciated
- The NIOAs to find optimal solutions became very popular as metaheuristic algorithms as they are much better in terms of efficiency and complexity than classical algorithms



NIOAs / 3

- Generally, these algorithms are based on biological, physical, and chemical phenomena of nature
- NIOAs have the distinct ability of finding the global minimum (or maximum) of certain objective functions (or cost functions) under specific conditions
- In addition, the analytical expression of the objective functions depending on other design (or tuning) parameters may be difficult or even impossible to formulate

Popular NIOAs in chronological order / 1

- 1965 Evolution Strategies (Kennedy et al., 1995)
- 1966 Evolutionary Programming (Melin et al., 2013)
- 1975 Genetic Algorithms (Holland, 1975)
- 1979 Cultural Algorithms (Reynolds, 1994)
- 1983 Simulated Annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983)
- 1989 Tabu Search (Glover, 1989)
- 1992 Ant Colony Optimization (Colorni, Dorigo, & Maniezzo, 1992)

Popular NIOAs in chronological order / 2

- 1995 Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995)
- 2002 Estimation of Distribution Algorithm (Ocenasek & Schwarz, 2002)
- 2002 Bacterial Foraging Algorithm (Tang, Wu, & Saunders, 2006)
- 2005 Honey Bee Algorithm (Teodorovic & Dell'orco, 2005)
- 2005 Harmony Search Algorithm (Geem, 2008)
- 2007 Intelligent water drops (Shah-Hosseini, 2009)
- 2007 Firefly Algorithm (Gandomi & Alavi, 2011)

Popular NIOAs in chronological order / 3

- 2009 Gravitational Search Algorithm (GSA) (Rashedi & Nezamabadi-pour, 2009)
- 2009 Cuckoo Search Algorithm (Gandomi et al., 2013)
- 2010 Artificial Bee Algorithm (Karaboga & Akay, 2009)
- 2010 Bat Algorithm (Yang, 2010)
- 2010 Charged System Search (CSS) (Kaveh & Talatahari, 2010)
- 2012 Krill Herd (Gandomi & Alavi, 2012)
- 2013 Backtracking Search Optimization Algorithm (BSOA) (Civicioglu, 2013)

Popular NIOAs in chronological order / 4

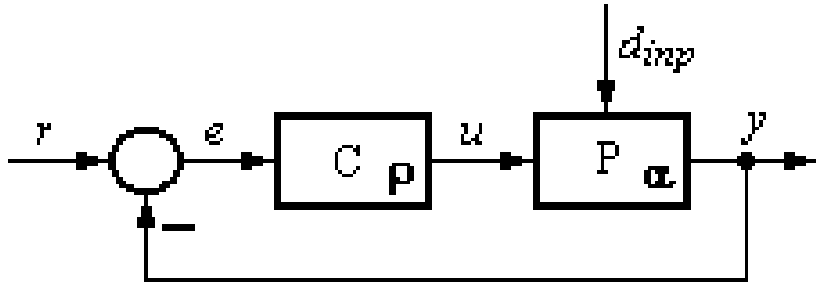
- 2014 Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014)
- 2016 Whale Optimization Algorithm (WOA) (Mirjalili and Lewis, 2016)
- 2017 Salp Swarm Algorithm (Mirjalili et al., 2017)
- 2017 Grasshopper Optimisation Algorithm (Saremi et al., 2017)
- 2017 Ideal Gas Molecular Movement Algorithm (Varaee & Ghasemi, 2017)
- 2020 Slime Mould Algorithm (SMA) (Li et al., 2020)
- 2021 African Vultures Optimization Algorithm (AVOA) (Abdollahzadeh et al., 2021)

Outline

- The Process Control research group of the Politehnica University of Timisoara
- Nature-inspired optimization algorithms (NIOAs)
- **NIOAs in the optimal tuning of fuzzy controllers**
- NIOAs in the optimal tuning of fuzzy models
- NIOAs in optimal path planning algorithms for mobile robots

Definition of optimization problems / 1

- Control system structure:



- Control error: $e = r - y$
- Process parameters: $\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2 \quad \ldots \quad \alpha_m]^T \in R^m$
- Tuning parameters: $\boldsymbol{\rho} = [\rho_1 \quad \rho_2 \quad \ldots \quad \rho_q]^T \in R^q$
- State vector of process: $\mathbf{x}_P = [x_{P,1} \quad x_{P,2} \quad \ldots \quad x_{P,n}]^T \in R^n$
- State vector of controller: $\mathbf{x}_C = [x_{C,1} \quad x_{C,2} \quad \ldots \quad x_{C,p}]^T \in R^p$

Definition of optimization problems / 2

- Differentiable state-space model of the control system – state vector of the control system:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_P \\ \mathbf{x}_C \end{bmatrix} = [x_1 \quad x_2 \quad \dots \quad x_{n+p}]^T \in R^{n+p}, \; x_b = \begin{cases} x_{P,i} & \text{if } b = 1 \dots n \\ x_{C,i-n} & \text{otherwise} \end{cases}, \; b = 1 \dots n+p$$

- State and output sensitivity functions:

$$\lambda_b^{\alpha_a} = \left[\frac{\partial x_b}{\partial \alpha_a} \right]_{\alpha_a, 0}, \; \sigma^{\alpha_a} = \left[\frac{\partial y}{\partial \alpha_a} \right]_{\alpha_a, 0}, \; b = 1 \dots n+p, \; a = 1 \dots m$$

- Objective functions and optimization problems:

$$I_{ISE}^{\alpha_a}(\boldsymbol{\rho}) = \sum_{t=0}^{\infty} \{ e^2(t) + (\gamma^{\alpha_a})^2 [\sigma^{\alpha_a}(t)]^2 \}, \; \boldsymbol{\rho}^* = \arg\min_{\boldsymbol{\rho} \in Do} I_{ISE}^{\alpha_a}(\boldsymbol{\rho}), \; a = 1 \dots m$$

GSA / 1

For *N* agents and a *q*-dimensional search space the position of i^{th} agent is defined by the vector

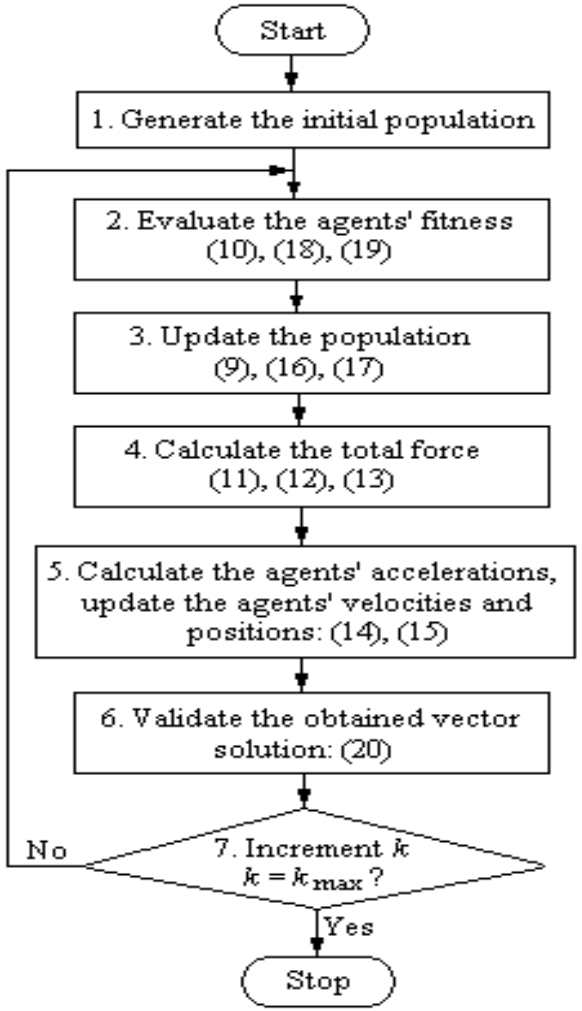
$$\mathbf{X}_i = [x_i^1 \quad \dots \quad x_i^d \quad \dots \quad x_i^q]^T, \; i = 1 \dots N$$

Mapping GSA onto optimization problem: relation fitness function *f* in terms of fitness value $f_i(k)$ in GSA – objective function in optimization problem:

$$f_j(k) = I_{ISE}^{\alpha_a}(\boldsymbol{\rho}), \; a = 1 \dots m, \; j = 1 \dots N$$

The relation between the agents' position vector in the GSA and the parameter vector of the fuzzy controller:

$$\mathbf{X}_i = \boldsymbol{\rho}, \; i = 1 \dots N$$



GSA / 2

Depreciation law of the gravitational constant with the advance of GSA's iterations:

$$g(k) = g_0 \exp(-\zeta\, k / k_{\max})$$

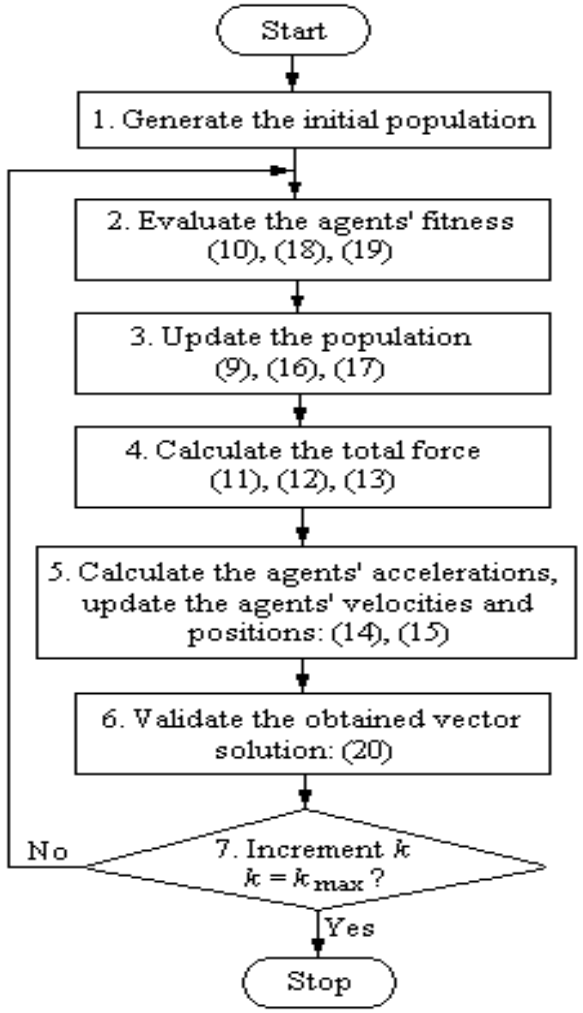
The passive gravitational mass and the inertial masses:

$$n_i(k) = \frac{f_i(k) - w(k)}{b(k) - w(k)},\ m_i(k) = \frac{n_i(k)}{\sum_{j=1}^{N} n_j(k)},$$

$$m_{Ai} = m_{Ii} = m_i$$

The fitness of the best and of the worst agent:

$$b(k) = \min_{j=1\ldots n} f_j(k),\ w(k) = \max_{j=1\ldots n} f_j(k)$$



Calculation and update of agents' accelerations, velocities and speeds:

$$a_i^d(k) = F_i^d(k) / m_{Ii}(k)$$

$$F_i^d(k) = \sum_{j=1, j\neq i}^{N} \rho_j F_{ij}^d(k)$$

$$F_{ij}^d(k) = g(k)\frac{m_{Pi}(k)m_{Aj}(k)}{\| \mathbf{X}_i(k) - \mathbf{X}_j(k) \| + \varepsilon}[x_j^d(k) - x_i^d(k)]$$

$$v_i^d(k+1) = \rho_i v_i^d(k) + a_i^d(k)$$

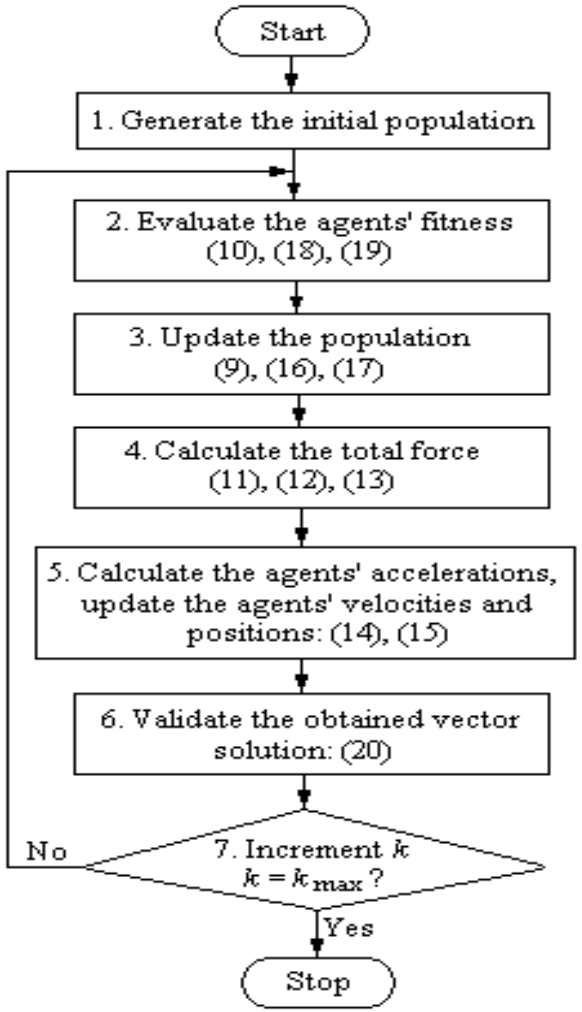
$$x_i^d(k+1) = x_i^d(k) + v_i^d(k+1)$$

New validation of obtained solution:

$$| y(t_f) - r(t_f) | \leq \varepsilon_y | r(t_f) - r(t_0) |$$

$\varepsilon_y = 0.001$ for a 2% settling time

Practically t_f – finite value (transients)



Tuning procedure

Step I: Derive the discrete-time sensitivity models of the fuzzy control system with respect a certain process parameter and express the output sensitivity function

Step II: Set the weighting parameters in the objective functions to meet the performance specifications imposed to the fuzzy control systems

Step III: Implement the steps of the GSA to solve the optimization problems that lead to the optimal controller tuning parameter vector $\boldsymbol{\rho}^*$

Example / 1

Position control of a nonlinear DC motor-based servo system (model: after six slides)

ESO method – tuning in the linear case:

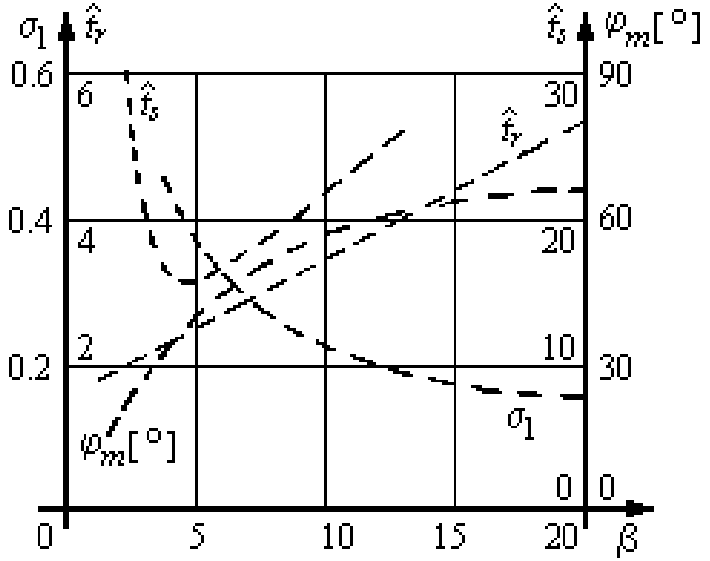
$$C(s) = k_c(1 + sT_i)/s = k_C[1 + 1/(sT_i)]$$

$$k_C = k_c T_i$$

$$k_c = 1/(\beta\sqrt{\beta}T_\Sigma^2 k_P),\ T_i = \beta T_\Sigma$$

$$F(s) = 1/(1 + T_i s)$$

One design parameter: β

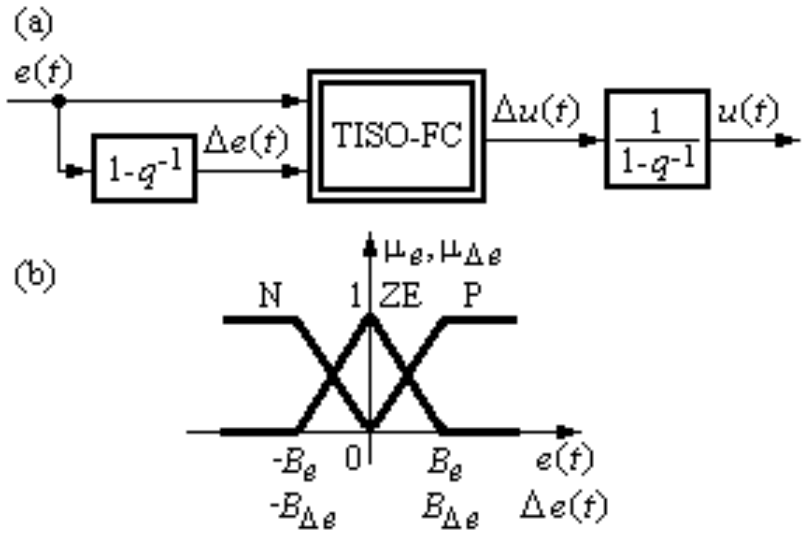


Example / 2

Structure of Takagi-Sugeno PI-FC:

$\Delta e(t)$	$e(t)$		
	N	ZE	P
P	$\Delta u(t) = f_1(t)$	$\Delta u(t) = f_1(t)$	$\Delta u(t) = f_2(t)$
ZE	$\Delta u(t) = f_1(t)$	$\Delta u(t) = f_1(t)$	$\Delta u(t) = f_1(t)$
N	$\Delta u(t) = f_2(t)$	$\Delta u(t) = f_1(t)$	$\Delta u(t) = f_1(t)$

$$f_1(t) = K_P[\Delta e(t) + \mu\, e(t)],\ f_2(t) = \eta\, f_1(t)$$



SUM and PROD operators,
weighted average method for defuzzification

Tustin's method $\Rightarrow$ digital PI controller parameters:

$$K_P = k_c\,(T_i - T_s/2),\ \mu = 2T_s/(2T_i - T_s)$$

Example / 3

Modal equivalence principle $\Rightarrow$ tuning condition:

$$B_{\Delta e} = \mu\, B_e$$

Parameter vectors of controller (i.e., vector variable of objective functions) and process:

$$\boldsymbol{\rho} = [\rho_1 = B_e \quad \rho_2 = \beta \quad \rho_3 = \eta]^T \in R^3$$

$$\boldsymbol{\alpha} = [\alpha_1 = k_P \quad \alpha_2 = T_\Sigma]^T \in R^2$$

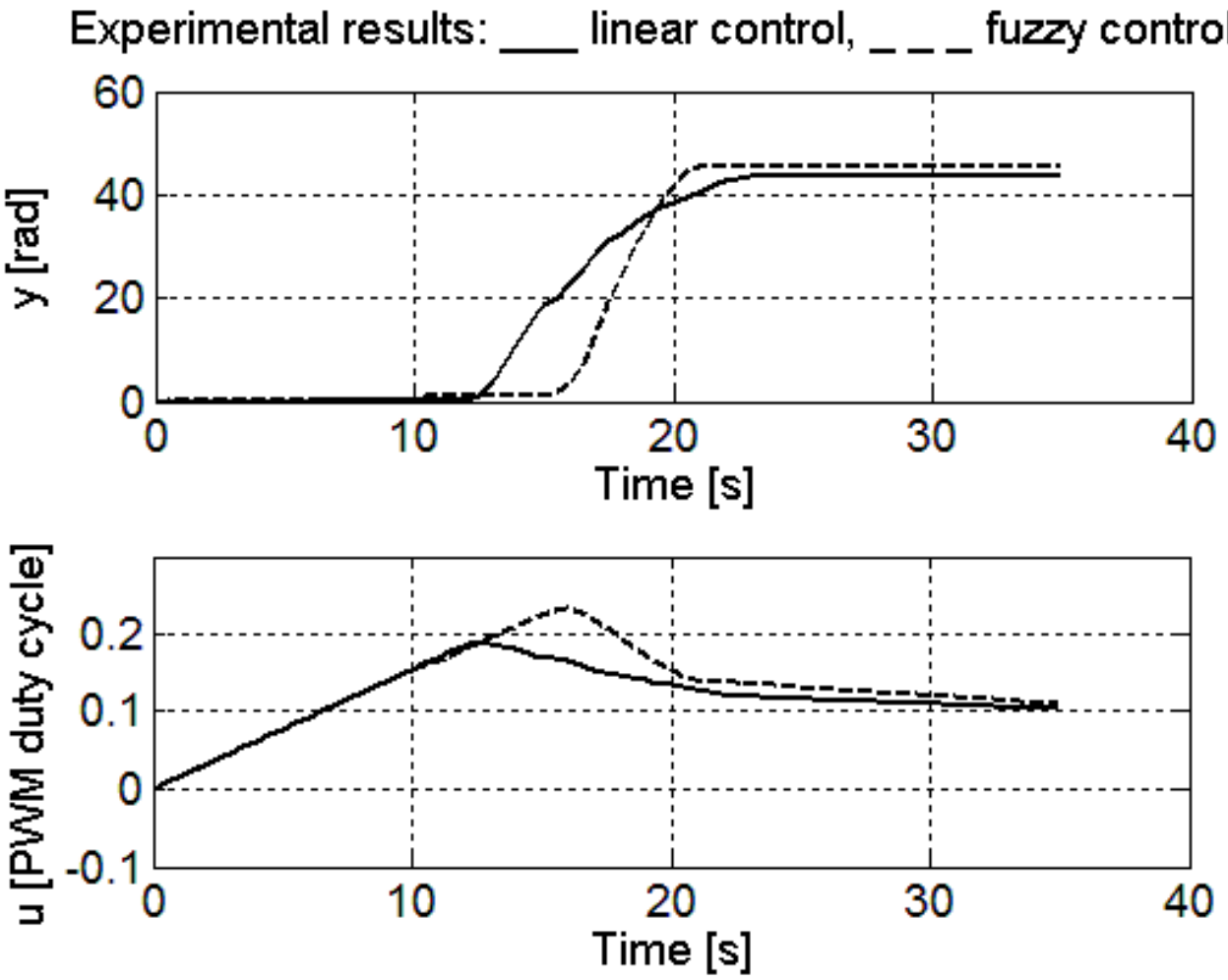
Weighting parameters of objective functions (for a reduced process gain sensitivity):

$$(\gamma^{k_P})^2 \in \{0,1000,10000,100000\}$$

Parameters of GSA: $N \in \{10,20,50\}, \quad k_{\max} \in \{75,100,150\}, \quad \delta = 0.5$

Search domain: $22.5 \le B_e \le 40, \; 0.55 \le \eta \le 1, \; 4 \le \beta \le 16$

Results / 1


Experimental results: ___ linear control, _ _ _ fuzzy control
y [rad]
60
40
20
0
0
10
20
30
40
Time [s]
u [PWM duty cycle]
0.2
0.1
0
-0.1
0
10
20
30
40
Time [s]

Results / 2

$N = 50 \quad k_{\max} = 100$

$(\gamma^{k_P})^2$	B_e^*	β^*	η^*
0	34.95422	7.333734	0.270214
1000	32.14256	7.301204	0.252184
10000	31.11246	6.806004	0.26495
100000	34.15234	6.680212	0.299149
$(\gamma^{k_P})^2$	k_C^*	T_i^*	$I_{ISE}^{k_P}$
0	0.002897	6.747032	9118.426
1000	0.002904	6.717108	9753.51
10000	0.003021	6.261524	17682.04
100000	0.003062	6.145796	97726.66
$(\gamma^{k_P})^2$	$Avg(I_{ISE}^{k_P})$	$StDev(I_{ISE}^{k_P})$	$StDev(I_{ISE}^{k_P})\%$
0	9118.426	306.2352	3.358422
1000	9753.51	136.4733	1.399222
10000	17682.04	624.9955	3.534634
100000	97726.66	2274.794	2.327711

Outline

- The Process Control research group of the Politehnica University of Timisoara
- Nature-inspired optimization algorithms (NIOAs)
- NIOAs in the optimal tuning of fuzzy controllers
- **NIOAs in the optimal tuning of fuzzy models**
- NIOAs in optimal path planning algorithms for mobile robots

Servo system models / 1

Nonlinear state-space model – for speed control:

$$m(t) = \begin{cases} 0, & \text{if } |u(t)| \le u_a \\ k_{u,m}[u(t) - u_a \operatorname{sgn}(u(t)], & \text{if } u_a < |u(t)| < u_b \\ k_{u,m}(u_b - u_a)\operatorname{sgn}(u(t)), & \text{if } |u(t)| \ge u_b \end{cases}$$

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -1/T_\Sigma \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ k_{P1}/T_\Sigma \end{bmatrix} m(t)$$

$$y(t) = [0 \quad 1]\,\mathbf{x}(t)$$

$$\mathbf{x}(t) = [\alpha(t) \quad \omega(t)]^T$$

Servo system models / 2

Linear continuous-time state space models – local models at n_R operating points:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_i\ \mathbf{x}(t) + \mathbf{B}_i\ u(t)$$
$$y(t) = \mathbf{C}_i\ \mathbf{x}(t)$$

$$\mathbf{A}_i = \begin{bmatrix} 0 & 1 \\ 0 & -1/(T_\Sigma)_i \end{bmatrix},\ \mathbf{B}_i = \begin{bmatrix} 0 \\ (k_P)_i/(T_\Sigma)_i \end{bmatrix},\ \mathbf{C}_i = [0 \quad 1],\ i = 1 \ldots n_R$$

$$P(s) = \frac{(k_P)_i}{1 + (T_\Sigma)_i s}$$

Optimization problem

$$\boldsymbol{\rho}^* = \arg\min_{\boldsymbol{\rho}\in D} J(\boldsymbol{\rho})$$

$$J(\boldsymbol{\rho}) = \frac{1}{t_f}\int_0^{t_f}[y(t,\boldsymbol{\rho}) - y_m(t,\boldsymbol{\rho})]^2\,dt = \frac{1}{t_f}\int_0^{t_f}[e_m(t,\boldsymbol{\rho})]^2\,dt$$

$\boldsymbol{\rho}$ – a part of the parameters of the input m.f.s of the TSK fuzzy models, i.e., the limits of the supports of the fuzzy sets which correspond to the input linguistic terms (i.e., their feet)

Mapping of SA algorithms onto the optimization problem:

$$J(\boldsymbol{\rho}) = C(\boldsymbol{\psi}),\ J(\boldsymbol{\rho}) = C(\varphi),\ \boldsymbol{\rho} = \boldsymbol{\psi},\ \boldsymbol{\rho} = \varphi$$

$\boldsymbol{\varphi}$ is the random initial solution vector, $\boldsymbol{\psi}$ is the next probable solution vector

SA-based approach to optimal tuning of T-S fuzzy models / 1

Step 1. Set the T-S fuzzy model structure, i.e., the number of operating points of the process = the number of input linguistic terms = the number of rules n_R such that to cover the range of $\omega(t)$. Define the input m.f.s such that their kernels match the n_R operating points and their supports avoid the problem of multiple overlapping m.f.s. Derive the n_R linear state-space models by least-squares identification

Step 2. Apply the modal equivalence principle $\Rightarrow$ the initial T-S fuzzy model

$$R^i : \text{IF } \omega(t) \text{ IS } LT_i \text{ THEN } \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}_i\, \mathbf{x}(t) + \mathbf{B}_i\, u(t) \\ y(t) = \mathbf{C}_i\, \mathbf{x}(t) \end{cases}, \; i = 1 \ldots n_R$$

Step 3. Set the values of the parameters in the SA algorithm: maximum number of iterations, maximum success rate, maximum rejection rate, initial success rate, initial rejection rate, minimum temperature, initial temperature

Step 4. Generate a random initial solution φ and evaluate its fitness value $C(\varphi)$ by digital simulation or by experiments

Step 5. Generate a probable neighboring solution ψ by disturbing φ, and evaluate the fitness value $C(\psi)$

Step 6. Calculate the fitness difference

$$\Delta C_{\psi\varphi} = C(\psi) - C(\varphi)$$

If $\Delta C_{\psi\varphi} \le 0$, accept $\varphi = \psi$ as the new vector solution

Otherwise, set the random parameter q_μ, $0 \le q_\mu \le 1$, and compute the probability of ψ to be the next solution, $p_\psi = \exp(-\Delta C_{\psi\varphi} / \theta_\mu)$

If $p_\psi > q_\mu$, $\varphi = \psi$ is the new solution. Otherwise, continue with the next step

Step 7. If the new solution is accepted, update the new solution and C, increment s_r and set $r_r = 0$

Otherwise, increment r_r. If r_r has reached $r_{r\max}$, the algorithm is stopped; otherwise, continue with the next step

Step 8. Increment s_r. If s_r has reached $s_{r\max}$, go to the next step.

Otherwise increment μ. If μ has reached $\mu_{\max}$, go to the next step; otherwise, go to step 4

Step 9. Reduce the temperature according to the temperature decrement rule $\theta_{\mu+1} = \alpha_{cs}\theta_\mu$, $\alpha_{cs} = \text{const}$, $\alpha_{cs} < 1$, $\alpha_{cs} \approx 1$

Step 10. If $\theta_\mu > \theta_{\min}$, go to step 5

Otherwise, the algorithm is stopped indicating that it has reached the solution φ

Results / 1

Input m.f.s:

$$\mu_{LT_i}(\omega) = \begin{cases} 0, & \text{if } \omega < a_i, \\ 1 + \dfrac{\omega - b_i}{b_i - a_i}, & \text{if } a_i \le \omega < b_i, \\ 1 - \dfrac{\omega - b_i}{c_i - b_i}, & \text{if } b_i \le \omega < c_i, \\ 0, & \text{if } \omega \ge c_i, \end{cases} \qquad i = 1 \ldots 14$$

Parameter vector (feet of triangular membership functions):

$$\boldsymbol{\rho} = [a_1 \quad c_1 \quad a_2 \quad c_2 \quad a_3 \quad c_3 \quad a_4 \quad c_4 \quad a_5 \quad c_5$$
$$a_6 \quad c_6 \quad a_7 \quad c_7 \quad a_8 \quad c_8 \quad a_9 \quad c_9 \quad a_{10} \quad c_{10}$$
$$a_{11} \quad c_{11} \quad a_{12} \quad c_{12} \quad a_{13} \quad c_{13} \quad a_{14} \quad c_{14}]^T$$

Results / 2

Matrices in rule consequents:

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 \\ 0 & -0.91524 \end{bmatrix}, \mathbf{B}_1 = \begin{bmatrix} 0 \\ 116.1905 \end{bmatrix}, \mathbf{C}_1 = [0 \quad 1]$$

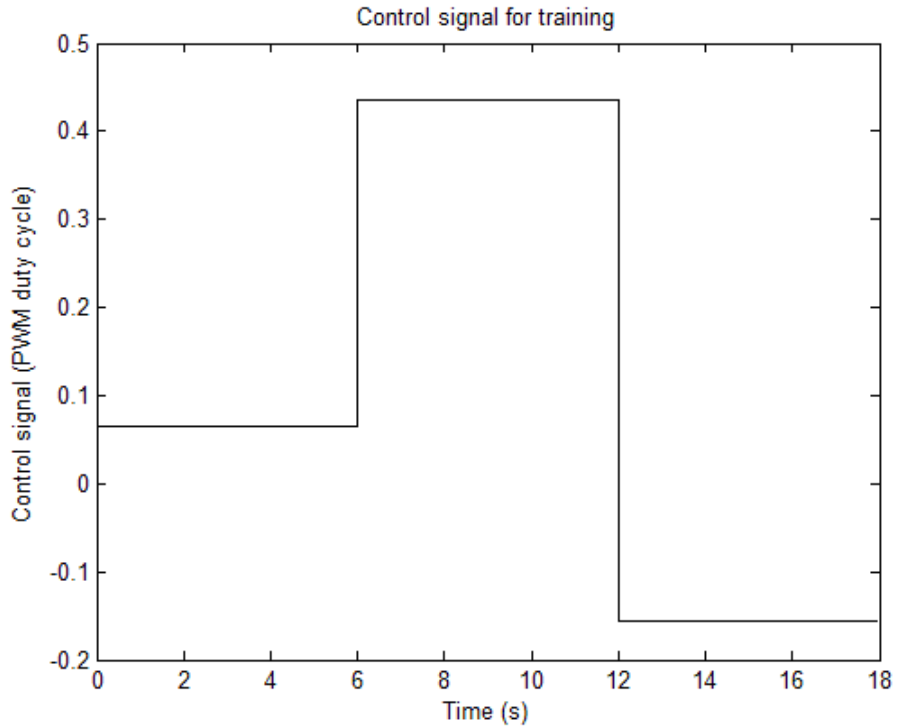
$$\mathbf{A}_{14} = \begin{bmatrix} 0 & 1 \\ 0 & -1.6949 \end{bmatrix}, \mathbf{B}_{14} = \begin{bmatrix} 0 \\ 201.5932 \end{bmatrix}, \mathbf{C}_{14} = [0 \quad 1]$$

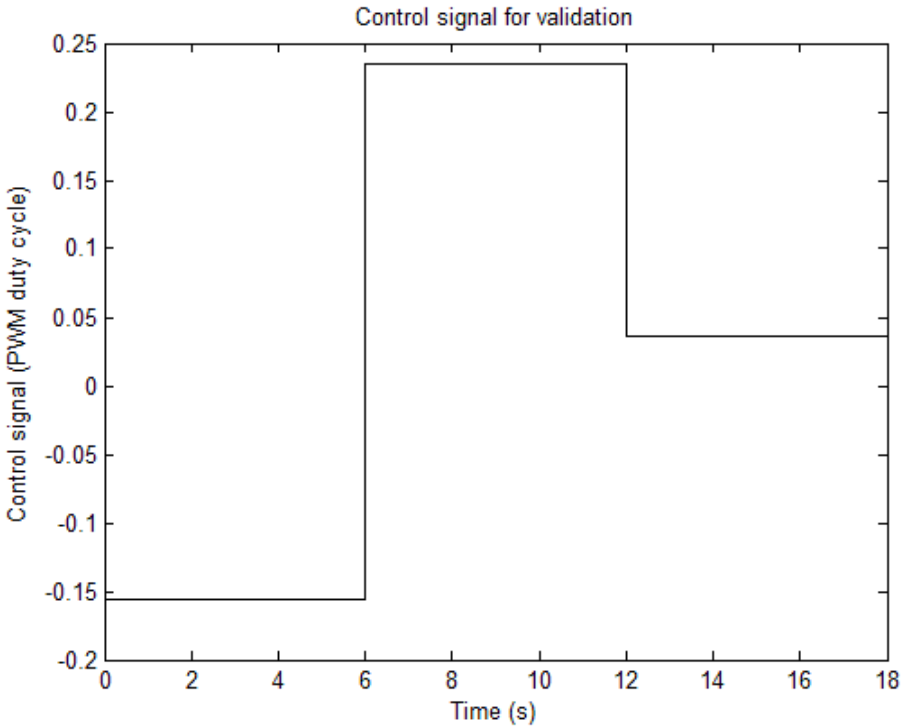
Initial parameter vector:

$$\boldsymbol{\rho} = [-130 \quad -83 \quad -99 \quad -66 \quad -83 \quad -51$$
$$-66 \quad -36 \quad -51 \quad -21 \quad -36 \quad -7.6$$
$$-21 \quad 7.6 \quad -7.6 \quad 21 \quad 7.6 \quad 36 \quad 21 \quad 51$$
$$36 \quad 66 \quad 51 \quad 83 \quad 66 \quad 99 \quad 83 \quad 115]^T$$

Results / 3

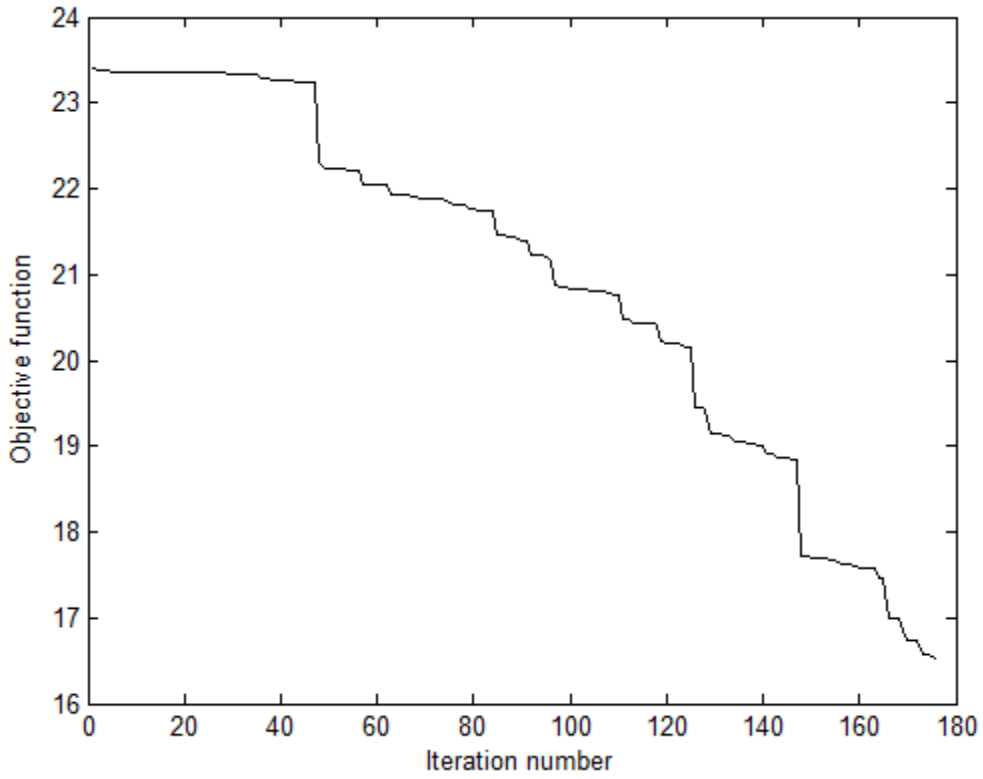
Test signals:





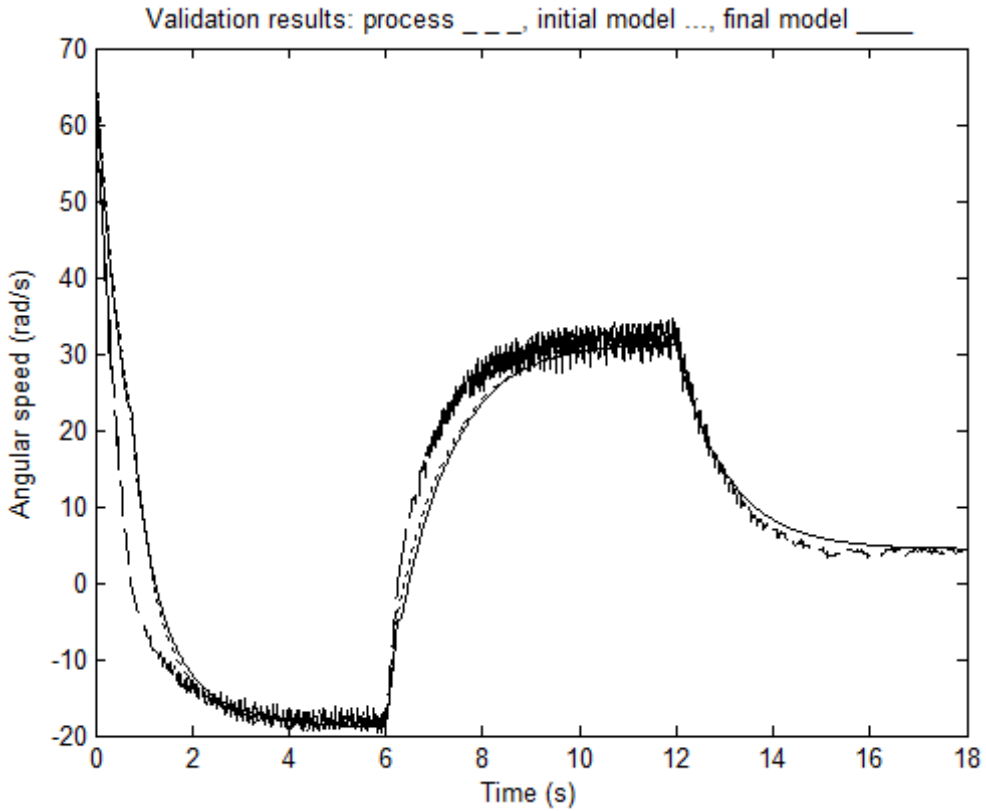
Results / 4

Objective function versus iteration number:



Results / 5

Experimental results for validation data:



Outline

- The Process Control research group of the Politehnica University of Timisoara
- Nature-inspired optimization algorithms (NIOAs)
- NIOAs in the optimal tuning of fuzzy controllers
- NIOAs in the optimal tuning of fuzzy models
- **NIOAs in optimal path planning algorithms for mobile robots**

Optimization problem / 1

- *R* – a mobile robot in a static environment (obstacles and danger zones)
- *P* – a population of *N* particles (agents)
- $\{O_k \mid k = 1...p\}$: set of obstacles
- $\{DZ_k \mid k = 1...r\}$: set of danger zones
- Generate a trajectory for the robot:

$$T_{i,t} = \{(X_i(0), Y_i(0)), (X_i(1), Y_i(1)), ..., (X_i(t), Y_i(t))\},\ i = 1...N$$

Optimization problem / 2

Two objectives:

- Maximize the distance between the trajectory and the danger zones:

$$f1_i(t) = 1 / \sum_{j=1, j\neq i}^{r} \sqrt{(X_{DZj} - X_i(t))^2 + (Y_{DZj} - Y_i(t))^2}, \; i = 1...N$$

- Minimize the distance traveled by the robot:

$$f2_i(t) = \sqrt{(X_i(t) - X_f)^2 + (Y_i(t) - Y_f)^2}, i = 1...N$$

- Objective function (multi-objective):

$$E_i(t) = f2_i(t) + \lambda_1 \cdot f1_i(t), i = 1...N$$

- Optimization problems (give the indices of best agents): $\hat{i}(t) = \arg\min_{i \in S_{cf,PRk}} E_i(t)$

$S_{cf,PRk} \subset \{1,2,...,N\}$ – set of agents that lead to collision-free paths for population P_{RK}

$T_{\hat{i}(t_{\max}),t_{\max}}$ – optimal paths

The idea (mapping the optimization problem onto the algorithms)

Algorithms (NIOAs)	**Mobile robots**
Search space	Solution space
Agent (particle)	Mobile robot
Population of agents	Set of robots
Fitness function	Objective function

Hybrid PSO-GSA

Aim: hybridization between GSA and PSO in order to combine the local search ability of GSA with the social thinking ability of PSO algorithm

Combine the PSO and GSA velocity update equations (weighted sums):

$$v_i^X(t+1) = w(t)\, v_i^X(t) + c_1\, r_1\, [P_i^X(t) - X_i^X(t)] + c_2\, r_2\, [P_g^X(t) - X_i^X(t)] + c_3 r_3 a_i^X,\ i = 1 \ldots N$$

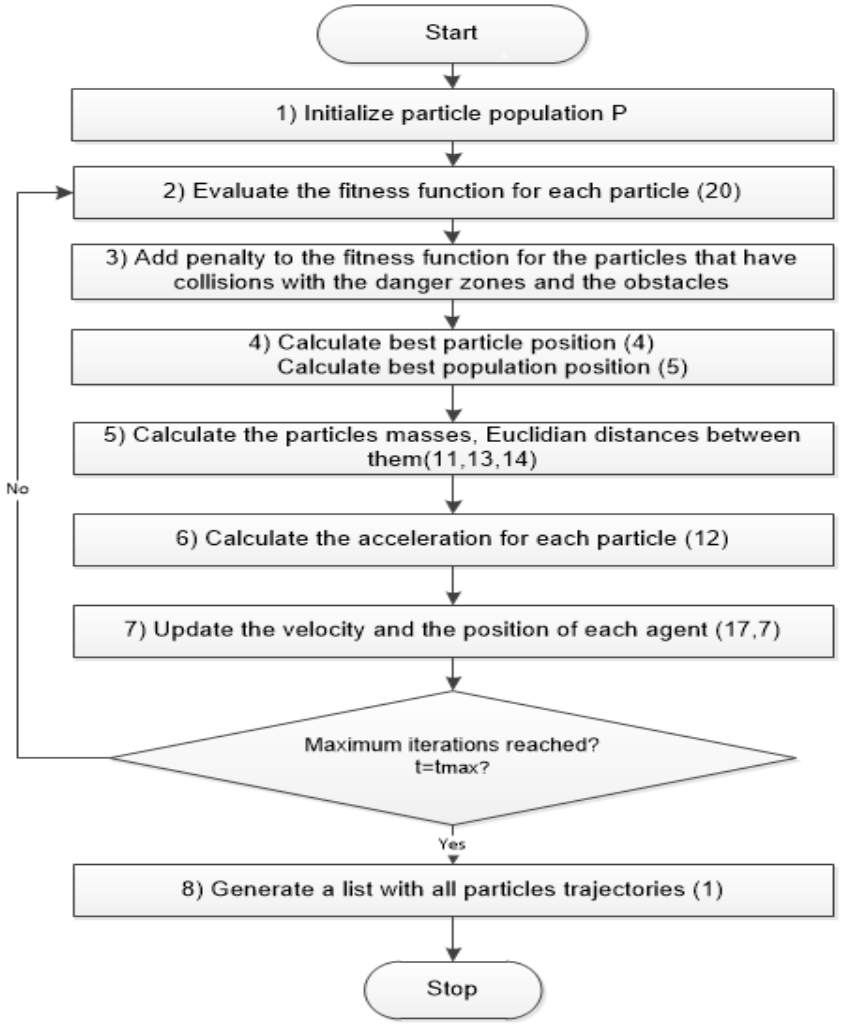
$$v_i^Y(t+1) = w(t)\, v_i^Y(t) + c_1\, r_1\, [P_i^Y(t) - X_i^Y(t)] + c_2\, r_2\, [P_g^Y(t) - X_i^Y(t)] + c_3 r_3 a_i^Y,\ i = 1 \ldots N$$

Keep the position update equations:

$$X_i(t+1) = X_i(t) + v_i^X(t+1)$$

$$Y_i(t+1) = Y_i(t) + v_i^Y(t+1)$$

Navigation algorithm



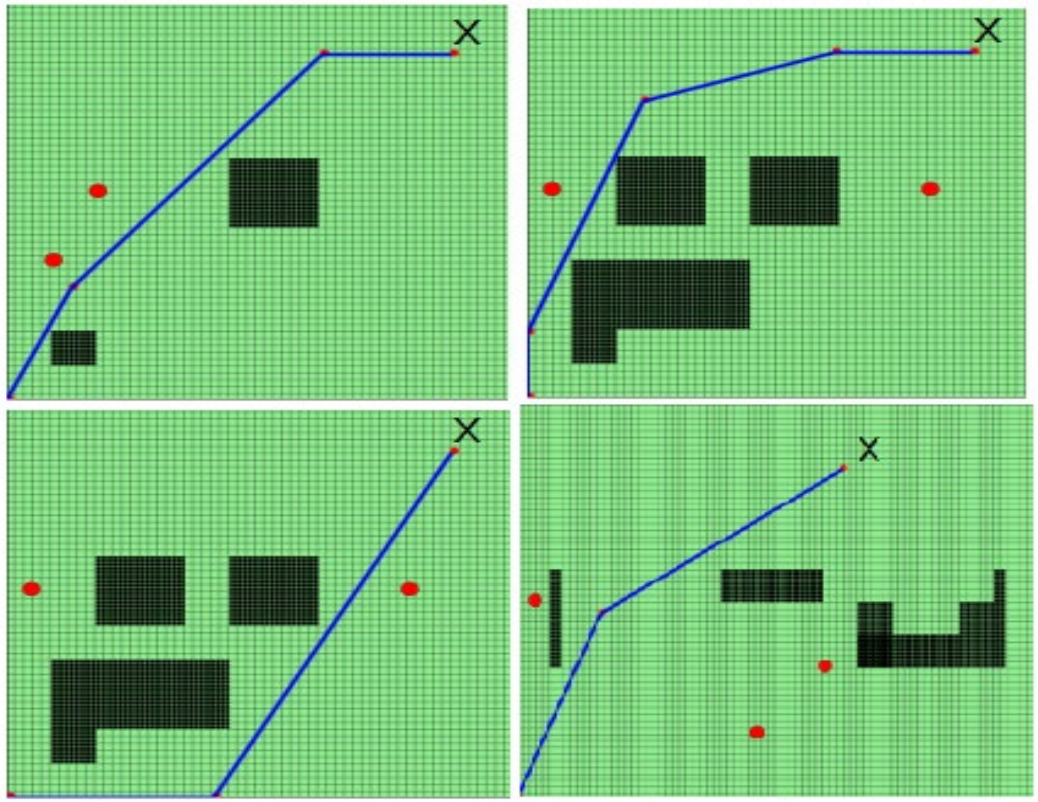
- Add penalty for the trajectories with collision points
- Add penalty for the trajectories that are very close to a danger zone
- Calculate the shortest collision-free trajectory for the robot

Results / 1

- Several experiments to test the navigation algorithm
- New mission added to the nRobotic platform
- Multiple obstacles, multiple danger zones
- Scope: avoid obstacles while arriving the target point on a shorter path and at a safe distance from the danger zones

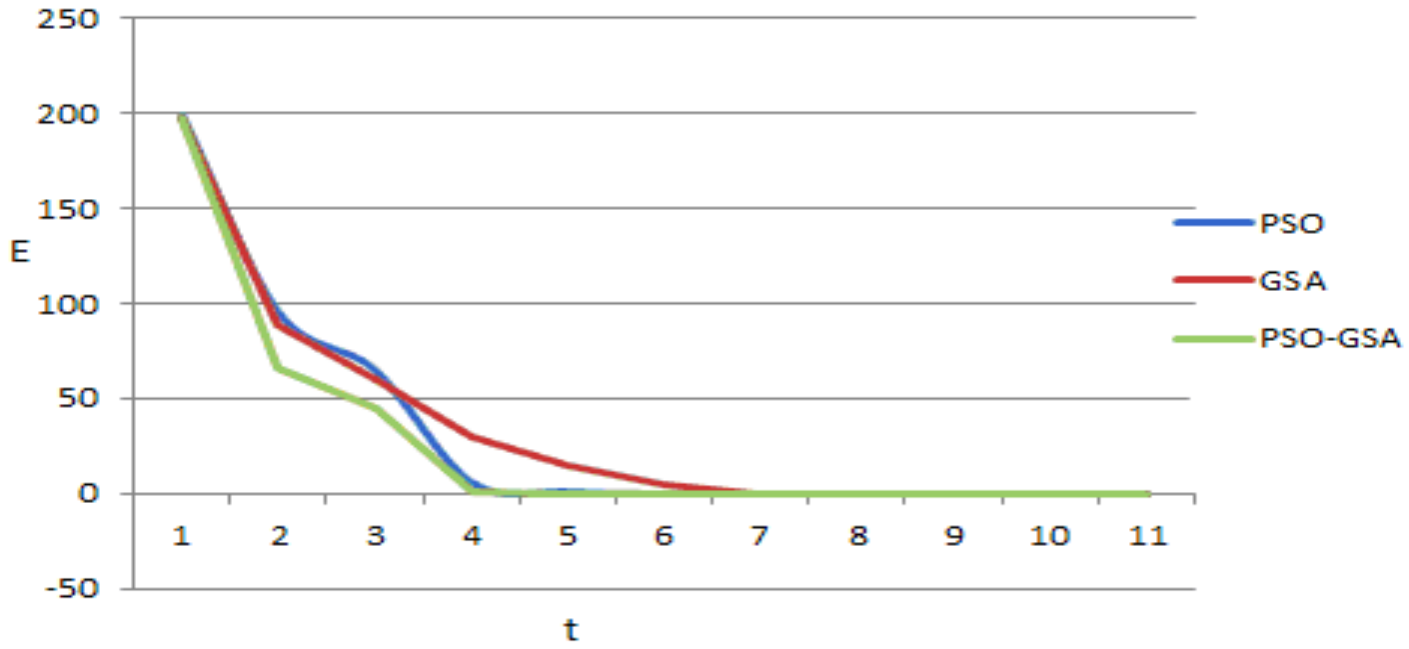
Results / 2

Several trajectories obtained by
the hybrid PSO-GSA path planning algorithm:

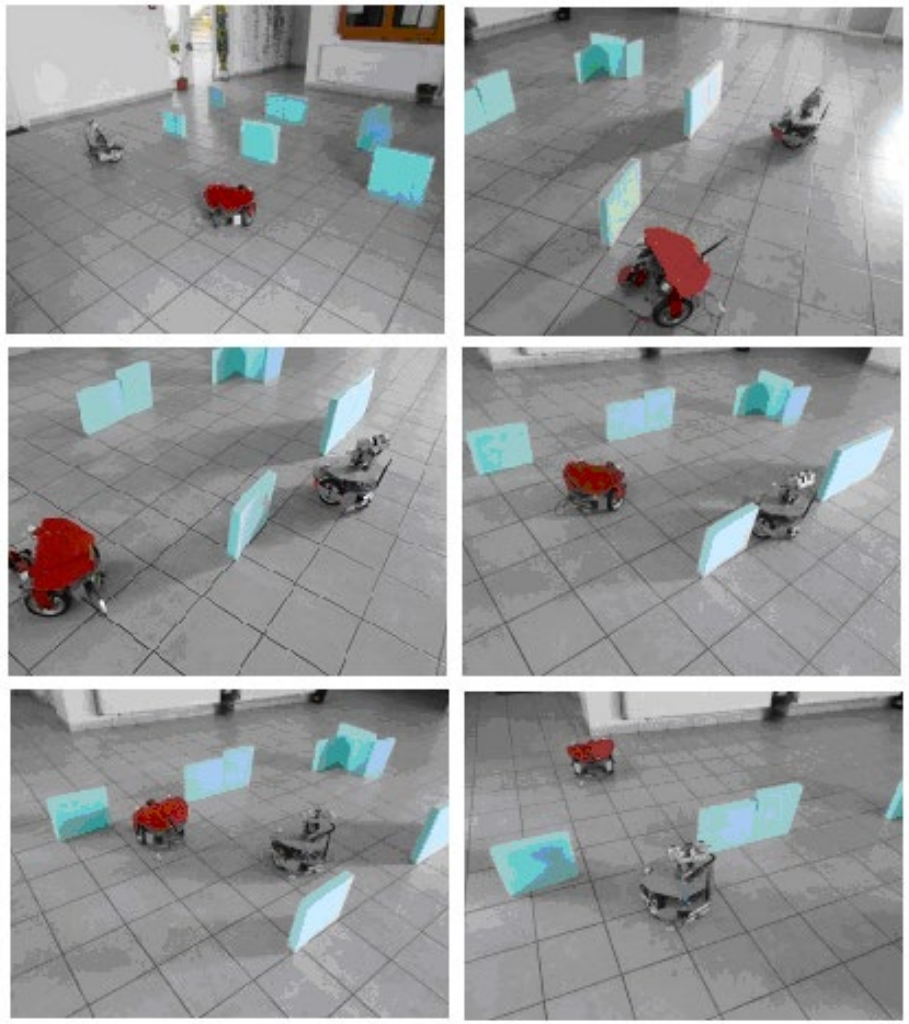


Results / 3

Objective function obtained by PSO, GSA, PSO-GSA algorithms for the best agents versus the iteration number:



Results / 4



- Experiments
- New mission added to the platform
- Known environment
- Multiple robots

Conclusions

- Advantage: performance improvement offered by metaheuristic algorithms including NIOAs for complicated optimization problems where analytical solutions cannot be found
- Shortcoming: there are not yet methods to know the best parameters of NIOAs to solve problems that can be set at the beginning when using the algorithms & sensitivity
- Shortcoming: large number of evaluations of the objective functions
- Solutions: use fuzzy logic to adapt parameters and achieve better results versus the initial algorithms + including gradient information used in classical algorithms (the gradient estimation is needed)

Any further questions?

Contact me:
radu.precup@upt.ro