Convergence Stabilization Modeling operating in Online mode

Estimating Stop Conditions of Swarm Based Stochastic Metaheuristic Algorithms

Peter Frank Perroni[1]
Daniel Weingaertner[1]
Myriam Regattieri Delgado[2]

[1]Federal University of Paraná

[2]Federal University of Technology - Paraná

GECCO - Jul 18, 2017

Inclusão Digital Integrada:
Tecnologias para as Cidades Digitais

When optimizing a problem:

1. How many evaluations?

When optimizing a problem:

1. How many evaluations?
   - Roughly $30 \times D$

When optimizing a problem:

1. How many evaluations?
   - Roughly $30 \times D$.. what if I change one parameter and worsen?
   - Roughly $50 \times D$

When optimizing a problem:

1. How many evaluations?
   - Roughly $30 \times D$.. what if I change one parameter and worsen?
   - Roughly $50 \times D$... what if improved?
   - "Roughly" is not a good measure.

When optimizing a problem:

1. How many evaluations?
   - Roughly $30 \times D$.. what if I change one parameter and worsen?
   - Roughly $50 \times D$... what if improved?
   - "Roughly" is not a good measure.
2. We can assume that:
   - Fitness evaluation frequently is more costly than take intermediate step to use it efficiently.
   - The solution representation should be derived as directly from the problem as possible.
   - Increasing the problem representation exponentially increase the optimization complexity.

Class of Problems
Convergence Modeling

Common Methods:

1. Mathematical: use asymptotic approximations to prove convergence and “roughly” the number of evaluations required.

Class of Problems
Convergence Modeling

Common Methods:

1. Mathematical: use asymptotic approximations to prove convergence and "roughly" the number of evaluations required.
2. Model Convergence: focus on population instead of aiming the global best.

Class of Problems
Convergence Modeling

Common Methods:

1. Mathematical: use asymptotic approximations to prove convergence and "roughly" the number of evaluations required.
2. Model Convergence: focus on population instead of aiming the global best.
3. Model-based optimization: simplify the problem through surrogate model or distribution.

Class of Problems
Convergence Modeling

Common Methods:

1. Mathematical: use asymptotic approximations to prove convergence and "roughly" the number of evaluations required.
2. Model Convergence: focus on population instead of aiming the global best.
3. Model-based optimization: simplify the problem through surrogate model or distribution.
4. Computing Budget Allocation: assign larger budget for more promising solutions estimated through sampling.

Class of Problems
Convergence Modeling

Problems with these methods:

1. Mathematical: use asymptotic approximations to prove convergence.
   → **Specific for the method and of restricted practical usage.**
2. Model Convergence: focus on population instead of aiming the global best.
   → **Slow convergence.**
3. Model-based optimization: simplify the problem through surrogate model or distribution.
   → **Specific for the method and for the problem.**
4. Computing Budget Allocation: assign larger budget for more promising solutions estimated through sampling.
   → **Specific for the method and waste too many evaluations.**

Wouldn't be interesting to have a method that tells us automatically a good moment to stop the optimization or to take some action to improve the convergence?

CSMOn

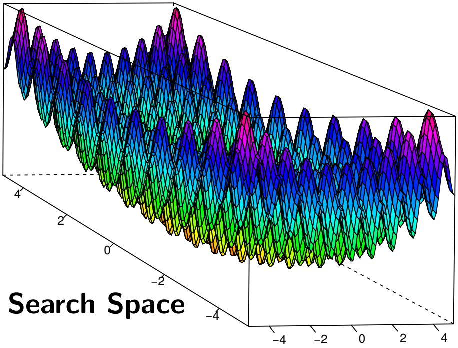
Convergence Stabilization Modeling operating in Online mode

The Cost/Benefit Trade-off Problem

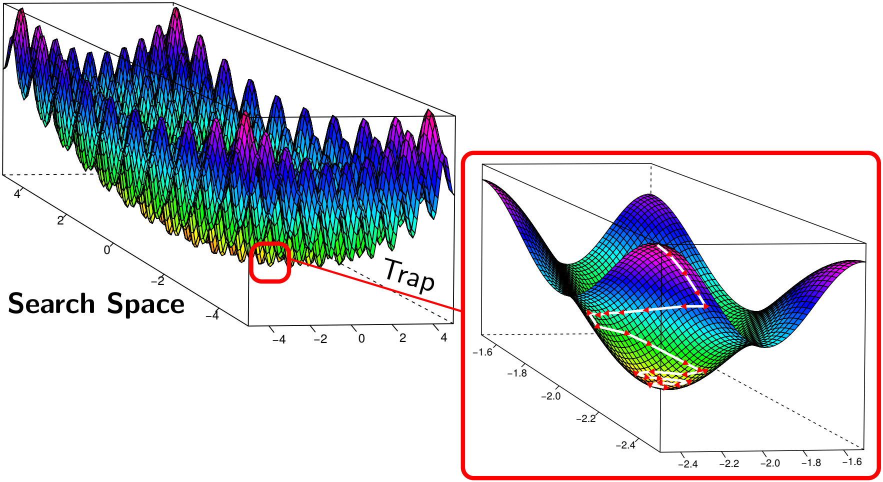
- Benefit is the saved computational effort (advantages)
- Cost is the performance loss (drawbacks)

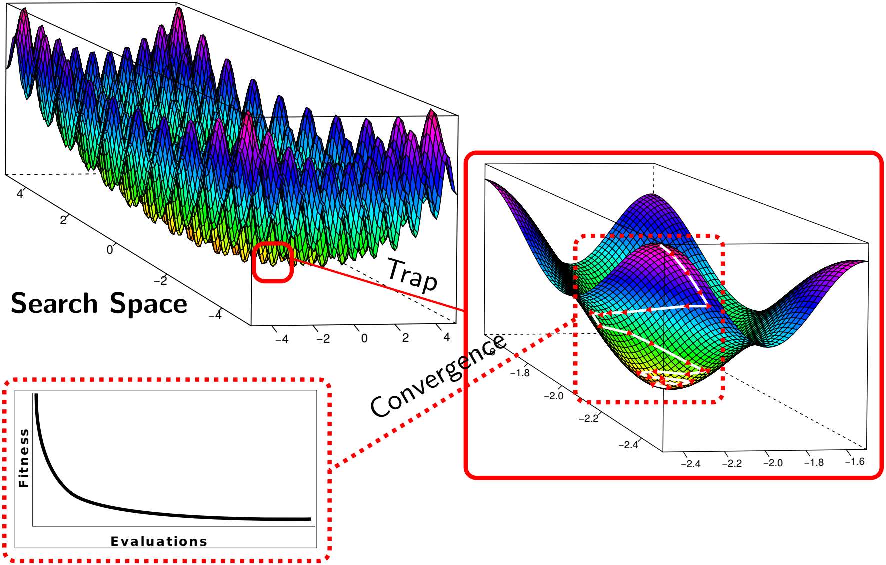
Objectives:

- Find automatically the advantages/drawbacks balancing to save fitness evaluations
- Focus on Local optimum
- Avoid changes to the original search algorithm (use it as-is)

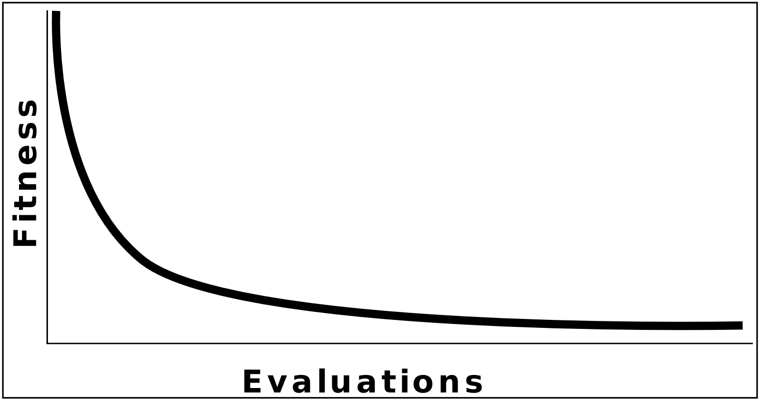


Search Space

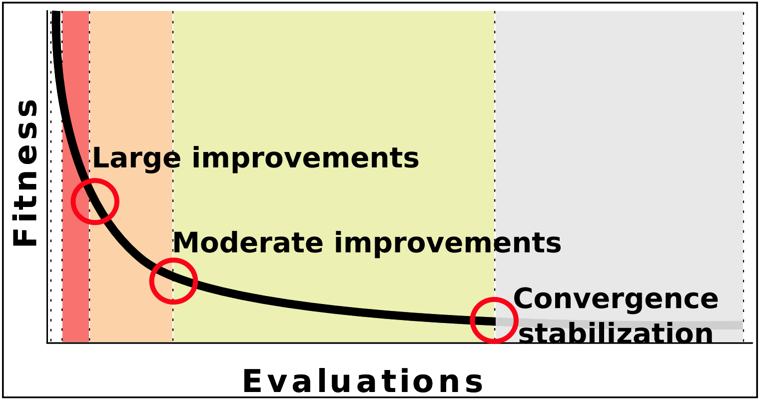

Search Space
Trap
4
2
0
−2
−4
−4
−2
0
2
4
−1.6
−1.8
−2.0
−2.2
−2.4
−2.4
−2.2
−2.0
−1.8
−1.6


Search Space
Trap
Convergence
Fitness
Evaluations

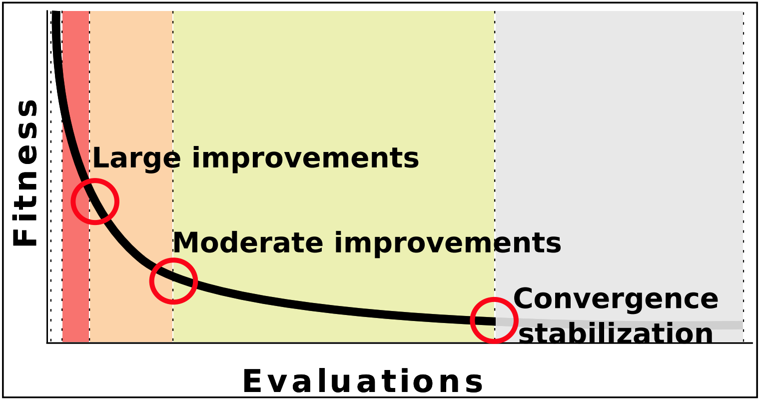
Convergence



Convergence Phases

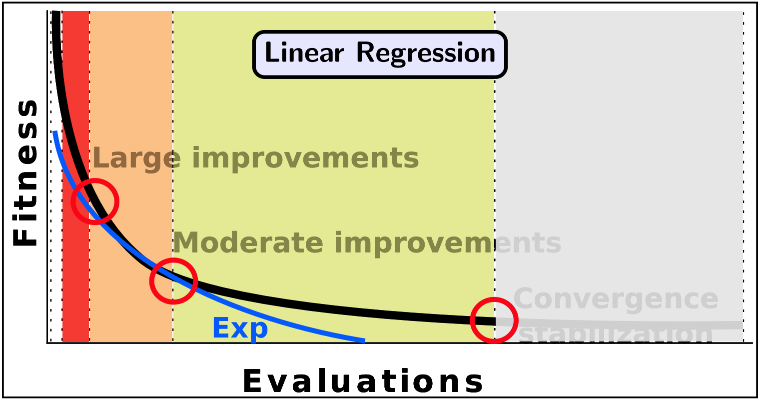


The Cost/Benefit Trade-off Problem

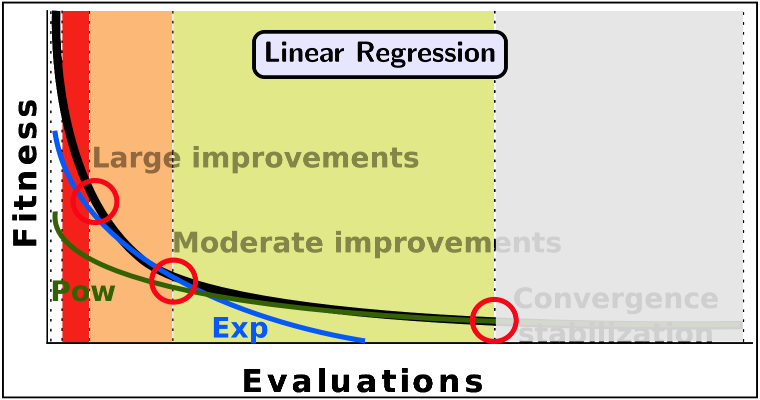


Evaluations are wasted when reaching convergence stabilization

Convergence Stabilization Modeling

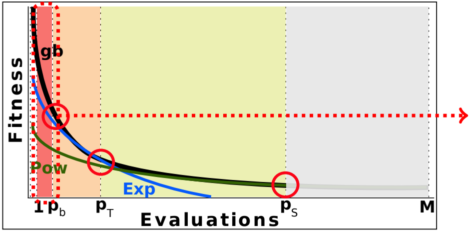


Convergence Stabilization Modeling



CSMON

Convergence Stabilization Modeling operating in Online mode



Convergence Decay triggers CSMOn

$$\lim_{s \to \infty} \frac{|y_s - L|}{|y_{s-1} - L|} \to 1$$

$$\lim_{s \to \infty} \frac{|y_s - y_{s-1}|}{|y_{s-1} - y_{s-2}|} \to 1$$

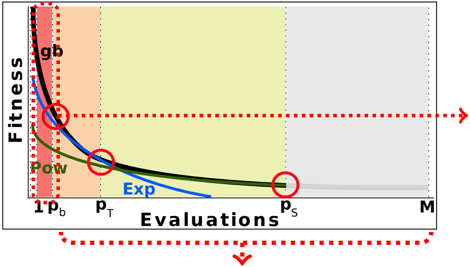
Where:

y_s is the current fitness

L is the last possible fitness

CSMON

Convergence Stabilization Modeling operating in Online mode



Convergence Decay triggers CSMOn

$$\lim_{s\to\infty} \frac{|y_s - L|}{|y_{s-1} - L|} \to 1$$

$$\lim_{s\to\infty} \frac{|y_s - y_{s-1}|}{|y_{s-1} - y_{s-2}|} \to 1$$

Where:

y_s is the current fitness

L is the last possible fitness

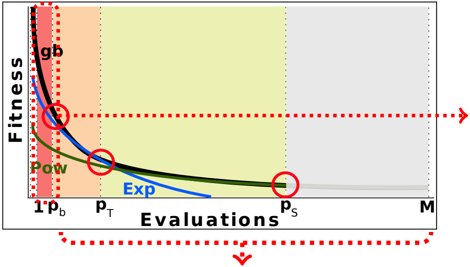
Online Modeling

$$\text{Decay is} \begin{cases} \mathbf{Exp(x)} = \alpha \mathbf{e}^{-\beta \mathbf{x}} & \text{if } nEvals \leq p_T \\ \mathbf{Pow(x)} = \alpha \mathbf{x}^{-\beta} & \text{otherwise} \end{cases}$$

$$x \in [p1, p2]$$
$$1 > p1 > p2 > M$$
$$M = \text{max.evals. per run}$$

CSMON

Convergence Stabilization Modeling operating in Online mode



Convergence Decay triggers CSMOn

$$\lim_{s \to \infty} \frac{|y_s - L|}{|y_{s-1} - L|} \to 1$$

$$\lim_{s \to \infty} \frac{|y_s - y_{s-1}|}{|y_{s-1} - y_{s-2}|} \to 1$$

Where:

y_s is the current fitness

L is the last possible fitness

Online Modeling

$$\text{Decay is} \begin{cases} \mathbf{Exp(x)} = \alpha \mathbf{e}^{-\beta \mathbf{x}} & \text{if } nEvals \leq p_T \\ \mathbf{Pow(x)} = \alpha \mathbf{x}^{-\beta} & \text{otherwise} \end{cases}$$

$$x \in [p1, p2]$$
$$1 > p1 > p2 > M$$
$$M = \text{max.evals. per run}$$

Limit sensibility is controlled by relaxation

$$R \in]0, 1[$$

CSMON

Convergence Stabilization Modeling operating in Online mode

We assume that the search algorithm:

1. Is swarm-based.
2. Presents some initial convergence.
3. Has a memory mechanism (strictly monotone global fitness improvement).

CSMOn

Input: $\{M, R\}$
$p_T \leftarrow -1, p_S \leftarrow -1$
$r \leftarrow 0.99$
$append(\mathbf{gb}, GetBest(1, M))$
repeat
    $r \leftarrow max(r^2, R)$ ← Relaxation
    if $p_S = -1$ **then**
        $p_T \leftarrow AdjustExp(\mathbf{gb}, M, r)$
    if $p_T > 0$ **then**
        $p_S \leftarrow AdjustLog(\mathbf{gb}, M, r, p_T)$
until $ne_{p_S} >= M$ **or** $(r = R$ **and** $p_S > 0)$

CSMOn

Input: $\{M, R\}$
$p_T \leftarrow -1, p_S \leftarrow -1$
$r \leftarrow 0.99$
$append(\mathbf{gb}, GetBest(1, M))$
repeat
  $r \leftarrow max(r^2, R)$
  if $p_S = -1$ **then**
    $p_T \leftarrow AdjustExp(\mathbf{gb}, M, r)$
  if $p_T > 0$ **then**
    $p_S \leftarrow AdjustLog(\mathbf{gb}, M, r, p_T)$
until $ne_{p_S} >= M$ **or** $(r = R$ **and** $p_S > 0)$

AdjustExp

Input: $\{\mathbf{gb}, M, r\}$
$s_{prev} \leftarrow s$
$append(\mathbf{gb}, GetBest(2, M))$
if $s - s_{prev} < 2$ **then return** -1
$p_b \leftarrow -1$
while $ne_s < M$ **do**
  if $\mathcal{D}_e(\mathbf{gb}) < r$ **and** $\mathcal{D}_l(\mathbf{gb}) < r$ **then**
    if $p_b = -1$ **then**
      $p_b \leftarrow s - 2$
      $\alpha_2 \leftarrow \alpha_e(\mathbf{gb}, p_b, s)$
    else
      $\alpha_1 \leftarrow \alpha_2$
      $\alpha_2 \leftarrow \alpha_e(\mathbf{gb}, p_b, s)$
      if $\alpha_2 < \alpha_1$ **then**
        return s
  else
    $p_b \leftarrow -1$
  $append(\mathbf{gb}, GetBest(1, M))$
return -1

AdjustLog

Input: $\{\mathbf{gb}, M, r, p_T\}$
$s_{prev} \leftarrow s$
$append(\mathbf{gb}, GetBest(3, M))$
if $s - s_{prev} < 3$ **then return** -1
$\alpha_1 \leftarrow \alpha_p(\mathbf{gb}, p_T, s - 1)$
$\alpha_2 \leftarrow \alpha_p(\mathbf{gb}, p_T, s)$
while $\alpha_2 \geq \alpha_1$ and $ne_s < M$ **do**
  if $\mathcal{D}_e(\mathbf{gb}) \geq r$ **or** $\mathcal{D}_l(\mathbf{gb}) \geq r$ **then**
    return -1
  $append(\mathbf{gb}, GetBest(1, M))$
  $\alpha_1 \leftarrow \alpha_2$
  $\alpha_2 \leftarrow \alpha_p(\mathbf{gb}, p_T, s)$
return s

CSMOn

Input: $\{M, R\}$
$p_T \leftarrow -1, p_S \leftarrow -1$
$r \leftarrow 0.99$
$append(\mathbf{gb}, GetBest(1, M))$
repeat
    $r \leftarrow max(r^2, R)$
    if $p_S = -1$ **then**
        $p_T \leftarrow AdjustExp(\mathbf{gb}, M, r)$
    if $p_T > 0$ **then**
        $p_S \leftarrow AdjustLog(\mathbf{gb}, M, r, p_T)$
until $ne_{p_S} >= M$ **or** $(r = R$ **and** $p_S > 0)$

AdjustExp

Input: $\{\mathbf{gb}, M, r\}$
$s_{prev} \leftarrow s$
$append(\mathbf{gb}, GetBest(2, M))$
if $s - s_{prev} < 2$ **then return** -1
$p_b \leftarrow -1$
while $ne_s < M$ **do**
    if $\mathcal{D}_e(\mathbf{gb}) < r$ **and** $\mathcal{D}_l(\mathbf{gb}) < r$ **then**
        if $p_b = -1$ **then**
            $p_b \leftarrow s - 2$
            $\alpha_2 \leftarrow \alpha_e(\mathbf{gb}, p_b, s)$
        else
            $\alpha_1 \leftarrow \alpha_2$
            $\alpha_2 \leftarrow \alpha_e(\mathbf{gb}, p_b, s)$
            if $\alpha_2 < \alpha_1$ **then**
                return s
    else
        $p_b \leftarrow -1$
    $append(\mathbf{gb}, GetBest(1, M))$
return -1

AdjustLog

Input: $\{\mathbf{gb}, M, r, p_T\}$
$s_{prev} \leftarrow s$
$append(\mathbf{gb}, GetBest(3, M))$
if $s - s_{prev} < 3$ **then return** -1
$\alpha_1 \leftarrow \alpha_p(\mathbf{gb}, p_T, s - 1)$
$\alpha_2 \leftarrow \alpha_p(\mathbf{gb}, p_T, s)$
while $\alpha_2 \geq \alpha_1$ and $ne_s < M$ **do**
    if $\mathcal{D}_e(\mathbf{gb}) \geq r$ **or** $\mathcal{D}_l(\mathbf{gb}) \geq r$ **then**
        return -1
    $append(\mathbf{gb}, GetBest(1, M))$
    $\alpha_1 \leftarrow \alpha_2$
    $\alpha_2 \leftarrow \alpha_p(\mathbf{gb}, p_T, s)$
return s

http://web.inf.ufpr.br/vri/software/csmon

https://gitlab.c3sl.ufpr.br/pfperroni/CSMOn

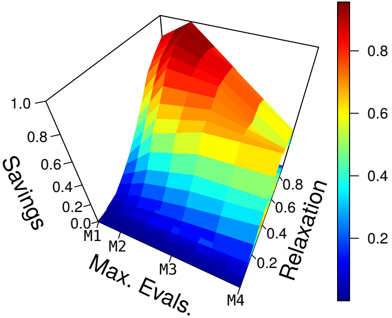
Experiments

- CEC13 competition benchmark functions:
  - 15 functions of 1000 dimensions.
  - Represent real-world problems.
  - Fully-separable Functions: F1, F2, F3.
  - Partially Additively Separable Functions: F4, F5, F6, F7, F8, F9, F10, F11.
  - Overlapping Functions: F12, F13, F14.
  - Non-separable Function: F15.

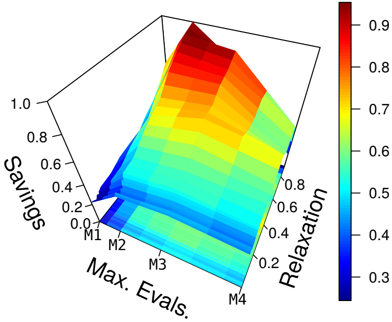
Cost / Benefit Trade-off Matrix for CEC13 Functions

15 functions averaged for CCPSO2-IP

$M1 = 1e6,\ M2 = 3e6,\ M3 = 6e6,\ M4 = 1e7$

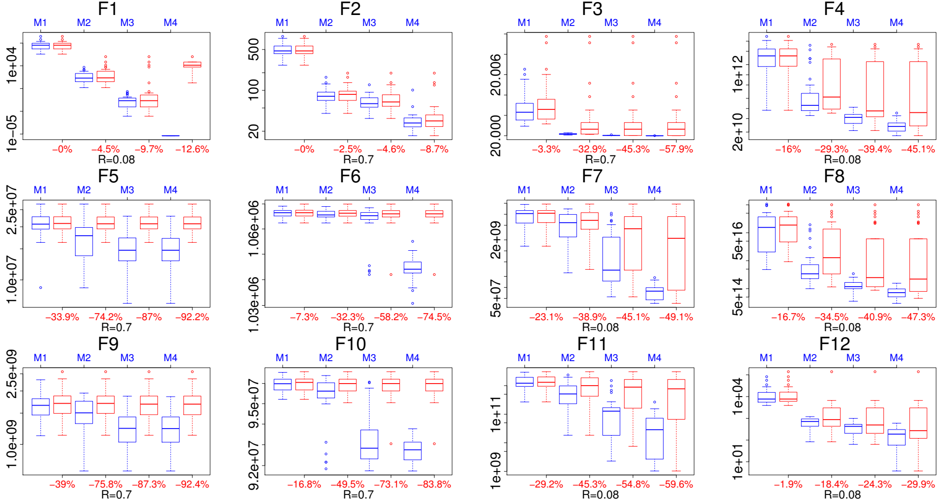


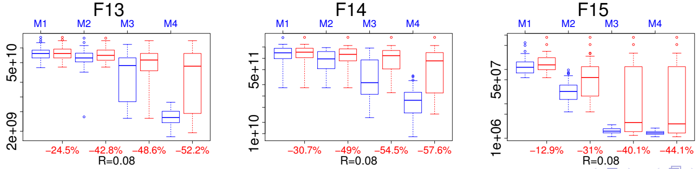
Fixed relaxation



Quadratic relaxation

CSMOn Economy on fitness function evaluation



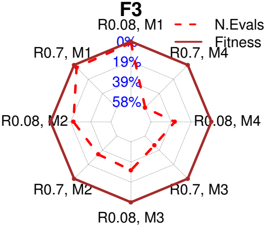


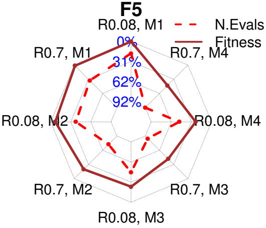
Quadratic $R = \{0.08, 0.7\}$

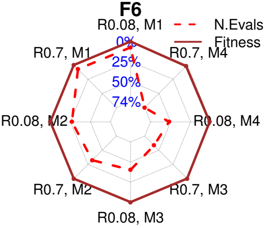
ne_{ps} = top axis − bottom axis

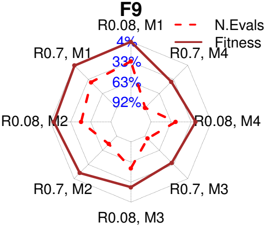
CCPSO2-IP search method

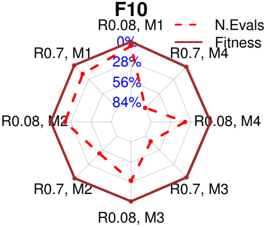
Best Averaged Results









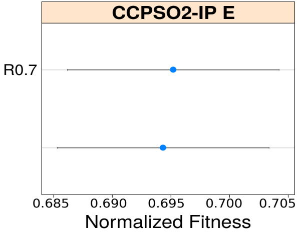


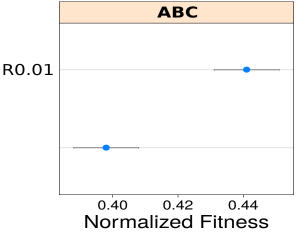
Percentage of Economy on Fitness Evaluations × Reduction on Fitness due to Quadratic Decreasing Relaxations

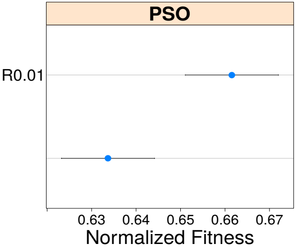
Best configurations are Fitness on borders and N.Evals on center

CCPSO2-IP search method

Paired comparison with and without CSMOn







MSG Landscape Generator

Fitness evaluations economy:
- CCPSO2-IP E: 6%
- ABC: 6.7%
- PSO: 27%

50 dimensions
$M = 5e4$ evaluations

Conclusions

Based on Results:

- CSMOn is able to effectively adapt to each optimization in progress (online)
- Best results are obtained with more stable convergences
- Fixed relaxation is prefered for erratic convergences
- CSMOn can indicate multistart points
- Most of configurations tested obtained saved evaluations
- Difference between advantages and drawbacks reached 70% on best case

Conclusions

Future Work:

- Consider long stagnation period of the search algorithm
- Test CSMOn with non-swarm memory-based metaheuristics
- Create new update mechanisms for the relaxation, like:
  - Consider the remaining budget in a multiple run scenario
  - React based on the distance to M

Thanks !

QUESTIONS