Automated Iterative Partitioning for Cooperatively Coevolving Particle Swarms in Large Scale Optimization

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

[1]Departamento de Informática
Universidade Federal do Paraná

[2]Pós-Graduação em Engenharia Elétrica e Informática Industrial
Universidade Tecnológica Federal do Paraná

Programa de Pós-graduação em Informática
UFPR

BRACIS, 2015

Summary

CCPSO2

- PSO variant developed to solve complex high scale optimization problems.
- Relative low cost and good performance when compared to counterparts.
- Grouping of swarms' dimensions is similar to the method used on Cooperative (multiswarm) PSO.

- Tackle high dimensionality by:
  - Permuting all n dimensions at every iteration t.
  - Randomly changing partition size s if no improvement is obtained.
  - Each swarm is assigned the same number of dimensions.

$$\begin{aligned}
&Given: \\
&\quad n = \text{number of dimensions} \\
&\quad S = \{s_1, s_2, ...\} \\
&\quad s \in S = \text{ Dimensions per swarm, randomly chosen}
\end{aligned}$$

$$\begin{aligned}
&Calculated :\rightarrow \mathbf{K} \times s = n \\
&\quad K = \text{number of Swarms}
\end{aligned}$$

- Convergence speed is controlled by using a *lbest* (local best) ring topology.
- Particle updates are performed by using:
  - Cauchy ($\mathcal{C}$) or Gaussian ($\mathcal{N}$) distributions.
  - Personal best, *lbest* and swarm's best to guide the direction.

$$x_{i,j}(t+1) = \begin{cases} y_{i,j}(t) + \mathcal{C}(1)|y_{i,j}(t) - \hat{y}'_{i,j}(t)|, & if \ rand \leq r \\ \hat{y}'_{i,j}(t) + \mathcal{N}(0,1)|y_{i,j}(t) - \hat{y}'_{i,j}(t)| & otherwise. \end{cases} \quad (1)$$

where:

$x_{i,j}$: Particle's dimension

$y_{i,j}$: Particle's personal best

$\hat{y}'_{i,j}$: Ring local best (*lbest*)

Algorithm 1 Pseudocode of CCPSO2

$\mathbf{b}(k, \mathbf{z}) = (P_1.\hat{\mathbf{y}}, \cdots, P_{k-1}.\hat{\mathbf{y}}, \mathbf{z}, P_{k+1}.\hat{\mathbf{y}}, \cdots P_K.\hat{\mathbf{y}})$
Create and initialize K swarms with s dimensions each
repeat
  if $f(\hat{\mathbf{y}})$ has not improved **then** randomly choose s from $\mathbf{S}$ and let $K = n/s$
  Randomly permutate all n dimension indices
  Construct K swarms, each with s dimensions
  for each swarm $k \in [1 \cdots K]$ **do**
    for each particle $i \in [1 \cdots p]$ **do**
      if $f(\mathbf{b}(k, P_k.\mathbf{x}_i)) < f(\mathbf{b}(k, P_k.\mathbf{y}_i))$ **then**
        $P_k.\mathbf{y}_i \leftarrow P_k.\mathbf{x}_i$
      if $f(\mathbf{b}(k, P_k.\mathbf{y}_i)) < f(\mathbf{b}(k, P_k.\hat{\mathbf{y}}))$ **then**
        $P_k.\hat{\mathbf{y}} \leftarrow P_k.\mathbf{y}_i$
    for each particle $i \in [1 \cdots p]$ **do**
      $P_k.\hat{\mathbf{y}}'_i \leftarrow \mathit{localBest}(P_k.\mathbf{y}_{i-1}, P_k.\mathbf{y}_i, P_k.\mathbf{y}_{i+1})$
    if $f(\mathbf{b}(k, P_k.\hat{\mathbf{y}})) < f(\hat{\mathbf{y}})$ **then** the kth part of $\hat{\mathbf{y}}$ is replaced by $P_k.\hat{\mathbf{y}}$
  for each swarm $k \in [1 \cdots K]$ **do**
    for each particle $i \in [1 \cdots p]$ **do**
      Update particle $P_k.\mathbf{x}_i$ using (1)
until termination criterion is met

Limitation Addressed

- Random rearrangement of swarm's dimensions is one of the strongest characteristics of CCPSO2.
- However, it can also be a weakness if S is not satisfactory.
- Manual setup of S is time consuming and mostly will not test many possibilities.
- Random selection of s will not consider search phase characteristics.

Proposed Approach

Search characteristics can greatly benefit results.

Well known behaviours include:

- Exploratory search reduces probability of local minima traps.
- Intensification search increases chance of finding better local results.
- Improved results can be obtained by:
  - Exploring at initial stages of the search.
  - Intensificating at later stages.

Well known behaviours include:

- Each swarm has its own partially independent state.
- The less the number of swarms, more dimensions will be dependent of the same swarm state.
- The more the number of swarms, less dimensions will restrict the swarms' movements.

- Considering that:
  - Intensification is usually implemented by restricting the swarm's movement.
- Then:
  - Hypothetically, since a small number of swarms restrict swarms' movement, it also could increase the likelihood of intensificating the search.
  - Likewise, a higher number of swarms could increase the probability of exploring the search space.

CCPSO2-IP

- Replace S by a boost function that controls the number of swarms $maxK$.
  - Aggressiveness of boost function is controlled by a boost rate parameter B_r.
  - $maxK$ is reduced iteratively a fixed number of times $maxTries$ by a static factor K_r.
  - Once $maxK$ is minimum, the boost function is called again to define a new $maxK$.

The process is repeated until the end of the search.

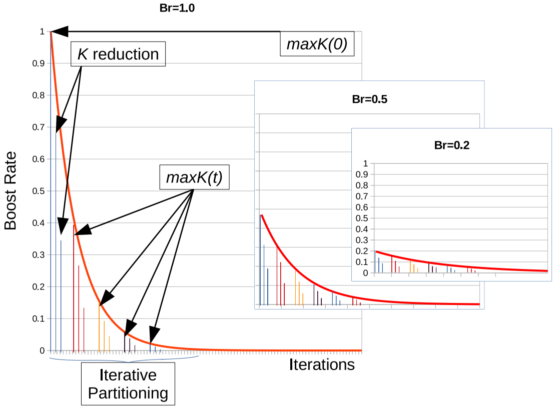

Figure : Iterative Partitioning method for Exponential boost function.

$$Boost_E(t) = \frac{B_r}{exp(12 * B_r * \left(\frac{t}{T_{max}}\right)}$$

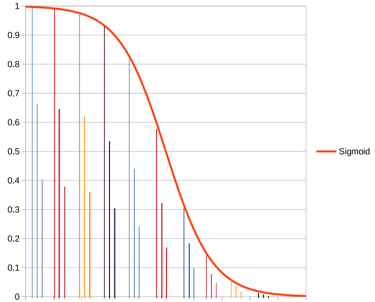

Figure : IP for Sigmoid boost function (for $B_r = 1.0$).

$$Boost_S(t) = \frac{B_r}{1.0 + \exp(12 * B_r * \left(\frac{t}{T_{max}}\right) - 6 * B_r)}$$

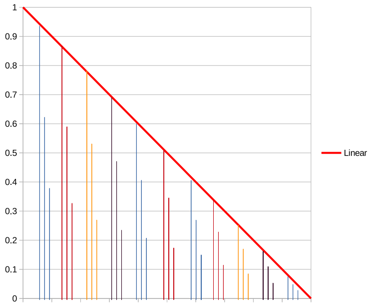

Figure : IP for Linear boost function (for $B_r = 1.0$).

$$Boost_L(t) = -B_r * \left(\frac{t}{T_{max}}\right) + B_r$$

Algorithm 2 CCPSO2-IP

$maxK(t) = MIN(MAX(n * Boost(t), 1), n)$
$K = maxK = maxK(0), K_r = 1/maxTries, fitImprovement = 1$
Create K swarms
for t in $[1 \cdots T_{max}]$ **do**
  if $fitImprovement < minImprovement$ **then**
    if $maxTries$ iterations without improvement **then**
      if $K \leq MAX(maxK * K_r, 1))$ **then**
        if $maxTries$ updates on maxK without improvement **then**
          $maxK = maxK(0)$ ▷ force exploration
        else ▷ Iteratively reduce maxK
          Calculate $maxK(t)$
        $K = maxK$
      else ▷ Iteratively reduce K
        $K = MIN(MAX(K - MAX(maxK * K_r, 1), 1), maxK)$
      if new K is different from previous K **then**
        Recreate swarms with new K
      else
        Permutate dimensions and resize swarms
      Recalculate PBest's and KBest's fitness values
    else ▷ give it a 50% chance of permutation
      if $rand < 0.5$ **then**
        Permutate dimension and resize swarms
        Recalculate PBest's and KBest's fitness
  Execute CCPSO2 search and Calculate fitImprovement

Results

- Benchmark used to validate the method:
  - Congress on Evolutionary Computation 2013/2015 (CEC13/15) for Large Scale Global Optimization (LSGO).
  - 15 Benchmark Functions
  - 1000 dimensions
- Iterative Partitioning method was compared to CCPSO2 and classic PSO.

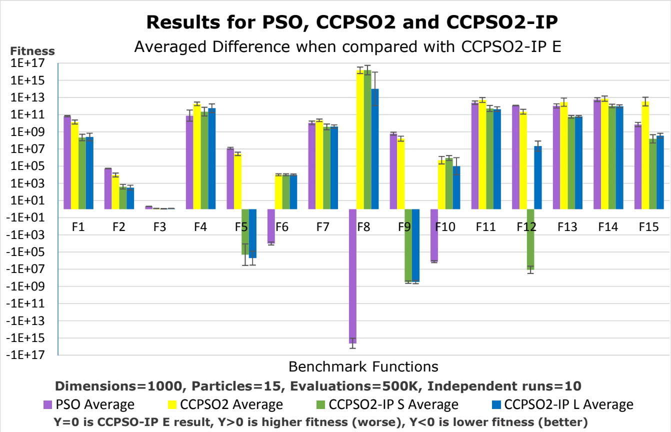

Figure : Averages and Std.Dev. on logarithmic scale compared to CCPSO2-IP E [15 benchmark functions, 500K fitness eval., 15 particles, 10 independent runs].

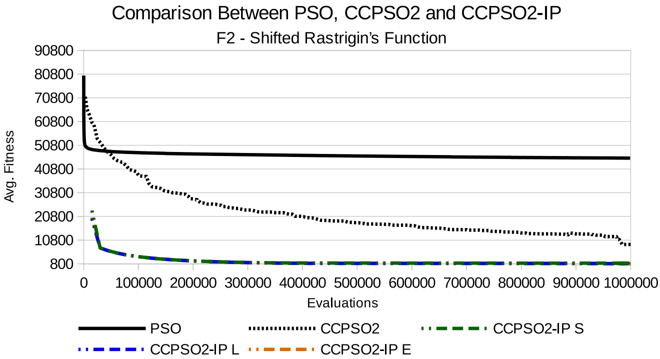

Figure : Comparison between methods for F2 benchmark function [1M fitness eval., 30 particles, 25 independent runs. PSO: [w=0.7; c1=0.8; c2=1.1]. CCPSO2: S={2,5,10,50,100,250}. CCPSO2-IP: E[Br=0.5, maxTries=2]; S[Br=0.521, maxTries=3]; L[Br=0.5, maxTries=5]].

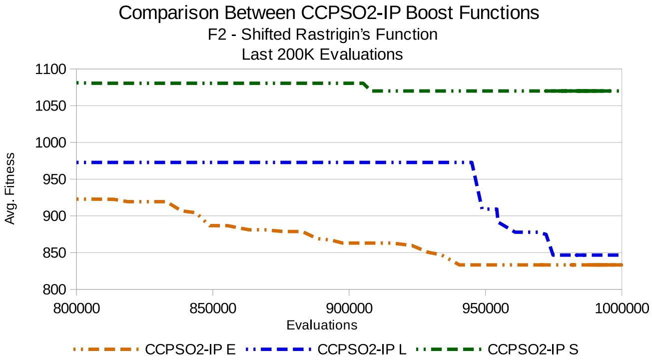

Figure : Last 200K fitness evaluations for CCPSO2-IP methods on F2 [1M fitness eval., 30 particles, 25 independent runs. CCPSO2-IP: E[Br=0.5, maxTries=2]; S[Br=0.521, maxTries=3]; L[Br=0.5, maxTries=5]].

Conclusion

CCPSO2-IP showed:

- Superior results when compared to CCPSO2 and PSO.
    - Specially on difficult functions.
- Good capacity of escaping from local minima.
    - Even after long stagnation periods.