Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

M2 AIC

TC2: Introduction to Optimization

Black-Box Optimization Benchmarking
with the COCO platform
-
Multiobjective Optimizer adaptive IBEA (ϵ-indicator)

*Group 1:* Martin BAUW, Robin DURAZ, Jiaxin GAO,
Hao LIU, Luca VEYRON-FORRER

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

IBEA: Indicator-Based Evolutionary Algorithm

- optimization: find decision space vectors leading to objective space minima
- multiobjective: the objective space is multidimensional
- evolutionary: decision space candidates follows an natural selection-like evolution
- indicator-based: binary quality indicators to compare two Pareto set approximations

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Overview of IBEA
Selection and variation

Successive steps of IBEA:

1. Initialization

2. Fitness assignment

3. Environmental selection

4. Termination

5. Mating selection

6. Variation

1

_____
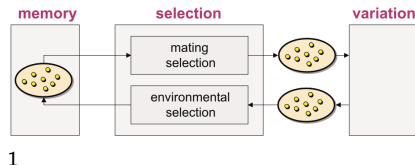
[1]Illustration from:

A Tutorial on Evolutionary Multiobjective Optimization - E. Zitzler,
M. Laumanns, and S. Bleuler

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Overview of IBEA
Selection and variation

- Binary quality indicators:

$$I_{\epsilon+(A,B)} = min_{\epsilon}\{\forall x^2 \in B \ \exists x^1 \in A : f_i(x^1) - \epsilon \leq f_i(x^2) \text{ for } i \in \{1, ..., n\}\} \tag{1}$$

- Fitness values:

$$F(x^1) = \sum_{x^2 \in P \setminus \{x^1\}} -e^{-\dfrac{I(\{x^1\},\{x^2\})}{ck}} \tag{2}$$
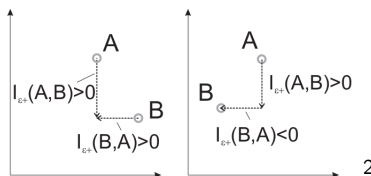
[1]Illustration from:

Indicator-Based Selection in Multiobjective Search - E. Zitzler and S. Künzli

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Overview of IBEA
Selection and variation

Recombination

For recombination, we use a Simulated Binary Crossover (SBX) operator. A uniform probability pick in $[0, 1]$ written u determines the parameter used in computing the features (decision space coordinates) of the children.

- if the uniform probability pick ≤ 0.5:

$$\beta_q = (2u)^{\frac{1}{\mu+1}} \tag{3}$$

- else:

$$\beta_q = (\frac{1}{2(1-u)})^{\frac{1}{\mu+1}} \tag{4}$$

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Overview of IBEA
Selection and variation

Recombination

Thus, we can compute the children's coordinates:

- first child:

$$child0[j] = 0.5((1 + \beta_q)parent0[j] + (1 - \beta_q)parent1[j]) \quad (5)$$

- second child:

$$child1[j] = 0.5((1 - \beta_q)parent0[j] + (1 + \beta_q)parent1[j]) \quad (6)$$

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Overview of IBEA
Selection and variation

Mating selection and mutation

Polynomial mutation operator:
The mutation operator modifies individuals by changing small parts in the associated vectors according to a given mutation rate.

- if the uniform probability pick ≤ 0.5:

$$\sigma_L = (2u)^{\frac{1}{\mu+1}} - 1 \qquad (7)$$

$$p_{mut}[j] = ind[j] + \sigma_L(ind[j] - Lo) \qquad (8)$$

- else:

$$\sigma_R = (2(1-u))^{\frac{1}{\mu+1}} \qquad (9)$$

$$p_{mut}[j] = ind[j] + \sigma_R(Up - ind[j]) \qquad (10)$$

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Code structure
Improvements regarding the execution speed

- Code build for the most general case.
- The IBEA code is in the class IBEA, where each method implements one step of the algorithm.
- No difficulty to get to the best asymptotic complexity.

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Code structure
Improvements regarding the execution speed

- Good data structures choices.
- The Indicatior fuction was the key.
- From the first implementation, to the last we divided by 4.5 the execution time.

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

CPU timing and results
Comparision with Random Search and NSGA-II

Computer specifications and batch options

- Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz
- Quad core CPU with 16GB RAM

Everything ran with a budget of 100

- Three batchs for dimensions 2, 3, 5, 10, 20
- First batch running alone, and two others together
- One batch for dimensions 40

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

CPU timing and results
Comparision with Random Search and NSGA-II

Options chosen to run the algorithm

- Population size : 100
- Maximum number of generation : 100
- Scaling factor : 0.05
- Mutation rate : 0.01
- Recombination and mutation mu : 1
- Population initialization in range (-5, 5)

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

CPU timing and results
Comparision with Random Search and NSGA-II

Dimension Batch	2	3		5
Batch 1 on 3	6.0e-04	6.3e-04		8.1e-04
Batch 2 and 3 on 3 run	8.6e-04	8.6e-04		9.1e-04
simultaneously	8.3e-04	8.4e-04		8.9e-04

Dimension Batch	10	20
Batch 1 on 3	8.3e-04	1.1e-03
Batch 2 and 3 on 3 run	1.1e-03	1.3e-03
simultaneously	1.0e-03	1.3e-03

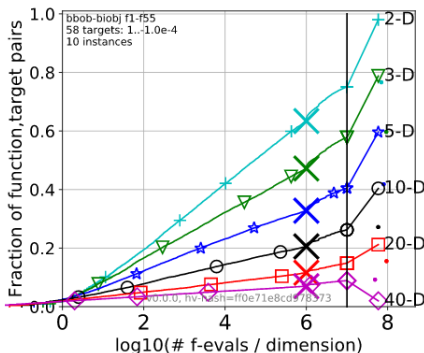| Dimension Batch	40
Whole test suite	4.2e-03

Results
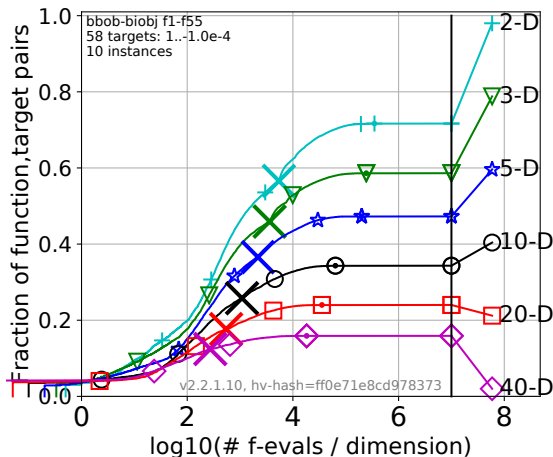
Results analysis

- Comparatively better in higher dimensions
- Results globally good for an EMOA

- More budget could have given better results
- A better initialization of population could lead to a sharper increase at the beginning
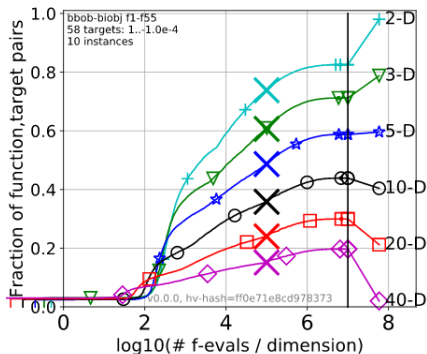
Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

CPU timing and results
Comparision with Random Search and NSGA-II

Random Search

- The introduce of Random Search:
- Its effect is directly proportional to its evalutions
- It doesn't work well when the dimension of input is too high

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

CPU timing and results
Comparision with Random Search and NSGA-II

NSGA-II

- The introduce of NSGA-II:
- The gap between our algorithm and it

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

In our experiments, we majorly focused on the comparison with NSGA-II and Random search:

IBEA VS Random search: IBEA outperformed Random search, a relatively good Pareto set approximation was given by IBEA.
IBEA VS NSGA-II: IBEA performed worse than NSGA-II.

Choosing a representation of the problem addressed, an initial population, a method of selection, a crossover operator, mutation operator, the probabilities of crossover and mutation, and the insertion method creates a variant of MOEAs algorithms.

Introduction
The algorithm
Our implementation
CPU timing and results
Conclusion
Bibliography

Non-exhaustive bibliography

- *Indicator-Based Selection in Multiobjective Search* - Zitzler, E. and Künzli, S.
- *A Tutorial on Evolutionary Multiobjective Optimization* - Zitzler, E. and Laumanns, M. and Bleuler, S.
- *Biobjective Performance Assessment with the COCO Platform* - Brockhoff, D. and Tušar, T. and Tušar, D. and Wagner, T. and Hansen, N. and Auger, A.