Applying Copeland Voting to Design an Agent-Based Hyper-Heuristic

Vinicius Renan de Carvalho and Jaime Simão Sichman

AAMAS 2017, Sao Paulo, Brazil

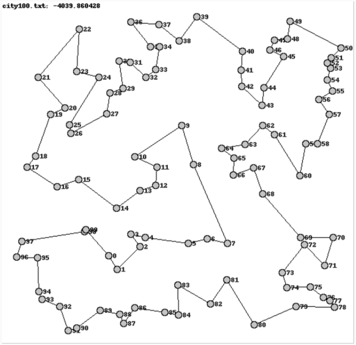
May 11th, 2017

Intelligent Techniques Laboratory
Computer Engineering Department
University of São Paulo (USP)

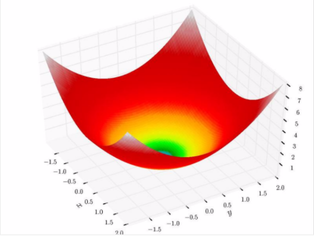
Evolutionary algorithms

- Evolutionary algorithms are algorithms which employ Darwin's theory of the survival of the fittest as their inspiration.
- They keep a population of solutions and generate new solutions using crossover and mutation operators;
- They needs a fitness function specification which tells how good is a solution;
- They are used to solve problems when there is not any problem-specific algorithm that gives a satisfactory solution in reasonable time.

Evolutionary algorithms - Applications

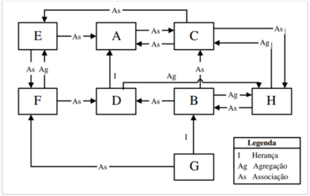


Logistics



Math

Industry



Software Development

Evolutionary algorithms - objective

Evolutionary algorithms can be classified according to their number of objectives (number of fitness function) as mono-objective and multi-objective algorithms.

- Mono-objective evolutionary algorithms:
    - Genetic Algorithm (GA) [5]
- Multi-objective evolutionary algorithms (MOEA):
    - Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [4]
    - Strength Pareto Evolutionary Algorithm 2 (SPEA2) [17]
    - Indicator-Based Evolutionary Algorithm (IBEA) [16]

Evolutionary algorithms - Pareto Front

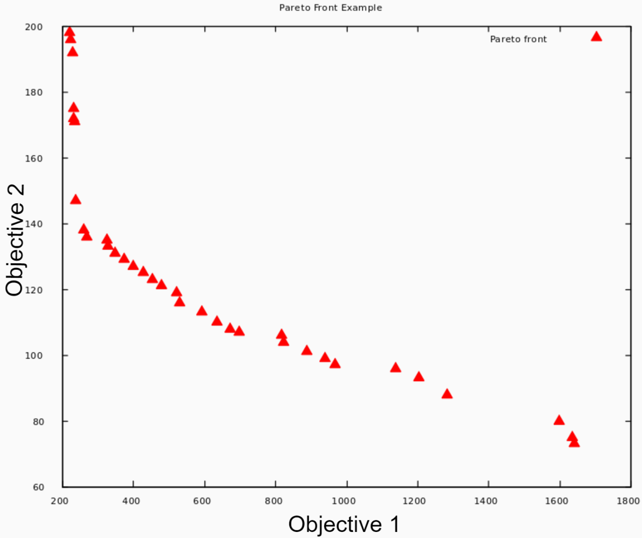


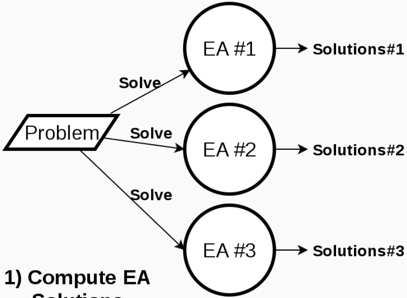
Figure 1: Two objectives Pareto Front

Evolutionary Algorithms - How to choose one?

Choosing an evolutionary algorithm is not a trivial task. Different evolutionary algorithms produce different results when applied to different problems. Thus to choose an Evolutionary algorithm we have to:

- Use literature recommendations;
- Perform a tuning and choose the best algorithm considering a quality indicator.

Evolutionary Algorithms - How to choose one?



1) Compute EA Solutions

3) Compare them to find the best one

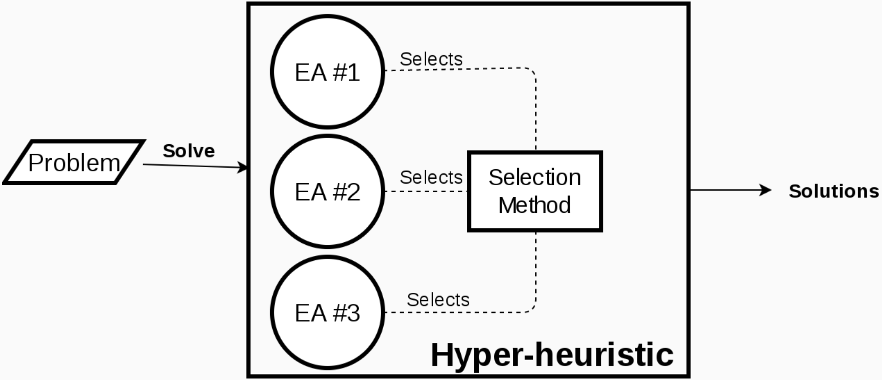
EA Results	Quality Indicator
Solutions#1	0.66
Solutions#2	0.33
Solutions#3	0.55

Evolutionary Algorithms - How to choose one?

Choosing an evolutionary algorithm is not a trivial task. Different evolutionary algorithms produce different results when applied to different problems. Thus to choose an Evolutionary algorithm we have to:

- Use literature recommendations;
- Perform a tuning and choose the best algorithm considering a quality indicator.
- Use a hyper-heuristic

Evolutionary algorithms - How to choose one?



Hyper-heuristic - Selection Method

Usually Hyper-heuristics employ a selection method. It can be:

- Roulette;
- A choice function;
- Multi-Armed Bandit approaches;

Related Work

- Multi-objective Selection hyper-heuristics, but not agent-based:
  - Vázquez-Rodríguez and Petrovic [14];
  - Maashi et al. [9];
- Mono-objective Selection agent-based hyper-heuristics:
  - Aydin and Fogarty [2];
  - Milano and Roli [10] al. [9];
  - Talbi and Bachelet [12].
- Multi-objective Selection agent-based hyper-heuristics:
  - Acan and Lotfi [1];

Motivations

- Choosing an EA is not a trivial task;
- Agent-based approaches seems suitable for this kind of problem;
- Multi-objective hyper-heuristics are on the state of art;
- Social Choice Theory provides interesting background that can be used to solve the algorithms selection problem.

Our proposal

We propose the Multi-Objective Agent-Based Hyper-Heuristic (MOABHH) which has the following characteristics:

- Share a population of solutions among a set of Multi-Objective Evolutionary Algorithms (MOEA);
- Gives a bigger population share to the best algorithm according to voting results;
- Perform a voting method using quality indicators as voters;
- Copeland voting method.

Copeland

In order to perform a Copeland voting [3], all candidates are ordered by the number of pairwise victories, minus the number of pairwise defeats.

Candidates	Wins	Losses	Wins-Losses	Final Rank
Candidate#1	4	-1	3	1
Candidate#2	3	-3	0	2
Candidate#3	1	-4	-3	3

MOABHH - Architecture

Four agents types:

- *Problem Manager agent* is responsible for all parameters.
- *EA Agent* contain a particular MOEA instance.
- *Indicator Voter* agent evaluates every EA Agent according to his own quality indicator metric.
- *Hyper-heuristic* agent defines how many solutions each *EA Agent* will receive.

Four artifacts types:

- *System variables artifact* keeps the problem specification and MOABHH parameters.
- *Population artifact*, keeps the main current population of solutions.
- *Population share artifact* contains which solutions will be used by each evolutionary algorithm during the next generation.
- *Copeland artifact* keeps all voting information.

MOABHH - Population Sharing

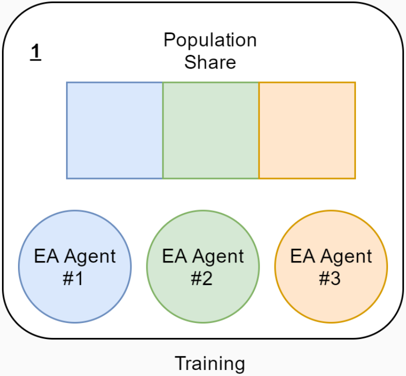


Figure 2: Population Sharing

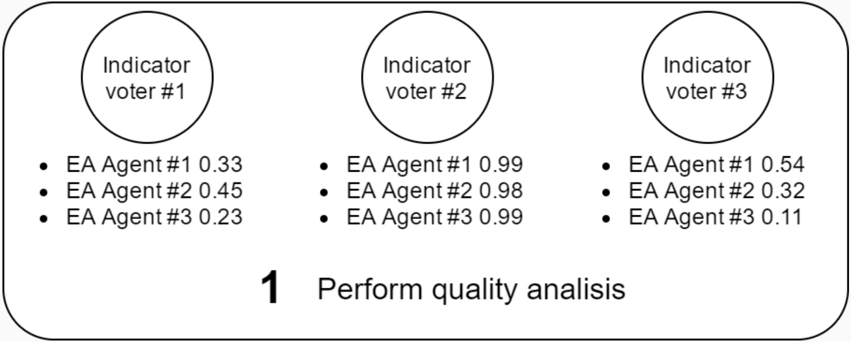


Figure 3: Voting method. First, all Indicator voter agents rank EA Agents based on their results.

MOABHH - Votation

2

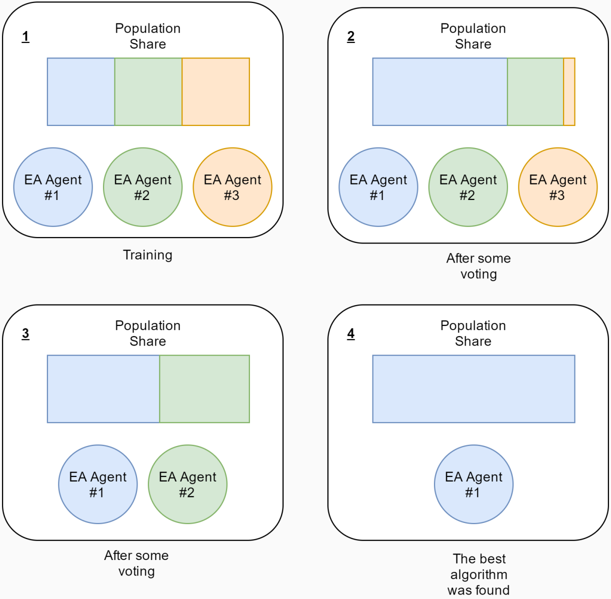
- One-on-one contest
- EA Agent#1 x EA Agent#2 on Voter #1
- EA Agent#1 x EA Agent#2 on Voter #2
- EA Agent#1 x EA Agent#2 on Voter #3
- EA Agent#1 x EA Agent#3 on Voter #1
- EA Agent#1 x EA Agent#3 on Voter #2
- EA Agent#1 x EA Agent#3 on Voter #3
- EA Agent#3 x EA Agent#2 on Voter #1
- EA Agent#3 x EA Agent#2 on Voter #2
- EA Agent#3 x EA Agent#2 on Voter #3

3

Candidate	Wins	Losses	Net	Final Rank
EA Agent#1	4	-1	3	1
EA Agent#2	3	-3	0	2
EA Agent#3	1	-4	-3	3

Figure 4: Voting method. In step 2 the Copeland voting is performed. In step 3 the Copeland ranking is generated.

MOABHH - Population Sharing



MOABHH - Implementation

- Agents developed in Java JDK 8;
- MOEAs from jMetal framework;
- Artifacts from Cartago framework.

Quality Indicators

There are different indicators to assess the quality of an algorithm:

Quality Indicator	Formula
Ratio of non-dominated solutions (RNI) [13]	$\frac{\lvert NonDominated(S)\rvert}{\lvert S\rvert}$
Hypervolume [18]	$volume(\cup_{i=1}^{\lvert S\rvert} v_i)$
Generational Distance (GD) [11]	$\frac{(\sum_{i=1}^{\lvert S\rvert} d_i^q)^{\frac{1}{q}}}{\lvert S\rvert}$
Inverted Generational Distance (IGD) [19]	$\frac{(\sum_{i=1}^{\lvert P\rvert} d_i^q)^{\frac{1}{q}}}{\lvert P\rvert}$
Spread [11]	$\frac{d_f+d_l+\sum_{i=1}^{\lvert S\rvert-1} \lvert d_i-\overline{d}\rvert}{d_f+d_l+(\lvert S\rvert-1)\overline{d}}$

Experiments - Configuration

- 5 algorithms:
  - IBEA;
  - SPEA2;
  - NSGA-II;
  - MOABHH
    - Random algorithm selection (*RDN*) among IBEA, SPEA2 and NSGA-II;
    - Copeland algorithm selection (*CPL*) among IBEA, SPEA2 and NSGA-II;
- 40 independent runs.
- Kruskal-Wallis test with 5% of significance level.

Experiments - Used Benchmark

In our experiments we employed the Walking Fish Group. (WFG) [7] benchmark.

Table 1: WFG characteristics, extracted from [7].

Problem	Separability	Modality	Bias	Geometry
WFG1	separable	uni	polynominal, flat	convex, mixed
WFG2	non-separable	uni	-	convex, disconnected
WFG3	non-separable	uni	-	linear, degenerate
WFG4	separable	multi	-	concave
WFG5	separable	deceptive	-	concave
WFG6	non-separable	uni	-	concave
WFG7	separable	uni	parameter dependent	concave
WFG8	non-separable	uni	parameter dependent	concave
WFG9	non-separable	multi, deceptive	parameter dependent	concave

Hypervolume results for two objectives

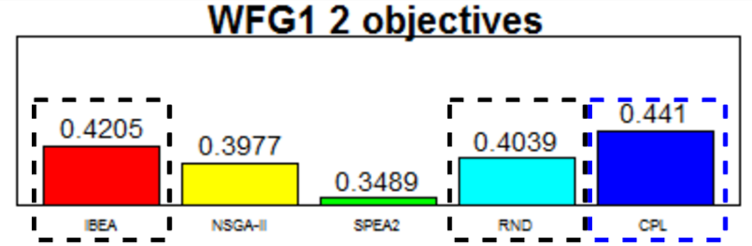
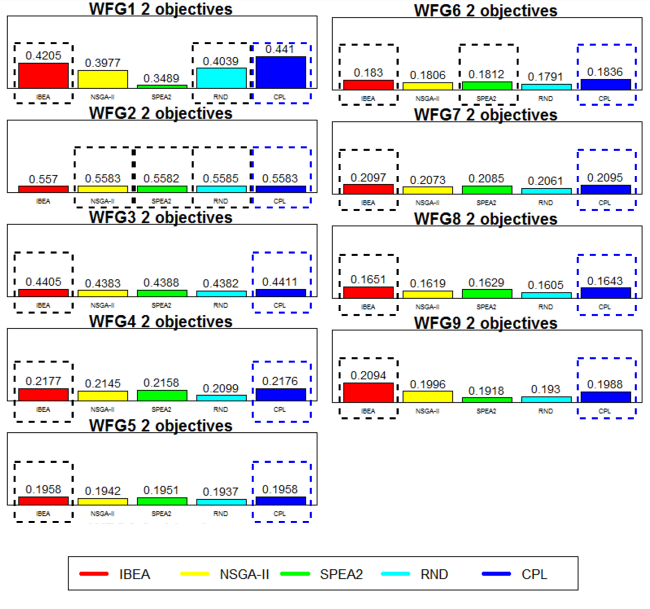
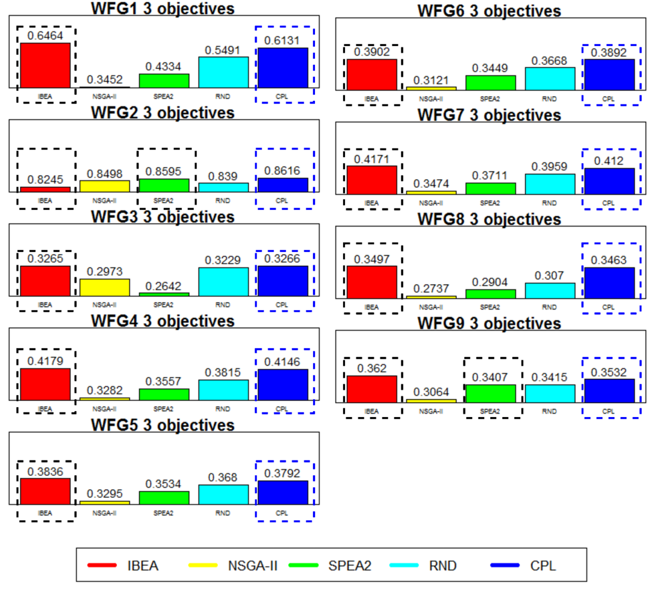


Figure 6: Result example where dashed lines means statistical ties

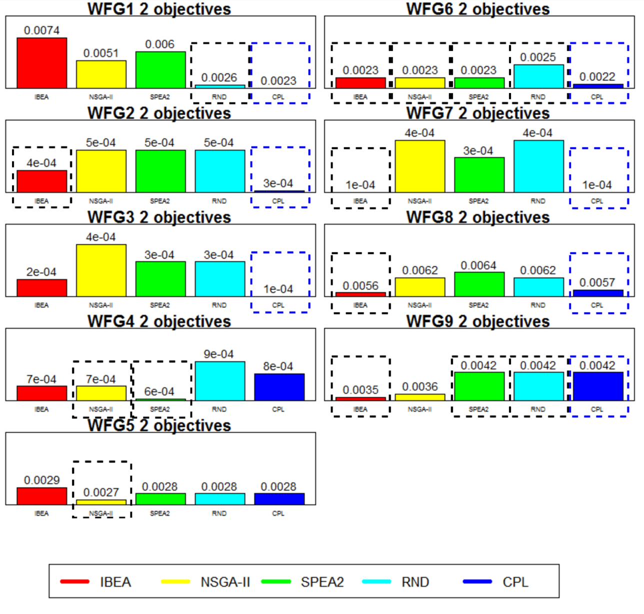
Hypervolume results for two objectives



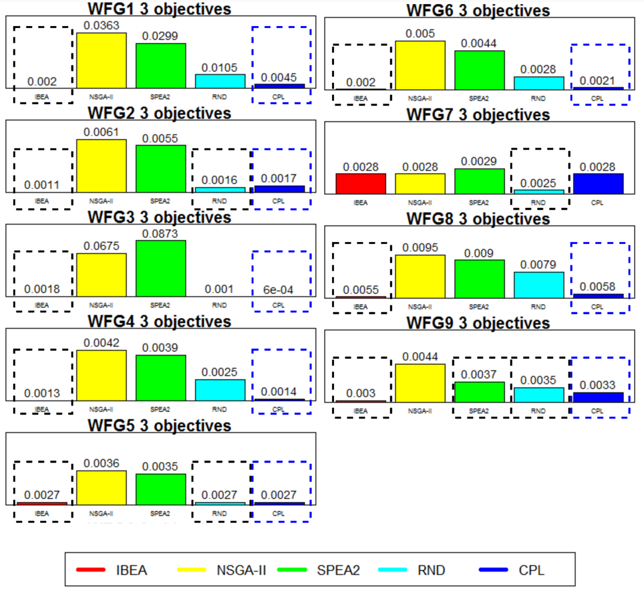
Hypervolume results for three objectives



IGD results for two objectives



IGD results for three objectives



Overall results

Quality Indicator	Two objectives	Three objectives
Hypervolume	9/9	9/9
IGD	7/9	8/9
GD	8/9	2/9
Spread	5/9	6/9
RNI	9/9	7/9

Table 2: Number of problems where CPL has achieved MOEA best results

Results - Time

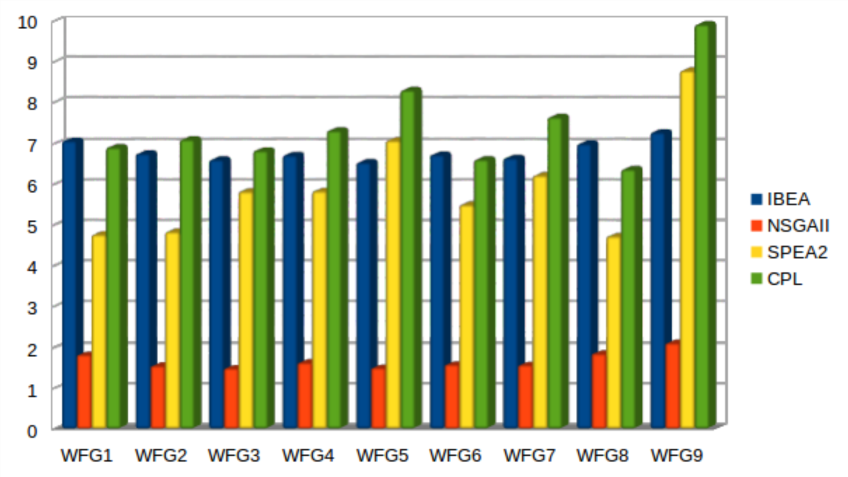


Figure 8: Average time (in minutes) for 2 objectives WFG suite

Results - Time

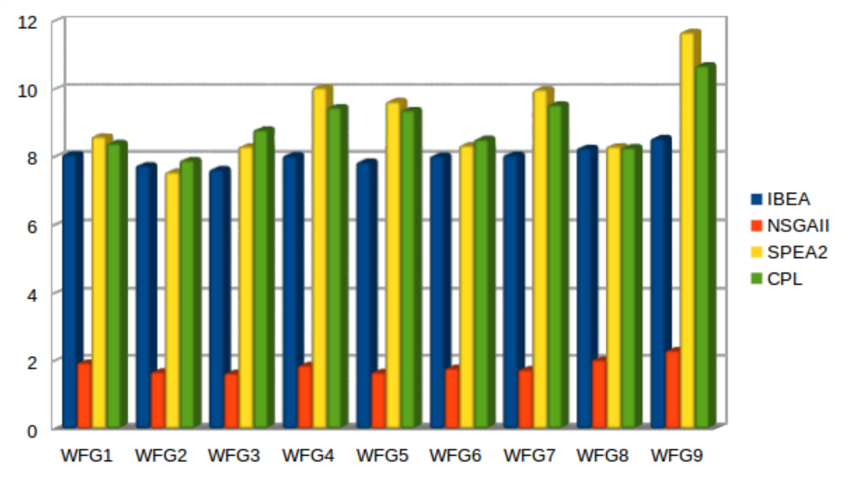


Figure 9: Average time (in minutes) for 3 objectives WFG suite

Conclusions

MOABHH-CPL has competitive results against the studied algorithm with little addition of computational effort. However, MOABHH-CPL removes from user the effort choosing the best MOEA.

Future work

- Use different meta-heuristic, such as MOEA/D-DRA [15], MOEA/D-DD [8] and MOMBI-II [6];
- Use different voting methods;
- Solve up to ten objectives problems;
- Apply MOABHH to different problems.

Acknowledgments

CNPq

Thank you!

MOABHH - Pseudocode

Algorithm 1: MOABHH Pseudocode.

```
Input: Problem, MOABHH params
begin
    Initialize agents and artifacts;
    Generate a random population of solutions;
    while Training do
        Uniformly share the population among EA Agents;
        EA Agents execute for one generation;
        Update the main population;
    end
    while Executing do
        Evaluate EA Agents qualities;
        Perform the voting method;
        Share population among EA Agents according to voting results;
        EA Agents execute for γ generations;
        Update the main population;
    end
    return Main population
end
```

Where $\gamma = 12$

HH Assign

Algorithm 2: HH Assign

```
begin
    if There is more than two MHAgents active then
        HH Agent assigns more (β ∗ 0.75) percent of the population share for
          the election winner, more (β ∗ 0.25) for second place winner and
          removes β percent from the last voted;
    end
    else
        HH Agent assigns more β percent of the population share for the
          election winner and removes β percent of the population share from
          the less voted;
    end
end
```

Where $\beta = 3$

Bibliography I

A. Acan and N. Lotfi.
A multiagent, dynamic rank-driven multi-deme architecture for real-valued multiobjective optimization.
Artificial Intelligence Review, pages 1–29, 2016.

M. E. Aydin and T. C. Fogarty.
Teams of autonomous agents for job-shop scheduling problems: An experimental study.
Journal of Intelligent Manufacturing, 15(4):455–462, 2004.

A. H. Copeland.
A reasonable social welfare function.
In *Mimeographed notes from a Seminar on Applications of Mathematics to the Social Sciences, University of Michigan*, 1951.

Bibliography II

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan.
A fast and elitist multiobjective genetic algorithm: NSGA-II.
Evolutionary Computation, IEEE Transactions on, 6(2):182–197, Apr 2002.

D. E. Goldberg and J. H. Holland.
Genetic algorithms and machine learning.
Machine learning, 3(2):95–99, 1988.

R. Hernández Gómez and C. A. Coello Coello.
Improved metaheuristic based on the r2 indicator for many-objective optimization.
In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 679–686, New York, NY, USA, 2015. ACM.

S. Huband, P. Hingston, L. Barone, and L. While.
A review of multiobjective test problems and a scalable test problem toolkit.
Evolutionary Computation, IEEE Transactions on, pages 477–506, 2006.

K. Li, K. Deb, Q. Zhang, and S. Kwong.
An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition.
IEEE Transactions on Evolutionary Computation, 19(5):694–716, oct 2015.

M. Maashi, E. Özcan, and G. Kendall.
A multi-objective hyper-heuristic based on choice function.
Expert Systems with Applications, 41(9):4475–4493, 2014.

Bibliography IV

- M. Milano and A. Roli.
  Magma: A multiagent architecture for metaheuristics.
  Trans. Sys. Man Cyber. Part B, 34(2):925–941, Apr. 2004.
- N. Srinivas and K. Deb.
  Multiobjective optimization using nondominated sorting in genetic algorithms.
  Evolutionary Computation, IEEE Transactions on, 2:221–248, 1994.
- E. Talbi and V. Bachelet.
  COSEARCH: A Parallel Cooperative Metaheuristic.
  Journal of Mathematical Modelling and Algorithms, 5(1):5–22, 2006.
- K. C. Tan, T. Lee, and E. Khor.
  Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons.
  Artificial Intelligence Review, 17(4):251–290, 2002.

Bibliography V

J. Vázquez-Rodríguez and S. Petrovic.
A mixture experiments multi-objective hyper-heuristic.
Journal of the Operational Research Society, 64(11):1664–1675, 2012.

Q. Zhang, W. Liu, and H. Li.
The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances.
Technical Report CES-491, School of CS & EE, University of Essex, Feb 2009.

E. Zitzler and S. Künzli.
Indicator-based selection in multiobjective search.
In *PPSN*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer, 2004.

Bibliography VI

E. Zitzler, M. Laumanns, and L. Thiele.
SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization.
In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100. International Center for Numerical Methods in Engineering, 2001.

E. Zitzler and L. Thiele.
Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach.
Trans. Evol. Comp, 3(4):257–271, nov 1999.

Bibliography VII

E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca.
Performance assessment of multiobjective optimizers: An analysis and review.
Trans. Evol. Comp, 7(2):117–132, Apr. 2003.