Multi-Objectivization of the Tool Selection Problem on a Budget of Evaluations

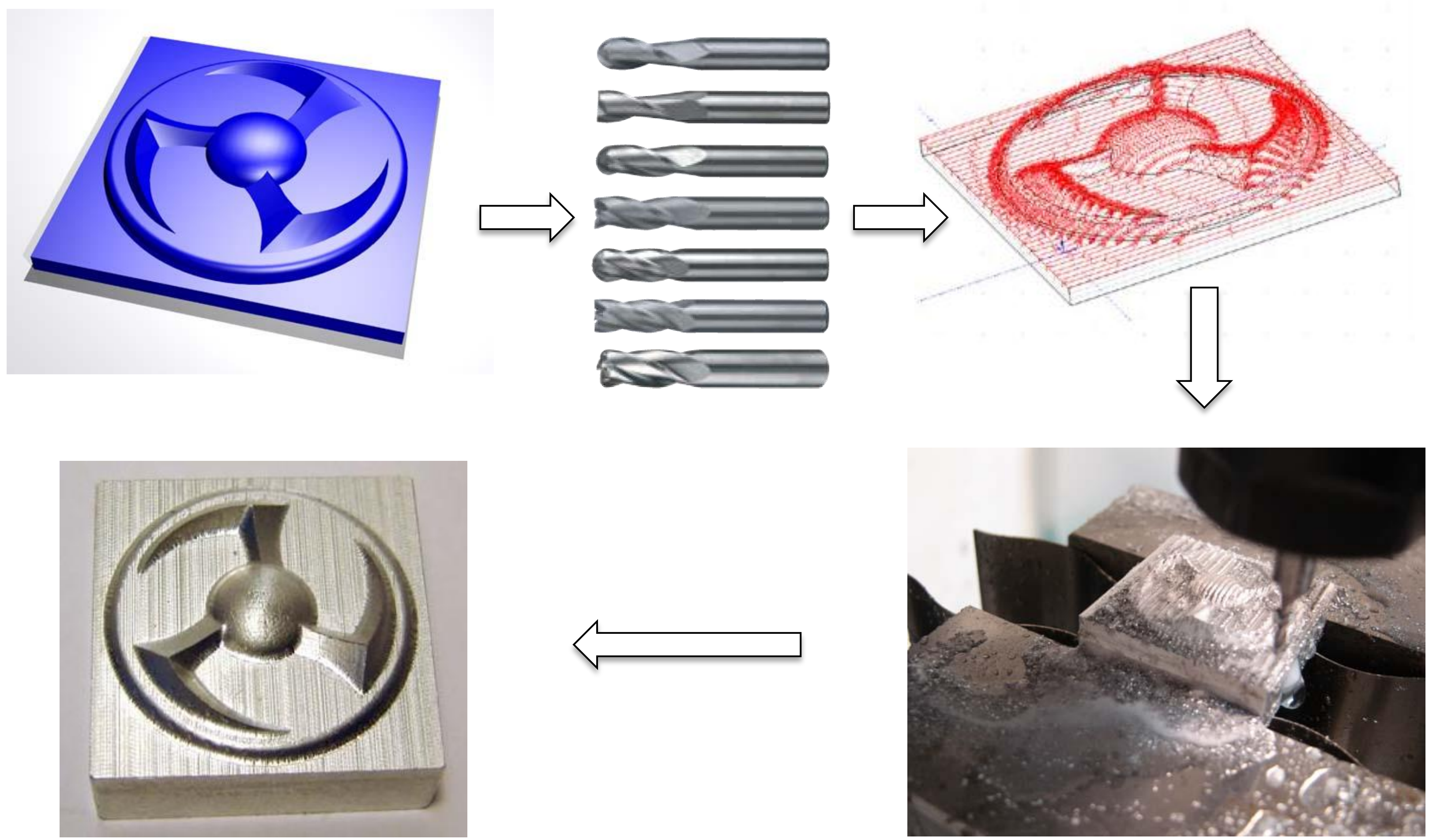
Alexander Churchill, Phil Husbands, Andy Philippides

University of Sussex
Centre for Computational Neuroscience and Robotics
EPSRC Engineering and Physical Sciences Research Council
vero Software

Introduction

- Previously, we evaluated the performance of single-objective metaheuristics on the Tool Selection Problem in machining
- Can we achieve better **single-objective** search performance using **multi-objective** techniques?
- Does preferential search improve performance?

CNC Milling



Tool Selection in Rough Machining

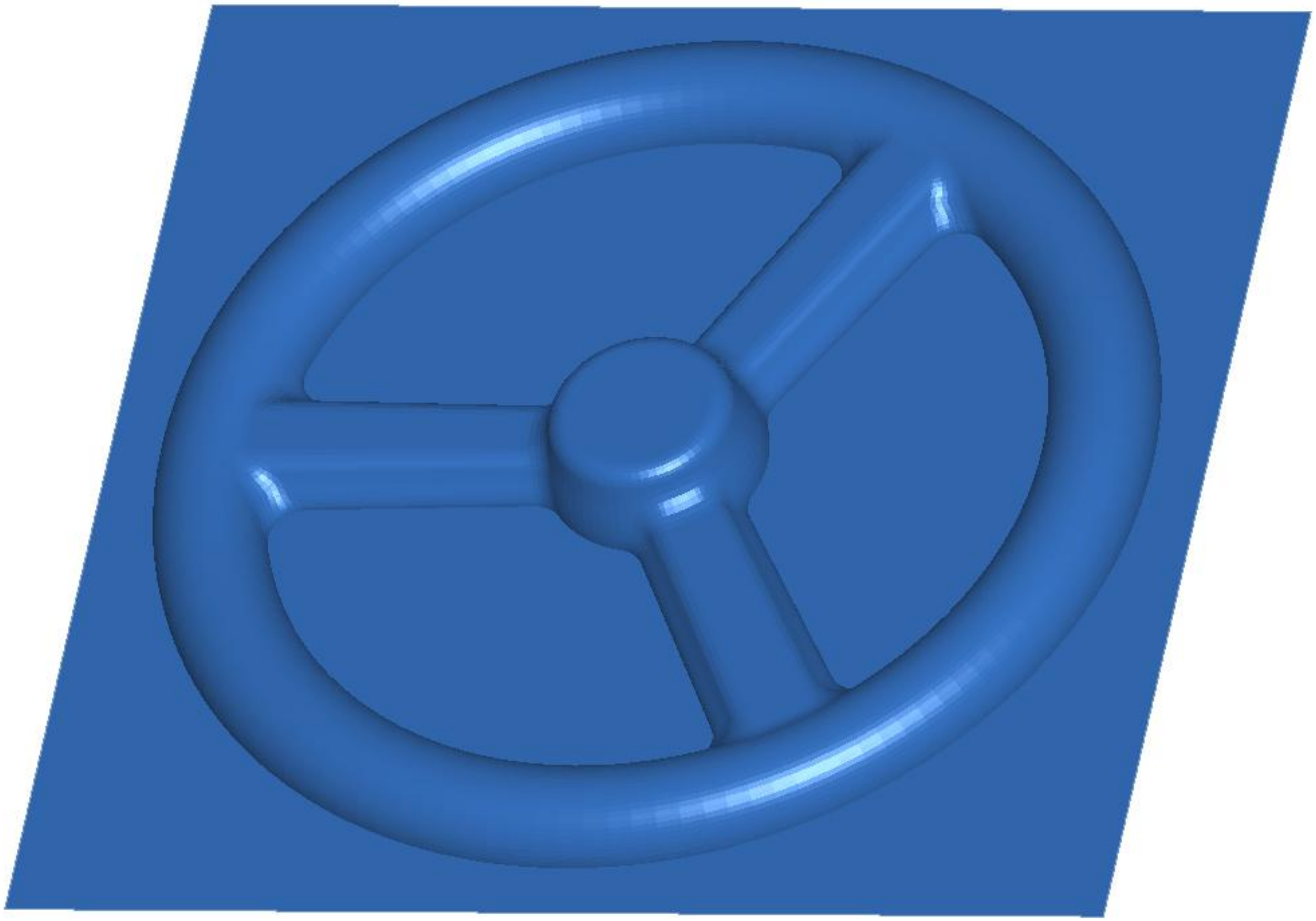
Tool Selection Experiment

- Produce a component using a sequence of up to 5 tools chosen from a library of 18.
- Tools have different geometrical properties and operate at different cutting speeds.
- The component has to have a **final surface tolerance < 1mm** in all places.
- The aim is to find the sequence that can achieve this in **the shortest amount of time**.

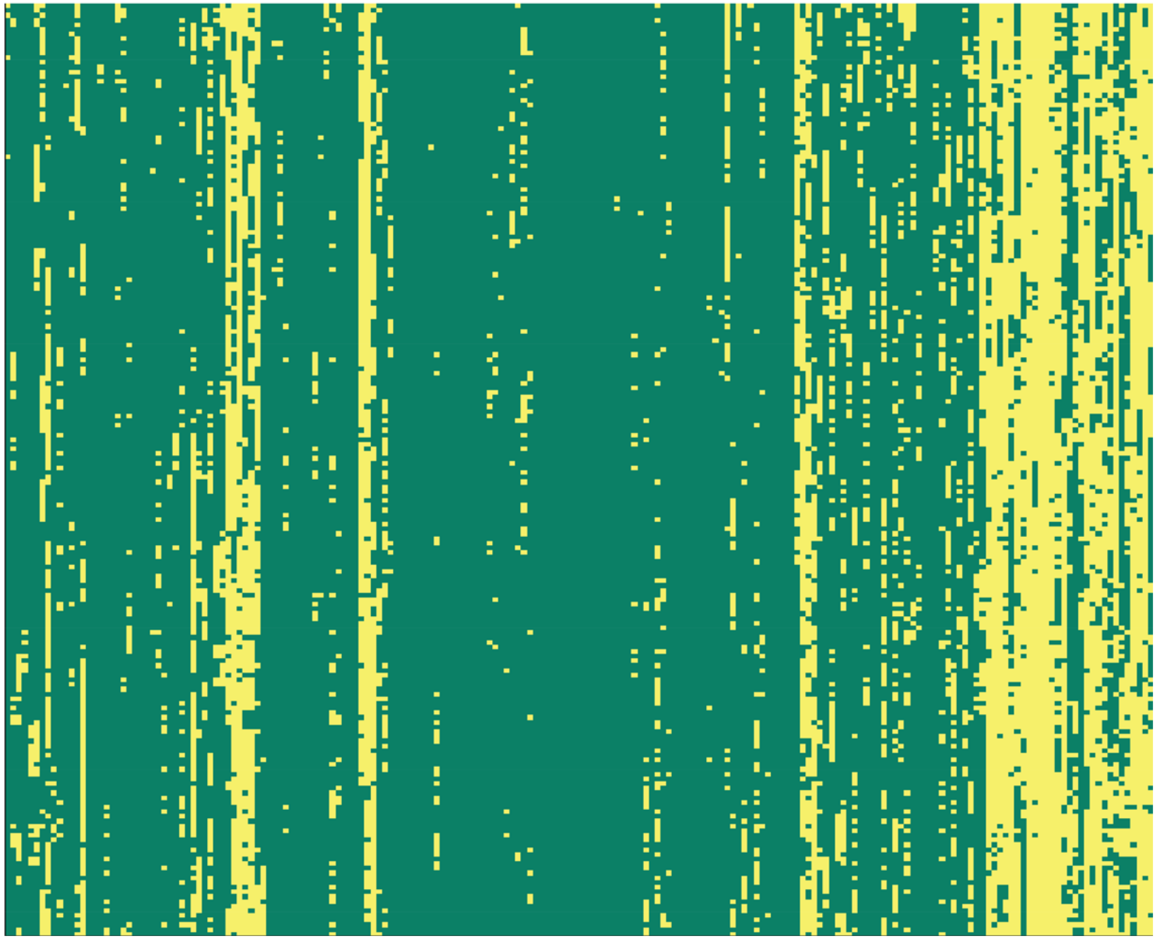
Simulations

- Support tools with different geometrical properties
- Simulations on the part used here can take from around 1 – 15 minutes to compute
- This is a big issue when trying to integrate this on the shop floor

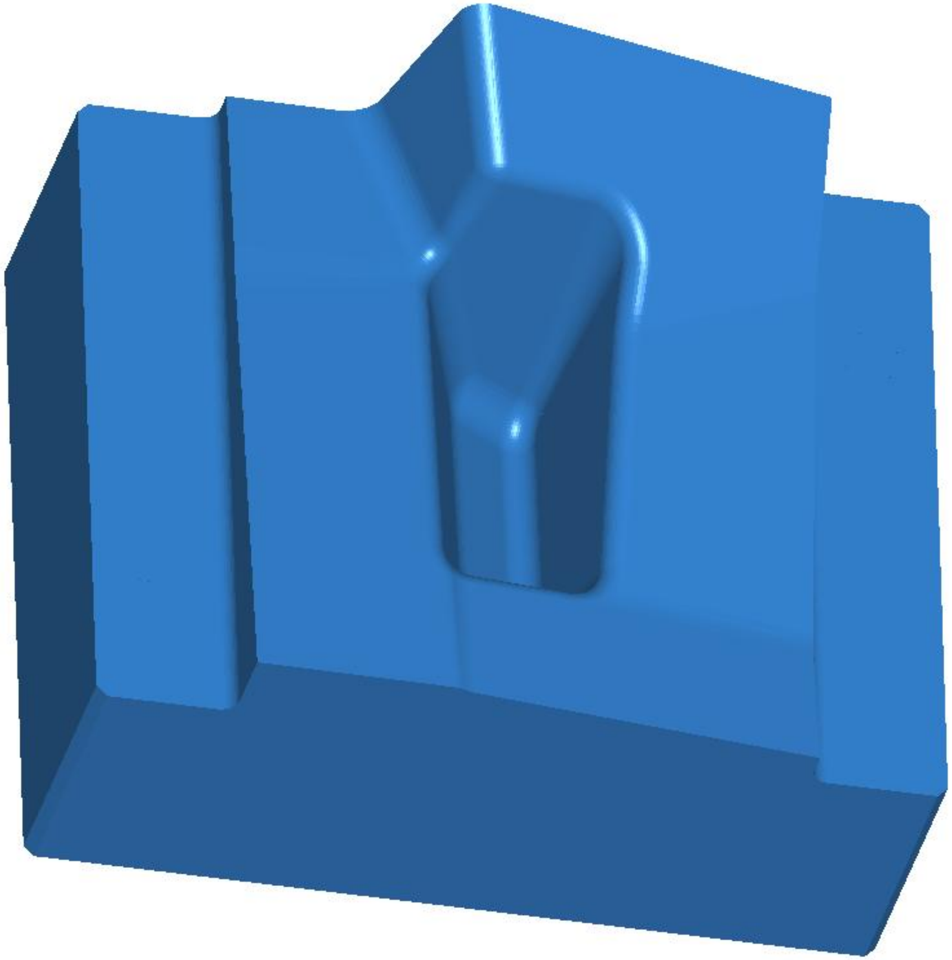
“Simple” Part



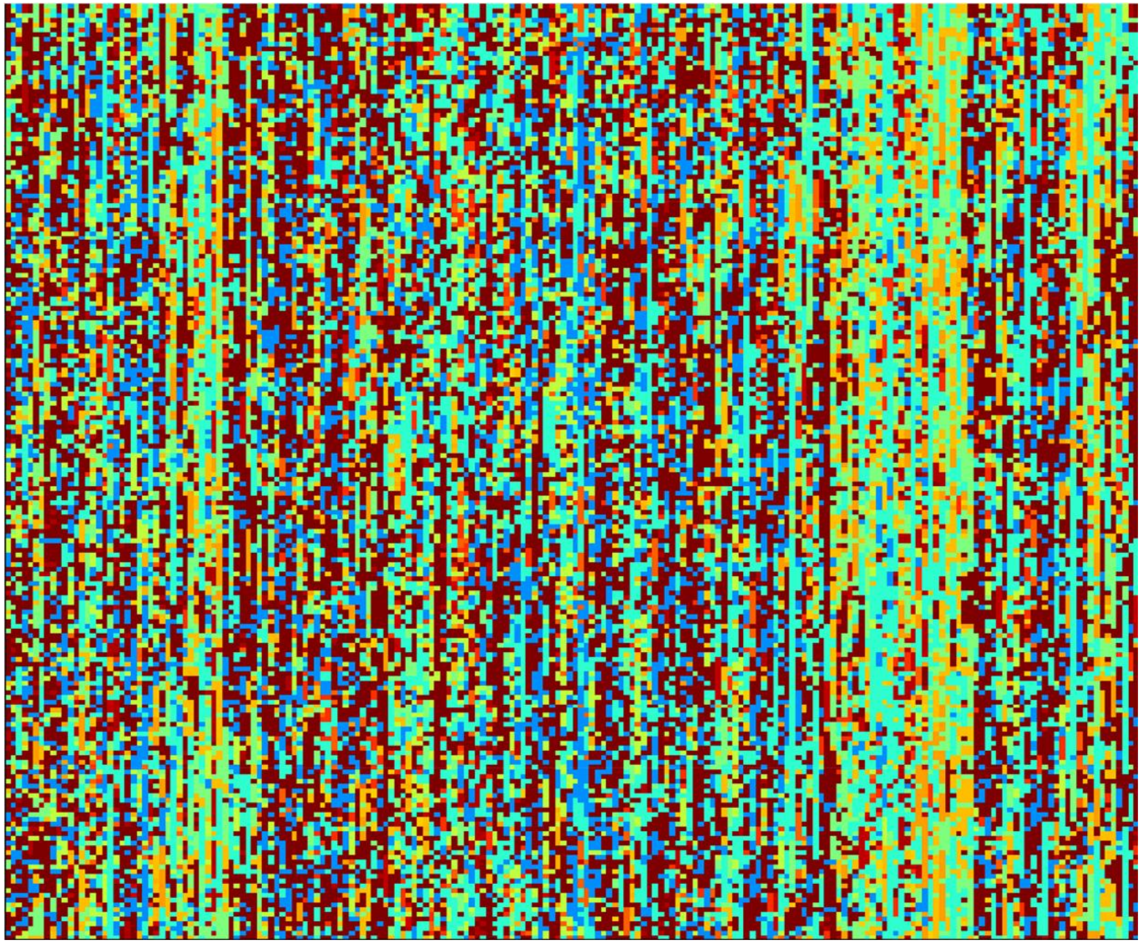
“Simple” Part



“Difficult” Part



“Difficult” Part



Multi-Objectivization

- Using multi-objective techniques on a single objective problem (Knowles et al., 2001)
- Can escape local optima by following multiple search gradients
- Reduces signal-to-noise ratio by isolating their good aspects from the ‘noise’ of their undesirable characteristics (Lochtefeld and Ciarallo, 2012)

University of Sussex
CCNR Centre for Computational Neuroscience and Robotics

Single Objective Fitness Function

$$f(x) = T_x + c_x$$

$$c_x = \begin{cases} 2k, & d_x > 1.5mm \\ k, & 1.0mm > d \leq 1.5mm \\ 0, & d_x < 1.0mm \end{cases}$$

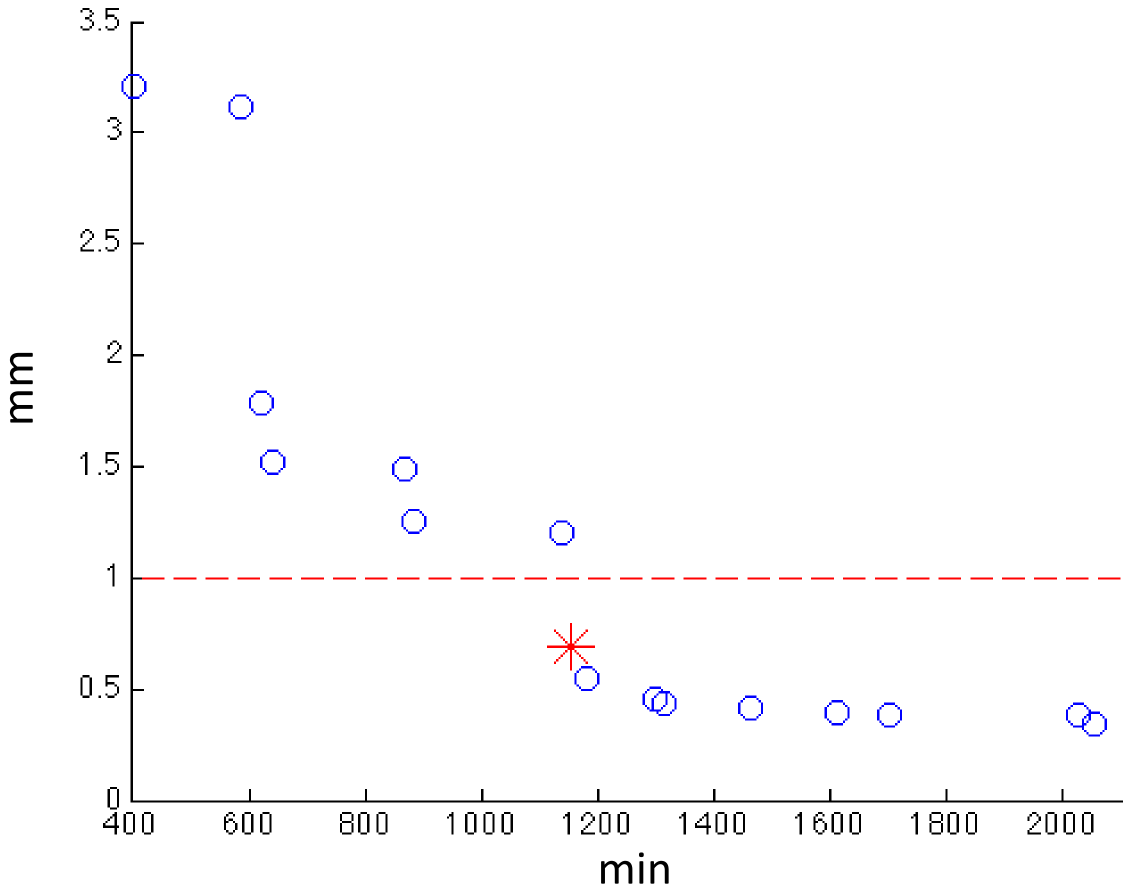
where *x* is a tool sequence, T_x is the total machining time, d_x is the excess material and *k* is a user defined value.

Multi-Objective Fitness Function

$$f_1(x) = T_x$$
$$f_2(x) = d_x$$

The first objective is the total machining time; the second objective is the excess material.

Pareto Front



Experiment

- Compare search performance of single-objective and multi-objective algorithms on the "difficult" component
- Test with different population sizes on four different evaluation budgets:
  150, 250, 350, 500
- For each population size and evaluation limit, count the number of times the "optimal solution" is found over 1,000 runs

Single-Objective Algorithms

- Simple Genetic Algorithm (GA)
- Random Restart Stochastic Hill Climbing (RRSHC)

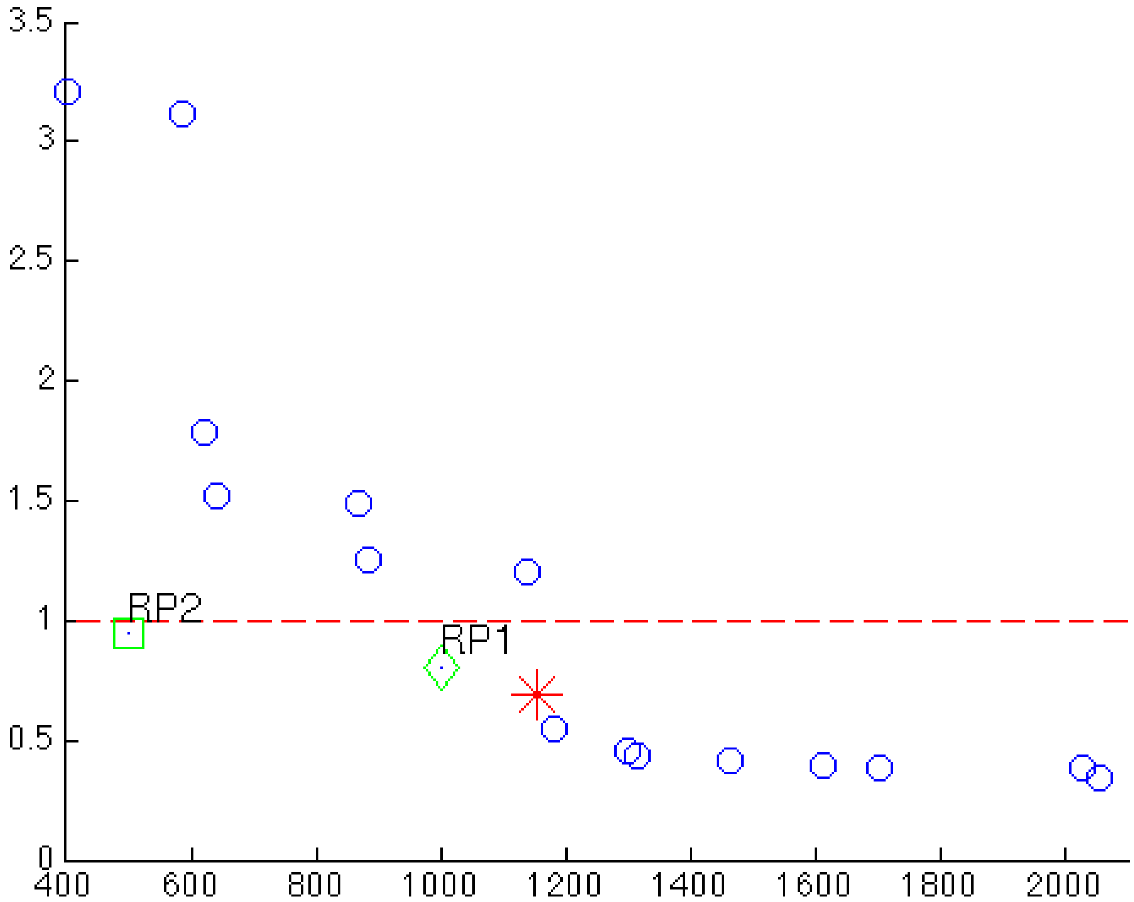
Multi-Objective Algorithms

- NSGA-II
- NSGA-II with duplication control (NSGA-II*)
- Reference Point NSGA-II (R-NSGA-II)
- Guided Elitism
- Guided Elitism with duplication control (GE*)

R-NSGA-II

- Reference Point modification to NSGA-II (Deb and Sundar, 2006)
- Crowding distance is modified to reflect closeness to a user-specified reference point
- Diversity maintained by an epsilon parameter
- 2 reference points were evaluated on this problem

R-NSGA-II



3.5
3
2.5
2
1.5
1
0.5
0
400
600
800
1000
1200
1400
1600
1800
2000
RP2
RP1

Guided Elitism

- Hybrid between the single and multi-objective approach
- Use the single-objective function that we already have to guide search

Guided Elitism

- Generate a child population
- Add to the current population, to create population, k
- Sort using a single-objective aggregate function, $f_s()$

Guided Elitism

- Remove the best 10% of members of k
- Assign these members the top dominance rank and a crowding distance method equal to their $f_s()$ value
- Add these “elite” members to the new population
- Add remaining members using normal Pareto methods

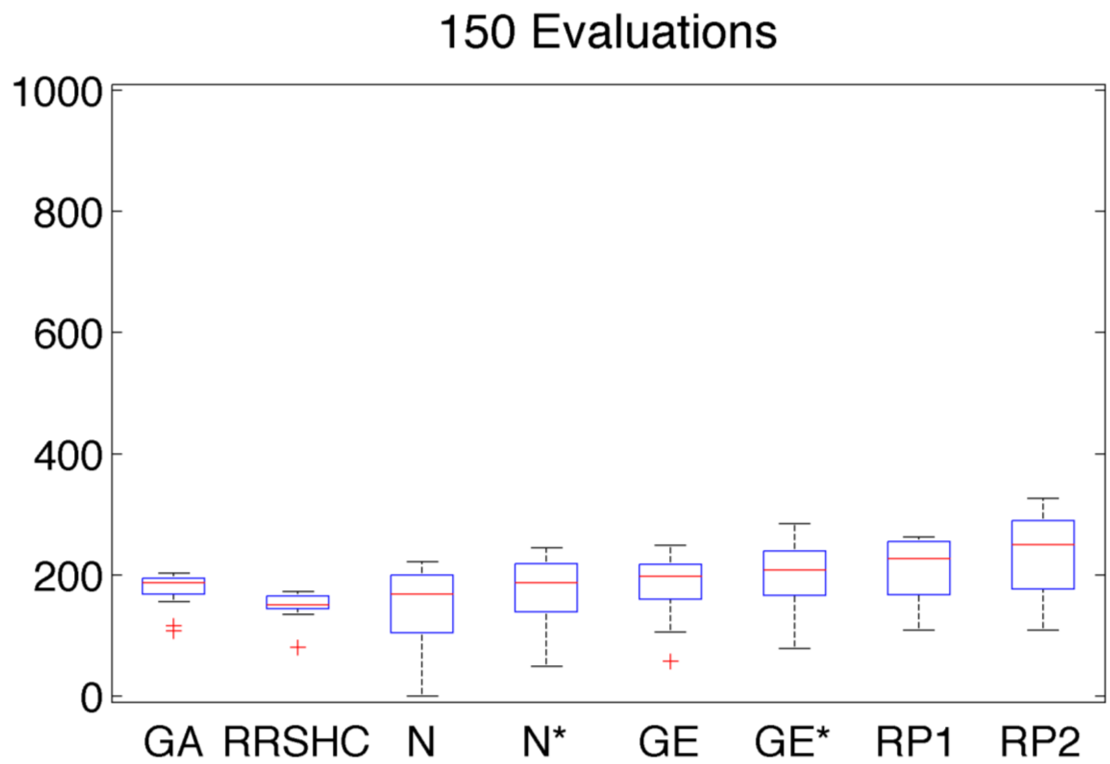
Guided Elitism

- Similar to (Ishibuchi et al., 2006)
- Is not probabilistic
- Guarantees the survival of preferred solutions and victory in binary tournaments against "non-elites"

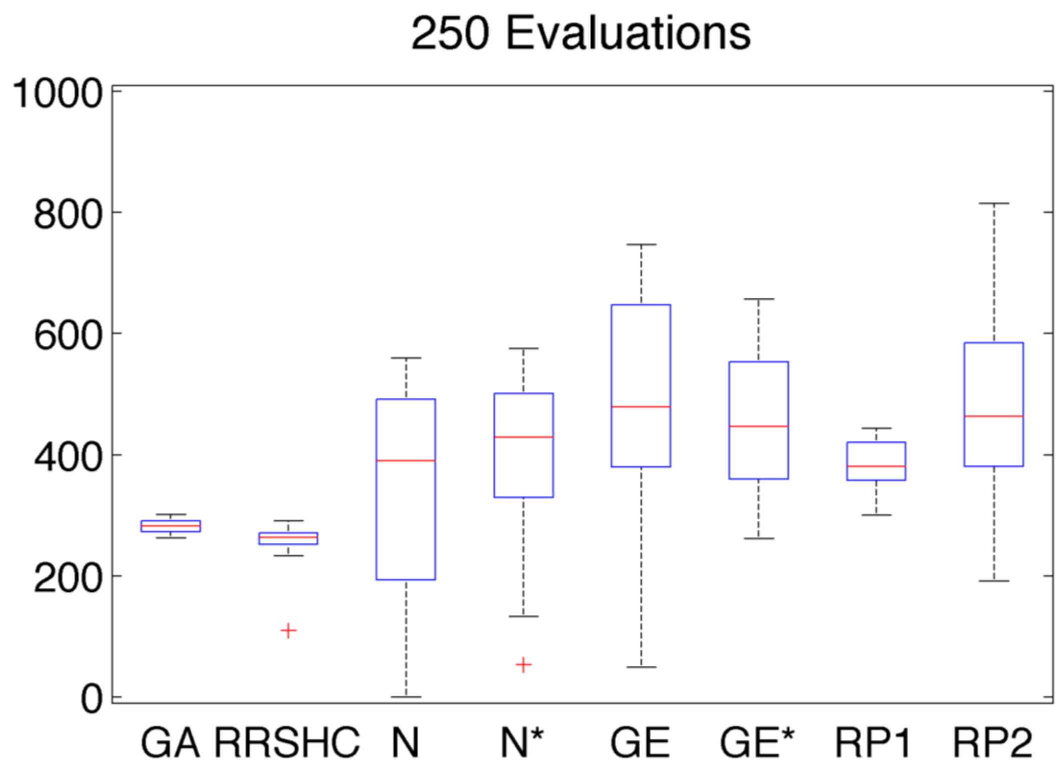
Experiment

- 16 configurations
- Population-based algorithms used population sizes: 5-15; 20; 25; 30; 35; 40
- RRSHC used restart limits:
  10 – 160 (in increments of 10)
- Four separate evaluations limits:
  150; 250; 350; 500

Results

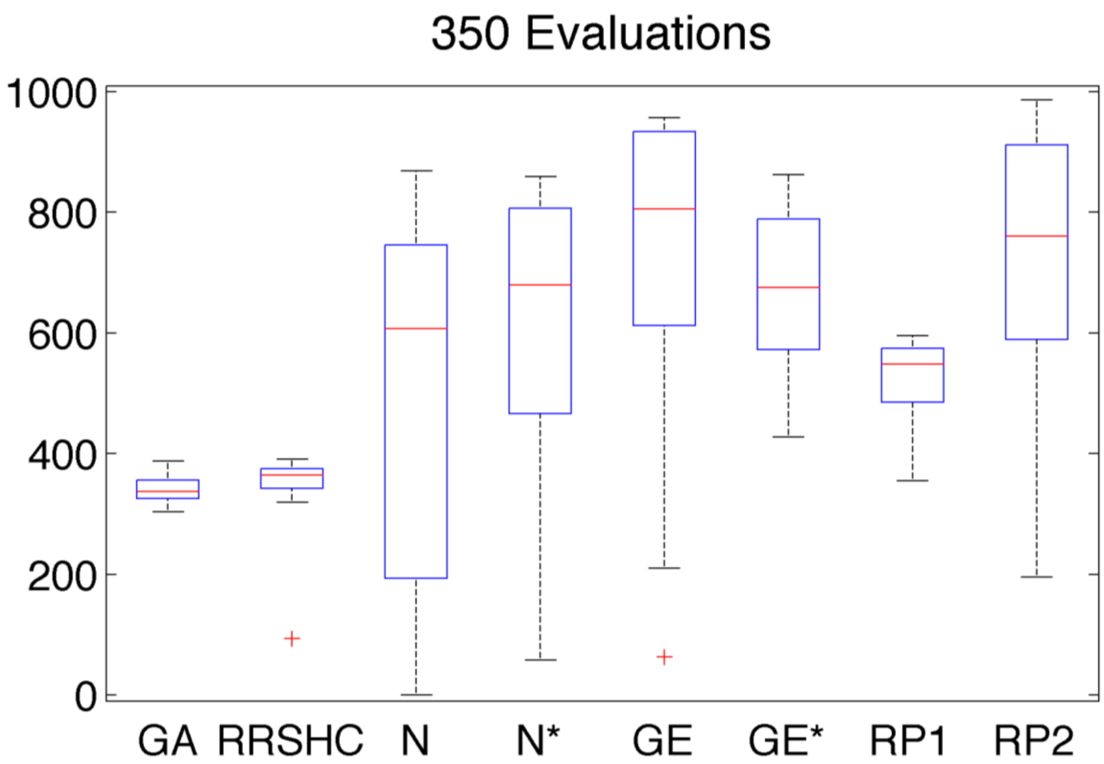

150 Evaluations
1000
800
600
400
200
0
GA
RRSHC
N
N*
GE
GE*
RP1
RP2

Results

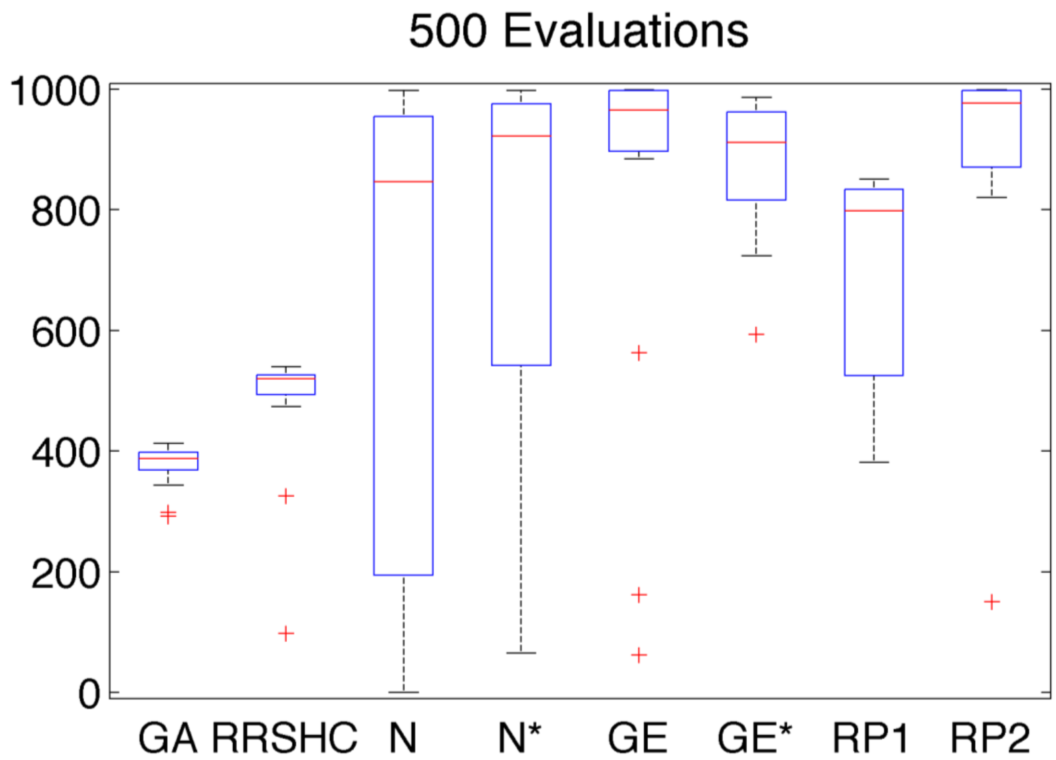


250 Evaluations
1000
800
600
400
200
0
GA
RRSHC
N
N*
GE
GE*
RP1
RP2

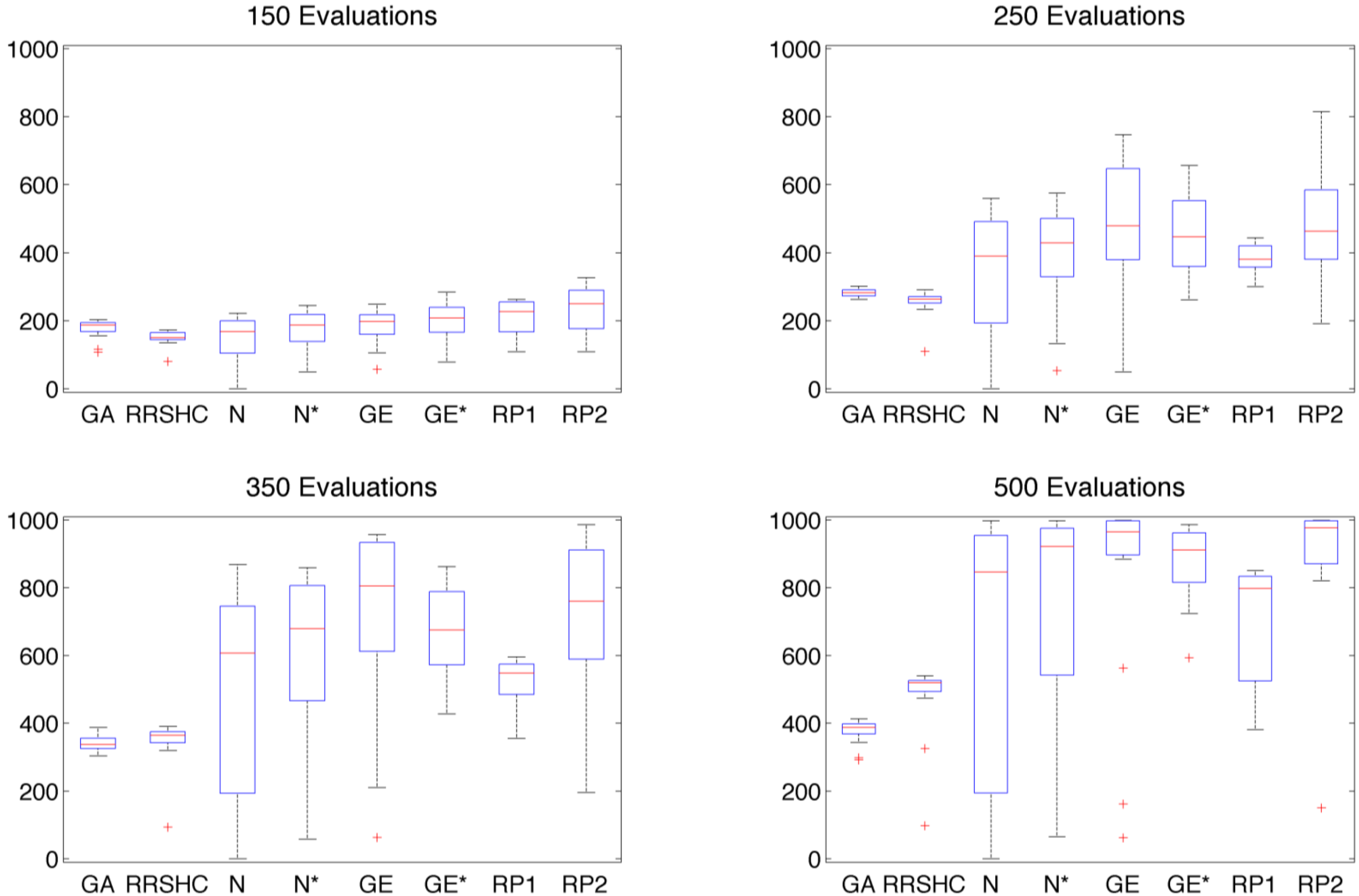
Results

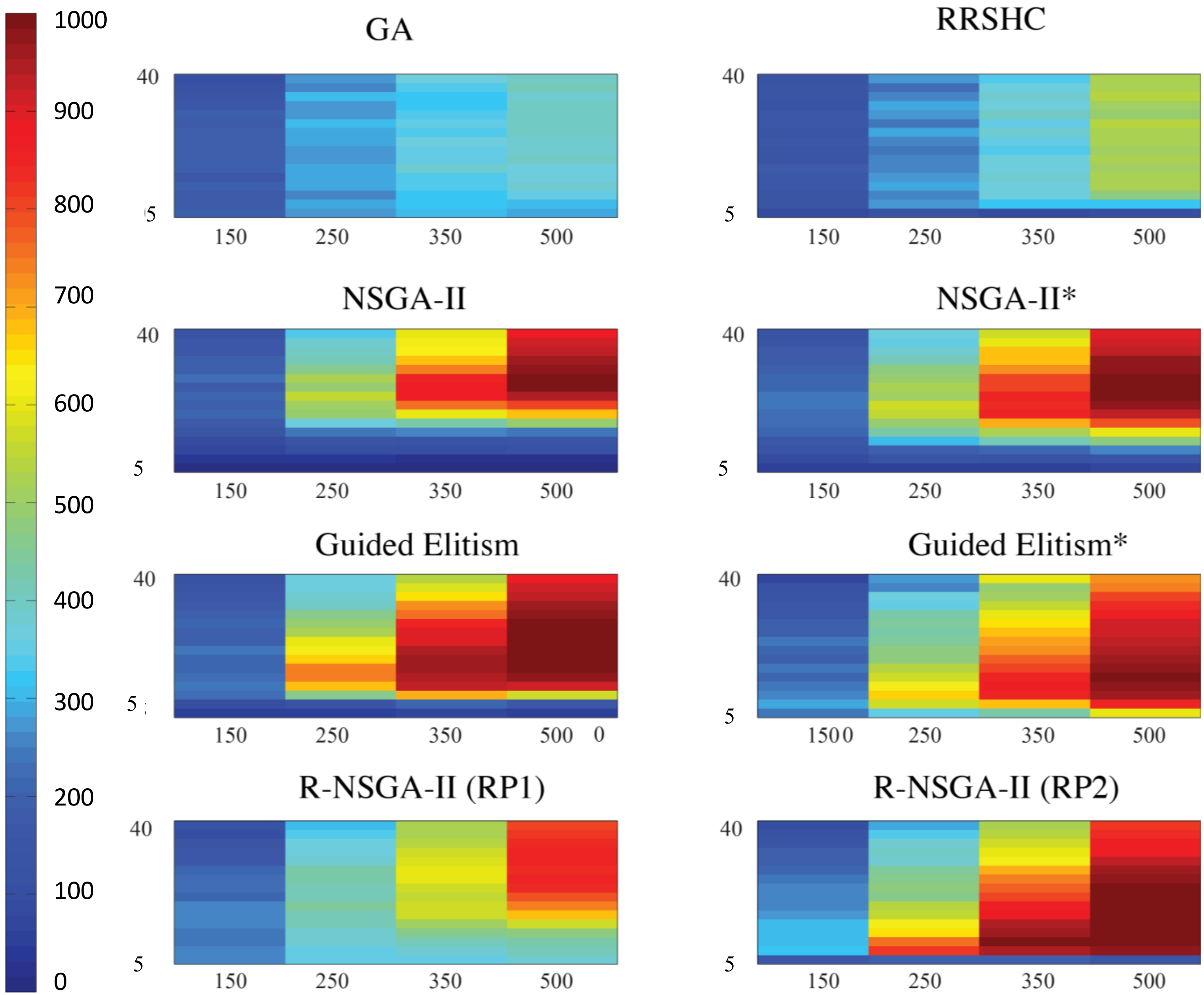

350 Evaluations
1000
800
600
400
200
0
GA
RRSHC
N
N*
GE
GE*
RP1
RP2

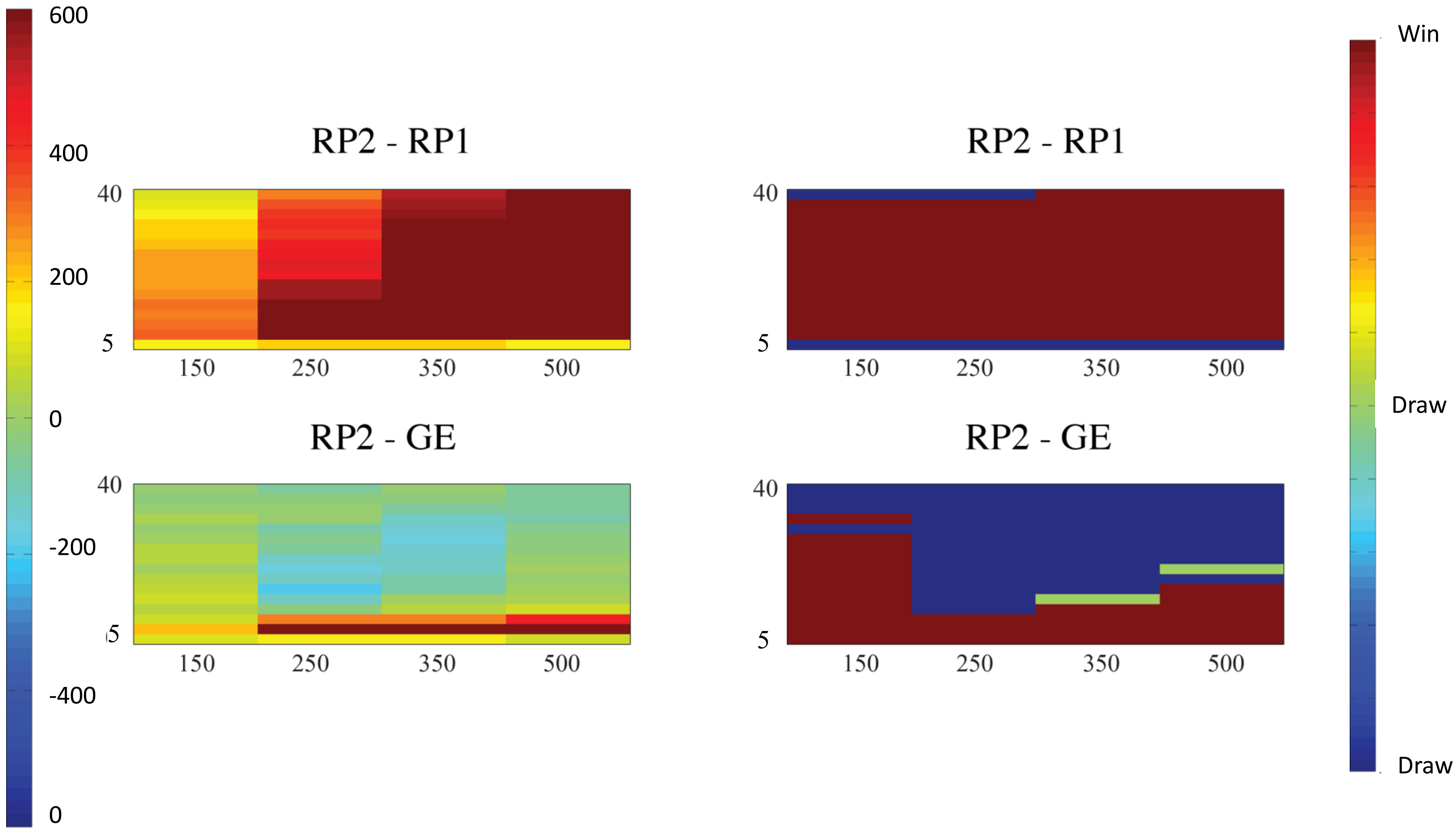
Results


500 Evaluations
1000
800
600
400
200
0
GA
RRSHC
N
N*
GE
GE*
RP1
RP2

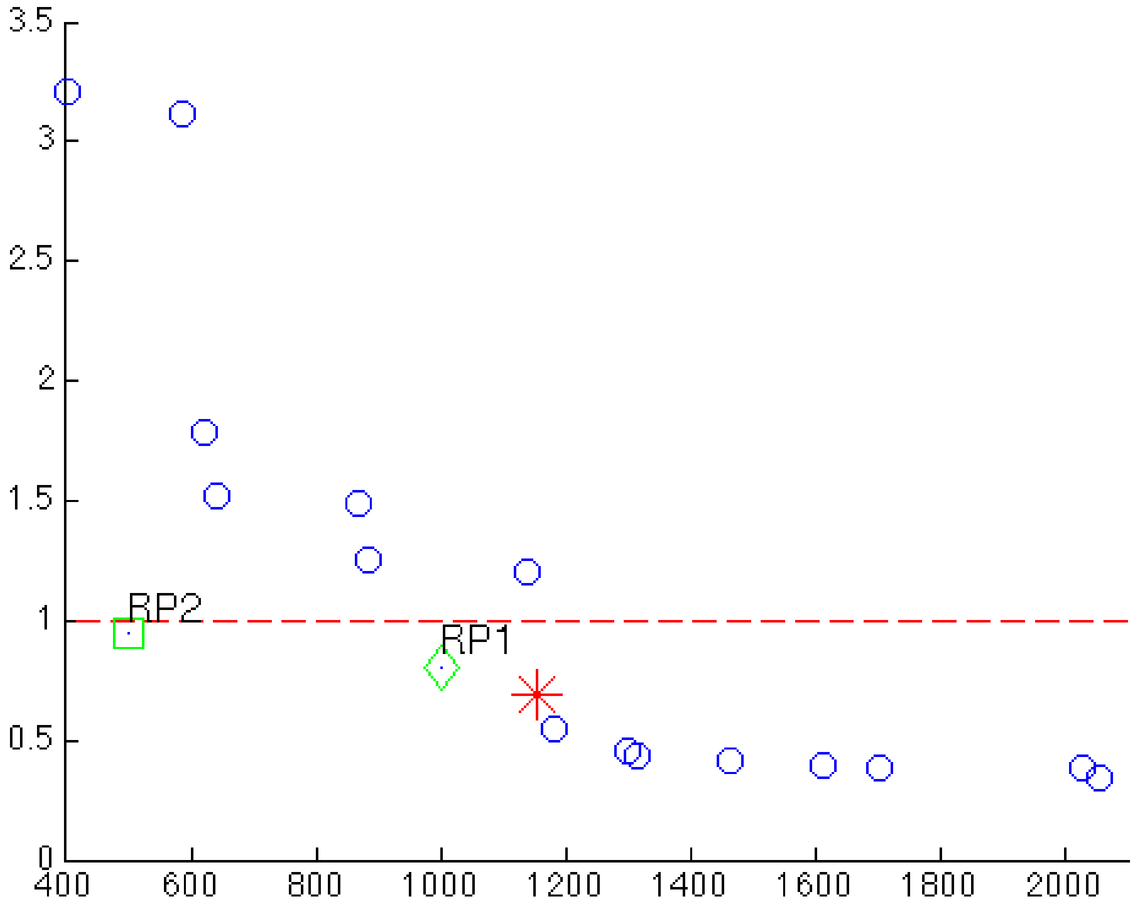
Results


150 Evaluations
250 Evaluations
350 Evaluations
500 Evaluations
0
200
400
600
800
1000
GA RRSHC N N* GE GE* RP1 RP2

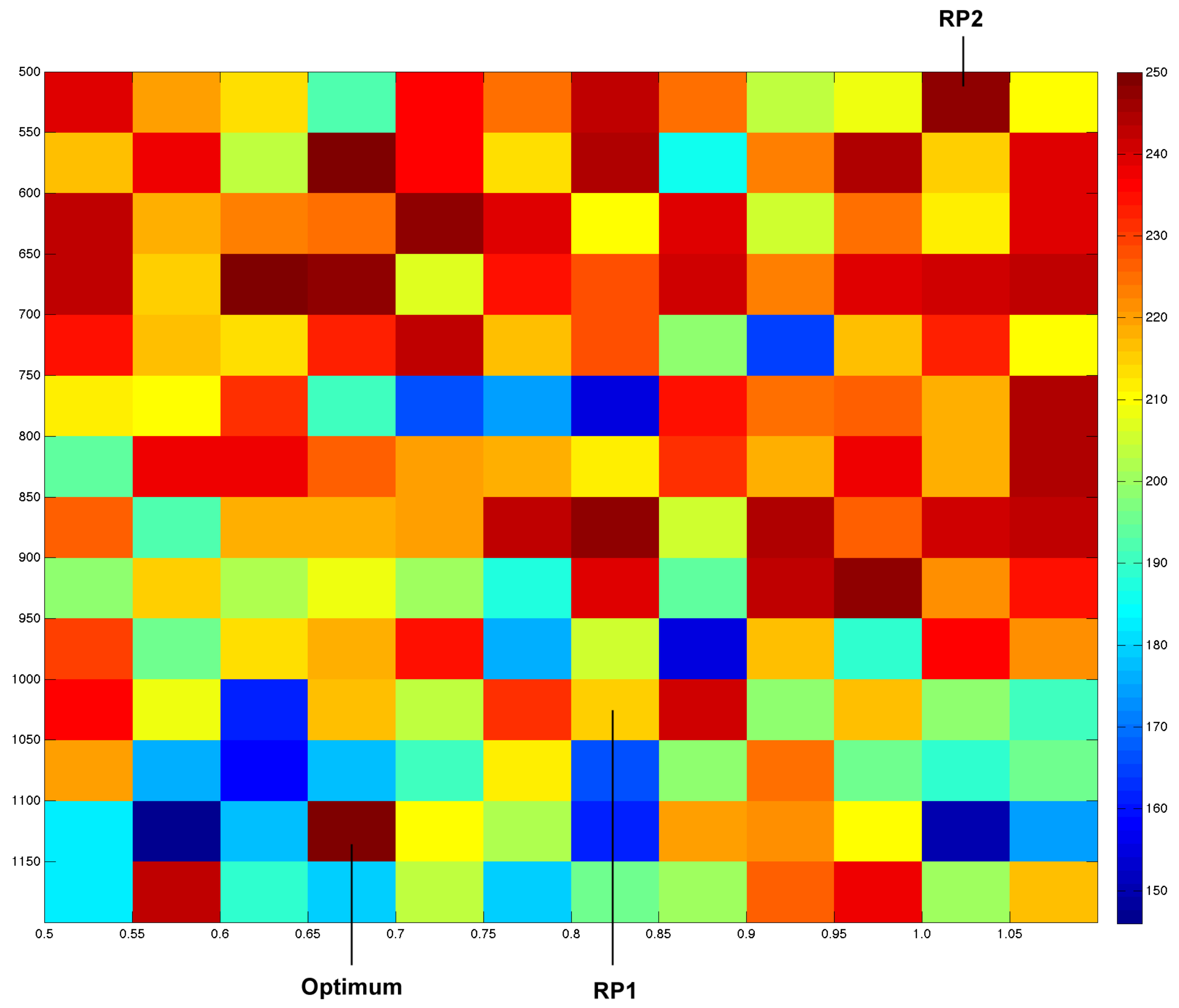

1000
900
800
700
600
500
400
300
200
100
0
GA
RRSHC
NSGA-II
NSGA-II*
Guided Elitism
Guided Elitism*
R-NSGA-II (RP1)
R-NSGA-II (RP2)
40
5
150
250
350
500


600
400
200
0
-200
-400
0
RP2 - RP1
40
5
150
250
350
500
RP2 - GE
40
5
150
250
350
500
RP2 - RP1
40
5
150
250
350
500
RP2 - GE
40
5
150
250
350
500
Win
Draw
Draw

Reference Points



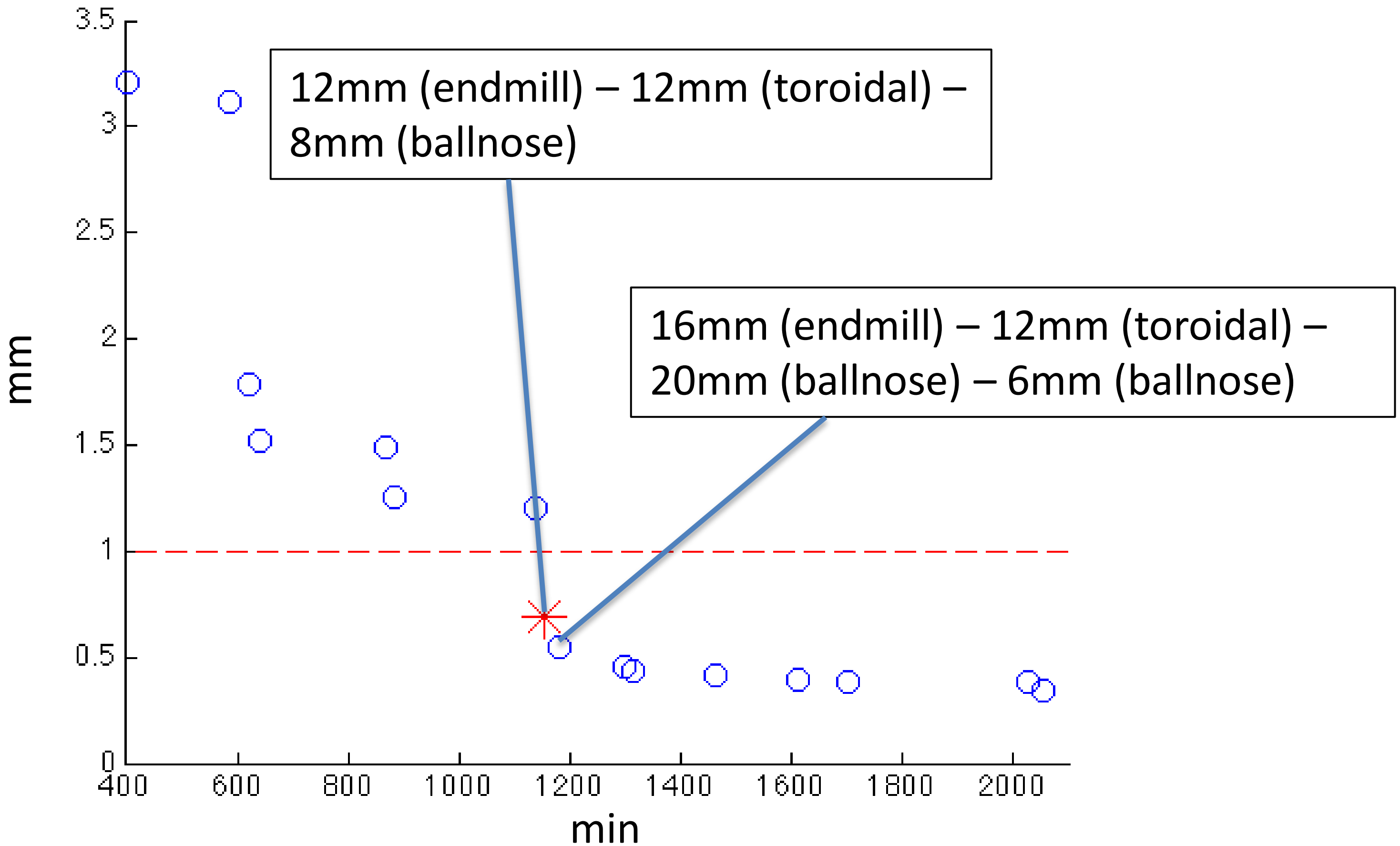
University of Sussex
CCNR Centre for Computational Neuroscience and Robotics


RP2
500
550
600
650
700
750
800
850
900
950
1000
1050
1100
1150
0.5
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9
0.95
1.0
1.05
250
240
230
220
210
200
190
180
170
160
150
Optimum
RP1

Conclusion

- In these experiments, multi-objective algorithms perform better than the single-objective ones but have more population size based variation
- Preferential search can lead to better single-objective search performance in the multi-objective algorithms
- Single-objective hybrid algorithm performs well
- Reference points have large differences in single-objective performance

Multi-Objective Tool Selection



Acknowledgements

- This work was supported by funding from EPSRC and Vero Software
- Very grateful for the assistance with simulations and clustering provided by Steve Youngs at Vero Software

EPSRC Engineering and Physical Sciences Research Council

vero Software

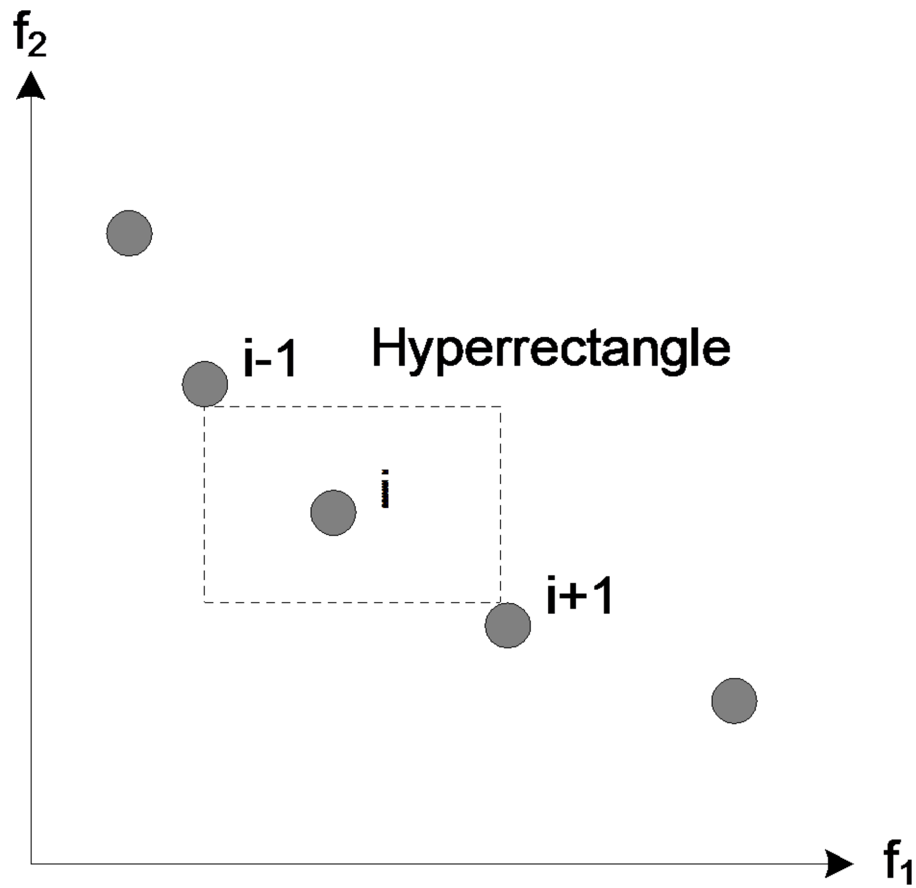
Thanks for listening!

Crowding Distance Modification

Crowding distance assignment

```
        set cd to 0
        for o in objectives:
                sort population by objective value
                best = worst = ∞
                for i = best + 1; i < worst; i++:
                        pop[i].cd += |pop[i-1].o –
pop[i+1].o|
```

Crowding Distance Modification



Crowding Distance Modification

- When a solution obtains the best score for one objective value and the worst for another, duplicates can be given an infinite crowding distance score and thus guaranteed survival in the next population
- This can be a problem when using very small population sizes

Crowding Distance Modification

id	F1(x)	F2(x)
1	1	10
2	1	10
3	3	5
4	5	1
5	5	1

Crowding Distance Modification

F1(x)

id	F1(x)	CD
1	1	0
2	1	0
3	3	0
4	5	0
5	5	0

Crowding Distance Modification

F1(x)

id	F1(x)	CD
1	1	∞
2	1	0
3	3	0
4	5	0
5	5	0

Crowding Distance Modification

<u>F1(x)</u>

id	F1(x)	CD
1	1	∞
2	1	2
3	3	0
4	5	0
5	5	0

Crowding Distance Modification

F1(x)

id	F1(x)	CD
1	1	∞
2	1	2
3	3	2
4	5	0
5	5	0

Crowding Distance Modification

F1(x)

id	F1(x)	CD
1	1	∞
2	1	2
3	3	2
4	5	4
5	5	0

Crowding Distance Modification

F1(x)

id	F1(x)	CD
1	1	∞
2	1	2
3	3	2
4	5	4
5	5	∞

Crowding Distance Modification

F2(x)

id	F2(x)	CD
4	1	4
5	1	∞
3	5	2
1	10	∞
2	10	2

Crowding Distance Modification

F2(x)

id	F2(x)	CD
4	1	∞
5	1	∞
3	5	2
1	10	∞
2	10	2

Crowding Distance Modification

F2(x)

id	F2(x)	CD
4	1	∞
5	1	∞
3	5	2
1	10	∞
2	10	2

Crowding Distance Modification

F2(x)

id	F2(x)	CD
4	1	∞
5	1	∞
3	5	2 + 4 = 6
1	10	∞
2	10	2

Crowding Distance Modification

F2(x)

id	F2(x)	CD
4	1	∞
5	1	∞
3	5	2 + 4 = 6
1	10	∞
2	10	2

Crowding Distance Modification

F2(x)

id	F2(x)	CD
4	1	∞
5	1	∞
3	5	2 + 4 = 6
1	10	∞
2	10	∞