A Hierarchical Evolutionary Approach to Multi-Objective Optimization

Christine L. Mumford

christine@cs.cardiff.ac.uk

Cardiff University

Introduction

- A hierarchical evolutionary approach to MO

Introduction

- A hierarchical evolutionary approach to MO
- Based on the *SEAMO* algorithm (a simple evolutionary algorithm for multi-objective optimization)

Introduction

- A hierarchical evolutionary approach to MO
- Based on the *SEAMO* algorithm (a simple evolutionary algorithm for multi-objective optimization)
- A better spread of solutions are obtained if subpopulations of various sizes are used

Introduction

- A hierarchical evolutionary approach to MO
- Based on the *SEAMO* algorithm (a simple evolutionary algorithm for multi-objective optimization)
- A better spread of solutions are obtained if subpopulations of various sizes are used
- Three alternative hierarchical models are tried and the results compared

Why Subpopulations?

Why Subpopulations?

Evolutionary algorithms (EAs) that utilize subpopulations are popular for two reasons:

Why Subpopulations?

Evolutionary algorithms (EAs) that utilize subpopulations are popular for two reasons:

- They adapt readily to parallel hardware, leading to faster execution

Why Subpopulations?

Evolutionary algorithms (EAs) that utilize subpopulations are popular for two reasons:

- They adapt readily to parallel hardware, leading to faster execution
- Isolated populations encourage genetic diversity and discourage premature convergence

Why Subpopulations?

Evolutionary algorithms (EAs) that utilize subpopulations are popular for two reasons:

- They adapt readily to parallel hardware, leading to faster execution
- Isolated populations encourage genetic diversity and discourage premature convergence

Premature convergence is a serious problem with EAs, and is encountered with single and multi-objective problems alike.

Why SEAMO?

Why SEAMO?

The SEAMO algorithms would appear to be particularly good candidates for hierarchical/parallel implementation for the following reasons:

Why SEAMO?

The SEAMO algorithms would appear to be particularly good candidates for hierarchical/parallel implementation for the following reasons:

- They perform well in comparison with other state-of-the-art multi-objective EAs

Why SEAMO?

The SEAMO algorithms would appear to be particularly good candidates for hierarchical/parallel implementation for the following reasons:

- They perform well in comparison with other state-of-the-art multi-objective EAs
- They are particularly simple to implement

Why SEAMO?

The SEAMO algorithms would appear to be particularly good candidates for hierarchical/parallel implementation for the following reasons:

- They perform well in comparison with other state-of-the-art multi-objective EAs
- They are particularly simple to implement
- No complex global calculations are required for fitness or dominance

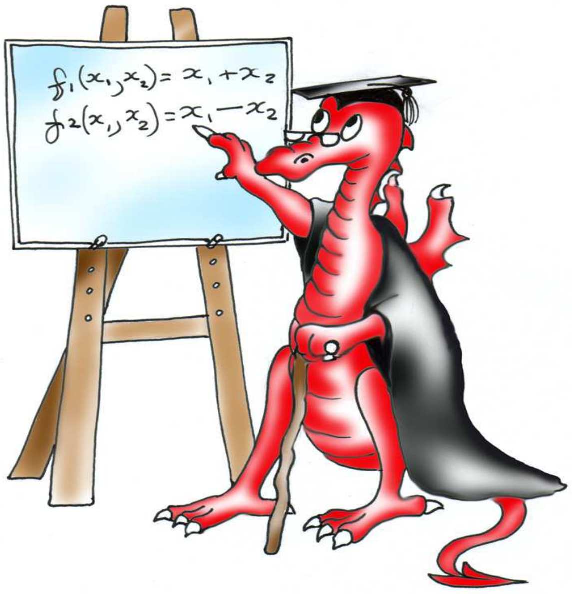
Test problems

Test problems

- Multiple knapsack problems (MKPs)

Test problems

- Multiple knapsack problems (MKPs)
- Continuous functions, SPH-2, ZDT6, QV and KUR



The SEAMO Framework

Procedure *SEAMO*
Begin
Generate N random individuals $\{N$ is the population size$\}$
Evaluate the objective vector for each population member and store it
Repeat
For each member of the population
This individual becomes the first parent
Select a second parent at random
Apply crossover to produce single offspring
Apply a single mutation to the offspring
Evaluate the objective vector produced by the offspring
if offspring qualifies
Then the offspring replaces a member of the population
else it dies
Endfor
Until stopping condition satisfied
Print all non-dominated solutions in the final population
End

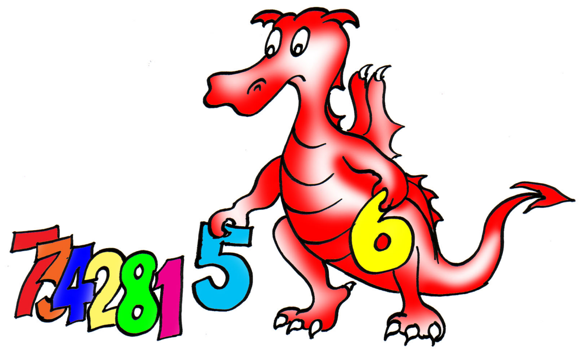
Replacement Strategy for SEAMO2

1. **if** offspring harbors a new best-so-far Pareto component
   (a) it replaces a parent, if possible
   (b) **else** it replaces another individual at random
2. **else if** offspring dominates either parent it replaces it
3. **else if** offspring is neither dominated by nor dominates either parent it replaces another individual that it dominates at random
4. **otherwise** it dies

Note: phenotypic duplicates are deleted

Representation for the MKP

Representation for the MKP

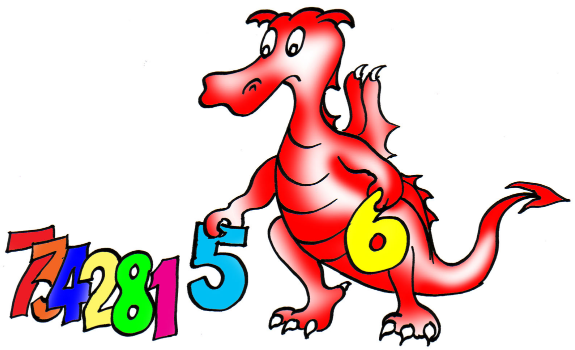


- Order-based representation with a first fit decoder

Representation for the MKP

- Order-based representation with a first fit decoder
- Cycle Crossover (CX)

Representation for the MKP



- Order-based representation with a first fit decoder
- Cycle Crossover (CX)
- A simple mutation operator swaps two arbitrarily selected objects within a single permutation list

The Continuous Functions

The Continuous Functions

- Solutions are coded as real vectors of length 100

The Continuous Functions

- Solutions are coded as real vectors of length 100
- One-point crossover

The Continuous Functions

- Solutions are coded as real vectors of length 100
- One-point crossover
- A non-uniform mutation

The Continuous Functions

- Solutions are coded as real vectors of length 100
- One-point crossover
- A non-uniform mutation
- Deletion of duplicates: component objective functions x_i and x'_i of $\mathbf{x}$ and $\mathbf{x}'$, are equal if and only if

  $x_i - \epsilon \leq x'_i \leq x_i + \epsilon$,

  where ϵ is an error term $(0.00001 \times x_i)$

The Effect of Population Size

The Effect of Population Size

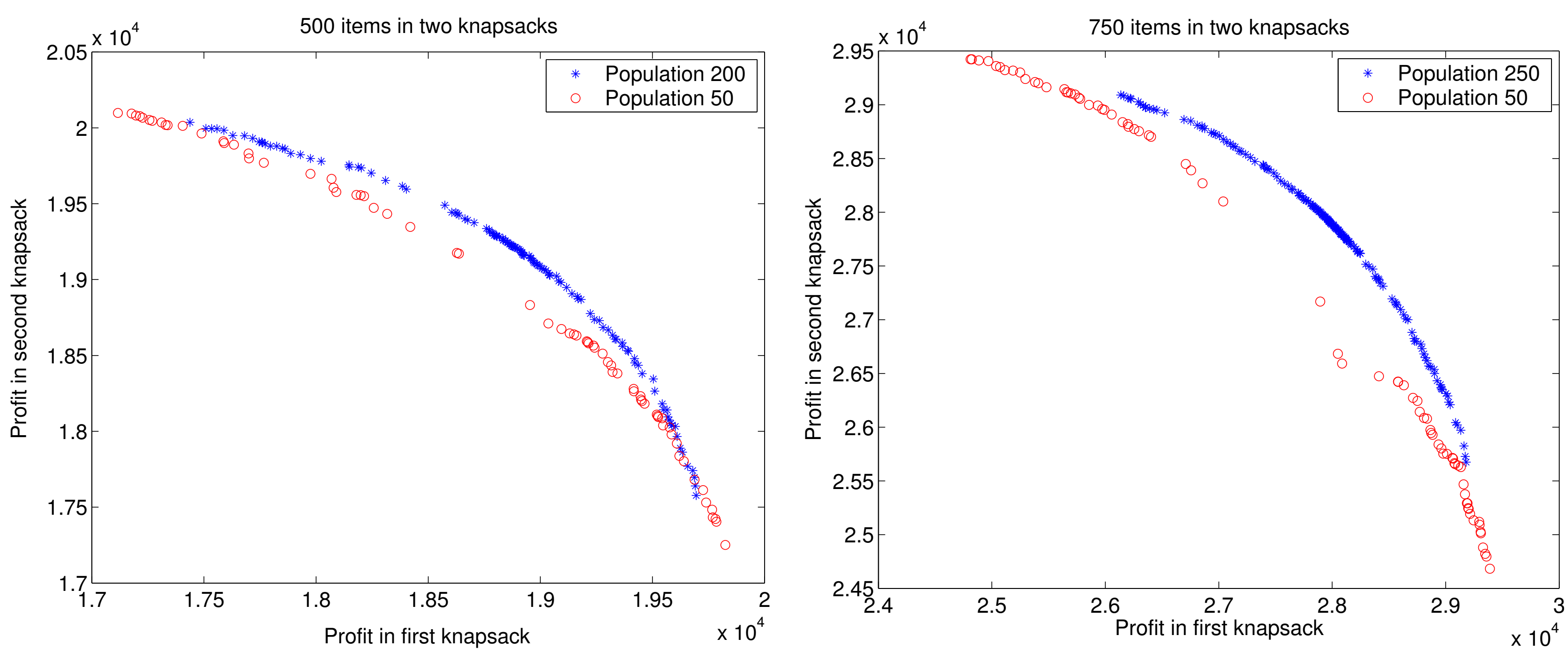
- Small populations gave a wider spread of results

The Effect of Population Size

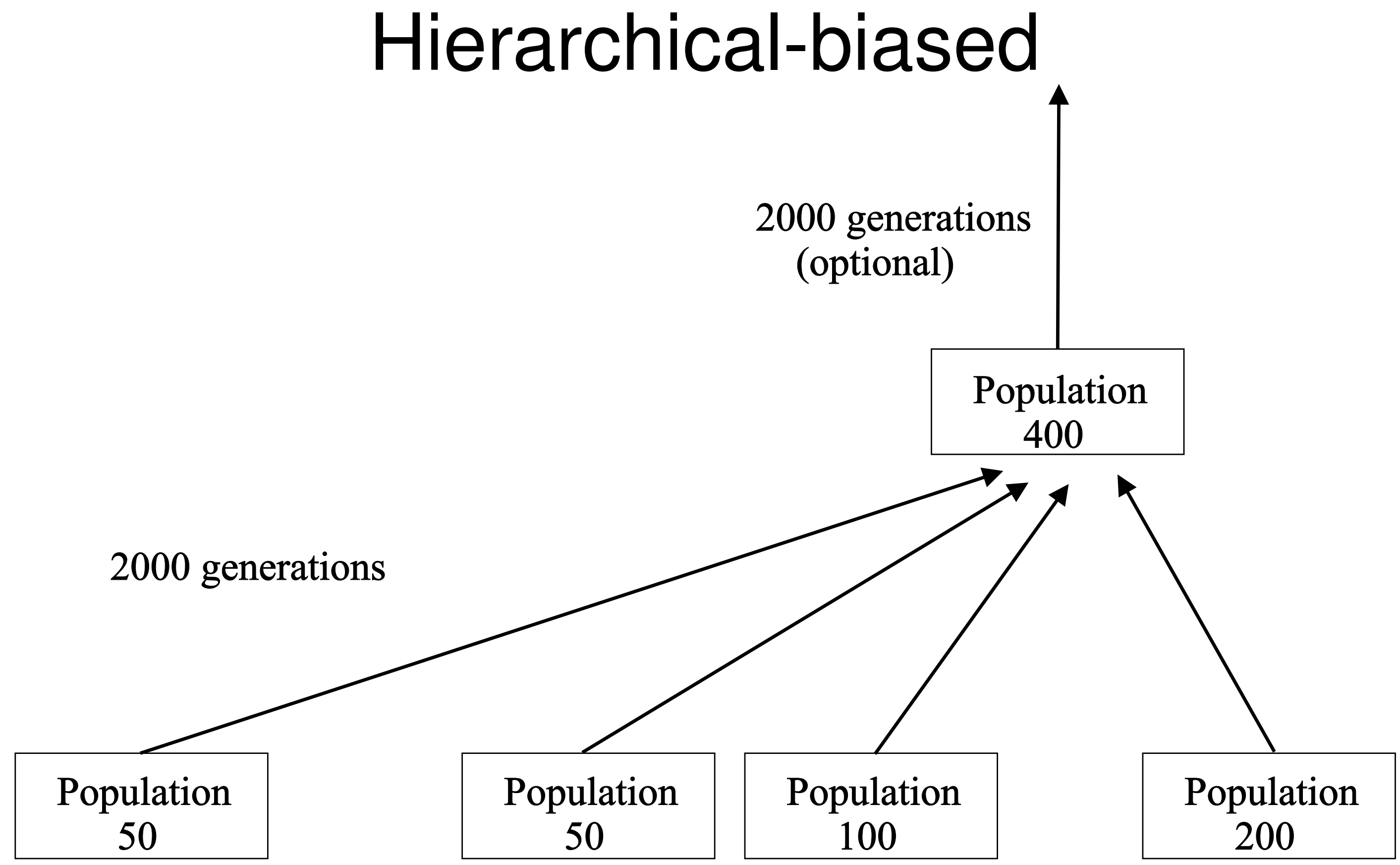
- Small populations gave a wider spread of results
- Large populations gave higher quality results in the center of the range

The Effect of Population Size

- Small populations gave a wider spread of results
- Large populations gave higher quality results in the center of the range

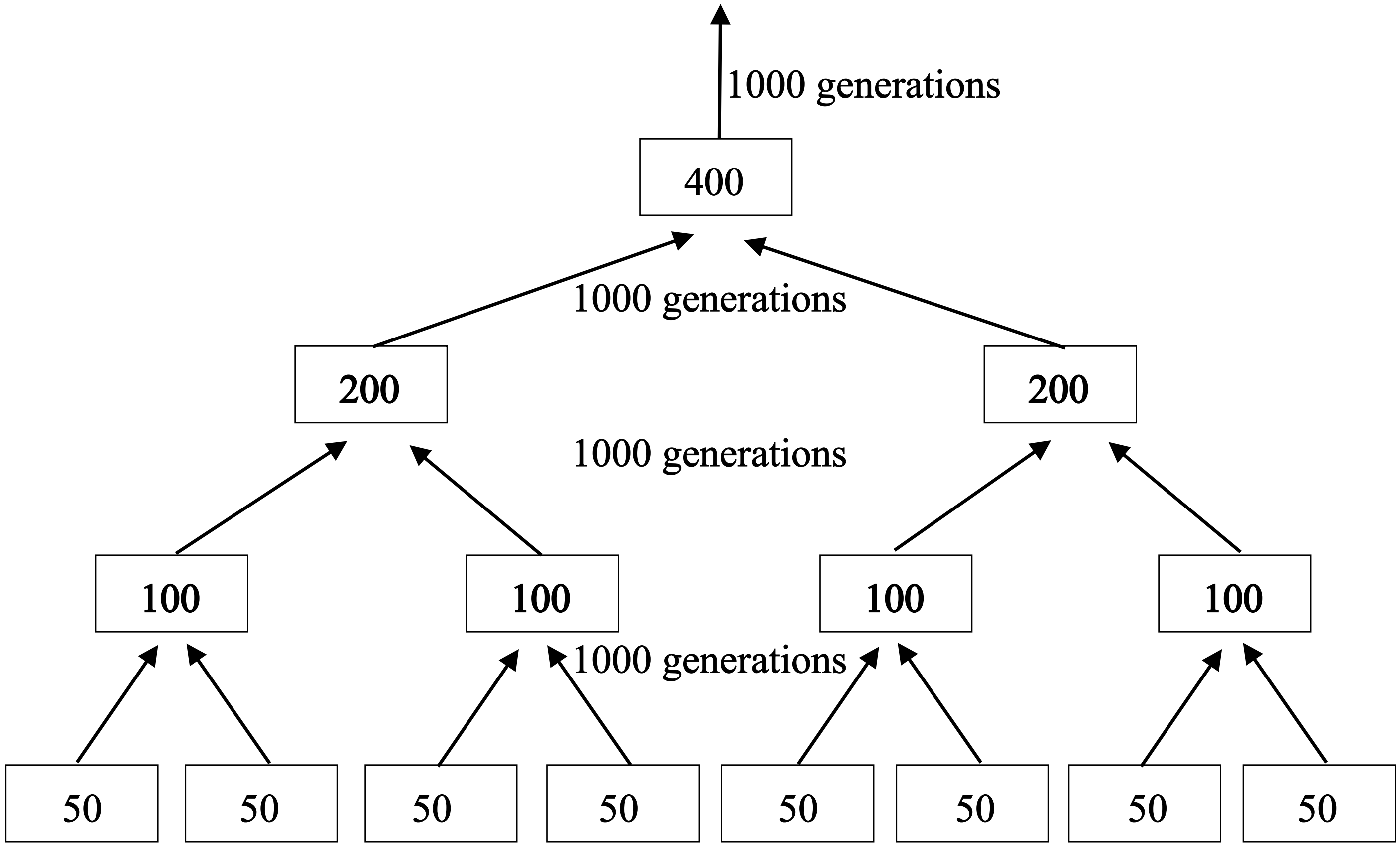


The Hierarchical Algorithms



The Hierarchical Algorithms

Hierarchical-balanced



Hierarchical Biased Algorithms

Procedure *Hierarchical-biased* ($population$)

Begin

if ($populationsize > threshold$)

split $population$ into $leftpop$ and $rightpop$

Hierarchical-biased ($leftpop$)

Run evolutionary algorithm on $rightpop$

else

Run evolutionary algorithm on (unsplit) $population$

End

The Hierarchical Balanced Algorithm

Procedure *Hierarchical-balanced* ($population$)

Begin

if ($populationsize > threshold$)

split population into $leftpop$ and $rightpop$

Hierarchical-balanced ($leftpop$)

Hierarchical-balanced ($rightpop$)

Recombine $leftpop$ and $rightpop$ into $population$

Run evolutionary algorithm on $population$

else

Run evolutionary algorithm on (unsplit) $population$

End

Hierarchical-Biased Algorithms

Hierarchical-Biased Algorithms

- *hierarchical-biased-2layer algorithm* (HBI2)

Hierarchical-Biased Algorithms

- *hierarchical-biased-2layer algorithm* (HBI2)
- *hierarchical-biased-flat algorithm* (HBIF)

Experimental Method

Experimental Method

- The three hierarchical algorithms are compared with the standard SEAMO2 algorithm on the the MKP

Experimental Method

- The three hierarchical algorithms are compared with the standard SEAMO2 algorithm on the the MKP
- Experiments are then extended to some continuous functions

Experimental Method

- The three hierarchical algorithms are compared with the standard SEAMO2 algorithm on the the MKP
- Experiments are then extended to some continuous functions
- 30 replicate runs carried out for each set of experiments, and all algorithms use the same total population size and number of generations

Comparisons with other EAs?

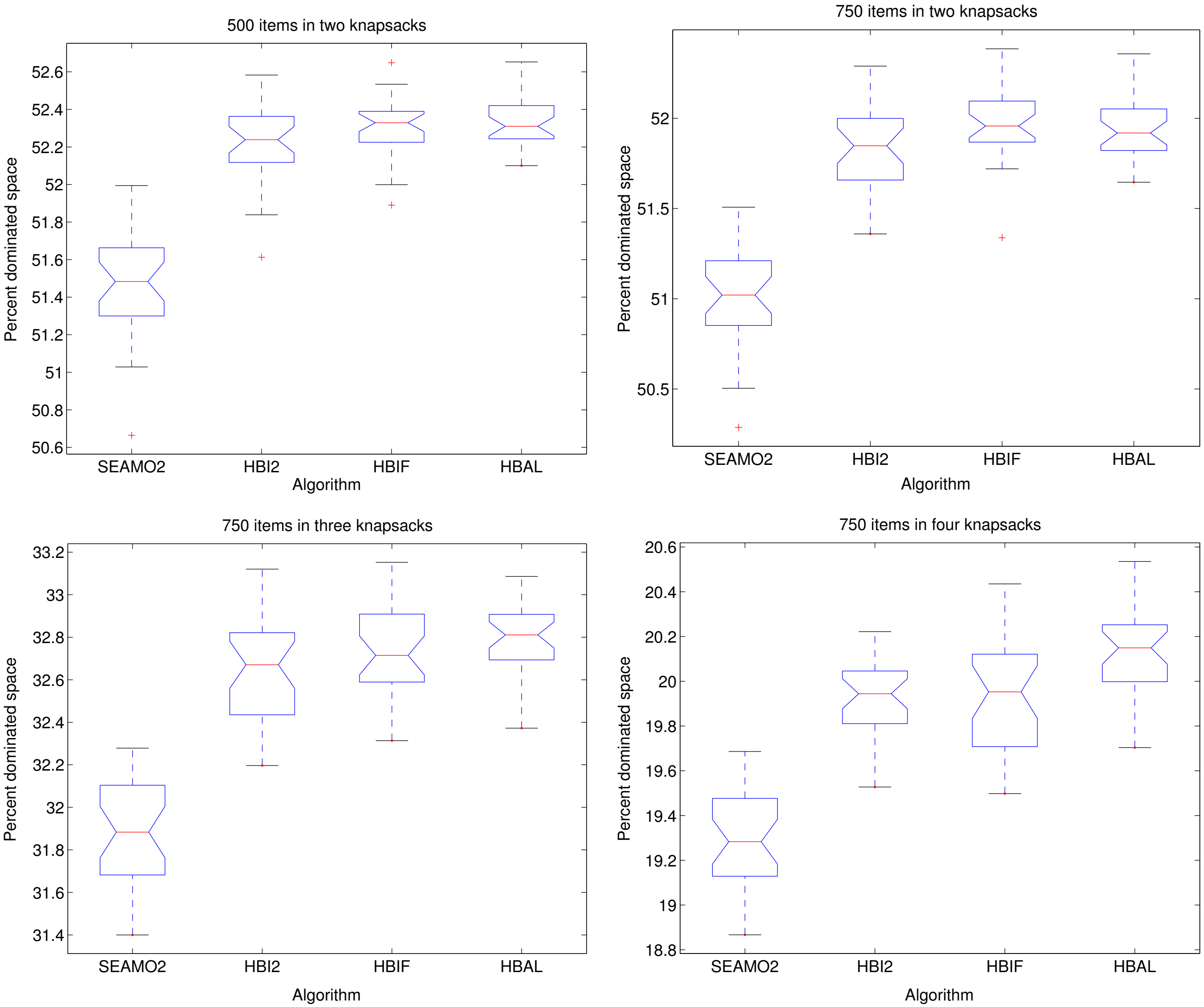
Comparisons with other EAs?

- The hierarchical algorithms are compared only with SEAMO2, and not with any other MOEAs

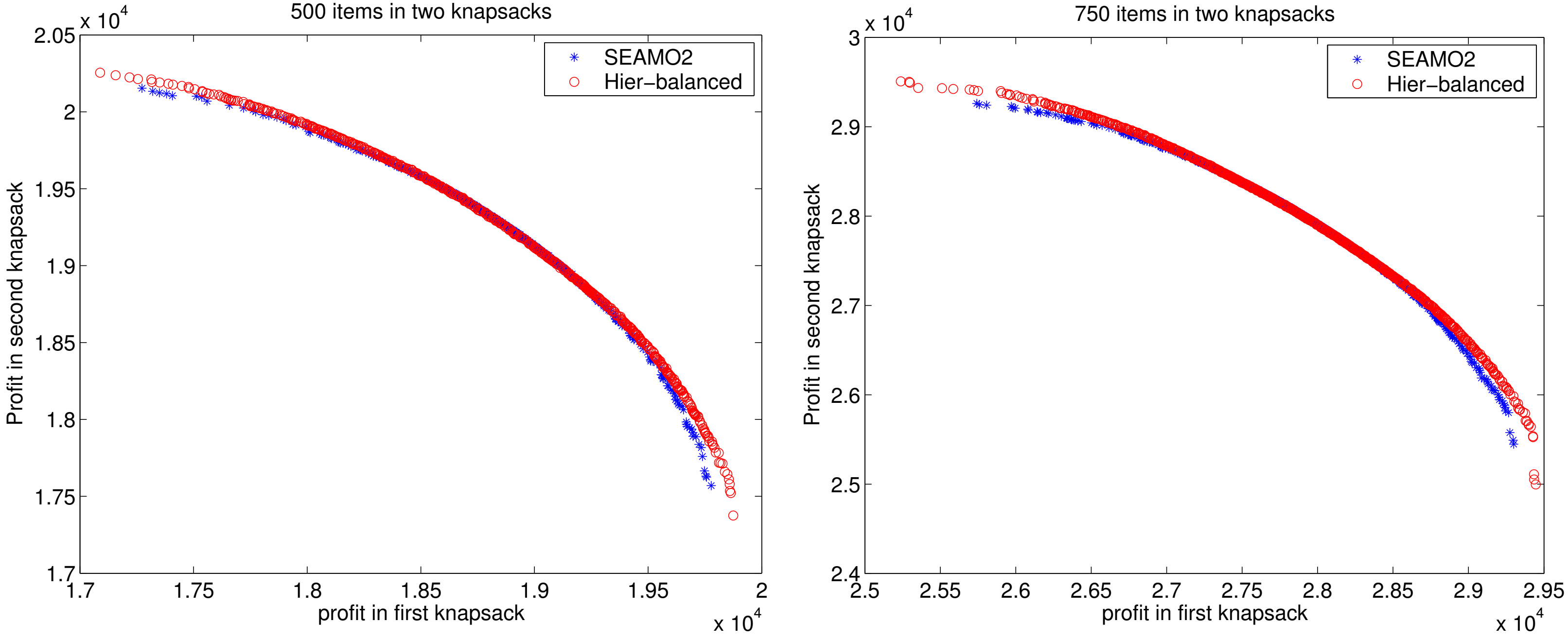
Comparisons with other EAs?

- The hierarchical algorithms are compared only with SEAMO2, and not with any other MOEAs
- SEAMO2 has demonstrated its strength in relation to other EAs elsewhere in a forthcoming GECCO 2004 paper

Results for the MKP, Dominate space, $\mathcal{S}$



Results for Multiple Knapsack Problems



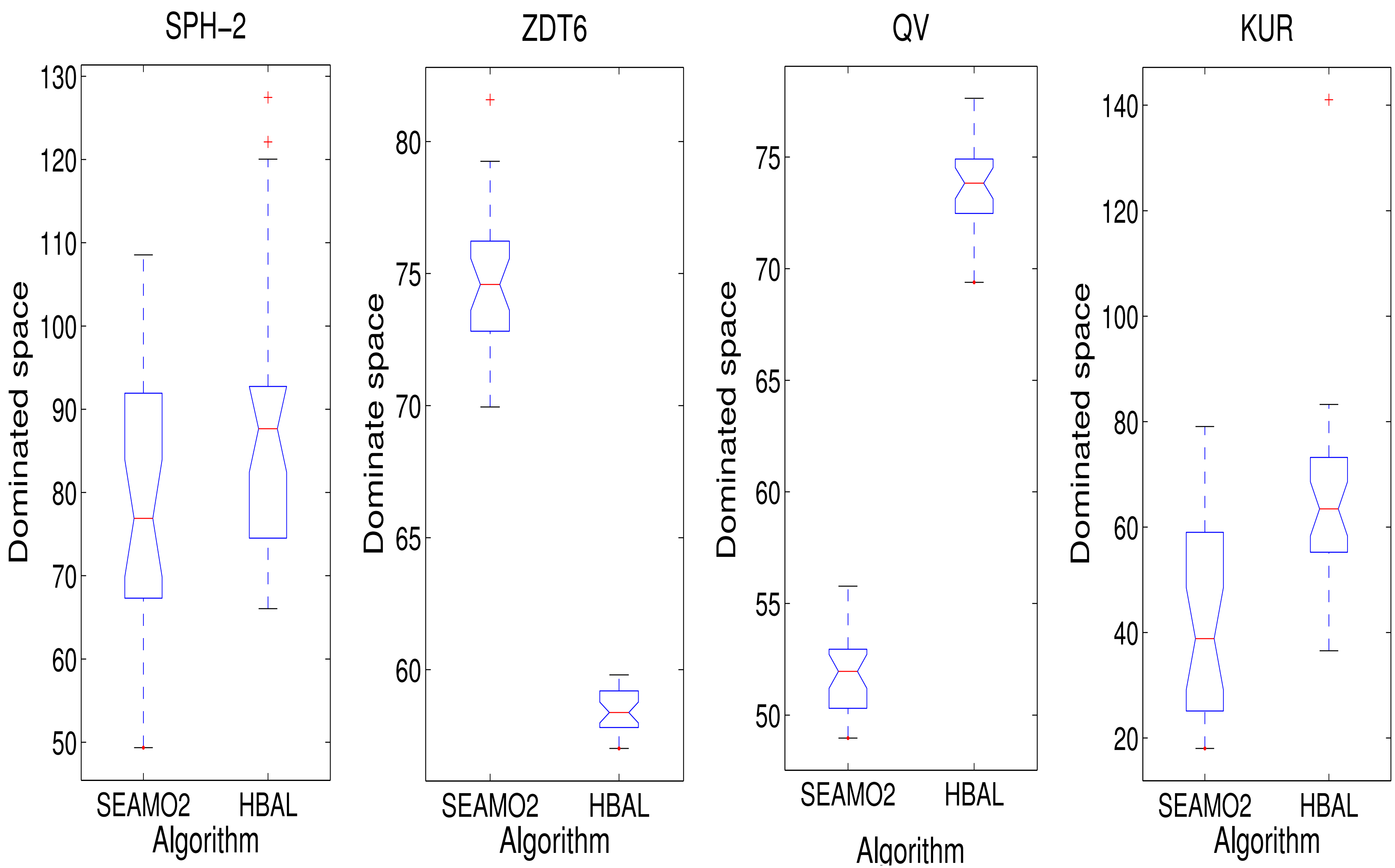
Comparing the performance of *SEAMO2* with

Hierarchical-balanced

Average Coverage, $(A \succeq B)$, on the MKP

Algorithm		Test problems			
A	B	kn500.2	kn750.2	kn750.3	kn750.4
SEAMO2	HBI2	37.6	50.1	46.4	46.9
	HBIF	64.9	70.8	61.4	60.1
	HBAL	22.2	32.8	19.1	21.9
HBI2	SEAMO2	25.4	15.0	5.2	4.4
	HBIF	75.4	68.1	50.1	37.2
	HBAL	20.0	14.8	4.0	6.8
HBIF	SEAMO2	5.8	2.8	0.8	1.1
	HBI2	7,7	14.9	7.3	4.9
	HBAL	5.4	6.1	1.5	2.3
HBAL	SEAMO2	28.9	21.8	12.8	6.4
	HBI2	54.7	72.0	44.4	25.2
	HBIF	72.4	77.1	60.6	50.8

Continuous Function Results, $\mathcal{S}$



Continuous Functions (cont)

Coverage $(A \succeq B)$					
Algorithm		Test problems			
A	B	SPH-2	ZDT6	QV	KUR
SEAMO2	HBAL	4.4	98.9	20.8	10.1
HBAL	SEAMO2	5.4	0	21.6	66.2

Conclusions

Conclusions

- Better solutions are achieved using large populations and long running times

Conclusions

- Better solutions are achieved using large populations and long running times
- Given large computational resources, how do we make best use of them?

Conclusions

- Better solutions are achieved using large populations and long running times
- Given large computational resources, how do we make best use of them?
- Do we use large single populations or utilize subpopulations?

Conclusions (cont)

Conclusions (cont)

- 3 Hierarchical algorithms based on SEAMO have been presented

Conclusions (cont)

- 3 Hierarchical algorithms based on SEAMO have been presented
- Incorporating runs on small and large populations

Conclusions (cont)

- 3 Hierarchical algorithms based on SEAMO have been presented
- Incorporating runs on small and large populations
- Improving the range of solutions, while maintaining their quality

Conclusions (cont)

- 3 Hierarchical algorithms based on SEAMO have been presented
- Incorporating runs on small and large populations
- Improving the range of solutions, while maintaining their quality
- The hierarchical balanced algorithm performed best

Future Work

Future Work

- Focus subpopulations on different regions of the Pareto space

Future Work

- Focus subpopulations on different regions of the Pareto space
- Try ternary, quadtree etc structures for the hierarchical balanced algorithm

Future Work

- Focus subpopulations on different regions of the Pareto space
- Try ternary, quadtree etc structures for the hierarchical balanced algorithm
- Implement a massively parallel version of SEAMO