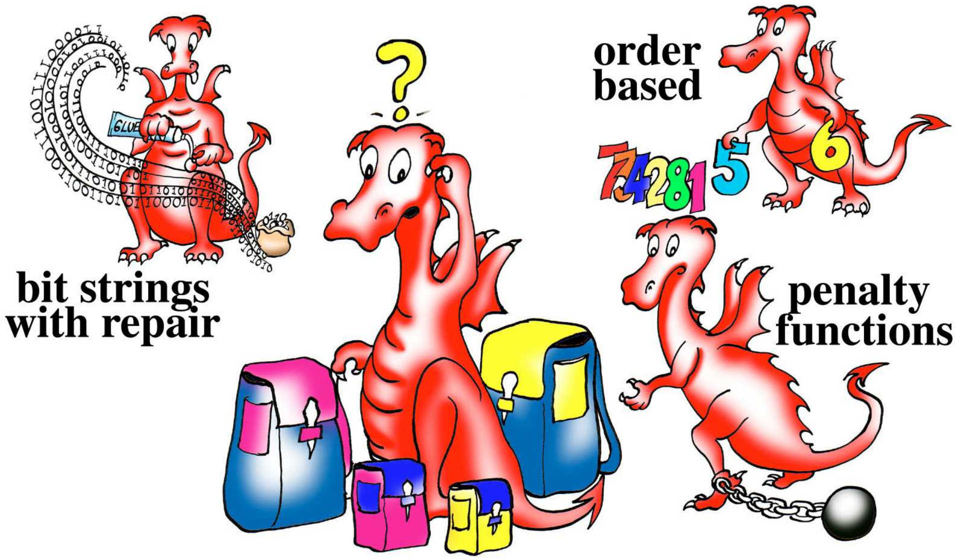
Comparing Representations and Recombination Operators for the Multi-Objective 0/1 Knapsack Problem

Christine L. Mumford
Cardiff University, Wales, U.K.
`C.L.Mumford@cs.cardiff.ac.uk`

November 28, 2003



Summary of Paper

- The Multiple Knapsack Problem (MKP) is a popular test-bed for multi-objective techniques
- This paper explores different representations and operators
- Adapted from the single objective knapsack case

The 0-1 Multiple Knapsack Problem

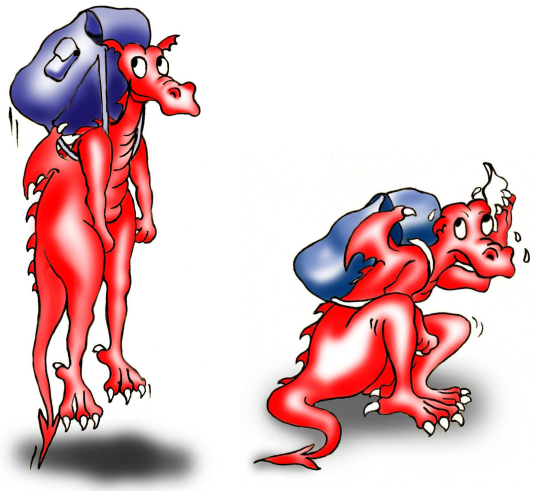
– a generalization of the 0-1 simple knapsack problem:

- A set of objects $O = \{o_1, o_2, o_3, ..., o_n\}$
- And a knapsack of capacity C are given.
- Each object o_i has an associated profit p_i and weight w_i.
- The objective is to find a subset $S \subseteq O$ such that the weight sum over the objects in S does not exceed the knapsack capacity
- And yields a maximum profit.

The 0-1 Multiple Knapsack Problem (MKP)

- The 0-1 MKP involves m knapsacks of capacities $c_1, c_2, c_3, \ldots, c_m$.
- Every selected object must be placed in all m knapsacks,
- Although neither the weight of an object o_i nor its profit is fixed,
- And will probably have different values in each knapsack.
- The present study is confined to problems involving two knapsacks, i.e. $m = 2$.

The same objects may have different weights in each knapsack



The same objects may have different profits in each knapsack

Representations for the Single Objective Problem

Robert Hinterding provides the following classification of representational techniques for the 0/1 knapsack problem:

- **Bit string representation** – where bit i is set if the i^{th} item from the list of items is included in the knapsack.
- **Numeric representation** – here the genes are numbers instead of bits, and a *decoder* is used to produce an ordering of the items.
- **Symbolic representation** – the genes represent the items themselves in a list. Usually consists of a *permutation* of all the items, using a heuristic to select items from the list to fill the knapsack.

Representations for the Multiple Objective Problem

For the MKP most researchers have used bit string representations with greedy repair.

This paper compares

- Bit string representations with penalty functions
- Bits strings with repair
- Order-based representations

The Bit String Representation with a Penalty Function

- A bit string vector, $\mathbf{x} = (x_1, x_2, x_3, ..., x_n)$,
- with $x_i = 1$ if item i is included in the knapsack(s), and $x_i = 0$, otherwise.
- Generating bits at random or using EAs, can easily produce over-full knapsacks
- The simplest way to deal with violated constraints is to apply a penalty function

Penalty Functions of Michalewicz and Arabus

The linear model adopted is defined below:

$$Pen_j(\mathbf{x}) = \rho_j.(\sum_{i=1}^{n} x_i.w_{ij} - C_j)$$

$$eval_j(\mathbf{x}) = \sum_{i=1}^{n} x_i.p_{ij} - Pen(x_j)$$

where $Pen_j(\mathbf{x})$ is the penalty applied to knapsack j, p_{ij} and w_{ij} represent the profit and weight respectively of the i^{th} item in the j^{th} knapsack, $\rho_j = max_{i=1..n}\{p_{ij}/w_{ij}\}$, C_j is the capacity of knapsack j, and $eval_j(\mathbf{x})$ is the adjusted value of the total profit in knapsack j.

Note: Penalty functions are applied only to knapsacks which are over capacity.

The Bit String Representation with Repair

- The repair mechanism begins with knapsacks already packed according to bits set,
- and sequentially removes items from the solution until no knapsack is over-filled

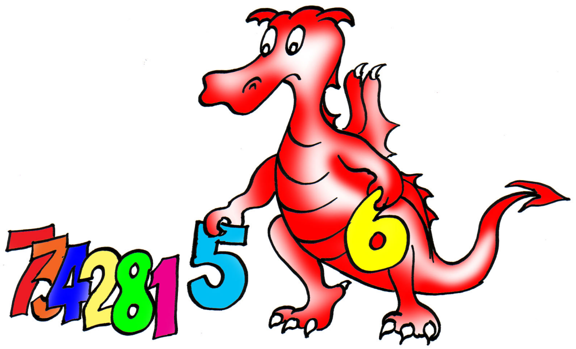
Two Variations of Repair Have Been Encoded

Both depend on sequential removal of objects with the least profitable, per unit weight, being deleted first.
However, the order in which the items are deleted is slightly different.
The two methods are:

1. the mechanism due to Zitzler and Thiele where items are deleted according to their *maximum* profit/weight ratio,
2. and an alternative mechanism where items are deleted according to their *average* profit/weight ratio.

One-point crossover and *point mutation* are used for all experiments using the bit string representation, for penalty as well as repair methods.

Order Based Representations



- Population of permutation lists
- e.g. {2,5,1,7,9,8,10,3,4,6}
- 2 is packed, then 5, then 1 etc.
- packing halts before knapsack capacity exceeded

A Sample Problem with Ten Objects and Two Knapsacks

Object number	Knapsack 1 Capacity = 38		Knapsack 2 Capacity = 35	
	Weight	Profit	Weight	Profit
1	9	2	3	3
2	8	7	4	9
3	2	4	2	1
4	7	5	4	5
5	3	6	9	3
6	6	2	5	8
7	1	7	4	2
8	3	3	8	6
9	9	7	3	1
10	3	1	7	3

Example Permutation

{2,5,1,7,9,8,10,3,4,6},

- Pack item 2, which weighs 8 units in knapsack 1 and 4 units in knapsack 2,
- Pack item 5 weighing 3 units in knapsack 1 and 9 units in knapsack 2, giving total weights of 11 and 13 for the two knapsacks.
- Carry on packing items from the permutation list until it had packed item 10, giving total weights of 36 and 39 for knapsacks 1 and 2 respectively.
- A weight of 39 units in knapsack 2 exceeds the capacity of the knapsack, and so item 10, is removed from both knapsacks
- weights are {33, 31}, profits are {32, 24}.

Two Decoders for Order Based Representation

1. Next Fit (NF)
2. First Fit (FF)

Next Fit Decoder

1. Items packed as permutation sequence
2. until a knapsack is overfull
3. then last to be added item is removed from all knapsacks

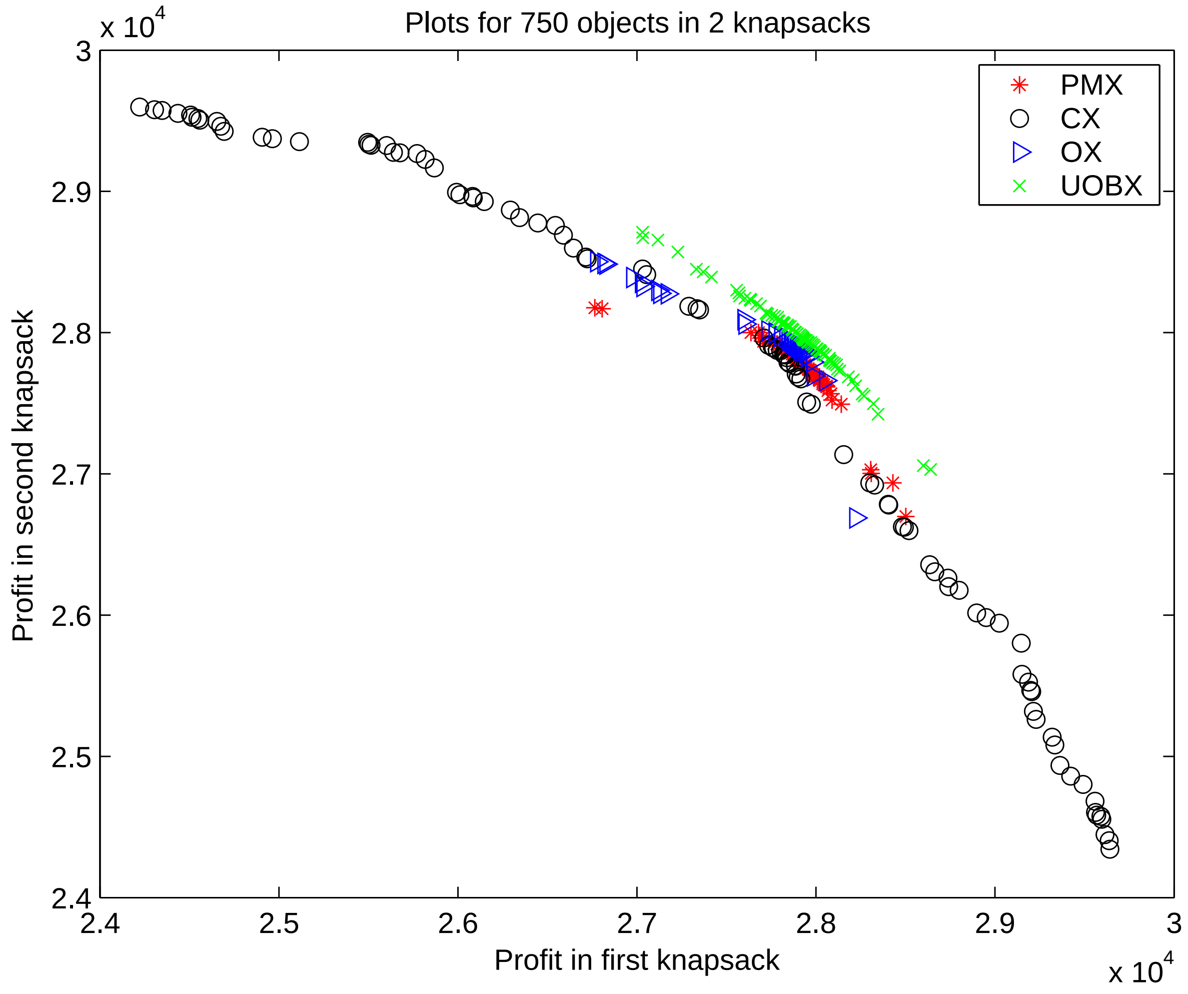
First Fit Decoder

1. Items packed as permutation sequence
2. until a knapsack is overfull
3. then last item to be added is removed from all knapsacks
4. process carries on to the end of the list
5. there may be room in the knapsacks for small items

Crossover and Mutation

- Cycle crossover CX
- the mutation swaps two arbitrarily selected objects within a single permutation list.

Non-dominated Solutions for Different Crossovers



Performance Measures Used

- The $\mathcal{C}$ Metric
- The $\mathcal{S}$ Metric
- 2D plots

The SEAMO Algorithm

- A very simple algorithm
- very few parameters to tune
- concentrate on representational issues

The Test Problems

- The four test problems of Zitzler and Thiele with two knapsacks are used here.
- With 100, 250, 500 or 750 objects.
- Restricting the test-bed to two knapsacks means that solutions can be plotted using standard 2D graphics, and their quality easily visualized.

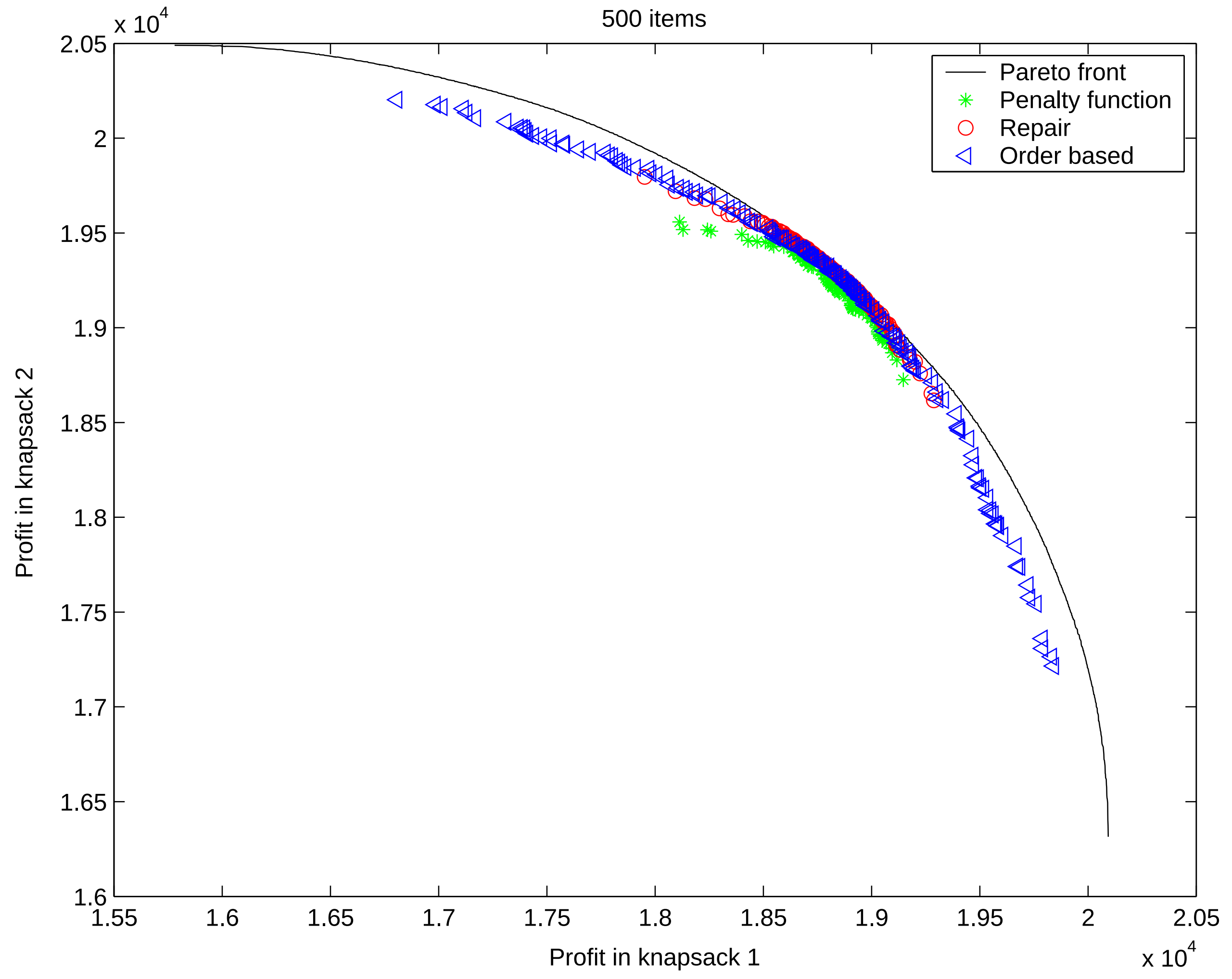
Experimental Method

- Populations of 250, for most experiments
- Long runs of 5,000 generations to ensure convergence
- 30 replicate runs for each experiment
- 2D plots used to visualize results

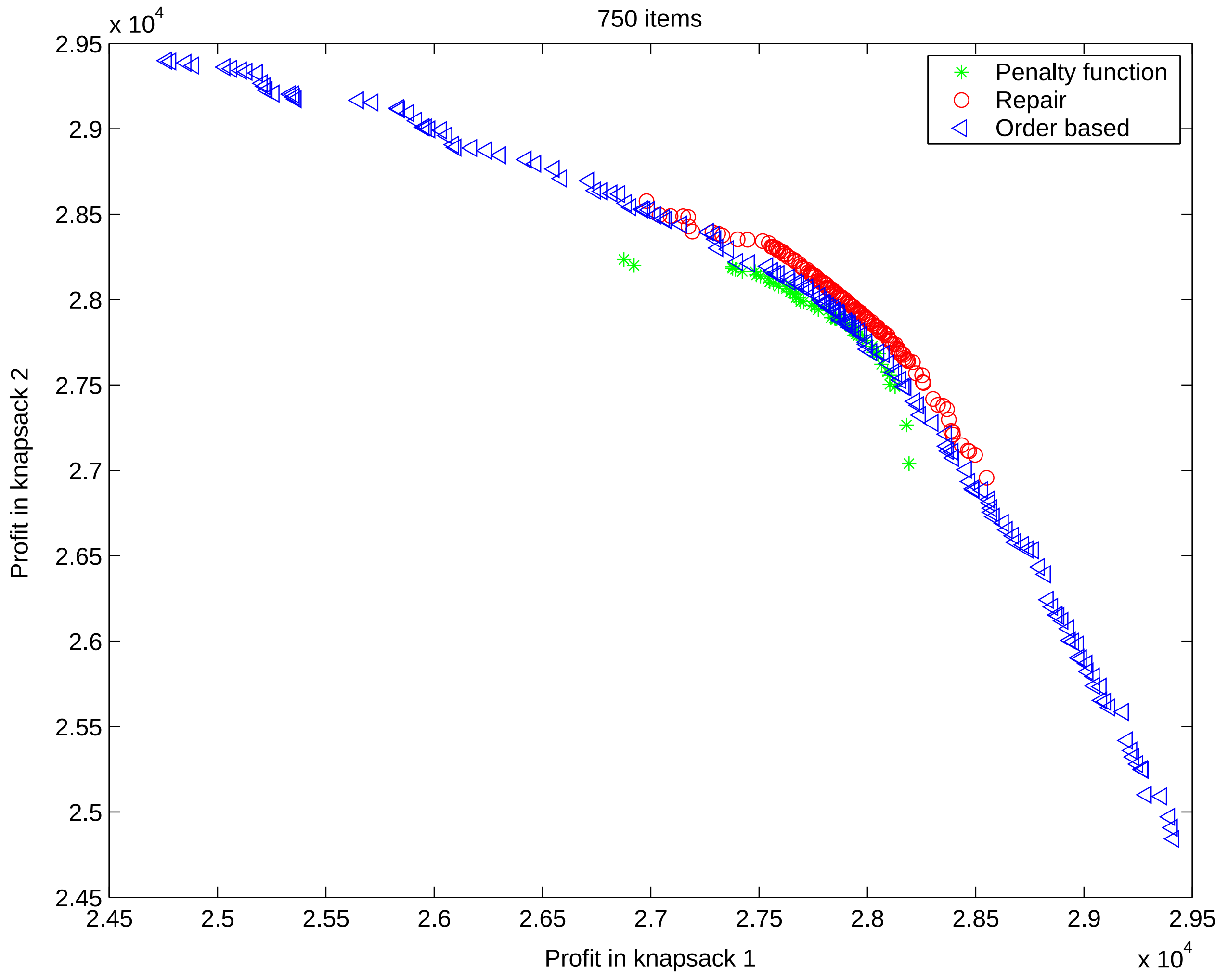
Summary of Results

- Order based representations give best results
- with first fit decoder better than next fit
- although choice of genetic operators play a vital role
- Penalty functions perform very poorly

2D Plots: 500 items



2D Plots: 750 items



Future Plans

- Try wider range of knapsack problems - e.g. more objectives
- Try the different approaches on other multi-objective EAs - not just SEAMO