A Meta-Greedy Approach for Multiobjectives Combinatorial Optimization

Louis-Claude CANON, Emmanuel JEANNOT

Project-Team AlGorille
Loria/INRIA/Université Henri Poincaré

ASTEC 2009, Les Plantiers, France

June 3, 2009

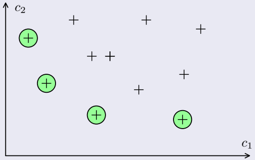
Outline

Outline

Motivation

Mutliobjective problem

Need to handle antagonist objectives.
Solutions can be incomparable (non-dominated).



Techniques

Fast method for generating a set of non-dominated solutions (possibly Pareto-optimal).
Existing method: multiobjective metaheuristics, espilon-method, Pareto set approximation (Papadimitriou, Yannakakis, 2000), ...

Beyond heuristics

General methodology

Method for solving multiobjective problems: takes a problem as input and produces a heuristic.
Similar to multiobjective metaheuristic and greedy strategy.
Restriction to the input problems: solutions can be constructed incrementally (as for greedy).

More precisely

Generalization of *greedy algorithms* when dealing with *multiple objectives*.

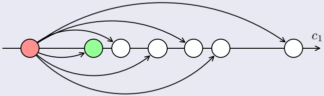
Related work

Framinan and Leisten (2007) considered a bicriteria scheduling problem (makespan and flowtime).

Multiobjective

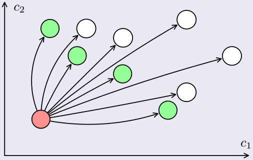
Classical greedy

Any following incremental modification to a partial solution is chosen according to one criterion c_1 (from red to green).



Considering multiple objectives

A set of non-dominated solutions is constructed at each step.

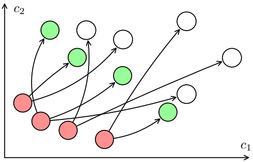


Main loop

Algorithm

Each incremented solution (red) is considered at the next iteration:

```
for each iteration
    for each solution in the population
        increment the solution in several ways
    keep the best generated partial solutions
```

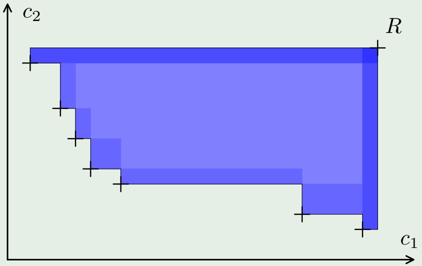


Set limitation size

Preference ordering

If too much generated solutions: selection among non-dominated solutions (active field of research).
Indicator-based proposition by Zitzler and Thiele (2009): keep a subset of solutions such that the indicator is maximized.
Parameters of the produced heuristics: indicator and maximum sizes.

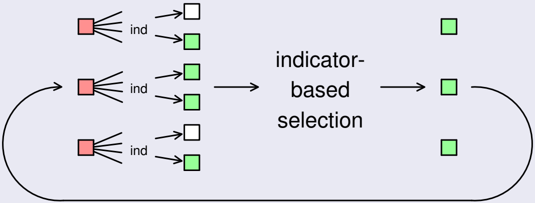
The hypervolume



Main loop revisited

Complete algorithm

Each new solution (red) is incremented in some ways (first limit on the number of build solutions)
Then, a subset of solutions are selected for next iteration.



Criteria specification

Required problem-specific specification

Similar to mutation and crossover operators for metaheuristic.
Since each partial solutions need to be evaluated, criteria for comparing partial solutions need to be defined.

Remark

Intermediate criteria $\neq$ final criteria.
How to compare partial solutions in a *fair* way (good intermediate criteria)?

Outline

Problem definition

Input

- a set of m elements $i \in [1..m]$
- each element has a profit p_i and a weight w_i
- a knapsack with a capacity C

Output

- m boolean decision variables x_i (if $x_i = 1$, element i is inserted)
- capacity constraint: $\sum_i x_i w_i \leq C$
- objective: maximize $\sum_i x_i p_i$

k-dimensional version

- each element has k profits p_{ij} and weights w_{ij} with $j \in [1..k]$
- the knapsack has k capacities C_j

Greedy heuristics

Greedy choice

At each step, a new element is added until no more element can be added.

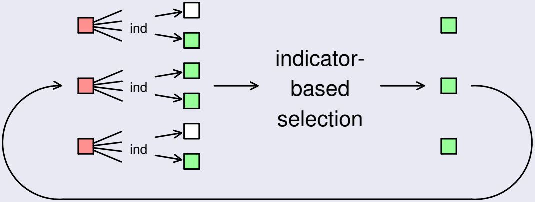
Criterion for selecting an element

- highest profit p_{ij}
- lowest weight w_{ij}
- highest ratio profit over weight $\frac{p_{ij}}{w_{ij}}$

Main loop revisited

Complete algorithm

Each new solution (red) is incremented in some ways (first limit on the number of build solutions)
Then, a subset of solutions are selected for next iteration.



Meta-greedy implementation

Preliminary observations

Each partial solution has the same number of elements.
Once a solution is complete, it is put in the final archive.

j-th intermediate criteria definition

- sum of assigned profits: $\sum x_{ij} p_{ij}$
- sum of assigned density: $\sum x_{ij} \frac{p_{ij}}{w_{ij}}$ or $\frac{\sum_i x_{ij} p_{ij}}{\sum_j x_{ij} w_{ij}}$
- ...

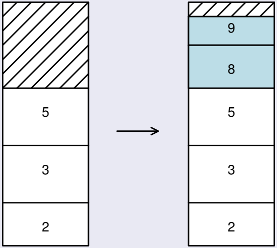
Meta-greedy implementation

Issue

Empty space is eluded.
What if there is enough space for inserting a good element?

Prediction-based criteria

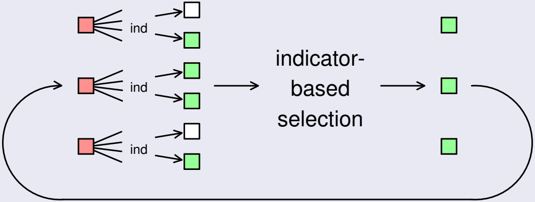
- complete with a mono-objective heuristic
- complete with an aggregation-based heuristic



Main loop revisited

Complete algorithm

Each new solution (red) is incremented in some ways (first limit on the number of build solutions)
Then, a subset of solutions are selected for next iteration.



Outline

Hypervolume indicator

Other multiobjective approaches

Aggregation Mono-objective heuristic with aggregated criteria.

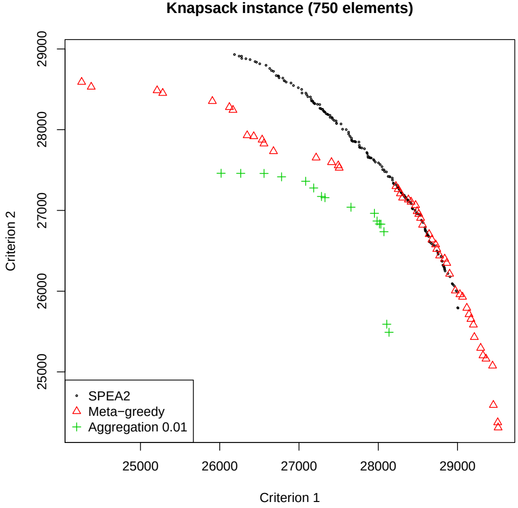
Optimal Exhaustive search (dynamic programming).

MOEA Multi-Objective Evolutionary Algorithm (NSGA-II, SPEA2, ...).

Test instances (Zitzler and Thiele, 1999)

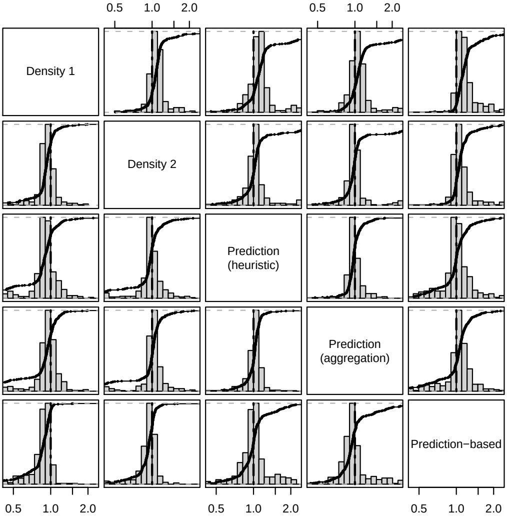
4 instances with 100, 250, 500 and 750 elements.
Population sizes: 2, 3, 5, 10, 20, 50, 100.

Specific case



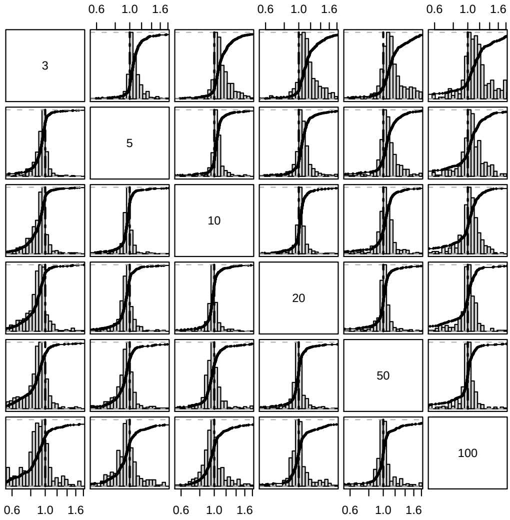
Intermediate criterion effect

Hypervolume of the column over the hypervolume of the row.



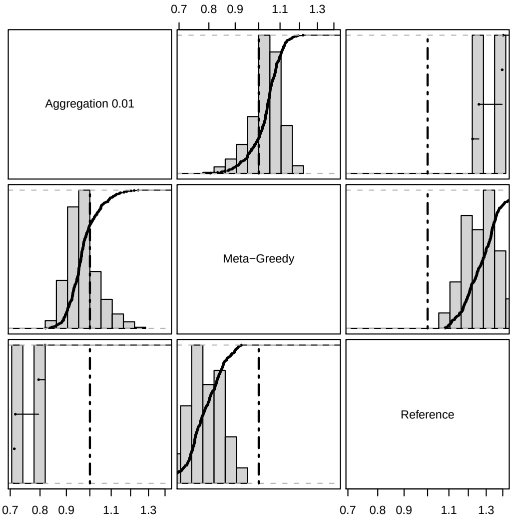
Population size effect

Hypervolume of the column over the hypervolume of the row.



Overall performance

Hypervolume of the column over the hypervolume of the row.



Outline

Conclusion and future directions

Main contributions

- Propose a generic approach (on the same level as greedy and metaheuristic designs) that can be applied to many problems.
- Assess its efficiency on a multiobjective combinatorial problem.
- Raise principal issue: intermediate criteria selection (comparing partial solutions).

Perspective

- Complete study of other scheduling problems.