A HyFlex Module for the 0-1 Knapsack Problem*

Steven Adriaensen, Gabriela Ochoa

March 11, 2015

1 Problem Formulation

Given a set of n items I, with associated weight $w : I \to \mathbb{R}$ and profit $p : I \to \mathbb{R}$ functions, and knapsack capacity V. Find the subset K that maximizes the total profit $\sum_{i \in K} p(i)$ and that satisfies the capacity constraint, i.e. $\sum_{i \in K} w(i) < V$. Here the search-space S is the subset of 2^I where each candidate solution satisfies the capacity constraint. The cost function $c : S \to \mathbb{R}$ maps each subset K to its negated total profit. This domain provides 10 benchmark instances, generated using the generator[1] used in [2]. Table 1 gives the optimal solution qualities (f_{opt}) as well as the parameters[2] used to generate each instance.

Table 1: Instances provided in the KP domain

index	**n**	**type (t)**	**seed (i)**	$f_{opt}*$
0	1K	no small weights (15)	12	104046
1	2K	uncorrelated (1)	13	1263861
2	2K	weakly corr. (2)	14	243145
3	2K	almost str. corr. (5)	17	431363
4	2K	no small weights (15)	23	396167
5	5K	uncorrelated (1)	24	4417737
6	5K	weakly corr. (2)	25	954172
7	5K	uncorr., similar weights (9)	32	1577175
8	5K	almost str. corr. (5)	28	1530536
9	5K	no small weights (15)	34	1467454

*This description is an extract from [1]

[1]available here `http://www.diku.dk/~pisinger/codes.html`

[2]the range of coefficients is $[1, r = 10\text{K}]$ and # tests is always 1K

2 Solution Initialisation

Constructs a candidate solution representing the empty set.

3 Low Level Heuristics

3.1 Local search heuristics

0. PACKRANDOM: Packs a randomly fitting item.
1. PACKBEST: Packs most valuable fitting item.
2. PACKPPP: Packs a randomly fitting item, improving the *Profit Per Pound* $\frac{\sum_{j \in K} p(j)}{\sum_{j \in K} w(j)}$ of the solution.
3. PACKLIGHT: Packs the lightest fitting item.
4. SWAPFIRST: Substitute a random packed piece by a random more expensive fitting piece.
5. SWAPBEST: Perform the substitution of 2 pieces, improving the total profit most.

3.2 Mutational heuristics

6. SWAPRANDOM: Randomly substitutes 2 pieces.
7. SWAPPPP: Randomly substitutes 2 pieces, improving the *Profit Per Pound* of the solution.
8. REMOVERANDOM: Unpacks a random piece.
9. REMOVEWORST: Unpacks the cheapest piece.
10. REMOVEHEAVY : Unpacks the heaviest piece.

The SWAP, REMOVE mutations are repeated $\lceil 5\alpha \rceil$ and $\lceil \alpha |K| \rceil$ times, respectively.

3.3 Ruin-Recreate heuristics

These heuristics first unpack $\lceil \alpha |K| \rceil$ randomly selected pieces, subsequently they pack pieces, till no more fit.

11. REPACKBEST: Each time packs the most valuable fitting piece.
12. REPACKPPP: Each time packs the fitting piece, improving the *Profit Per Pound* of the solution most.

3.4 Crossover heuristics

13. INTERSECTION: The output solution packs the items packed in both input solutions.

14. RANDOMUNION: Starting from the empty solution, each time packs a random fitting piece, packed in either of the input solutions.

15. BESTUNION: Starting from the empty solution, each time packs the fitting piece, packed in either of the input solutions, improving the total profit most.

References

[1] Steven Adriaensen, Gabriela Ochoa, and Ann Nowé. A benchmark set extension and comparative study for the hyflex framework. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 2015.

[2] Silvano Martello, David Pisinger, and Paolo Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999.