PROBE: a Metaheuristic for Hybridization
Extended Abstract

P. Chardaire and G. P. McKeown
School of Computing Sciences
UEA Norwich NR4 7TJ

1 Introduction

In this paper we illustrate the use of Population Reinforced Optimization Based Exploration (PROBE) as a framework for developing hybrid metaheuristic algorithms. We discuss three algorithms for solving the Multiconstraint Knapsack Problem (MKP) and two for the Graph Bisection Problem (GBP). Our first algorithm for the MKP combines PROBE with an approach based on a general heuristic, OCTANE (OCTAhedral NEighbourhood Enumeration), presented in [2], for solving 0-1 integer programs. The second algorithm for the MKP combines ideas from Greedy Randomized Adaptive Search Procedure (GRASP) with PROBE. Our final algorithm combines Greedy Randomized Adaptive Path Relinking (GRAPR) with PROBE. Our algorithms for the GBP also combine GRASP and GRAPR with PROBE. Path Relinking was introduced by Glover et al. [10] and was first used in conjunction with GRASP, as a form of intensification, by Laguna and Marti [12]. GRAPR was developed by Resende et al., see, for example, [14].

PROBE [4, 6] is a population based metaheuristic technique and directs optimization algorithms, towards good regions of the search space using some ideas from genetic algorithms (GAs). For each run of PROBE, an initial population is constructed and at each generation, their PROBE maintains a population of feasible solutions $S_g^i$, $i = 1, 2, \ldots, p$. Let $\sigma$ be a permutation function with domain $\{1, 2, \ldots, n\}$ ($\sigma$ may vary with $g$). Solution $S_{g+1}$ is computed from its parents $(S_{\sigma(i)}, S_{\sigma(i+1)})$ (with $S_{g(p+1)} = S_{g(1)}$) by the sequence of steps in figure 1.

Steps 1 and 2 allow a control of the size of the space searched. Too large a subspace may result in a prohibitive solution time, especially when an exact algorithm is used to search that subspace, while too small a subspace often leads to the return of a replica of the fittest parent. PROBE may use shrink and expand operations to avoid these situations. A shrink operation fixes additional bits according to the corresponding bit values of one of the parents (step 2). An expand operation frees some variables previously fixed (i.e. only some variables are fixed in step 1). There is much latitude in the ways of extending or shrinking subspaces. For example, variables may be selected randomly or according to intensification or diversification strategies. The size of the subspace searched may also be fixed. Steps 1 and 2 of PROBE guarantee that the subspace searched in step 3 contains a feasible solution if both parent solutions are feasible. The
1. Fix some of the bits that have the same value in \((S^g_{\sigma(i)} \text{ and } S^g_{\sigma(i+1)})\).
2. Fix some of the bits which do not have the same value in \((S^g_{\sigma(i)} \text{ and } S^g_{\sigma(i+1)})\) to the value they have in one of the two parent solutions.
3. Find an instantiation of the remaining bits using an appropriate search algorithm.
4. Use the instantiation obtained as a starting point for a local optimizer. The solution obtained is \(S^{g+1}\).

Figure 1: Generation of solutions in PROBE.

use of a permutation function \(\sigma\) when deriving a new generation allows for the possibility of shuffling the population between generations.

If the sub-spaces generated in these ways are sufficiently small, we could use an exact algorithm, for example Branch-and-Bound (B&B), to determine optimal solutions in these subspaces. A nice feature of PROBE is that when the selected subspace search algorithm is an exact algorithm, a version of PROBE can be implemented to guarantee that the average fitness of the solutions in the pool increases until all solutions have the same fitness [4].

Rather than using a general exact algorithm such as B&B for searching the sub-space, we could use a customized solver or some heuristic method.

2 Application of PROBE to the Multi-constraint Knapsack Problem

A maximization instance of the MKP is specified in Figure 2.

Each of the entries in the \(m \times n\) matrix \(R = [r_{ij}]\) is a non-negative integer, as is each entry in the \(m\)-component right-hand-side vector, \(b\). We assume that \(p_j > 0\), and \(r_{ij} \leq b_i < \sum_{j=1}^{n} r_{ij}\), for all \(1 \leq i \leq m, 1 \leq j \leq n\).

Given an instance of the MKP, each solution, \(S^g_i\), \(i = 1, \ldots, n\), at each generation, \(g\), is represented by a binary string which corresponds to a 0-1 solution, \(x\), to the given MKP. The fitness is simply the objective value, \(Z = px\).
associated with $S_i$. Any algorithm designed to solve the MKP is a candidate algorithm for sub-space search as the sub-problems to be solved are themselves smaller instances of the MKP.

We now give a brief description of a number of hybrid techniques instantiated within the PROBE framework for solving the MKP. Further details and some results will be given in the full paper.

In our first instantiation of PROBE for the MKP, we exploit the fact that the MKP has a 0-1 integer programming formulation, so that any of a number of general-purpose heuristics for 0-1 integer linear programming could be used for searching the sub-space. We have used an approach based on the OCTANE (OCTAgedral NEighbourhood Enumeration) heuristic, presented in[2]. One alternative would be to use the pivot and complement P&C heuristic of Balas and Martin [3]. OCTANE selects a fractional solution, $y$, to the linear programming relaxation of the 0-1 integer program and translates it to obtain the point $x = y - 1/2e$ where $e = (1, \ldots, 1)$. $x$ is in the interior of the regular octahedron, $K$, say, that circumscribes the unit hypercube centred at the origin. Each extreme point of the unit hypercube is the centre of one facet of $K$. From $x$, a ray is chosen that cuts the supporting hyperplanes of the facets of $K$. There is a one-one correspondence between the 0-1 points in the space and the facets of $K$, and so the points corresponding to the first $k$ facets intersected by the ray may serve as heuristic solutions to the problem instance. The structure of OCTANE is summarised in figure 3.

1. Solve the LP relaxation of the current sub-problem and translate the solution to get a fractional point, $x$.
2. Select an appropriate direction, $a$.
3. Determine the first $k$ facets intersecting the half-line
$$\Omega = \{x + \Lambda a | \Lambda \geq 0\}.$$ 
4. Take the corresponding 0-1 points as heuristic solutions.

Figure 3: Basic Structure of OCTANE.

An efficient algorithm for facet enumeration is presented in [2] based on the reverse search paradigm of Avis and Fukuda [1]. In our application of PROBE, we compute a fixed number of facets in the subspace to be searched. The fractional feasible point and ray used to initialize the computation in OCTANE are respectively the solution to the LP relaxation of the MKP associated with the subspace and the objective vector of that MKP. Each facet computed gives a 0-1 solution vector which when supplemented with the fixed variables provides a solution to the original problem. If necessary, a repair operator is used to make the solution feasible. After repairing the infeasible solutions, we apply a limited local optimization (step 4 in table 1) to all solutions. The scheme used is the same as the one used in [8] except for the order in which variables are considered.
We have performed a few experiments with the ordering based on the utility ratios of Pirkul’s method [9] and used in [8] but this ordering does not give any significant improvement over the one adopted. The best solution is then selected as a member of the next generation.

For our second hybrid metaheuristic for the MKP, the initial population, is constructed using the construct function defined for one of the GRASPs (Greedy Randomized Adaptive Search Procedure [13]) presented in [7]. This GRASP constructs solutions starting from the all-zero solution. Each child solution is generated by first fixing all of the bits in the child solution that do not have the same value in its two parents, $A$ and $B$ say, to the value they have in $A$. The construct function from another GRASP presented in [7] is used to complete the child solution. The latter construct function can start from any initial solution.

Our final hybrid metaheuristic for the MKP uses both GRASP and path relinking [10, 14]. The initial population for each run is constructed as in our second method but child solutions are now generated by applying path relinking to the parent solutions.

3 Application of PROBE to the Graph Bisection Problem

The graph bisection problem (GBP) may be specified formally as in figure 4.

A graph $G = (V, E)$ of $2n$ vertices with an associated edge cost matrix $C = (C_{ij})$, is to be bisected into two disjoint subgraphs, $G_A = (A, E_A)$, and $G_B = (B, E_B)$, such that $|A| = |B|$ and the cut-size is minimized. The cut-size, $T_{AB}$, is given by:

$$T_{AB} = \sum_{i \in A, j \in B} C_{ij}.$$  

Figure 4: The Graph Bisection Problem.

We represent a bisection by a bit string, with bit $i$ set to 0 if vertex $i$ is in one half of the bisection but set to 1 if vertex $i$ is in the other half of the bisection. We have instantiated two hybrid algorithms for the GBP. These are analogous to the second and third methods given for the MKP: they both use the construct function from a GRASP to construct the initial population. This function is based on the differential greedy algorithm [5]. Our first method then uses a modified form of Kernighan and Lin’s algorithm [11] to complete the child solutions in each generation. Our second method uses path relinking. Further details will be given in the full paper.
References


