A Hybrid Metaheuristic Based on Heuristic Problem Instance Reduction

Christian Blum[1,2]

[1] University of the Basque Country UPV/EHU, San Sebastian, Spain
[2] IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
christian.blum@ehu.es

1 Introduction

Hybrid metaheuristics [2, 6] are algorithms that combine components of different techniques for optimization. Examples are combinations of metaheuristics with dynamic programming, contraint programming, and branch & bound. In this work we present a hybrid metaheuristic which is based on an iterative probabilistic reduction of the problem instance size, with a subsequent application of an integer linear programming (ILP) solver in order to find the best solution contained in the reduced problem instance. A similar idea can be found in [1, 3] for the classical traveling salesman problem (TSP). The first algorithm phase consists in generating a bunch of high-quality TSP solutions using a metaheuristic. These solutions are then merged, and the TSP is solved to optimality in the resulting reduced graph. Another related idea is described in [4, 5] for cutting and packing problems. The authors introduce a framework labelled *Generate-and-Solve* which makes use of a metaheuristic engine to generate reduced problem instances, which are then solved to optimality by means of a complete technique.

2 Proposed Algorithm

In the following we assume that, given a problem instance $\mathcal{I}$, set C represents the set of all possible solution components. Moreover, given a (valid) solution s, set $C(s) \subseteq C$ represents the set of solution components that are contained in solution s. Finally, set $C' \subseteq C$ contains the solution components that belong to the restricted problem instance.

The main loop of the proposed algorithm, which is executed while the CPU time limit is not reached, consists of the following actions. First, the best-so-far solution s_{bsf} is initialized to NULL, and the restricted problem instance (C') to the empty set. Then, at each iteration a number of n_a solutions is probabilistically generated (see function ProbabilisticSolutionGeneration(C) in line 6 of Algorithm 1). The components of all these solutions are added to set C'. The age age[c]) of a newly added component is set to 0. After solution construction an ILP solver is applied to find the best solution s_{ilp} in the restricted problem instance C' (see function ApplyILPSolver(C') in line 12 of Algorithm 1). In case s_{ilp} is better than the current best-so-far solution s_{bsf}, solution s_{ilp} is stored as the new best-so-far solution (line 13). Next, the age values of the solution components are updated in function UpdateComponentAge(C', s_{ilp}) (see line 14), that is, the age of each solution component in C' is incremented, and, subsequently, the age of each solution component in $C(s_{\text{ilp}}) \subseteq C'$ is re-initialized to zero. Finally, those solution components from C' whose age has reached the maximum component age (age_{max}) are deleted from C' in function RemoveOldComponents($C', \text{age}_{\text{max}}$); see line 15. The motivation behind the ageing mechanism is that components which never appear in an optimal solution of C' should be removed from C' after some while, because they slow down the ILP solver. On the other side, components which appear in optimal solutions seem to be useful and should therefore remain in C'. This completes the description of the algorithm.

3 Results

The technique proposed in this paper is applied to two NP-hard combinatorial optimization problems. The so-called *minimum weight rooted arborescence* (MWRA) problem is a problem with applications in computer vision and multistage production planning, whereas the *minimum common string partition* (MCSP) problem has applications in computational biology. In the case of the

Algorithm 1 Hybrid Metaheuristic (HYBRID) Proposal

input: problem instance $\mathcal{I}$, values for parameters n_a and $\text{age}_{\max}$
$s_{\text{bsf}} :=$ NULL, $C' := \emptyset$
age$[c] := 0$ for all $c \in C$
while CPU time limit not reached **do**
  for $i = 1, \ldots, n_a$ **do**
    $s :=$ ProbabilisticSolutionGeneration(C)
    for all $c \in C(s)$ **and** $c \notin C'$ **do**
      age$[c] := 0$
      $C' := C' \cup \{c\}$
    end for
  end for
  $s_{\text{ilp}} :=$ ApplyILPSolver(C')
  if s_{ilp} is better than s_{bsf} **then** $s_{\text{bsf}} := s_{\text{ilp}}$
  UpdateComponentAge(C', s_{ilp})
  RemoveOldComponents$(C', \text{age}_{\max})$
end while
output: s_{bsf}

MWRA problem, problem instances consist of directed graphs (digraphs) with a designated root node and real-valued weights on the arcs. Solutions are (not necessarily spanning) arborescences rooted in the root node. The objective function value of an arborescence is the sum of the weights of its arcs. The optimization objective concerns minimization. We applied the proposed algorithm to 270 acyclic digraphs and to 270 digraphs that possibly contains directed cylces. Moreover, the results of our approach were compared to the results obtained by applying the ILP solver to the original problems, and to the results obtained by an ant colony optimization approach and a heuristic algorithm, both from the related literature. They show, first, that our approach outperforms the ant colony optimization approach and the heuristic. Moreover, they show that our approach has important advantages over the direct application of the ILP solver especially when the problem instance size grows. In the case of the MCSP problem, a problem instance consists of two (related) strings of the same size. Note that two strings are related if each letter appears the same number of times in each string. The optimization objective consists in finding a minimum size partition of each input string such that the two partitions are equal. The proposed approach was applied to 45 problem instances from the literature, and the results were compared to a simple greedy algorithm and to an ant colony optimization approach from the liteature, which is the current state of the art. Our approach was able to improve over the results of the ant colony optimization algorithm in all 45 cases.

Acknowledgements. This work was supported by grant TIN2012-37930-02 of the Spanish Government. In addition, support is acknowledged from IKERBASQUE (Basque Foundation for Science).

References

1. D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding tours in the TSP. Technical report, Forschungsinstitut für Diskrete Mathematik, University of Bonn, Germany, 1999.
2. C. Blum, J. Puchinger, G. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.
3. W. Cook and P. Seymour. Tour merging via branch-decomposition. *INFORMS Journal on Computing*, 15(3):233–248, 2003.
4. N. Nepomuceno, P. Pinheiro, and A. L. V. Coelho. *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, volume 153 of *Studies in Computational Intelligence*, chapter A Hybrid Optimization Framework for Cutting and Packing Problems, pages 87–99. Springer Verlag, Berlin, Germany, 2008.
5. P. R. Pinheiro, A. L. V. Coelho, A. B. de Aguiar, and T. O. Bonates. On the concept of density control and its application to a hybrid optimization framework: Investigation into cutting problems. *Computers & Industrial Engineering*, 61(3):463–472, 2011.
6. E.-G. Talbi, editor. *Hybrid Metaheuristics.* Number 434 in Studies in Computational Intelligence. Springer Verlag, Berlin, Germany, 2013.