Automatic Design of Heuristic Algorithms for Binary Optimization Problems

Marcelo de Souza
Federal University of Rio Grande do Sul
Santa Catarina State University
marcelo.desouza@udesc.br

Abstract

In this work we present AutoBQP, a heuristic solver for binary optimization problems. It applies automatic algorithm design techniques to search for the best heuristics for a given optimization problem. Experiments show that the solver can find algorithms which perform better than or comparable to state-of-the-art methods, and can even find new best solutions for some instances of standard benchmark sets.

1 Introduction

The manual design of heuristic algorithms for optimization problems is time-consuming and often biased. First, we choose a heuristic method and evaluate it experimentally to determine the best algorithmic components and parameter values. Then, we iteratively introduce new problem-specific components and tune their parameters in order to improve performance. In addition to the time spent in this process, the design space is often not explored systematically, and good heuristic strategies may be overlooked.

To minimize these problems, we propose a heuristic solver for binary optimization problems that automatically determines the best algorithm and its parameter values. We define a search space of heuristic components and their input parameters, and then apply automatic algorithm configuration techniques to search for the best algorithm for a given problem. The flexibility of this approach allows the solver to generate hybrid heuristics of very different types for the specific problem at hand.

Previous studies use similar approaches to automatically generate algorithms for different optimization problems [Stützle and López-Ibáñez, 2019]. Despite the good results reported, all of them focus on specific problems. On the other hand, we propose a generic solver to deal with a wide range class of optimization problems: those that represent the solution by a binary string. This increases the applicability of the solver and makes it an initial step for researchers interested in binary optimization.

2 Proposed Approach

The general idea behind AutoBQP is depicted in Figure 1. In order to generate algorithms for a new problem, we need

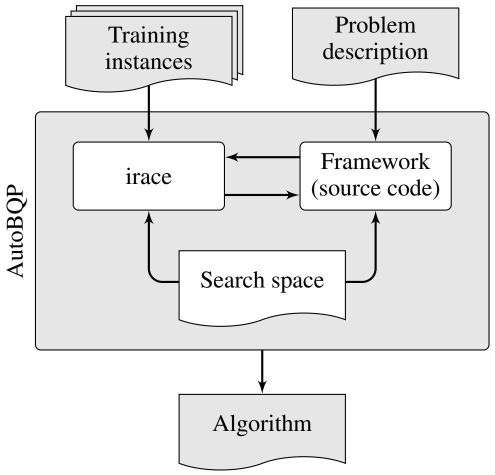

Figure 1: Structure of the AutoBQP solver.

to provide a problem description, i.e. implementations for instance reading and objective function, and also a set of training instances. AutoBQP implements a framework with several heuristic components and their input parameters, which define the search space of algorithms. We use irace [López-Ibáñez *et al.*, 2016] to explore this search space, producing the algorithm that optimizes the performance on the given training instances.

The framework component is based on the unconstrained Binary Quadratic Programming (BQP). Given a matrix $Q = (q_{ij}) \in \mathbb{R}^{n \times n}$ BQP asks to

$$\begin{array}{ll} \textbf{maximize} & x^t Q x, \\ \textbf{subject to} & x \in \{0,1\}^n. \end{array}$$

We focus on BQP because several optimization problems can be reduced to it. We extracted the heuristic components from the state-of-the-art algorithms of BQP to build our framework: the iterated tabu search (ITS) of Palubeckis [2006], the diversity-driven tabu search (D^2TS) of Glover et al. [2010], and the path relinking recombination strategies (PR1 and PR2) of Wang et al. [2012]. We also included common heuristic components, like local search strategies (e.g. hill climbing) and constructive methods (e.g. GRASP). The selected components allow our solver to generate more than 3000 different algorithms.

Instance set	ITS	D^2TS	PR1	PR2	AAC_P	AAC_M	AAC_R	TS	AAC_T^1	AAC_T^2
BEASLEY	2.0	**0.0**	1.34	5.84	**0.0**	34.0	**0.0**	-	-	-
PALUBECKIS	1109.6	2082.9	457.1	690.4	**186.8**	2424.3	211.7	-	-	-
MAXCUT	-	-	5.6	4.7	25.0	**4.4**	20.3	-	-	-
TESTASSIGNMENT	-	-	-	-	-	-	-	0.94	0.23	**0.18**

Table 1: Average absolute gap on the different instance sets. The best values for each instance set are shown in bold.

3 Experimental Results

We analyze the ability of AutoBQP to generate good heuristic algorithms for different problems. Table 1 summarizes the results of our experiments on different instance sets, presenting the average absolute gap from the best known solutions of running each algorithm 20 times. We compare the algorithms produced by AutoBQP (named AAC) to those from the literature of each problem considered.

Our first experiment consists in executing AutoBQP on a subset of the PALUBECKIS instances, producing algorithm AAC_P. We observe that this algorithm reaches the best possible performance on the BEASLEY instances, as well as the D^2TS algorithm. For the PALUBECKIS instances, AAC_P performs better than all other approaches. Our second experiment considers solving the MaxCut problem. Algorithm AAC_M is produced by running AutoBQP using a subset of the MAXCUT instances. We observe that AAC_M is worse than the rest on the BQP instances, but presents a slightly better performance on the MAXCUT instances, in comparison to the other algorithms. Algorithm AAC_M also found new best solutions for 7 out of 54 instances (namely G25, G26, G27, G28, G30, G31, and G38).

In order to find a good algorithm for BEASLEY and PALUBECKIS instance sets, which also performs good for the MAXCUT instance set, we executed AutoBQP using a set of randomly generated training instances. They follow the structure of PALUBECKIS instances, but were not selected from their original distribution. The resulting AAC_R algorithm still performs better than the state-of-the-art approaches for BEASLEY and PALUBECKIS instances, and better than AAC_P on the MAXCUT instances, although algorithms PR1 and PR2 present smaller gaps on MAXCUT.

Finally, our last experiment applies AutoBQP to solve the test-assignment problem [Duives *et al.*, 2013]. Given a set of desks in a classroom and a set of test variants, the test-assignment problem asks to assign tests to desks, in order to minimize the chance of copies between students. Each pair of desks has a physical proximity, and each pair of test variants has a similarity. The chance of copies is given by the product of proximity and similarity. This problem is a generalization of the vertex coloring and can be modeled as an unconstrained a binary quadratic problem using a penalization strategy. Duives et al. [2013] present a tabu search (TS) algorithm for solving the problem and provide a set of benchmark instances (TESTASSIGNMENT instance set).

After reducing the test-assignment problem to BQP, we applied AutoBQP using a subset of the TESTASSIGNMENT instances for training. Two algorithms were produced by AutoBQP with similar performance, namely AAC_T^1 and AAC_T^2. As we can see in Table 1, both algorithms perform better than the TS algorithm. AAC_T^1 and AAC_T^2 also found new best solutions for 8 out of 36 instances (with the following number of desks, empty desks and tests: $\{47, 10, 3\}$, $\{47, 0, 4\}$, $\{47, 10, 4\}$, $\{79, 0, 3\}$, $\{79, 10, 3\}$, $\{79, 20, 3\}$, $\{79, 0, 4\}$, and $\{79, 10, 4\}$).

4 Concluding Remarks

We present a solver for binary optimization problems based on automatic algorithm design techniques. It defines a search space of components and their input parameters, and then uses irace to search for the best combination of components and parameter values. Our experiments show that the proposed solver is able to produce algorithms competitive to state-of-the-art approaches for different problems that can be modeled as binary optimization. The proposed solver reduces the human effort in developing heuristic algorithms and provides a baseline method for comparison with other approaches.

As future work, we plan to extend the experiments to other problem scenarios. We also aim at producing algorithms better than state-of-the-art approaches by introducing new algorithmic components in the solver's framework.

References

[Duives *et al.*, 2013] Jelle Duives, Andrea Lodi, and Enrico Malaguti. Test-assignment: a quadratic coloring problem. *Journal of heuristics*, 19(4):549–564, 2013.

[Glover *et al.*, 2010] Fred Glover, Zhipeng Lü, and Jin-Kao Hao. Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR: A Quarterly Journal of Operations Research*, 8(3):239–253, 2010.

[López-Ibáñez *et al.*, 2016] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[Palubeckis, 2006] Gintaras Palubeckis. Iterated tabu search for the unconstrained binary quadratic optimization problem. *Informatica*, 17(2):279–296, 2006.

[Stützle and López-Ibáñez, 2019] Thomas Stützle and Manuel López-Ibáñez. Automated design of metaheuristic algorithms. In *Handbook of Metaheuristics*, volume 272 of *International Series in Operations Research & Management Science*, pages 541–579. Springer, 2019.

[Wang *et al.*, 2012] Yang Wang, Zhipeng Lü, Fred Glover, and Jin-Kao Hao. Path relinking for unconstrained binary quadratic programming. *EJOR*, 223(3):595–604, 2012.