Evolving Instances of Unconstrained Binary Quadratic Programming that Challenge a Tabu Search Heuristic

Michael Porta
Computer Science and Information Technology
St. Cloud State University
St. Cloud, MN 56301
pomi0201@stcloudstate.edu

Bryant A. Julstrom
Computer Science and Information Technology
St. Cloud State University
St. Cloud, MN 56301 USA
julstrom@stcloudstate.edu

ABSTRACT

A Tabu Search heuristic for unconstrained binary quadratic programming performs perfectly on a range of random problem instances. A genetic algorithm searches spaces of UBQP instances for instances that challenge the heuristic. The GA's evaluation step compares the performance of the Tabu Search to that of a memetic algorithm on the candidate instance being evaluated. On UBQP instances evolved by the GA, the TS heuristic returns solutions that are inferior to those of the memetic algorithm by significant margins.

Categories and Subject Descriptors

G.2.1 [**Mathematics of Computing**]: Discrete Mathematics—*Combinatorics*; I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic Methods

General Terms

Algorithms

Keywords

Tabu Search, difficult instances, unconstrained binary quadratic programming, genetic algorithm

1. INTRODUCTION

Tabu Search (TS) [4] is a heuristic for problems of combinatorial optimization. TS defines the neighborhood of a solution as those solutions that can be reached via one small change, and a change can be specified as tabu; that is, temporarily prohibited. The algorithm repeatedly examines all solutions in the neighborhood of the current solution that are reachable via a permitted change. If a better solution is found, it becomes current, and the change that was made to reach it is tabu for a specified number of steps.

Associated with a collection of n objects are values that may be positive, negative, or zero. Each object i has an individual value v_i, and each pair of objects (i, j) has a joint value $v_{i,j}$. Unconstrained binary quadratic programming (UBQP) seeks a subset of the objects for which the sum of their individual and joint values is a maximum. A candidate solution to an instance of UBQP can be specified by a binary string $c[\cdot]$: $c[i] = 1$ indicates that object i is included in the selection; 0 that it is not. UBQP is NP-hard [3], and thus an appropriate target for heuristics such as Tabu Search.

A genetic algorithm evolves instances of unconstrained binary quadratic programming that challenge a Tabu Search heuristic; that is, on which the TS heuristic performs poorly. Julstrom [6] described a similar project in which a GA successfully searched for instances of the quadratic knapsack problem (QKP) that challenged a simplified form of a greedy heuristic described by Hammer and Rader [5]. That project used an exact algorithm for the QKP by Caprara, Pisinger, and Toth [2] to determine how well the greedy heuristic was doing; here, an exact algorithm is impractical and the standard is set by a memetic algorithm for UBQP.

The TS heuristic identifies optimum solutions on ten random UBQP instances from Beasley's OR-Library[1] [1] of $n = 50, 100, 250$, and 500 objects, but the GA evolves ten instances of similar sizes on which the memetic algorithm outperforms the Tabu Search by 2.5% to 9.3%.

2. A TABU SEARCH HEURISTIC

A Tabu Search heuristic for UBQP maintains one candidate solution to the problem in the obvious way: as a binary string $c[\cdot]$: $c[i] = 1$ indicates that object i is chosen; $c[i] = 0$ indicates that it is not. The neighborhood of $c[\cdot]$ consists of all the strings that differ from it in exactly one bit; that is, the strings within Hamming distance 1 of $c[\cdot]$. The algorithm repeatedly flips all the non-tabu bits in the current string. The best of these neighbors becomes the current solution and the bit flipped to reach it is tabu—immune from being flipped again—for a specified number of iterations. The heuristic begins with a string that is entirely zeros, and bits remain tabu for 20 iterations.

3. REPRESENTING UBQP INSTANCES

An instance of unconstrained binary quadratic programming is an $n \times n$ array of rational values that is symmetric about its diagonal. Thus an instance can be represented by the upper right triangle of such an array, including the diagonal. Moreover, only the non-zero values need be explicitly recorded, so candidate UBQP instances are represented by lists of non-zero values, with their positions in the array.

4. EVALUATION

A chromosome's fitness, which we seek to maximize, is the difference between the value returned by the Tabu Search

[1]http://people.brunel/ac/uk/~mastjjb/leb/info.html

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

heuristic on the UBQP instance the chromosome represents and an estimate of the optimum value for that instance. The estimate is generated by a memetic algorithm: a genetic algorithm that incorporates a local search step.

The memetic algorithm is generational, with parents chosen in 2-tournamemnts. Each pair of parents is recombined in one-point crossover to produce two offspring. Bit-flip mutation is applied to the offspring, which are then subjected to local search. The local search step seeks an improved solution by flipping each of the offspring's bits in turn. This process continues for each offspring until no further improvement is possible. The new generation replaces the old, with the exception of 1-elitism. The memetic algorithm runs through a fixed number of generations.

In the trials reported here, the memetic algorithm's population contained 50 chromosomes, and it ran through 100 generations. Such runs are relatively short, but the local search does most of the work.

5. OPERATORS

A random UBQP instance is generated by choosing random pairs of objects and giving them random integer values in the range [-100,100], excluding zero. One-point crossover recombines representations of UBQP instances, and mutation either moves a value to a new location or randomly replaces a (non-zero) value.

6. A GENETIC ALGORITHM

A genetic algorithm uses the representation and operators just described to search for instances of unconstrained binary quadratic programming that are difficult for the Tabu Search heuristic. The GA is generational, with 1-elitism. A pool of parent chromosomes twice the size of the population is formed through k-tournament selection. Crossover is applied to each pair of chromosomes to yield one offspring, which is then mutated. The algorithm runs through a fixed number of generations.

In the tests reported below, the GA's population size was set to 50, the size of its selection tournaments to $k = 5$. Crossover was performed to generate 85% of offspring chromosomes. The GA ran through 500 generations (300 in the largest case due to time constraints).

Evaluating a candidate UBQP instance requires running both the Tabu Search heuristic and the memetic algorithm on it, so a GA that searches a space of UBQP instances is necessarily slow. The GA was therefore written in C and used the Message Passing Interface, so that it could be executed in parallel on the University of Minnesota's Calhoun cluster.

7. TESTS

To establish the performance of the Tabu Search heuristic, it was run on ten randomly generated instances from Beasley's OR-Library: three with $n = 50$ objects, three with $n = 100$, three with $n = 250$, and one with $n = 500$. The TS heuristic found the known optimum solution value on every one of these instances.

The genetic algorithm was then run to develop instances that challenge the TS heuristic, one such instance for each of the OR-Library instances. Table 1 shows the results of these trials. For each trial, the table reports the size of the instance, the value returned by the memetic algorithm, the value returned by the TS heuristic, and their percent difference.

Table 1: Performance of the Tabu Search heuristic on ten UBQP instances evolved by the GA to be difficult for the TS. The memetic algorithm identified the best known solutions to which the TS was compared.

n	Best Known	Tabu Search	% diff
50	3849	3491	9.3
50	3989	3691	7.5
50	4568	4186	8.4
100	9517	8721	8.4
100	14564	14204	2.5
100	11106	10821	2.6
250	46248	45102	2.5
250	40696	39451	3.1
250	40236	38995	3.1
500	120310	117629	2.6

The results indicate the experiment's success. On the evolved instances, the TS heuristic missed the standard set by the memetic algorithm by between 2.5% and 9.3%, in contrast to its perfect performance on the random instances.

8. CONCLUSION

A Tabu Search heuristic for unconstrained binary quadratic programming performs perfectly on a set of random problem instances of moderate sizes. A genetic algorithm evolves UBQP instances intended to challenge the heuristic. In the GA, the key step is the evaluation of each candidate UBQP instance; in this step, a memetic algorithm is executed on the instance, and its results compared to those of the TS heuristic. The GA successfully identifies problem instances on which the TS heuristic significantly underperforms the standard set by the memetic algorithm.

9. REFERENCES

[1] J. E. Beasley. Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[2] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11:125–137, 1999.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[4] F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[5] P. L. Hammer and J. David J. Rader. Efficient methods for solving quadratic 0-1 knapsack problems. *INFOR*, 35(3):170–182, 1997.

[6] B. A. Julstrom. Evolving heuristically difficult instances of combinatorial problems. In G. Raidl et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO-2009*, pages 279–285, New York, 2009. ACM Press.