An optimisation based approach to neighbourhood search

J.E. Beasley

Mathematics, Brunel University, Uxbridge UB8 3PH, UK
john.beasley@brunel.ac.uk
http://people.brunel.ac.uk/~mastjjb/jeb/jeb.html

July 2022

Abstract
In this paper we present an optimisation based approach to neighbourhood search. This approach completely eliminates the need to define a problem specific search neighbourhood for any particular (zero-one) optimisation problem. It does this by incorporating a generalised Hamming distance neighbourhood into the problem, and this leads naturally to an appropriate neighbourhood search procedure.

We apply our approach to the non-unicost set covering problem. Computational results are presented for 65 test problems that have been widely considered in the literature. Our results indicate that our optimisation based approach to neighbourhood search is very competitive in terms of solution quality with other approaches from the literature.

Keywords: discrete optimisation; Hamming distance; integer programming; neighbourhood search; set covering

1 Introduction

As the reader may be aware a common approach to the heuristic solution of many zero-one integer programming problems is to apply neighbourhood search. By this we mean that given a (typically feasible) solution to the problem at hand we examine “small” changes to this solution. So we examine solutions in the “neighbourhood” of this feasible solution. If we find a better feasible solution then this typically becomes the new solution and the process repeats until some termination condition is satisfied (e.g. computational time limit, or failure to improve on the solution).

There are a number of general neighbourhood search approaches in the literature such as simulated annealing [11], tabu search [9] and variable neighbourhood search [10][14] that can be applied. Such approaches set out a general search procedure, but need particularisation for the problem at hand, e.g. in defining the neighbourhood of a solution. Typically the neighbourhood of a solution is defined by specifying the possible moves away from a solution. These neighbourhood search approaches have been extensively used in the literature. For example a recent search using Web of Science (http://www.webofscience.com) listed approximately 27,000 papers referring to simulated annealing, 10,000 referring to tabu search and 6,000 papers referring to variable neighbourhood search.

In this paper we present an optimisation based approach to neighbourhood search. As far as we are aware such an approach has not been presented previously in the literature. This approach completely eliminates the need to define a problem specific search neighbourhood for
any particular (zero-one) optimisation problem. It does this by incorporating a generalised Hamming distance neighbourhood into the problem, and this leads naturally to an appropriate neighbourhood search procedure.

The structure of this paper is as follows. In Section 2 we define what we mean by the neighbourhood of a feasible solution to a general (zero-one) optimisation problem. We then go on to outline a search procedure that we can adopt to successively search for improved solutions. In Section 3 we consider the example optimisation problem, the non-unicost set covering problem, to which we are going to apply our optimisation based approach to neighbourhood search. We define the problem and consider relevant literature on the problem with especial reference to papers in the literature which report good computational results. In Section 4 we present computational results based on applying our optimisation based approach to neighbourhood search to 65 non-unicost set covering problems that have been extensively considered by others in the literature. Finally in Section 5 we present our conclusions.

2 Optimisation based neighbourhood search

In this section we first define what we mean by the neighbourhood of a feasible solution to a general (zero-one) optimisation problem. We then go on to outline a search procedure that we can adopt to successively search for improved solutions.

2.1 Neighbourhood

To illustrate our approach suppose that we have a general zero-one integer programming problem involving \( n \) zero-one variables \( x_i, \ i = 1, \ldots, n \) and \( m \) constraints where the optimisation problem is:

\[
\text{minimise} \quad \sum_{i=1}^{n} c_i x_i
\]  

subject to:

\[
\sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i = 1, \ldots, m
\]

\[
x_i \in \{0, 1\} \quad i = 1, \ldots, n
\]

Equation (1) is a minimisation objective. Without significant loss of generality we shall henceforth assume that all the objective function coefficients \( c_i \) are integer. Equation (2) represents the constraints of the problem and Equation (3) the integrality condition.

Let \( [X_i] \) be some feasible solution to the problem. Then in this paper we define the neighbourhood of \( [X_i] \) to be any set of zero-one variable values \( [x_i] \) satisfying \( 1 \leq \sum_{i=1}^{n} |x_i - X_i| \leq K \), where \( K \) is a known positive constant of our choice. In other words the neighbourhood of \( [X_i] \) is any set of zero-one values \( [x_i] \) such that the Hamming distance between \( [x_i] \) and \( [X_i] \) lies between one and \( K \).

Then consider the optimisation problem:

\[
\text{minimise} \quad \sum_{i=1}^{n} c_i x_i
\]
subject to Equations (2), (3) and:

\[ 1 \leq \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} (1 - x_i) \leq K \]  

\[ \sum_{i=1}^{n} c_i x_i \leq \sum_{i=1}^{n} c_i X_i - 1 \]  

Here we have added two constraints to our original optimisation problem. In Equation (5) the expression seen is a linearisation of the nonlinear Hamming distance \( \sum_{i=1}^{n} |x_i - X_i| \). Equation (5) ensures that the Hamming distance between \([x_i]\) and \([X_i]\) is at least one (so we have a solution different from \([X_i]\)) and is also less than or equal to \(K\).

Equation (6) implies that we are only interested in improved feasible solutions in the neighbourhood of \([X_i]\), i.e. those that strictly improve on the solution value \(\sum_{i=1}^{n} c_i X_i\) associated with the current solution. Improving on the current feasible solution cannot be achieved by Equation (5) since it only constrains the structural (Hamming distance) difference between two solutions, it does not address their solution values.

With regard to a minor technical issue here we have that in integer programming terms use of Equation (6) automatically implies that the Hamming distance between \([x_i]\) and \([X_i]\) is at least one. This is because any improved feasible solution must be different from \([X_i]\). However it could be that including an explicit lower limit on the Hamming distance of one (as in Equation (5)) improves computational performance, e.g. by improving the linear programming relaxation solution, so we include it here.

Our Neighbourhood Search Optimisation Programme, NSOP is then optimise Equation (4) subject to Equations (2), (3), (5), (6). In essence in NSOP we amend the original optimisation problem to restrict attention to distinctly different (and improved) solutions within the \(K\) neighbourhood of \([X_i]\). Note here that because of the extra constraints added to the original optimisation problem any feasible solution to NSOP must be an improved solution as compared to \([X_i]\).

In general terms it is clear that if we solve NSOP to proven global optimality (e.g. using a package such as Cplex [8]) then we will either:

- find an improved feasible solution within the neighbourhood of \([X_i]\), technically the minimum feasible solution within the neighbourhood; or
- prove that there is no improved feasible solution within the neighbourhood.

Obviously computational considerations may mean that we do not solve NSOP to proven global optimality, but within computational limits we may still find an improved feasible solution. As noted above any feasible solution to NSOP must by definition be an improved solution as compared to \([X_i]\). Obviously, as with standard neighbourhood search procedures, an improved feasible solution can be used to replace \([X_i]\) and the process repeated in a natural way. The search procedure we adopted based upon NSOP is detailed below.

2.2 Search procedure

Our search procedure requires an initial feasible solution \([X_i]\) as well as three parameter values. These are: an initial value for \(K\); a value \(\delta\) for incrementing \(K\) so as to increase the size of the neighbourhood and a value \(L\) for the number of successive iterations we allow without improving
the solution before terminating the search.

Our search procedure is:

(a) Initialise \([X_i], K, \delta\) and \(L\). Set \(t \leftarrow 0\), where \(t\) is the iteration counter.
(b) Set \(t \leftarrow t + 1\) Solve NSOP. If we find an improved feasible solution replace \([X_i]\) with this solution.
(c) If \(L\) successive iterations have been performed without improving the current feasible solution then stop, else set \(K \leftarrow K + \delta\) and go to step (b).

In this procedure we increase the size of the neighbourhood (increment \(K\) by \(\delta\)) at each iteration, irrespective as to whether an improved solution has been found or not. This ensures that we continually expand the search space around the (current) feasible solution. The procedure only terminates once \(L\) successive iterations have been performed without finding an improved feasible solution.

2.3 Comment

We would make a number of comments as to our optimisation based approach to neighbourhood search:

- Our NSOP draws directly on the mathematical formulation of the problem and hence can be classed as a matheuristic [5].
- It completely eliminates the need to design problem specific search neighbourhoods, since the neighbourhood is automatically incorporated into NSOP using the Hamming distance (Equation (5)) as discussed above.
- If the NSOP associated with the final feasible solution found has been solved to proven global optimality then we have an absolute guarantee that there is no improved feasible solution within the Hamming distance \(K\) neighbourhood associated with that final feasible solution.

3 Application - set covering

It is clear that to investigate the worth of the optimisation based approach to neighbourhood search given above we need to apply it to an example optimisation problem. For this purpose we choose to use a classical zero-one optimisation problem, the set covering problem. Referring back to Equation (2) above we have that \([a_{ij}]\) is a known matrix with \(a_{ij} = 1\) if column \(j\) covers row \(i\), \(a_{ij} = 0\) otherwise. The values \([b_i]\) are all one.

There are two variants of the problem, one where the column costs \([c_i]\) are all one (known as the unicot set covering problem), one where the column costs \([c_i]\) are general non-negative values (referred to as the non-unicost set covering problem, or more commonly as just the set covering problem). In the results given below we apply our approach to the non-unicost problem. Of the two variants of the problem the non-unicost variant has attracted greater attention in the literature.
The (non-unicost) set covering problem has been considered by a number of authors in the literature as discussed below. It is not our intention here to give a comprehensive and detailed review of the literature for the set covering problem. Rather our intention is to highlight significant papers in the literature which report good computational results on set covering instances. This is because the focus of this paper is whether, for the specific example problem (set covering) considered, our optimisation based approach to neighbourhood search can yield good quality results comparable with those already reported in the literature.

3.1 Relevant literature

Caprara and Toth [7] give a survey of algorithms for the set covering problem prior to 2000. As is clear from their paper most of the authors in the literature since 1990 have made use of the test problems publicly available from OR-Library [2], see http://people.brunel.ac.uk/~mastjjb/jeb/info.html. In this paper we also make use of these test problems.

Lan et al [13] presented a heuristic approach Meta-RPS (Meta-heuristic for Randomized Priority Search). They stressed the use of randomness to avoid local optima. Their approach is a repeated application of: firstly a constructive heuristic to find a feasible solution (but including randomisation); secondly a local improvement heuristic based on neighbourhood search. Their approach also included preprocessing, both to exclude columns from consideration and to include them if a column is the only one that covers a row. To reduce the computation time associated with their neighbourhood search procedure they defined a core problem consisting of a small subset of columns.

Lan et al [13] reported that their heuristic is one of only two to find all optimal/best-known solutions for non-unicost instances. They gave a table illustrating the effectiveness of different heuristics on 65 non-unicost set covering problems. Of note there is the indirect genetic algorithm of Aickelin [1]; the genetic algorithm of Beasley and Chu [3]; and the lagrangian heuristic of Caprara et al [6]. We consider each of these three approaches below.

Aickelin [1] presented a genetic algorithm approach with a decoder which works on a permuted list of the rows to be covered, with hill-climbing to improve the solution applied after the decoder has provided a suitable solution. For 65 non-unicost set covering problems they compare their approach with other approaches, [3,6], taken from the literature.

Beasley and Chu [3] presented a genetic algorithm approach including a new fitness-based crossover operator (fusion), a variable mutation rate and a heuristic feasibility operator tailored specifically for the non-unicost set covering problem. They reported computational experience for their approach on 65 non-unicost set covering problems.

Caprara et al [6] presented a lagrangian heuristic approach using dynamic pricing for the variables and systematic use of column fixing to improve the solution. They made use of a number of improvements in the subgradient optimisation procedure as well as a refining procedure to improve upon any given solution. They gave a table illustrating the effectiveness of different heuristics on 65 non-unicost set covering problems showing that their approach performs well.

Naji-Azimi et al [15] presented an electromagnetic metaheuristic approach drawing on the work of Birbil and Fang [4]. Their approach involves an initial preprocessing step and then repetitively adjusting a pool of solutions to which local search is first applied and where the solutions are then changed based on the force generated by the “charge” associated with each solution. Mutation was also applied to perturb solutions. They considered 65 non-unicost
problems and compared their results with those of Lan et al \cite{13}. Their results indicated that their approach was competitive with that of Lan et al \cite{13}.

In recent years a number of papers in the literature have applied algorithms based upon paradigms drawn from the natural world (sometimes referred to as bio-inspired metaheuristics). One example of work of this kind is Soto et al \cite{17} who presented approaches based on cuckoo search and black hole optimisation.

Cuckoo search (see Yang and Deb \cite{18}) is a population based approach where each “nest” in the population contains a number of “eggs” (solutions) and “cuckoos” lay eggs in randomly chosen nests. The best nests carry over to the next generation. Black hole optimisation (see Kumar et al \cite{12}) is also a population based approach where a “black hole” attracts “stars” (solutions). Stars change locations as they are attracted by the black hole. Some stars are absorbed by the black hole and replaced by newly generated stars (solutions).

Soto et al \cite{17} applied these two approaches to the non-unicost set covering problem. They applied preprocessing and presented results for 65 non-unicost set covering problems (based on 30 trials for each instance).

4 Computational results

In this section we first discuss the non-unicost set covering test problems which we used. We then give computational results for our optimisation based approach to neighbourhood search when applied to these test problem. We also give a comparison between the results from our approach and seven other approaches presented previously in the literature.

4.1 Test problems

We used the standard set of 65 non-unicost set covering test problems that are available from OR-Library \cite{2}, see \url{http://people.brunel.ac.uk/~mastjjb/jeb/info.html}. We used a Windows pc with 8GB of memory and an Intel Core i5-1137G7 2.4Ghz processor, a multi-core pc with four cores. The initial value for $K$ was set to 5, the neighbourhood increment $\delta$ was set to 5 and the number successive iterations $L$ allowed without improving the current feasible solution before termination was set to 5. These values were set based on limited computational experimentation.

The initial feasible solution required to start the search was produced using a simple greedy heuristic for the set covering problem: repetitively choosing a column with the minimum value of the ratio (column cost/number of uncovered rows covered by the column). Once a solution covering all rows had been found we removed any redundant columns (those columns for which all rows which they cover are also covered by other columns). Note here that, unlike a number of papers in the literature (e.g. \cite{13,15,17}), we made no use of problem-specific preprocessing to eliminate columns.

Table 1 shows the characteristics of the 65 non-unicost problems considered. In that table we show the problem set name, the number of instances in that set, the number of rows ($m$) and columns ($n$) and the problem density ($\left((\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}/mn\right]$ expressed as a percentage). We set a time limit of 15 seconds for the solution of each NSOP for all the problems with $m \leq 500$. For the larger problems with $m = 1000$ we increased this time limit to 45 seconds.
Table 1: Test problem characteristics

| Problem set name | Number of instances | Number of rows ($m$) | Number of columns ($n$) | Density (%) |
|------------------|---------------------|----------------------|-------------------------|-------------|
| 4                | 10                  | 200                  | 1000                    | 2           |
| 5                | 10                  | 200                  | 2000                    | 2           |
| 6                | 5                   | 200                  | 1000                    | 5           |
| A                | 5                   | 300                  | 3000                    | 2           |
| B                | 5                   | 300                  | 3000                    | 5           |
| C                | 5                   | 400                  | 4000                    | 2           |
| D                | 5                   | 400                  | 4000                    | 5           |
| NRE              | 5                   | 500                  | 5000                    | 10          |
| NRF              | 5                   | 500                  | 5000                    | 20          |
| NRG              | 5                   | 1000                 | 10000                   | 2           |
| NRH              | 5                   | 1000                 | 10000                   | 5           |

4.2 Results

Table 2 shows the results obtained by our optimisation approach to neighbourhood search. In that table we show the optimal/best-known solution value for each problem (as taken from Lan et al [13]), the solution value as obtained by our approach, the total computation time (in seconds), the value for $K$ at termination and whether the NSOP for this final value of $K$ was solved to proven optimality or not. We have not given in Table 2 the number of iterations made in our procedure as this can easily deduced by dividing the final value of $K$ by the neighbourhood increment $\delta$, where we used $\delta = 5$.

So for example consider problem 4.10 in Table 2. The optimal/best-known solution for this instance is 514 and the “o” signifies that this was the solution found by our approach. The value of $K$, the size of the neighbourhood at the final iteration, was 45 and the total solution time was 0.4 seconds. The “yes” in the solution guarantee column signifies that the NSOP associated with this final value of $K$ was solved to proven optimality, indicating that we have an absolute guarantee that there is no improved solution within a Hamming distance of $K = 45$ from the solution associated with the value of 514 as found by our approach.

Considering Table 2, it is clear that for all but one of the 65 problems we find the optimal/best-known solution. From Table 1, these problems are of increasing size and for all 45 problems up to and including problem set D we have the guarantee on solution quality. However for the 20 larger problems we only have one instance in which this is the case (recall here that we impose a time limit for the solution of each and every NSOP encountered during the process).

In order to compare our results with previous results in the literature we have taken the detailed results given by various authors and computed percentage deviation from the optimal/best-known value, OBKS, as shown in Table 2. In other words using the solution values given by authors in their papers we computed 100(solution value - OBKS)/OBKS for each individual problem and then averaged the percentage deviations. Note here that whilst previous authors may have used this percentage deviation approach their OBKS values may be different from those shown in Table 2 (since obviously best-known values may be updated over time).

Table 3 shows this comparison where we give in each case the average percentage deviation and the average computation time (in seconds) as calculated using solution details in the papers
cited. All of these averages relate to the same 65 instances which we considered, as detailed in Table 1.

As we would expect different authors have used different hardware and so a direct comparison between computation times is difficult, but the times given are an indication of how quickly problems are solved on average. In order to help set the times shown in context, as the papers cited range from 1996-2022, so over 25 years during which hardware has improved immensely, we also show in Table 3 the year each paper was published.

With regard to Table 3:

- Aickelin [1] and Beasley and Chu [3] used ten trials of their algorithms. So for both these papers in Table 3 we have used the best solution found over the ten trials in calculating percentage deviation, and the time seen is the total time for these ten trials.
- For Caprara et al [6] the detailed times given in their paper are the times as to when the final best solution was first found. As such we have no information as to the total time taken, which is the value given for the other papers cited.
- For Lan et al [13] the detailed times given in their paper are the times as to when the final best solution was first found. For this reason the time given in Table 3 is taken from the best performing of three different variants of their approach (the variant Meta-RaPs w/randomized priority rules, see Table 4 in [13]) and the time given is as reported by them in their paper as the total time taken for that variant.
- For Soto et al [17] the times given in their paper are the average time for each trial [16], where they used 30 trials. So in Table 3 we give the total time for 30 trials and in calculating percentage deviation we used the best solution found over these 30 trials.

Considering Table 3 it seems reasonable to conclude that our optimisation based approach for neighbourhood search is very competitive in terms of solution quality with other approaches given in the literature.

5 Conclusions

In this paper we have presented an optimisation based approach to neighbourhood search. This approach completely eliminated the need to define a problem specific search neighbourhood for any particular (zero-one) optimisation problem by incorporating a generalised Hamming distance neighbourhood into the problem. As far as we are aware such an approach has not been presented previously in the literature.

We applied our approach to the non-unicost set covering problem and presented computational results for 65 test problems that have been widely considered in the literature. Our results indicated that our optimisation based approach to neighbourhood search is very competitive in terms of solution quality with other approaches from the literature.
| Instance | Optimal/best-known | Solution | Final K | Time (secs) | Solution guarantee |
|----------|--------------------|----------|---------|-------------|-------------------|
| 4.1      | 429                | o        | 45      | 0.1         | yes               |
| 4.2      | 512                | o        | 50      | 0.6         | yes               |
| 4.3      | 516                | o        | 55      | 0.5         | yes               |
| 4.4      | 494                | o        | 50      | 0.5         | yes               |
| 4.5      | 512                | o        | 45      | 0.4         | yes               |
| 4.6      | 560                | o        | 55      | 0.7         | yes               |
| 4.7      | 430                | o        | 50      | 0.4         | yes               |
| 4.8      | 492                | o        | 45      | 0.8         | yes               |
| 4.9      | 641                | o        | 55      | 1.0         | yes               |
| 4.10     | 514                | o        | 45      | 0.4         | yes               |
| 5.1      | 253                | o        | 50      | 1.1         | yes               |
| 5.2      | 302                | o        | 50      | 1.3         | yes               |
| 5.3      | 226                | o        | 50      | 0.7         | yes               |
| 5.4      | 242                | o        | 45      | 1.1         | yes               |
| 5.5      | 211                | o        | 45      | 0.7         | yes               |
| 5.6      | 213                | o        | 45      | 0.8         | yes               |
| 5.7      | 293                | o        | 50      | 1.1         | yes               |
| 5.8      | 288                | o        | 45      | 1.0         | yes               |
| 5.9      | 279                | o        | 50      | 0.8         | yes               |
| 5.10     | 265                | o        | 50      | 0.7         | yes               |
| 6.1      | 138                | o        | 60      | 2.0         | yes               |
| 6.2      | 146                | o        | 40      | 1.1         | yes               |
| 6.3      | 145                | o        | 45      | 1.5         | yes               |
| 6.4      | 131                | o        | 45      | 1.4         | yes               |
| 6.5      | 161                | o        | 50      | 1.8         | yes               |
| A1       | 253                | o        | 50      | 3.7         | yes               |
| A2       | 252                | o        | 55      | 4.1         | yes               |
| A3       | 232                | o        | 50      | 2.9         | yes               |
| A4       | 234                | o        | 50      | 2.4         | yes               |
| A5       | 236                | o        | 65      | 3.5         | yes               |
| B1       | 69                 | o        | 40      | 3.5         | yes               |
| B2       | 76                 | o        | 45      | 9.4         | yes               |
| B3       | 80                 | o        | 45      | 4.7         | yes               |
| B4       | 79                 | o        | 45      | 10.2        | yes               |
| B5       | 72                 | o        | 40      | 3.4         | yes               |
| C1       | 227                | o        | 55      | 4.0         | yes               |
| C2       | 219                | o        | 60      | 5.8         | yes               |
| C3       | 243                | o        | 80      | 19.6        | yes               |
| C4       | 219                | o        | 50      | 4.0         | yes               |
| C5       | 215                | o        | 55      | 5.0         | yes               |
| D1       | 60                 | o        | 50      | 7.8         | yes               |
| D2       | 66                 | o        | 50      | 23.0        | yes               |
| D3       | 72                 | o        | 55      | 23.5        | yes               |
| D4       | 62                 | o        | 45      | 10.6        | yes               |
| D5       | 61                 | o        | 45      | 5.1         | yes               |
| NRE1     | 29                 | o        | 40      | 78.3        | no                |
| NRE2     | 30                 | o        | 65      | 160.2       | no                |
| NRE3     | 27                 | o        | 60      | 139.8       | no                |
| NRE4     | 28                 | o        | 50      | 105.3       | no                |
| NRE5     | 28                 | o        | 40      | 76.8        | no                |
| NRF1     | 14                 | o        | 45      | 114.2       | no                |
| NRF2     | 15                 | o        | 35      | 78.5        | no                |
| NRF3     | 14                 | o        | 45      | 44.4        | yes               |
| NRF4     | 14                 | o        | 40      | 93.4        | no                |
| NRF5     | 13                 | o        | 45      | 109.4       | no                |
| NRG1     | 176                | o        | 65      | 308.3       | no                |
| NRG2     | 154                | o        | 60      | 219.7       | no                |
| NRG3     | 166                | o        | 85      | 565.3       | no                |
| NRG4     | 168                | o        | 115     | 821.2       | no                |
| NRG5     | 168                | o        | 75      | 475.4       | no                |
| NRH1     | 63                 | o        | 64      | 344.9       | no                |
| NRH2     | 63                 | o        | 70      | 501.0       | no                |
| NRH3     | 59                 | o        | 100     | 773.0       | no                |
| NRH4     | 58                 | o        | 85      | 618.6       | no                |
| NRH5     | 55                 | o        | 55      | 342.0       | no                |

Table 2: Computational results
| Approach                  | Average % deviation | Average time (secs) | Year published |
|---------------------------|---------------------|---------------------|----------------|
| This paper               | 0.02                | 94.6                | 2022           |
| Aickelin [4]             | 0.13                | 1179.5              | 2002           |
| Beasley and Chu [3]      | 0.07                | 14694.3             | 1996           |
| Caprara et al [6]        | 0                   | not known           | 1999           |
| Lan et al [13]           | 0                   | 878.4               | 2007           |
| Naji-Azimi et al [15]    | 0.18                | 118.4               | 2010           |
| Soto et al [17] cuckoo search | 1.01                | 151.0               | 2017           |
| Soto et al [17] black hole | 1.55                | 184.5               | 2017           |

Table 3: Comparison of results
References

[1] U. Aickelin. An indirect genetic algorithm for set covering problems. *Journal of the Operational Research Society*, 53(10):1118–1126, 2002.

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

[3] J. E. Beasley and P.C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392–404, 1996.

[4] S. I. Birbil and S. C. Fang. An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization*, 25(3):263–282, 2003.

[5] M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. B. Rohler. *Matheuristics: Optimization, simulation and control*, volume 5818 of *Lecture Notes in Computer Science*, pages 171–177. Springer, Berlin, 2009.

[6] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations Research*, 47(5):730–743, 1999.

[7] A. Caprara, P. Toth, and M. Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–371, 2000.

[8] CPLEX Optimizer. IBM. https://www.ibm.com/products/ilog-cplex-optimization-studio last accessed May 2022, 2019.

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

[10] P. Hansen and N. Mladenovic. Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[12] S. Kumar, D. Datta, and S. K. Singh. Black hole algorithm and its applications. In A. T. Azar and S. Vaidyanathan, editors, *Computational Intelligence Applications in Modeling and Control*, pages 147–170. Springer International Publishing, 2015.

[13] G. Lan, G. W. DePuy, and G. E. Whitehouse. An effective and simple heuristic for the set covering problem. *European Journal of Operational Research*, 176(3):1387–1403, 2007.

[14] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[15] Z. Naji-Azimi, P. Toth, and L. Galli. An electromagnetism metaheuristic for the unicost set covering problem. *European Journal of Operational Research*, 205(2):290–300, 2010.

[16] R. Olivares. Private communication, 2022.

[17] R. Soto, B. Crawford, R. Olivares, J. Barraza, I. Figueroa, F. Johnson, F. Paredes, and E. Olguin. Solving the non-unicost set covering problem by using cuckoo search and black hole optimization. *Natural Computing*, 16(2):213–229, 2017.
[18] X.-S. Yang and S. Deb. Cuckoo search via Lévy flights. In A. Abraham, F. Herrera, A. Carvalho, and V. Pai, editors, 2009 World Congress on Nature & Biologically Inspired Computing (NABIC 2009), pages 210–214, 2009.