IJIO
(International Journal of Industrial Optimization)
ISSN 2723-3022 (online) | 2714-6006 (printed)
Vol 6, No. 1, 17-27
https://doi.org/10.12928/ijio.v6i1.10836

Filling the gap in weighted set covering problem test instances: implications for both researchers and practitioners

Francis J. Vasko [a,*], Yun Lu [a], Myung Soon Song [a], Dominic Rando [b]

[a] Department of Mathematics, Kutztown University, United State of America
[b] Department of Computer Science, Kutztown University, United State of America
* Corresponding Author: vasko@kutztown.edu

ARTICLE INFO

Article history
Received: June 18, 2024
Revised: October 1, 2024
Accepted: October 23, 2024

Keywords
metaheuristics;
operations research;
weighted set covering problems;
integer programming software;
Beasley's OR-Library.

ABSTRACT

Since 1990, the quality of approximate solution methods for solving weighted set covering problems (WSCPs) has been measured based on how well they solve 65 WSCPs available in Beasley's OR-Library. In a 2024 paper, it has been shown that guaranteed optimal solutions can easily be obtained for 55 weighted set covering problems (WSCPs) in Beasley's OR-Library using general-purpose integer programming software. These 55 WSCPs have 500 rows and 5,000 columns or less and were solved in a few seconds on a standard PC. However, the remaining 10 WSCPs have 1000 rows and 10,000 columns and either required considerably more than 1000 seconds to obtain guaranteed optimums (data set NRG) or no optimums were obtained (NRH). The purpose of this short paper is to try to quantify the solution times needed to solve WSCPs using general-purpose integer programming software that are larger than 500 rows and 5,000 columns, but less than 1,000 rows and 10,000 columns. This is important because the size and solution time gap is so large that solution times go from a few seconds for the 55 "smaller" WSCPs to very large solution times for the two largest data sets. To fill this gap, 40 new WSCP instances are defined and their solution times are analyzed to determine when to expect that WSCPs, based on size and density, can be solved to optimality in a timely manner using general-purpose integer programming software like Gurobi or CPLEX.

1. Introduction

The weighted set covering problem (WSCP) is a well-known NP-hard [1-2] binary integer programming problem that has many and diverse industrial and business applications [3-15]. In the steel industry alone, the WSCP has been used to model the optimal design of ingot sizes (an ingot is a block of steel) [3], optimal selection of metallurgical grades (steel recipes for meeting customer requirements) [4], and product consolidation to reduce inventory costs [5-6]. Additional WSCP applications include: wireless networks [7], vehicle routing [8], unmanned aerial vehicles [9], gas detectors in chemical process plants [10], ambulance location routing [11], facility location models [12], traffic counting location [13], multi-depot train driver scheduling [14], and railway crew scheduling [15].

We will now provide the usual mathematical formulation of the WSCP. The WSCP is the problem of covering the rows of an m-row, n-column, zero-one matrix (aij) by a subset of the

columns at minimum cost. A mathematical formulation for the WSCP is represented in Eq. (1) – Eq. (3).

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1, \qquad i = 1,...,m, \tag{2}$$

$$x_j \in \{0,1\}, \qquad j = 1,...,n. \tag{3}$$

where x_j is one if column j is in the solution and zero otherwise. Eq. (2) ensures that each row is covered by at least one column and Eq. (3) ensures that the x_j's take on only the values zero or one.

Given that the WSCP is NP-hard, combined with the state of computer hardware and software in 1990, operations research (OR) practitioners could not expect at that time to be able to solve industrial applications of the WSCP to proven optimality. Hence for the many industrial applications mentioned previously, OR practitioners relied on approximate solution methods (heuristic and metaheuristic) to solve applications that were formulated as WSCPs. The 65 WSCP instances in Beasley's OR-Library [16] were made available to the OR community so that they could be used to measure the performance of approximate solution methods. At that time, exact methods that would guarantee optimal solutions for WSCPs usually required excessive computer resources (time and memory) for large (industrial-size) WSCPs.

The 65 WSCPs in Beasley's OR-Library [16] have been used extensively for more than three decades to test WSCP solution methods. The quality of approximate solution methods for solving WSCPs has been measured based on how well they solve these 65 WSCPs. In a 2024 article [17], these instances are used to test the performance of a local branching solution approach. In a 2022 article [18], these instances were used to test the performance of the following eight nature-inspired metaheuristics: binary fruit fly swarm algorithms (BFFSA), Binary cuckoo search (BCS), Binary black hole (BBH), Binary cat swarm optimization (BCSO), Binary firefly optimization (BFO), Binary shuffled frog leap algorithm (BSFLAS), Binary electromagnetism-like algorithm (BELA), and the Binary artificial bee colony (BABC). Also, since 2020, these instances were used to test the performance of the following metaheuristics: the intelligent water drop (IWD) [19], the greedy randomized adaptive search procedure (GRASP) [20], the improved binary monkey search algorithm (IBMSA) [21], a new direct coefficient-based heuristic [22], a greedy heuristic that incorporates the computation of a "surprisal" measure when selecting the solution columns [23], and an effective local search algorithm denoted NuSc [24]. Hence, we are aware of 15 solution methods that have appeared in the literature since 2020 and have used *all* of these 65 WSCPs to test their performance (numerous other articles have used only a subset of these 65 WSCPs for algorithm test purposes [25-28]). Additionally, two articles that address the set covering problem with conflicts [29], reference [30] derive test instances based on the 65 WSCPs from Beasley's OR-Library. The details of these 65 WSCP instances are given in Table 1.

As can easily be seen from Table 1, there is a huge gap in instance size between data sets NRE and NRF which have problems with 500 rows by 5,000 columns and the two largest data sets which have 1,000 rows by 10,000 columns. This paper is the first to acknowledge the gap in the test instance size in the literature. Even the 15 approximate solution methods [17-24] discussed in the literature since 2020 simply use these 65 WSCPs to test the performance of their solution methods and there is no recognition that there is a significant gap in instance size between the NRE and NRF data sets which have 500 rows by 5,000 columns and data sets NRG and NRH that have 1,000 rows by 10,000 columns.

Essentially all the solution methods tested on these 65 WSCPs are approximate in nature—ones since 2020 are largely based on nature-inspired metaheuristics. In other words, they do not guarantee that their solutions are the optimums. Additionally, solution techniques that are based on mathematical programming are essentially heuristic in nature and do not generate proven optimal solutions. The paper by Yelbay et al. [31] provides extensive background on so called primal-dual

methods that make use of dual information. Even the two best performing solution methods tested on these 65 WSCPs do not guarantee optimal solutions. Specifically, the Lagrangian heuristic by Caprara et al. [32] and the randomized priority search heuristic by Lan et al. [33]. In fact, it was reported in [18] that the optimums were still unknown for data sets NRE, NRF, NRG, and NRH.

Table 1. 65 Weighted set covering problem instances from Beasley's OR-library

Set	No. of instances	No. of rows	No. of Columns	Range of cost	Density	Optimal solution
4	10	200	1000	1-100	2%	Known
5	10	200	2000	1-100	2%	Known
6	5	200	1000	1-100	5%	Known
A	5	300	3000	1-100	2%	Known
B	5	300	3000	1-100	5%	Known
C	5	400	4000	1-100	2%	Known
D	5	400	4000	1-100	5%	Known
NRE	5	500	5000	1-100	10%	Known
NRF	5	500	5000	1-100	20%	Known
NRG	5	1000	10000	1-100	2%	Known
NRH	5	1000	10000	1-100	5%	Unknown

However, it is the authors' opinion that if the required computer resources are not excessive, then it is *always* preferred to use solution methods (especially for industrial applications) that either guarantee that the optimum has been found or that the solution generated is guaranteed to be very close to the optimum. When such solutions are obtained, the OR practitioner can confidently present results to management. No matter how well approximate solution methods performed on Beasley's 65 WSCPs, solutions based on such methods do not typically provide any guarantees on the solution quality for industrial applications and should only be used if solutions that are guaranteed to be optimum or near-optimum cannot be generated in a timely manner.

Recent advances in integer programming software [34-40] has made it possible to obtain optimal solutions for large integer programming problems such as the WSCP. In fact, Koch et al. [40] state that from 2001 to 2020 there has been a 1000-fold speed-up in the solution of mixed integer linear programming problems. In 2024 [17] it was shown that all but five (data set NRH) of Beasley's 65 WSCPs can be solved optimally using CPLEX (version 12.10). In fact, 15 WSCPs (NRE, NRF, and NRG) that were reported to have unknown optimums in as recent as a 2022 article [18], were solved to optimality. In this article, we first summarize our experience using both Gurobi (10.0) and CPLEX (22.1.1) to solve Beasley's 65 WSCPs (both fail to obtain optimums for data set NRH—same as in [17]). Our results using both Gurobi and CPLEX to solve these 65 WSCPs are in total agreement with the CPLEX results presented in [17]. However, it is important to note that CPLEX was used in [17] to try to solve the 65 WSCPs strictly to demonstrate that the approximate solution approach based on local branching discussed in [17] was generating high quality solutions even though these solutions were not guaranteed to be optimal solutions. In [17] there is absolutely no mention of the size gap in these 65 WSCP instances.

However, a significant research contribution of this paper is in "filling the gap" that has existed since 1990 between Beasley's data sets NRE and NRF (500 rows by 5,000 columns) and NRG and NRH (1,000 rows by 10,000 columns) with 40 new WSCP instances. It is important to realize that the existence of this gap has never been previously discussed or even acknowledged in the literature. This paper is the first to acknowledge this gap and to fill it with appropriately sized WSCP instances. Furthermore, we will demonstrate how efficiently these new WSCP instances can be solved to proven optimality. As will be discussed shortly, determining how well these 40 new WSCPs can be solved is important because all of Beasley's problems that are smaller in size than these 40 new WSCPs can be solved to proven optimality in a few seconds, but the 10 Beasley problems that are larger than these 40 new WSCPs either required extensive solution times or have unknown optimums. These 40 WSCPs are defined in the same manner as Beasley's 65 WSCP. Another significant research contribution of this article, based on these 40 new WSCP instances and Beasley's 65 WSCP

10.12928/ijio.v6i1.10836

instances, is to indicate which size WSCP instances can most likely be solved to proven optimality using general-purpose integer programming software on a standard PC. WSCPs that might not be solved to proven optimality can still be input to general-purpose integer programming software. If the software does not find the proven optimum in the maximum allotted execution time, a feasible solution and lower bound will be returned that can prove very useful to OR practitioners that need solutions to real-world problems.

In the next section we will provide the methodology that we used. Next, we will summarize our results using both Gurobi and CPLEX to solve Beasley's 65 WSCPs. This will be followed by an analysis of 40 new WSCPs. We will conclude with ranges on expected solution times based on density and WSCP size.

2. Method

To demonstrate the need for filling the gap between data sets with 500 rows and 5000 columns (NRE and NRF) and data sets with 1,000 rows and 10,000 columns (NRG and NRH), we propose the following methodology:

1) Use general-purpose integer programming software to solve the 65 WSCPs in Beasley's OR Library
2) Using the same methodology that was used to generate these 65 WSCP instances, generate 40 new WSCP instances that have between 500 rows by 5,000 columns and 1,000 rows by 10,000 columns.
3) Use general-purpose integer programming software to solve these 40 new WSCP instances.
4) Analyze the solutions of these 40 WSCP instances based on the time required to obtain optimal solutions.

In the next section we will discuss the results obtained from using this methodology.

3. Results and Discussion

3.1 Solving Beasley's 65 WSCP instances using Gurobi and CPLEX

Unless otherwise stated, all Gurobi (version 10.0) [41] and CPLEX (version 22.1.1) [42] runs were executed on a PC with the following specifications: Intel® Core™ i3-1005G1 CPU at 1.20GHz with 8 GB of RAM at 2667 Mhz. Default parameter settings were used for all Gurobi and CPLEX runs. In [17] CPLEX (version 12.10) [43] was used on a PC with 8 GB of RAM, a 512 GBSSD disk and an Intel ® Core™ i5-1135G7 2.4 GHz processor. In Table 2 we summarize the results of using these general-purpose integer programming software packages to solve the first 55 of Beasley's 65 WSCPs.

Although different processors and software are being used the results are very similar. Although we do not try to quantify how much faster the PC used in [17] is compared to the one that we used, we believe the faster PC used in [17] is why the execution times are somewhat smaller than ours. However, the average times to obtain guaranteed optimal solutions differ very little regardless of PC or software used. In our opinion, the important result is that all 55 of these WSCPs can be solved in less than 2 minutes to proven optimality using standard PCs and general-purpose integer programming software with all default parameters. Contrast this result with the results given in the 2022, Reference [18] for eight nature-inspired metaheuristics when solving the same 55 WSCPs. For these eight metaheuristics, the largest number of optimums found was 44 out of the 55 WSCP instances by the binary fruit fly swarm algorithm and the binary electromagnetism-like algorithm found none of the 55 optimums. It is important to note that even when these methods happen to find the optimum, it is unknown that the optimum has been found. The next largest data sets are NRG and NRH which are much larger than the data sets NRE and NRF. Table 3 shows the execution times for the 5 WSCPs from NRG.

Table 2. Obtaining optimal solutions for the first 55 of the 65 weighted set covering problem instances from Beasley's OR-library

WSCP instances	CPLEX (12.10) [6]	Gurobi (10.0) this paper	CPLEX (22.1.1) this paper
Data sets 4-6 (25 instances)			
Average solution time (seconds)	0.09	0.12	0.10
Maximum solution time (seconds)	0.30	0.26	0.55
Data sets A-D (20 instances)			
Average solution time (seconds)	0.65	0.94	0.84
Maximum solution time (seconds)	1.60	2.73	2.11
Data sets NRE-NRF (10 instances)			
Average solution time (seconds)	15.64	31.18	25.63
Maximum solution time (seconds)	41.00	93.18	76.25
Data sets A-NRF (55 instances)			
Average solution time (seconds)	3.12	6.07	5.01
Maximum solution time (seconds)	41.0	93.18	76.25

Table 3. Obtaining optimal solutions for the five weighted set covering problem instances from Beasley's OR-library in data set NRG

Solution times (seconds) for WSCP instances in data set NRG	CPLEX (12.10) [6]	Gurobi (10.0) this paper	CPLEX (22.1.1) this paper
NRG1	2,071	2,841	2,430
NRG2	335	412	361
NRG3	9,351	8,920	8,239
NRG4	4,339	8,201	7,081
NRG5	16,484	24,343	36,285
Minimum	335	412	361
Average	6,516	8,943	10,879
Maximum	16,484	24,343	36,285

As can be seen from Tables 2 and 3, there is a tremendous increase in execution time between data sets NRE and NRF each with 500 rows and 5,000 columns and data set NRG with 1,000 rows and 10,000 columns. The average execution time averaged over the three solution strategies (CPLEX (12.10), Gurobi (10.0), and CPLEX (22.1.1)) for the 10 instances in NRE and NRF is only 24 seconds. However, this same average execution time for the five instances in NRG is 8779 seconds. In other words, when the WSCP instance size is increased from 500 rows by 5,000 columns to 1,000 rows by 10,000 columns, the average execution time is increased 364 times! Put another way, the average execution time goes from less than 30 seconds to over 2.4 hours. The situation is even worse for data set NRH which has 1,000 rows by 10,000 columns but the density (percentage of non-zero elements in the matrix) is now 5%. For data set NRH, both CPLEX (12.10) as reported in [17] and our experience using CPLEX (22.1.1), had memory issues trying to solve the NRH instances. We executed Gurobi (10.0) on these five NRH instances, but optimal solutions could not be obtained after 24 hours of execution time. After 24 hours of execution, NRH2, NRH4, and NRH5 had obtained the best-known solutions as reported in the literature, but had not proven optimality. Since there is a tremendous jump in execution time between WSCPs with 500 rows and 5,000 columns and WSCPs with 1,000 rows and 10,000 columns, the vital question we seek to answer is what kind of

10.12928/ijio.v6i1.10836

solution times can be expected for WSCPs with the number of rows between 500 and 1,000 and the number of columns between 5,000 and 10,000.

In the next section, we will define 40 new WSCP instances (20 at 2% density and 20 at 5% density) that have sizes strictly between 500 rows by 5,000 columns and 1,000 rows by 10,000 columns. Because our results when using Gurobi (10.0) and CPLEX (22.1.1) were similar when solving WSCPs from Beasley's OR-Library, all 40 new instances will be solved using Gurobi (10.0) only.

3.2 Empirical Results: 40 Weighted Set Covering Problems Introduced in this Article

As noted previously, there is a large gap in WSCP instance sizes between data set NRF (500 rows by 5,000 columns) and data set NRG (1,000 rows by 10,000 columns). If solution times to obtain optimal solutions for NRF instances and NRG instances did not differ much, then "filling the gap" with WSCP instances would not be a major concern. However, the fact that, for WSCPs with up to 500 rows by 5,000 columns, Gurobi solved these problems in under two minutes but, for WSCPs with 1,000 rows by 10,000 columns (the next size in Beasley's WSCP collection), these large problems now either took hours to solve to optimality or optimums were still not obtained after 24 hours of Gurobi execution time is the motivation for defining, solving and analyzing 40 WSCPs with sizes between 500 rows by 5,000 columns and 1,000 rows and 10,000 columns.

The 40 new WSCPs are summarized in Table 4. There are 8 data sets with five WSCPs in each. Four data sets have problems at 2% density and four have problems at 5% density. The notation NFG signifies that these data sets are between data sets NRF and NRG in instance sizes. All 40 WSCPs were randomly generated in the same manner as Beasley's 65 WSCPs [16]. Specifically, column costs are integers randomly generated from [1,100]; every column covers at least one row; and every row is covered by at least two columns. Access to these data sets will be provided at https://github.com/dnr0915/WSCP/. The problems are located in the folder, "NFG"

Table 4. Weighted set covering problem instances introduced in this article

Set	No. of instances	No. of rows	No. of columns	Range of cost	Density	Optimal solution
NFG1	5	600	6000	1-100	2%	Known
NFG2	5	600	6000	1-100	5%	Known
NFG3	5	700	7000	1-100	2%	Known
NFG4	5	700	7000	1-100	5%	Known
NFG5	5	800	8000	1-100	2%	Known
NFG6	5	800	8000	1-100	5%	Known
NFG7	5	900	9000	1-100	2%	Known
NFG8	5	900	9000	1-100	5%	Known

Gurobi solution information for these eight data sets is given in Table 5. Gurobi solved all 40 WSCPs to optimality. For data sets with less than 9000 columns (NFG1, …, NFG6), the maximum solution time was 1,325 seconds (NFG64) and the average Gurobi execution time was 274 seconds. For data sets with 9,000 columns (NFG7 and NFG8), the maximum solution time was 69,590 seconds (NFG82) and the average Gurobi execution time was 12,497 seconds. Although the average execution time over all WSCPs with 9,000 columns was 12,497 seconds (3.47 hours), Gurobi was able to solve all of the largest 900 rows by 9,000 columns by 5% density WSCPs (NFG8). However, the largest execution time of 69,590 seconds (19.3 hours) was a 5% density WSCP with 900 rows and 9,000 columns. The next largest execution time was 11,643 seconds (3.2 hours) for WSCP NFG74 which has 2% density and 900 rows by 9,000 columns.

Table 6 provides minimum, mean, and maximum solution time information for each of the 8 data sets. It illustrates the impact of both instance size and density on solution time. Of the 40 WSCP instances defined in this article with sizes and densities from 600 rows by 6000 columns and 2% density up to 900 rows by 9000 columns and 5% density, only 10 instances can be considered to

require excessive solution times: the 900 rows by 9000 columns for both 2% and 5% densities. The average execution time for these 10 WSCPs is about 208 minutes. Looking more closely at Table 6, one notices execution times for NFG1 (600 rows by 6,000 columns at 2% density), NFG2 (600 rows by 6,000 columns at 5% density), and NFG3 (700 rows by 7,000 columns at 2% density), are very similar to the execution times for Beasley's 500 rows by 5,000 columns instances at both 2% and 5% density (NRE and NRF)—less than 2 minutes. However, for data sets NFG4 (700 rows by 7,000 columns at 5% density) and NFG5 (800 rows by 8,000 columns at 2% density) the execution time starts to increase into the 3-to-6-minute range. Finally, for data set NFG6 (NFG7 and NFG8 were discussed earlier) with 800 rows by 8,000 columns at 5% density, the execution time is now in the 15-to-22-minute range.

Table 5. Gurobi optimal solutions for data sets FG1, FG2, FG3, FG4, FG5, FG6, FG7, and FG8 introduced in this article

Problem instances	Objective function	Gurobi execution time (seconds)	Problem instances	Objective function	Gurobi execution time (seconds)
SCPNFG11	217	25.52	SCPNFG51	181	226.46
SCPNFG12	190	12.35	SCPNFG52	182	293.01
SCPNFG13	184	6.51	SCPNFG53	188	75.69
SCPNFG14	203	16.74	SCPNFG54	186	409.77
SCPNFG15	212	11.02	SCPNFG55	183	438.78
SCPNFG21	66	87.38	SCPNFG61	62	830.27
SCPNFG22	61	14.82	SCPNFG62	62	897.34
SCPNFG23	59	31.88	SCPNFG63	57	1184.41
SCPNFG24	62	58.96	SCPNFG64	59	1324.88
SCPNFG25	64	21.15	SCPNFG65	59	1072.89
SCPNFG31	195	13.04	SCPNFG71	182	3561.52
SCPNFG32	182	39.98	SCPNFG72	165	913.40
SCPNFG33	188	44.02	SCPNFG73	174	418.08
SCPNFG34	187	13.94	SCPNFG74	184	11642.64
SCPNFG35	182	10.16	SCPNFG75	178	562.40
SCPNFG41	60	71.01	SCPNFG81	61	11189.09
SCPNFG42	58	158.54	SCPNFG82	61	69590.00
SCPNFG43	61	91.66	SCPNFG83	60	4540.93
SCPNFG44	61	217.33	SCPNFG84	59	10936.47
SCPNFG45	65	531.33	SCPNFG85	57	11620.29

Table 6. Minimum, mean, and maximum execution times by density for the 40 weighted set covering problem instances introduced in this article

Sets-density	No. of instances	No. of rows	No. of columns	2% density min/mean/max execution time(sec)	5% density min/mean/max execution time (sec)
NFG1-2% **NFG2-5%**	5 5	600	6000	7 14 26	21 43 87
NFG3-2% **NFG4-5%**	5 5	700	7000	13 24 44	71 214 531
NFG5-2% **NFG6-5%**	5 5	800	8000	76 289 439	830 1,062 1,325
NFG7-2% **NFG8-5%**	5 5	900	9000	418 3420 11,643	4,541 21,575 69590

10.12928/ijio.v6i1.10836

3.3. Implications for OR Practitioners

The solution times required to solve 105 randomly generated WSCPs using general-purpose integer programming software with default parameters on a standard PC can be used to guide OR practitioners on what solution times to expect as well as if optimums can likely be obtained when solving industrial applications that are formulated as WSCPs. The 105 WSCP instances discussed in this paper consist of 65 WSCPs from Beasley's OR-Library and 40 new instances defined in this paper to fill the gap in Beasley's instances between problems with 500 rows by 5,000 columns (data sets NRE and NRF) and problems with 1,000 rows and 10,000 columns (data sets NRG and NRH). Filling this gap (which has existed since 1990) was important both academically and practically because the solution times jump dramatically when going from data sets NRE and NRF to data sets NRG and NRH. However, because these 105 instances are randomly generated, the authors as practitioners feel that these execution times are probably upper bounds for the given instance size and density. The reason for this conjecture is that WSCPs that model real-world applications many times have structures of which the software algorithms can take advantage. The solution times discussed in this paper can be used as guidelines for OR practitioners in terms of expected solution times. However, even if the software terminates because the maximum execution time is reached, the best answer found and the best lower bound can be very useful in providing an answer to a real-world problem.

4. Conclusions

Since the year 2000, there has been a tremendous speed-up in integer programming software with most of this speed-up due to algorithm improvements. Hence many problems that had previously only been solved with approximate solution methods can now be solved exactly using general-purpose integer programming software such as Gurobi and CPLEX. Since 1990, 65 weighted set covering problems (WSCPs) accessible by researchers from Beasley's OR-Library have been used to test the performance of approximate solution methods for the WSCP. In this article, we first confirm, using both Gurobi (10.0) and CPLEX (22.1.1), the CPLEX (12.10) results from [17] that demonstrated that all but five of these 65 WSCPs could be solved using general-purpose integer programming software with default parameter settings on a standard PC. In this article, fifty-five of these WSCPs were solved in less than two minutes with an average solution time of only 6 seconds. However, the significant and novel contribution of this paper is that, because there was a large gap in both instance sizes and solution times for Beasley's 65 WSCPs between data set NRF with 500 rows by 5,000 columns and data set NRG with 1,000 rows by 10,000 columns, we defined 40 new WSCPs (8 data sets with five WSCPs each) to fill this size gap. All 40 of these new WSCPs were solved to optimality on a standard PC with 22 of them requiring less than 5 minutes of execution time each. For these 40 new WSCPs, a detailed analysis was provided demonstrating how the execution time was impacted by instance size and density.

Based on Gurobi's performance on these 105 WSCPs ranging in size from 200 rows by 1,000 columns up to 1,000 rows by 10,000 columns, OR practitioners that need to solve WSCPs that model real-world problems have some idea of what to expect in terms of execution times. Even when execution times are expected to be excessive, executing the software for an acceptable amount of time and using the best solution obtained (even if not proven to be optimum) may very well be an acceptable strategy for industrial problems. Given that the OR practitioner's corporation has already invested in the general-purpose integer programming software, this strategy may be preferred to using some WSCP-specific approximate solution method that will need to be coded, tested, and provides no guarantees on solution quality.

For the WSCP instances that required excessive execution time, since all default parameter settings were used when solving these instances, exploring fine-tuning of these parameters might reduce execution times. In particular, Gurobi has a parameter tuning tool that can be used to efficiently tune parameters.

References

[1] R. M. Karp, "Reducibility among combinatorial problems," in Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds) Complexity of Computer Computations. The IBM Research Symposia Series. Springer, Boston, MA, 1972, pp. 85–103, doi: 10.1007/978-1-4684-2001-2_9.

[2] M. R. Garey, D.S. Johnson. "Computers and Intractability". Freeman, San Francisco. 1979.

[3] F. J. Vasko, F. E. Wolf, and K. L. Stott, "Optimal Selection of Ingot Sizes via Set Covering", *Operations Research,* 35, 346-353, 1987, doi: 10.1287/opre.35.3.346.

[4] F. J. Vasko, F. E. Wolf, and K. L. Stott, "A Set Covering Approach to Metallurgical Grade Assignment," *European Journal of Operational Research,* 38, 27-34, 1989, doi: 10.1016/0377-2217(89)90465-7.

[5] D. D. Newhart, K. L. Stott, and F. J. Vasko, "Consolidating Product Sizes to Minimize Inventory Levels for a Multi-stage Production and Distribution System. *Journal of the Operational Research Society* 44, 7, 637-644, 1993, doi: 10.1038/sj/jors/0440701

[6] F. J. Vasko, F. E. Wolf, K. L. Stott, and O. Ehrsam, "Bethlehem Steel Combines Cutting Stock and Set Covering to Enhance Customer Service". *Mathematical Computer Modelling* 16, 9-17, 1992, doi: 10.1016/0895-7177(92)90075-V.

[7] Z. Feng, H. Okamura, T. Dohi, W.Y. Yun, "Reliability Computing Methods of Probabilistic Location Set Covering Problem Considering Wireless Network Applications", *IEEE Transactions on Reliability*, August 21, 2023, doi: 10.1109/TR.2023.3301929.

[8] J. Bramel, D, Simchi-Levi, "On the Effectiveness of Set Covering Formulations for the Vehicle Routing Problem with Time Windows", *Operations Research*, 45, 2, March-April 1997, doi: 10.1287/opre.45.2.295.

[9] Y. Park, C. S. Ko, I. Moon, "Unmanned aerial vehicle radius set covering problem for emergency wireless network", *Computers and Operations Research,* 170, 2024, doi: 10.1016/j.cor.2024.106765.

[10] S. S. V. Vianna, "The set covering problem applied to optimization of gas detectors in chemical process plants", *Computers and Chemical Engineering,* 121, 388-395, 2019, doi: 10.1016/j.compchemeng.2018.11.008.

[11] F. Khosghgehbari, S, Mohammand, J. Mirzapour Al-e-Hashem, "Ambulance location routing problem considering all sources of uncertainty: Progressive estimating algorithm", *Computers and Operations Research,* 160, 2023, doi: 10.1016/j.cor.2023.106400.

[12] B. Erdebilli, S. G. A. Ozsahin, "Uncertainty management with an autonomous approach to fuzzy set-covering facility location models'', *Journal of Intelligent and Fuzzy Systems*, 46, 8233-8246, 2022, doi: 10.3233/JIFS-213220.

[13] B. S. Vierira, T. Ferrari, G. M. Ribiero, L. Bahiense, R. D. O. Filho, C. A. Abramides, N. F. R. Campos Junior, "A progressive hybrid set covering based algorithm for the traffic counting location problem", *Expert Systems with Applications*, 160, 2020, doi: 10.1016/j.eswa.2020.113641.

[14] M. Yaghini, M. Karimi, M. Rahbar, "A set covering approach for multi-depot train driver scheduling", *Journal of Combinatorial Optimization*, 29, 636-654, 2015, doi: 10.1007/s10878-013-9612-1.

[15] J. Heil, K. Hoffmann, and U. Buscher, "Railway crew scheduling: Models, methods and applications," *European Journal of Operational Research*, 283, 2, pp. 405–425, 2020, doi: 10.1016/j.ejor.2019.06.016.

[16] J. E. Beasley, OR Library, 1990, available at: http://people.brunel.ac.uk/mastjjb/jeb/info.html.

[17] J. E. Beasley, "A Heuristic for the Non-unicost Set Covering Problem Using Local Branching", *International Transactions in Operational Research*, 1-19, 2024. doi: 10.1111/itor.13446, https://doi.org/10.1111/itor.13446.

[18] B. Crawford, R. Soto, H. Mella, de la Fuente, C. Elortegui, W. Palma, C. Torres-Rojas, C. Vasconcellos-Gaete, M. Becerra, J. Peña, S. Misra, "Binary Fruit Fly Swarm Algorithms for the Set Covering Problem", *Computers, Materials & Continua,* 71, 3, 4295-4318, 2022, doi: 10.32604/cmc.2022.023068.

[19] B, Crawford, R. Soto, N. G. Astorga, J. Lemus-Romani, S. Misra, M. Castillo, F. Cisternas-Caneo, D. Tapia, M. Becerra-Rozas, "Balancing Exploration-Exploitation in the Set Covering Problem Resolution with a Self-adaptive Intelligent Water Drops Algorithm", *Advances in Science, Technology and Engineering Systems Journal*, 6. 1. 134-145, 2021, doi: 10.25046/aj060115.

[20] V. Reyes, I. Araya, "A GRASP-based Scheme for the Set Covering Problem", *Operational Research*, 21, 2391-2408, 2021, doi: 10.1007/s12351-019-00514-z.

[21] B. Crawford, R. Soto, R. Olivares, G. Embry, D. Flores, W. Palma, C. Castro, F. Paredes, J. M. Rubio, "A Binary Monkey Search Algorithm Variation for Solving the Set Covering Problem", *Natural Computing*, 19, 825-841, 2020, doi: 10.1007/s11047-019-09752-8.

[22] A. Hashemi, H. Gholami, U. Venkatadri, S. S. Karganroudi, S. Khouri, A. Wojceiechowski, and D. Streimikiene, "A New Direct Coefficient-Based Heuristic Algorithm for Set Covering Problems", *International Journal of Fuzzy Systems,* 24, 2,1131-1147, 2022, doi: 10.1007/s40815-021-01208-5.

[23] T.Adamo, G. Ghiani, E. Guerriero, and D. Pareo, "A Surprisal-Based Greedy Heuristic for the Set Covering Problem", *Algorithms,*16, 321, 2023, doi: 10.3390/a16070321.

[24] C. Luo, W. Xing, S. Cai, and C. Hu, "NuSC: An Effective Local Search Algorithm for Solving the Set Covering Problem", *IEEE transactions on cybernetics*, 54, 3, 2024, doi: 10.1109/TCYB.2022.3199147.

[25] J. E. Beasley, "An algorithm for set covering problem," *European Journal of Operational Research*, vol. 31, no. 1, pp. 85–93, 1987, doi: 10.1016/0377-2217(87)90141-X.

[26] S. Sundar and A. Singh, "A hybrid heuristic for the set covering problem," *Operational Research*, 12, 3, 345–365, 2012, doi: 10.1007/s12351-010-0086-y.

[27] J. E. Beasley and K. Jørnsten, "Enhancing an algorithm for set covering problems," *European Journal of Operational Research,* 58, 2, 293–300, 1992, doi: 10.1016/0377-2217(92)90215-U.

[28] E. Balas and M. C. Carrera, "A dynamic subgradient-based branch-and-bound procedure for set covering," *Operational Research*, vol. 44, no. 6, pp. 875–890, 1996, doi: 10.1287/opre.44.6.875.

[29] F. Carrabs, R. Cerulli, R. Mansini, L. Moreshini, and D. Serra, "Solving the Set Covering Problem with Conflicts on Sets: A new parallel GRASP", *Computers and Operations Research,* 2024, doi: 10.1016/j.cor.2024.106620.

[30] S.Saffari, Y. Fathi, "Set covering problem with conflict constraints", *Computers and Operations Research,* 2022, doi: 10.1016/j.cor.2022.105763.

[31] B. Yelbay, S. I. Birbil, and K. Bulbul. "The Set Covering Problem Revisited: An Empirical Study of the Value of Dual Information", *Journal of Industrial and Management Optimization,* 11, 2, 575-594, 2015, doi: 10.3934/jimo.2015.11.575.

[32] A. Caprara, M. Fischetti, and P. Toth, "A Heuristic Method for the Set Covering Problem", *Opns Res*, 47, 5, 730-743, 1999, doi: 10.1287/opre.47.5.730.

[33] G. Lan, G. DePuy, and G. Whitehouse, "An Effective and Simple Heuristic for the Set Covering Problem", *European Journal of Operational Research,* 176, 3, 1387-1403, 2007, doi: 10.1016/j.ejor.2005.09.028.

[34] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R.E. Bixby, E. Danna, G. Gamrath, A.M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D.E. Steffy, K. Wolter, MIPLIB 2010, *Mathematical Programming Computation,* 3, 2011, doi: 10.1007/s12532-011-0025-9.

[35] T. Koch, A. Martin, M.E. Pfetsch, Progress in academic computational integer programming, in: M. Jünger, G. Reinelt (Eds.), Facets of Combinatorial Optimization, Springer, Berlin, Heidelberg, 483–506, 2013, doi: 10.1007/978-3-642-38189-8_19.

[36] A. Gleixner, G. Hendel, G. Gamrath, T. Achterberg, M. Bastubbe, T. Berthold, P.M. Christophel, K. Jarck, T. Koch, J. Linderoth, M. Lübbecke, H.D. Mittelmann, D. Ozyurt, T.K. Ralphs, D. Salvagnin, Y. Shinano, "MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library," *Mathematical Programming Computation*, 13, pp. 443-490, 2021, doi: 10.1007/s12532-020-00194-3.

[37] E.R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, R. Wunderling, MIP: theory and practice — closing the gap, in: M.J.D. Powell, S. Scholtes (Eds.), System Modelling and Optimization, Springer US, Boston, MA, 2000, pp. 19–49, doi: 10.1007/978-0-387-35514-6_2.

[38] E.R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, R. Wunderling, Mixed-integer programming: a progress report, in: M. Grötschel (Ed.), The Sharpest Cut: The Impact of Manfred Padberg and His Work, SIAM, Philadelphia, PA, 2004, pp. 309–325, doi: 10.1137/1.9780898718805.ch18.

[39] R. E. Bixby, "A Brief History of Linear and Mixed-Integer Programming Computation." *Documenta Mathematica,* 107–121, 2012, doi: 10.4171/dms/6/16.

[40] T. Koch, T. Berthold, J. Pedersen, "Progress in Mathematical Programming Solvers from 2001 to 2020", *EURO Journal on Computational Optimization* 10, 1-18, 2022, doi: 10.1016/j.ejco.2022.100031.

[41] GUROBI Optimizer Reference Manual, 10.0 version, Gurobi Optimization, Beaverton, Oregon, U.S.A. 2023.

[42] User's Manual for CPLEX, 22.1.1 version, IBM, Armonk, New York, U.S.A. 2022.

[43] User's Manual for CPLEX, 12.10 version, IBM, Armonk, New York, U.S.A. 2021.