Review of Computer Engineering Research
2021 Vol. 8, No. 2, pp. 76-95.
ISSN(e): 2410-9142
ISSN(p): 2412-4281
DOI: 10.18488/journal.76.2021.82.76.95
© 2021 Conscientia Beam. All Rights Reserved.

A SIMPLE AND EFFECTIVE METHODOLOGY FOR GENERATING BOUNDED SOLUTIONS FOR THE SET K-COVERING AND SET VARIABLE K-COVERING PROBLEMS: A GUIDE FOR OR PRACTITIONERS

Bryan McNally[1]
Yun Lu[2]
Emre Shively-Ertas[3]
Myung Soon Song[4]
Francis J. Vasko[5+]

[1,3]*Computer Science Department, Kutztown University, Kutztown, PA, USA.*
[1]*Email: bryanjames@mcnally.io Tel: 484-425-0419*
[3]*Email: eshiv901@live.kutztown.edu Tel: 610-360-8734*
[2,4,5]*Department of Mathematics, Kutztown University, Kutztown, PA, USA.*
[2]*Email: lu@kutztown.edu Tel: 610-207-7133*
[4]*Email: song@kutztown.edu Tel: 412-477-2313*
[5]*Email: vasko@kutztown.edu Tel: 610-704-5469*

(+ Corresponding author)

Article History
Received: 13 October 2021
Revised: 1 December 2021
Accepted: 7 December 2021
Published: 10 December 2021

Keywords
Simple sequential increasing tolerance matheuristic
Set K-covering problem
Set variable K-covering problem
General purpose integer programming software.

.

ABSTRACT

As generalizations of the classic set covering problem (SCP), both the set K-covering problem (SKCP) and the set variable (K varies by constraint) K-covering problem (SVKCP) are easily shown to be NP-hard. Solution approaches in the literature for the SKCP typically provide no guarantees on solution quality. In this article, a simple methodology, called the simple sequential increasing tolerance (SSIT) matheuristic, that iteratively uses any general-purpose integer programming software (Gurobi and CPLEX in this case) is discussed. This approach is shown to quickly generate solutions that are guaranteed to be within a tight tolerance of the optimum for 135 SKCPs (average of 67 seconds on a standard PC and at most 0.13% from the optimums) from the literature and 65 newly created SVKCPs. This methodology generates solutions that are guaranteed to be within a specified percentage of the optimum in a short time (actual deviation from the optimums for the 135 SKCPs was 0.03%). Statistical analyses among the five best SKCP algorithms and SSIT demonstrates that SSIT performs as well as the best published algorithms designed specifically to solve SKCPs and SSIT requires no time-consuming effort of coding problem-specific algorithms—a real plus for OR practitioners.

Contribution/Originality: This study documents a methodology that iteratively uses integer programming software to efficiently generate solutions that are guaranteed to be very close to the optimums for the set K-covering problem. A significant benefit of this methodology is that no problem specific algorithm needs to be coded by the user.

1. INTRODUCTION

The set k-covering problem (SKCP) is a generalization of the classic set covering problem. The SKCP involves finding the columns of an m x n 0-1 matrix that cover each row of the matrix at least k times at minimal cost (the classic set covering problem only covers each row at least once). The SKCP can be mathematically represented as the following integer program:

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j \qquad (1)$$

Subject to $\sum_{j=1}^{n} a_{ij}x_j \geq k, \ \forall i \in \{1,2,\ldots,m\}$ (2)

$x_j \in \{0,1\}\ , \ \forall j \in \{1,2,\ldots,n\}$ (3)

Algebraic expression (1) is the objective function which must be minimized. Inequalities (2) are the constraints that guarantee each row *i* is covered by at least k columns, and conditions (3) force each variable x_j to be either 0 or 1. The set variable K-covering problem (SVKCP) has the same formulation as the SKCP except that the value of K can be different for each of the *m* constraints in (2). Note that, to the authors' knowledge, no solution procedures or problem instances currently exist in the literature for the SVKCP.

The SKCP has a number of important real-world applications and plays a significant role in many areas, such as scheduling, location, marketing, logistics, computational biology, and wireless networks [1]. For example, in communication or distribution problems where reliability is important, it is not sufficient to cover each element only once. Another application of the SKCP comes from the domain of bioinformatics. Researchers divide chromosomes into two regions: hot spots, where chromosome recombination takes place, and haplotype blocks, which complement the chromosomes. If we want each pair of the haplotype patterns to be distinguished by k or more mutations or single nucleotide polymorphisms (SNPs), then we have a robust tagging SNP problem which can be modeled as a SKCP [2].

The SVKCP is a straightforward extension of the SKCP in which the K value can differ for each row constraint. Although, to the authors' knowledge, the SVKCP has not been explicitly studied in the literature, one can easily envision real-world applications of the SVKCP. For example, if a geographic area needed to be serviced by fire stations, the optimal location of the fire stations could be formulated as a SVKCP. Each row of the matrix in this SVKCP would represent a neighborhood that should be serviced by at least two fire stations (one primary and one backup), but some neighborhoods might need to be serviced by more than two fire stations (for example, a neighborhood with a hospital in it).

It is important to note that the problems (SKCP and SVKCP) that are being addressed in this research both require that all rows of a matrix be covered by K columns at minimum cost (K varies by row for the SVKCP). However, there are related problems in the literature that have similar names, but different objectives. Specifically, the maximum set K-covering problem (MKCP) consists in selecting a subset of K columns from a given set of n columns, in such a way that the number of rows covered by the selected columns is maximized (see Lin and Guan [3]). Another related problem is the K-set cover problem (K-scp) that seeks to cover all rows of a matrix with the minimum number of columns such that each column chosen can cover at most K rows (see HAE, et al. [4]).

As extensions of the classic set covering problem, both the SKCP and the SVKCP can easily be shown to be NP-hard. To avoid the potential for excessive computing times, up to this point, solutions reported in the literature for the SKCP (note that the SVKCP has not previously been discussed in the literature) have not guaranteed the quality of their solutions.

Normally in the literature, for a set of test problem instances, approximate solution method results are compared to optimal or best-known results that were determined by executing an exact algorithm for a long period of time—sometimes up to 24 hours [5] or more! Researchers assume that, if an approximate solution method performs well on a limited set of problem instances, it will perform well on other problems—this is the weakness of using approximate solution methods with no guaranteed bounds on solution quality. Such approximate solution methods will be discussed in the next section and include methods by Al-Shihabi [6]; Pessoa, et al. [5]; Salehipour [7] and Wang, et al. [8]; Wang, et al. [9].

In this article a methodology first discussed in McNally [10] referred to as the simple sequential increasing tolerance (SSIT) matheuristic is used to quickly solve 135 SKCPs commonly used in the literature and 65 SVKCPs introduced in this article. Lu, et al. [11] used SSIT to quickly (average of 88 seconds on a standard PC) generate solutions guaranteed to be close (average within 0.094% of the optimums) on 270 multidimensional knapsack problem (MKP) instances commonly used in the literature. These results are far better than other published metaheuristic results for the MKP. Dellinger, et al. [12] used SSIT to quickly (average of 63 seconds on a standard PC) generate solutions guaranteed to be close (average within 0.080% of the optimums) on 51 generalized assignment problems (GAP) instances commonly used in the literature. These results are very competitive with the best published solution methods for the GAP.

The key feature of the SSIT solutions is that they are guaranteed to be within a tight tolerance of the optimum. Another important feature of SSIT is its iterative use of any general-purpose integer programming software (Gurobi and CPLEX in this article, but other software could be used just as easily). This combined with a user-defined sequence of loosening tolerances and maximum execution times for each tolerance makes SSIT a very flexible solution methodology. SSIT can be considered a matheuristic because it uses math programming combined with a heuristically determined sequence of tolerances and execution times. Since SSIT takes advantage of the power of a general-purpose exact solution method, no problem-specific algorithm is required which can be a substantial time saver for operations research (OR) practitioners.

In the next section, the relevant literature will be reviewed. Then, a brief overview of the SSIT matheuristic will be provided in Section 3. In Sections 4 empirical results obtained from using the SSIT matheuristic to solve 135 SKCPs will be discussed and statistically compared to the leading published solution approaches specifically designed to solve the SKCP. In Section 5 empirical results for 65 SVKCPs will be presented. This article will close with some conclusions and suggested future work.

2. RELEVANT LITERATURE

The five main solution approaches designed specifically to solve the SKCP that appear in the literature are:

DLLCCSM (diversion local search based on configuration checking and scoring mechanism). This algorithm is presented in Wang, et al. [8] and uses a local search algorithm. First, to overcome the cycling problem in local search, the set k-covering configuration checking (SKCC) strategy is proposed. Second, a cost scheme of elements is used to define a scoring mechanism. Then, the SKCC strategy and scoring mechanism are combined to decide which subset should be selected as a candidate solution component.

MLQCC (multilevel score heuristic with quantitative configuration checking). This algorithm is presented in Wang, et al. [9] and uses a local search algorithm. To overcome the cycling problem in local search, a new quantitative configuration checking (QCC) is proposed. Also, a subset property called subscore, redefines property score [8]. A new multilevel (ML) score heuristic, which is a linear combination of subscore and score is developed.

SALEHIPOUR_HEURISTIC. This algorithm is presented in Salehipour [7] and is a heuristic that first generates a lower bound and then builds a feasible solution from the lower bound. The feasible solution is improved through a removal local search.

LP-MMAS_LS. This algorithm is presented in Al-Shihabi [6] and uses a hybrid algorithm consisting of linear programming, max-min ant system and local search. The algorithm exploits the LP-relaxation solution by classifying the columns based on their reduced costs.

LAGRASP. This algorithm is presented in Pessoa, et al. [5] and uses a hybrid GRASP (greedy randomized adaptive search procedure) Lagrangean heuristic that uses GRASP with a path-relinking heuristic that uses modified costs to obtain approximate solutions.

The performance of these five approximate solution methods on 135 SKCPs will be compared both empirically and statistically in a later section to results obtained from using several SSIT scenarios with Gurobi to solve these same 135 SKCPs. However, it is important to remember that all the above-mentioned solution procedures are explicitly designed to solve the SKCP and more importantly provide no guarantees on solution quality! This is in sharp contrast to our use of SSIT to solve SKCPs because SSIT uses strictly general-purpose software with no time-consuming algorithm coding required and *does* provide bounds on the solutions that it generates.

3. OVERVIEW OF THE SIMPLE SEQUENTIAL INCREASING TOLERANCE MATHEURISTIC

The motivation [10] behind the simple sequential increasing tolerance (SSIT) matheuristic is to try to have the best of two worlds. Namely, SSIT makes use of state-of-the-art optimization software (such as CPLEX or Gurobi) combined with loosening tolerances to obtain solutions that are guaranteed within *known and relatively tight* tolerances of the optimum in a timely manner. By using commercially available and state-of-the-art optimization software instead of highly complex specialized codes for the particular combinatorial optimization problem (COP) being solved, SSIT can be used in a straightforward manner by both OR practitioners as well as researchers with no problem-specific coding required. The SSIT matheuristic is very flexible and robust because the user can specify the number of tolerances as well as their specific values based on their needs. The maximum execution time for each tolerance is also specified based on the specific needs of the user. Although the SSIT concept is very intuitive, this is the first article to discuss and quantify the benefits of using SSIT specifically to solve SKCPs.

As indicated earlier, SSIT can be considered a multi-pass methodology in which the program terminates if the goal tolerance is met. If it is not met, then the tolerance is "loosened" and the *current best solution is used as input for the next step in the solution process.* The "loosened" tolerance allows the branch-and-bound tree in the commercial software to be pruned more quickly. The worst-case scenario for SSIT is that it does not terminate until the sum of the maximum execution times for each tolerance is reached. In this case, the software gap at termination will indicate how close the best SSIT solution is to the optimum. Specifically, for a minimization COP, the optimization software provides the gap between the best lower bound and the best solution.
The pseudo code below summarizes the SSIT methodology for a generic COP.

3.1. SSIT Matheuristic

```
1. Begin
2. Input the number of phases N
3. Input tolerance T_i and maximum execution time t_i for phases i=1, ..., N
4. Input COP details
5. Run integer programming software program to solve COP
6. For 1<=i<=N-1,
7. **IF** integer programming software running time in phase i is less than t_i or i=N, **FINISH**
8. **ELSE,**
9. Take best solution obtained from Phase i and save it as SOL_i.
10. Run integer programming software program with SOL_i as the warm start and tolerance T_{i+1} and
    maximum execution time t_{i+1}.
11. i=i+1
12. **LOOP** through step 7-11 until **FINISH**.
```

The flow chart of SSIT is also provided in Figure 1.

The benefit of SSIT using general purpose integer programming software such as CPLEX or Gurobi and, at the same time, requiring no problem-specific coding is significant. For the problems discussed in this article, all the software default settings were kept except the time and tolerance per SSIT pass. In particular, the OR practitioner or researcher does not need to develop code or test a problem-specific algorithm. Furthermore, practitioners will find that there is a wealth of examples that come with most optimization software (definitely CPLEX and Gurobi), which are ready to run out of the box.

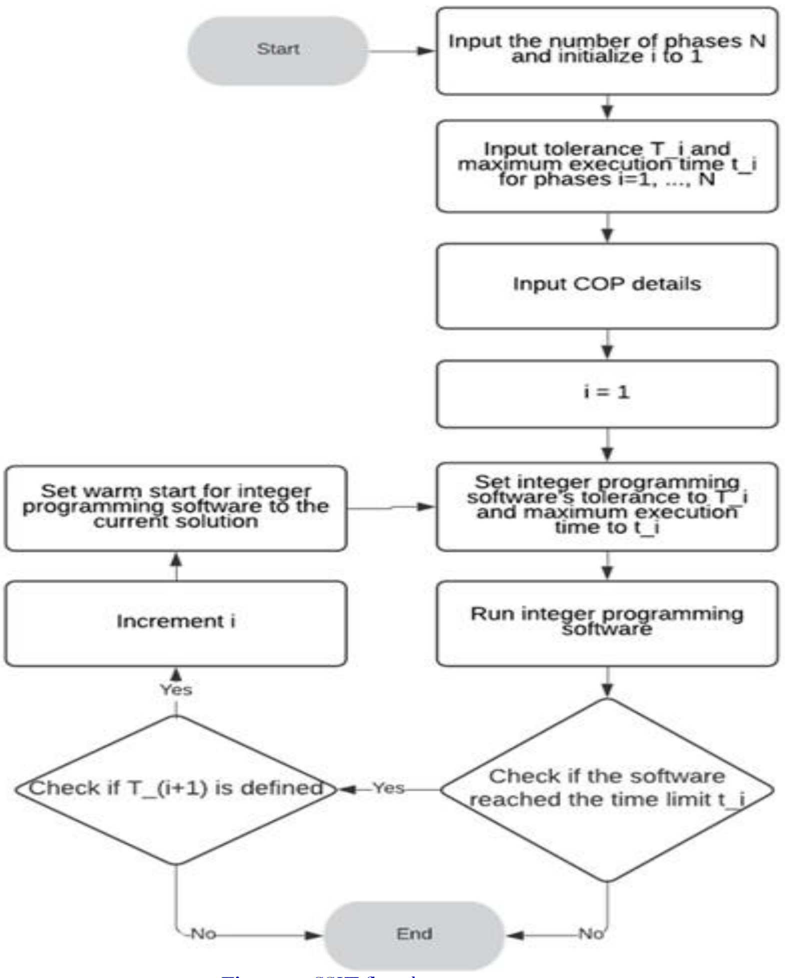
Start
Input the number of phases N and initialize i to 1
Input tolerance T_i and maximum execution time t_i for phases i=1, ..., N
Input COP details
i = 1
Set integer programming software's tolerance to T_i and maximum execution time to t_i
Run integer programming software
Check if the software reached the time limit t_i
Yes
Check if T_(i+1) is defined
Yes
Increment i
Set warm start for integer programming software to the current solution
No
End
No

Figure-1. SSIT flowchart.

These templates often only require a few adjustments before they are ready to run domain specific combinatorial optimization problems. Practitioners can also quickly find answers to many software specific questions in the online forums and extensive manuals. Additionally, for industrial systems that use SSIT, the performance of these systems is "automatically" improved when new versions of the optimization software are installed.

It is important to note that there is no need to "optimize" either the number of tolerances used or their values as well as the execution times for each tolerance. These values are both user and problem specific and can be easily adjusted to meet the users' needs!

Although it is common for OR practitioners to use commercial software at the default tolerance for a fixed amount of time and use the best solution generated when the execution time "runs out", SSIT provides an alternate to this approach that will be shown to provide bounded solutions quickly.

4. SKCP EMPIRICAL RESULTS

4.1. Problem Instances

In Beasley's OR library there are set covering problems that are used by researchers to test algorithms developed to solve the set covering problem (SCP). Sixty-five SCP instances are commonly used to solve weighted set covering problems (WSCP) in which the objective function coefficients are positive integers. Specifically, data sets 4, 5, 6, A, B, C, D, E, F, G, and H are commonly used by researchers. These problem instances are summarized in Table 1 below. In contrast, if the cost coefficients are all the same or typically all equal to one, then the problem is called a minimum cardinality set covering problem (MCSCP) or a uni-cost set covering problem. This current research will focus on solving weighted SKCP and SVKCP instances.

Table-1. WSCP Instances from Beasley's or Library.

Data Set	Rows	Columns	Density	Instances
SCP4	200	1000	2	10
SCP5	200	2000	2	10
SCP6	200	1000	5	5
SCPA	300	3000	2	5
SCPB	300	3000	5	5
SCPC	400	4000	2	5
SCPD	400	4000	5	5
SCPE	50	500	20	5
SCPNRE	500	5000	10	5
SCPNRF	500	5000	20	5
SCPNRG	1000	10,000	2	5
SCPNRH	1000	10,000	5	5

In the literature, it is common for researchers to use the following 135 SKCP instances based on Beasley's SCP instances. Specifically, researchers use the 45 problem instances in data sets 4, 5, 6, A, B, C, and D with three different K values. How the K values are determined will now be given. KMIN = 2 for all 45 problem instances. KMAX has the same K value for all rows of a problem, but can differ for the 45 problem instances. For a given problem instance (from data sets 4, 5, 6, A, B, C, and D), the KMAX value is equal to the sum of the ones in a row with the minimum number of ones. KMED is equal to the ceiling of (KMIN + KMAX)/2. Hence, researchers typically report algorithm results by K value, KMIN, KMED, and KMAX.

4.2. SSIT Results for the SKCP

In order to use the SSIT matheuristic to solve SKCPs, a sequence of increasing tolerances and corresponding maximum execution times must be specified and an integer programming software package must be selected. The purpose of this article is to demonstrate that the SSIT matheuristic works with any integer programming software package. Obviously, the better the selected optimization software is in terms of generating good solutions quickly, the better the solution generated by SSIT.

The authors considered two leading optimization software packages: CPLEX (12.9) and Gurobi (9.1). Both of these are highly sophisticated general purpose optimization packages that are easy for practitioners and researchers to use to solve large-scale integer programming problems in particular. Empirical testing on more than 50 combinatorial optimization problems using both CPLEX and Gurobi resulted in no statistical difference in either solution quality or execution time required given the same parameter settings for both software packages. Hence, the authors decided to use both: Gurobi for the empirical analysis of the SKCPs and CPLEX for the empirical analysis of the new SVKCPs (discussed in Section 5). Also, to demonstrate that SSIT is not PC dependent, two different PCs were used—one for the SKCP analyses and a different one for the SVKCP analyses. The SKCPs will be analyzed using Gurobi on a computer with the following specifications: an AMD Ryzen 7 3700X 8-Core Processor and 16 GB RAM on Windows 10 Home 64-bit. The number of threads is 8. The SVKCPs will be analyzed using CPLEX on a computer with the following specifications: 16 GB RAM on Windows 10, Intel processor with 2.9 GHz, and 1000 GB hard drive. By default, CPLEX uses a number of threads equal to the number of cores or 32 threads (whichever number is smaller). The operating system manages any contention for processors. The PC used has 4 cores, so the number of threads is 4. A thorough discussion of the performance of the latest versions of CPLEX versus Gurobi for solving combinatorial optimization problems is a topic for another article. In this article the power and robustness of SSIT regardless of the integer programming software used or the PC used is demonstrated.

For the SSIT analysis of the SKCP, three SSIT scenarios (SSIT1, SSIT2, and SSIT3) were used. Limited preliminary empirical analysis using the PC specified above for SKCPs indicated that tight bounds (less than 1%) could be achieved in under 600 seconds for even the most difficult problems tested. Hence for this application, the total execution times would add to 600 seconds. Three SSIT scenarios will demonstrate how shifting more execution time to the looser tolerances will for these 135 SKCPs reduces the average execution time. The specifics of these three SSIT scenarios are given in Table 2.

Table-2. SSIT scenario execution times (seconds) for each tolerance.

	Tolerances					
	0.0001	**0.001**	**0.003**	**0.005**	**0.007**	**0.009**
SSIT1	60	60	120	120	120	120
SSIT2	60	60	60	120	120	180
SSIT3	30	60	90	120	120	180

The results of executing the three SSIT scenarios for the 135 SKCPs are summarized in Table 3, 4, and 5. Detailed results for all 135 SKCPs are available upon request.

Table-3. Summary results averaged over 135 SKCPs.

SSIT scenario	Average guaranteed maximum deviation from optimum (%)	Average actual deviation from optimum (%)	Average execution time (seconds)
SSIT1	0.136	0.032	84
SSIT2	0.137	0.032	75
SSIT3	0.131	0.030	67

In Table 3, results for each of the three SSIT scenarios (SSIT1, SSIT2, and SSIT3) are averaged over all 135 SKCPs. The guaranteed maximum deviation from the optimum column shows the *farthest* away the SSIT solutions can be without knowing the exact value of the optimums. Over all 135 SKCPs, the average guaranteed farthest deviations the SSIT solutions are from the optimums are 0.136%, 0.137%, and 0.131% for SSIT1, SSIT2, and SSIT3 respectively. However, comparing these 135 SSIT solutions to known optimums or best-known solutions, the SSIT solutions, on average, actually only deviated 0.032%, 0.032%, and 0.030% from the optimums for SSIT1, SSIT2, and SSIT3 respectively.

Additionally, these very impressive results required, on average, only 84 seconds, 75 seconds, and 67 seconds for SSIT1, SSIT2, and SSIT3 respectively. For SSIT3, the execution times were 180 seconds or less for 120 of the 135 SKCPs (89%) and 120 seconds or less for 113 of the 135 SKCPs (84%). There is next to no differences among the three SSIT scenarios in terms guaranteed maximum deviation from the optimum and actual deviation from the optimum. However, the SSIT3 scenario has the smallest average execution time of 67 seconds which is a 20% reduction over the SSIT1 average execution time and an 11% reduction over the SSIT2 average execution time.

Table 4 shows average execution time and deviations for each SSIT scenario by KMIN, KMED, and KMAX. The results in Table 4 indicate that, regardless of the SSIT scenario, the KMED SKCPs require the most effort to solve and the KMIN SKCPs require the least effort to solve. However, regardless of SSIT scenario, even for the KMED problems, the average times are less than 180 seconds. Except for time, the results differ very little by SSIT scenario.

Table 5 shows the distribution of the tolerances at which SSIT terminated based on SSIT scenario and K value. As was evident from earlier analyses, the results differ very little based on the particular SSIT scenario. For example, regardless of SSIT scenario, all 45 KMIN SKCPs terminated at the tightest tolerance of 0.0001. As previously observed, again regardless of SSIT scenario, the KMED problems required the most effort to solve.

Table-4. Average execution time and deviations for each SSIT scenario by KMIN, KMED, and KMAX.

	Average guaranteed maximum deviation from optimum (%)	Average actual deviation from optimum (%)	Average execution time (seconds)
KMIN			
SSIT1	0	0	0.5
SSIT2	0	0	0.5
SSIT3	0	0	0.5
KMED			
SSIT1	0.296	0.071	178.6
SSIT2	0.298	0.070	151.4
SSIT3	0.295	0.065	145.0
KMAX			
SSIT1	0.111	0.025	73.0
SSIT2	0.113	0.027	71.9
SSIT3	0.114	0.023	54.4

The robustness of SSIT is that knowing the difficulty of the KMED instances (for example, from preliminary empirical analysis), if the OR practitioner wanted tighter guaranteed bounds for these problems and more computer time was not an issue, more time could be spent at the tighter tolerances. Exactly how much time at each tolerance would depend on the particular application. However, regardless of the SSIT scenario, the KMED SKCPs had a guaranteed maximum deviation from the optimum of under 0.3% (actual deviation of 0.07%) and average execution times under 180 seconds.

Table-5. Termination tolerances distributions for each SSIT scenario by KMIN, KMED, and KMAX.

	Tolerances					
	0.0001	**0.001**	**0.003**	**0.005**	**0.007**	**0.009**
KMIN						
SSIT1	45					
SSIT2	45					
SSIT3	45					
KMED						
SSIT1	19	2	6	3	10	5
SSIT2	19	2	4	5	11	4
SSIT3	19	1	6	4	11	4
KMAX						
SSIT1	20	2	23			
SSIT2	20	3	20	2		
SSIT3	16	7	21	1		

Note: Comparisons of our SSIT results with the best published algorithms specialized specifically to solve the SKCP will now be given.

4.3. SKCP SSIT Results Compared to Other Metaheuristics

Although, as operations research practitioners, the authors appreciate the guaranteed bounds that the SSIT matheuristic provides, there may be readers that are interested in seeing how the SSIT solutions compared to the five best performing metaheuristics for the SKCP reviewed earlier in this article. In Table 6, 7, and 8, the SSIT results for the 135 SKCPS typically used for empirical experiments by researchers will be compared to Wang, et al. [8]; Wang, et al. [9]; Salehipour [7]; Al-Shihabi [6] and Pessoa, et al. [5] discussed earlier in this article.

Table-6A. Comparison of SSIT with Other Metaheuristics.

PROBLEM	KMIN	OPT/BKS	ALSHIHIBA	PESSOA	SALEH	WANG 2017	WANG 2019	SSIT1	SSIT2	SSIT3
SCP41	2	1148	1150	1150	1150	1148	1148	1148	1148	1148
SCP 42	2	1205	1205	1205	1205	1205	1205	1205	1205	1205
SCP 43	2	1213	1213	1214	1214	1213	1213	1213	1213	1213
SCP 44	2	1185	1189	1185	1185	1185	1185	1185	1185	1185
SCP 45	2	1266	1266	1266	1266	1266	1266	1266	1266	1266
SCP 46	2	1349	1349	1349	1352	1349	1349	1349	1349	1349
SCP 47	2	1115	1115	1115	1115	1115	1115	1115	1115	1115
SCP 48	2	1225	1225	1225	1225	1225	1225	1225	1225	1225
SCP 49	2	1485	1485	1485	1485	1485	1485	1485	1485	1485
SCP 410	2	1356	1360	1356	1359	1356	1356	1356	1356	1356
SCP 51	2	579	580	579	579	579	579	579	579	579
SCP 52	2	677	677	679	677	677	677	677	677	677
SCP 53	2	574	576	574	575	574	574	574	574	574
SCP 54	2	582	584	587	585	582	582	582	582	582
SCP 55	2	550	550	550	550	550	550	550	550	550
SCP 56	2	560	560	560	561	560	560	560	560	560
SCP 57	2	695	695	695	695	695	695	695	695	695
SCP 58	2	662	664	662	664	662	662	662	662	662
SCP 59	2	687	687	687	687	687	687	687	687	687
SCP 510	2	672	672	672	672	672	672	672	672	672
SCP 61	2	283	283	283	283	283	283	283	283	283
SCP 62	2	302	302	302	302	302	302	302	302	302
SCP 63	2	313	313	313	313	313	313	313	313	313
SCP 64	2	292	292	292	294	292	292	292	292	292
SCP 65	2	353	353	353	353	353	353	353	353	353

Table-6B. Comparison Of SSIT With Other Metaheuristics Results For Kmin Skcp Data Sets A, B, C, and D.

PROBLEM	KMIN	OPT/BKS	AL-SHIHIBA	PESSOA	SALEH	WANG 2017	WANG 2019	SSIT1	SSIT2	SSIT3
SCPA1	2	562	562	563	563	562	562	562	562	562
SCPA2	2	560	564	560	560	560	560	560	560	560
SCPA3	2	524	526	524	524	524	524	524	524	524
SCPA4	2	527	527	527	527	527	527	527	527	527
SCPA5	2	557	558	559	558	557	557	557	557	557
SCPB1	2	149	150	149	149	149	149	149	149	149
SCPB2	2	150	150	151	150	150	150	150	150	150
SCPB3	2	165	165	165	165	165	165	165	165	165
SCPB4	2	157	157	157	157	157	157	157	157	157
SCPB5	2	151	152	152	151	151	151	151	151	151
SCPC1	2	514	516	515	515	514	514	514	514	514
SCPC2	2	483	490	486	483	483	483	483	483	483
SCPC3	2	544	546	544	545	544	544	544	544	544
SCPC4	2	484	485	485	484	484	484	484	484	484
SCPC5	2	488	488	490	489	488	488	488	488	488
SCPD1	2	122	123	122	122	122	122	122	122	122
SCPD2	2	127	127	127	127	127	127	127	127	127
SCPD3	2	138	138	138	138	138	138	138	138	138
SCPD4	2	122	123	123	122	122	122	122	122	122
SCPD5	2	130	131	130	130	130	130	130	130	130

Table-7A. Comparison Of SSIT With Other Metaheuristics Results For Kmed Skcp Data Sets 4, 5, and 6.

PROBLEM	KMED	OPT/BKS	AL-SHIHIBA	PESSOA	SALEH	WANG 2017	WANG 2019	SSIT1	SSIT2	SSIT3
SCP41	7	8350	8373	8366	8363	8352	8352	8350	8350	8350
SCP 42	6	6111	6123	6117	6118	6111	6111	6111	6111	6111
SCP 43	5	4676	4681	4690	4681	4676	4676	4676	4676	4676
SCP 44	5	4670	4683	4679	4674	4670	4670	4670	4670	4670
SCP 45	7	8389	8400	8409	8398	8392	8389	8389	8389	8389
SCP 46	6	6416	6427	6432	6419	6416	6416	6416	6416	6416
SCP 47	6	6281	6282	6284	6282	6281	6281	6281	6281	6281
SCP 48	7	8421	8435	8439	8427	8427	8424	8421	8421	8421
SCP 49	6	7101	7127	7121	7106	7101	7101	7101	7101	7101
SCP 410	5	5355	5367	5364	5358	5355	5355	5355	5355	5355
SCP 51	13	11205	11226	11239	11213	11209	11206	11205	11205	11205
SCP 52	14	14418	14443	14473	14436	14428	14424	14418	14418	14418
SCP 53	13	11476	11532	11513	11488	11487	11476	11476	11476	11476
SCP 54	12	9944	9970	9965	9956	9950	9948	9944	9944	9944
SCP 55	12	10880	10888	10918	10898	10895	10881	10880	10880	10880
SCP 56	12	10581	10609	10629	10597	10591	10582	10581	10581	10581
SCP 57	14	14919	14940	14984	14934	14946	14924	14919	14919	14923
SCP 58	12	10622	10539	10687	10635	10623	10622	10622	10622	10622
SCP 59	12	11042	11071	11081	11053	11049	11047	11042	11042	11042
SCP 510	13	12436	12469	12475	12451	12450	12436	12436	12436	12436
SCP 61	17	7653	7679	7692	7669	7653	7653	7653	7653	7653
SCP 62	16	6739	6760	6773	6752	6739	6739	6747	6747	6746
SCP 63	18	8309	8350	8365	8317	8309	8309	8309	8309	8309
SCP 64	18	8546	8569	8585	8567	8546	8546	8546	8546	8546
SCP 65	18	9038	9068	9070	9060	9038	9038	9038	9038	9038

Table-7B. Comparison Of SSIT With Other Metaheuristics Results For Kmed Skcp Data Sets A, B, C, and D

PROBLEM	KMED	OPT/BKS	AL-SHIHIBA	PESSOA	SALEH	WANG 2017	WANG 2019	SSIT1	SSIT2	SSIT3
SCPA1	21	21224	21297	21324	21281	21241	21224	21236	21236	21241
SCPA2	21	21739	21810	21820	21793	21750	21740	21749	21753	21744
SCPA3	21	20095	20165	20155	20148	20126	20097	20113	20115	20102
SCPA4	22	22865	22959	22985	22916	22880	22880	22875	22866	22864
SCPA5	20	18643	18709	18706	18694	18660	18648	18641	18641	18646
SCPB1	61	29145	29214	29234	29218	29184	29145	29201	29211	29188
SCPB2	60	28075	28175	28187	28196	28124	28075	28135	28128	28109
SCPB3	59	27825	27934	27944	27899	27852	27825	27891	27878	27878
SCPB4	58	25664	25764	25742	25773	25695	25664	25707	25694	25703
SCPB5	60	28188	28295	28297	28310	28262	28188	28239	28247	28245
SCPC1	30	32613	32730	32763	32761	32648	32613	32646	32676	32639
SCPC2	31	32705	32837	32871	32848	32745	32705	32749	32776	32777
SCPC3	31	34428	34553	34610	34542	34451	34428	34496	34487	34477
SCPC4	30	31329	31466	31495	31472	31372	31329	31397	31364	31374
SCPC5	29	30030	30116	30196	30177	30061	30030	30082	30077	30087
SCPD1	82	38935	39091	39132	39073	38991	38935	39012	39022	39022
SCPD2	83	38935	39090	39098	39116	39038	38935	39035	39035	39037
SCPD3	81	39134	39256	39271	39314	39221	39134	39209	39209	39209
SCPD4	82	38723	38835	38879	38894	38814	38723	38798	38794	38814
SCPD5	83	40268	40401	40409	40404	40362	40268	40338	40342	40332

Table-8A. Comparison Of SSIT With Other Metaheuristics Results For Kmax Skcp Data Sets 4, 5, and 6.

PROBLEM	KMAX	OPT/BKS	AL-SHIHIBA	PESSOA	SALEH	WANG 2017	WANG 2019	SSIT1	SSIT2	SSIT3
SCP41	11	18265	18265	18290	18273	18265	18265	18265	18265	18265
SCP 42	9	12360	12378	12405	12369	12370	12360	12360	12360	12360
SCP 43	8	10396	10397	10398	10396	10403	10396	10396	10396	10396
SCP 44	8	10393	10415	10427	10401	10396	10395	10393	10393	10393
SCP 45	11	18856	18861	18856	18863	18856	18856	18856	18856	18856
SCP 46	10	15394	15426	15419	15411	15404	15394	15394	15394	15394
SCP 47	10	15233	15261	15280	15249	15236	15233	15233	15233	15233
SCP 48	11	18602	18635	18628	18610	18613	18612	18602	18602	18602
SCP 49	10	16558	16586	16591	16563	16568	16562	16558	16558	16558
SCP 410	8	11607	11615	11618	11616	11607	11607	11607	11607	11607
SCP 51	24	35663	35722	35749	35679	35716	35685	35670	35670	35671
SCP 52	26	45396	45449	45433	45412	45428	45409	45396	45396	45396
SCP 53	24	36329	36374	36388	36349	36368	36343	36329	36329	36329
SCP 54	21	28017	28044	28051	28037	28035	28026	28017	28017	28017
SCP 55	22	32779	32808	32878	32795	32802	32788	32779	32779	32779
SCP 56	21	29608	29656	29653	29632	29632	29618	29608	29608	29608
SCP 57	25	41930	41964	41954	41944	41956	41946	41930	41930	41930
SCP 58	22	32320	32358	32405	32344	32344	32332	32320	32320	32320
SCP 59	22	33584	33600	33655	33602	33608	33599	33584	33584	33584
SCP 510	24	38709	38779	38807	38737	38756	38730	38709	38709	38709
SCP 61	31	23510	23559	23534	23536	23510	23510	23525	23523	23510
SCP 62	29	19934	19980	20025	19964	19940	19934	19952	19953	19956
SCP 63	34	27983	28021	28027	28014	27983	27983	27983	27983	27994
SCP 64	33	26442	26477	26530	26475	26446	26446	26456	26449	26450
SCP 65	33	27069	27090	27124	27084	27069	27069	27071	27071	27071

Table-8B. Comparison Of SSIT With Other Metaheuristics Results For Kmax Skcp Data Sets A, B, C, and D.

PROBLEM	KMAX	OPT/BKS	AL-SHIHIBA	PESSOA	SALEH	WANG 2017	WANG 2019	SSIT1	SSIT2	SSIT3
SCPA1	40	68507	68620	68669	68579	68590	68528	68518	68518	68518
SCPA2	39	65842	65940	65922	65881	65927	65863	65861	65861	65860
SCPA3	40	66829	66978	67016	66879	66891	66864	66839	66860	66863
SCPA4	41	72334	72436	72465	72398	72398	72351	72342	72338	72342
SCPA5	38	60491	60609	60625	60553	60539	60498	60503	60503	60503
SCPB1	119	105491	105580	105636	105522	105560	105491	105532	105532	105532
SCPB2	118	102883	102988	103046	103007	102941	102883	102937	102937	102912
SCPB3	115	98255	98334	98445	98400	98347	98255	98311	98311	98311
SCPB4	114	93729	93797	93836	93807	93800	93729	93783	93783	93760
SCPB5	118	102761	102878	102905	102822	102867	102761	102829	102832	102832
SCPC1	58	112471	112595	112667	112557	112565	112476	112538	112524	112501
SCPC2	59	113916	114017	114145	113974	114012	113925	113950	113950	113948
SCPC3	59	117416	117505	117680	117544	117501	117446	117454	117454	117449
SCPC4	58	110823	110945	111091	110935	110938	110851	110869	110869	110894
SCPC5	56	104428	104509	104591	104506	104518	104428	104457	104488	104427
SCPD1	162	144815	144891	145060	145055	144961	144815	144891	144935	144941
SCPD2	163	144020	144165	144218	144177	144138	144020	144182	144172	144141
SCPD3	159	140450	140542	140685	140655	140589	140450	140530	140609	140533
SCPD4	162	143391	143470	143582	143544	143488	143391	143504	143504	143497
SCPD5	163	146249	146308	146452	146373	146342	146249	146294	146294	146294

To try to determine how good heuristic solutions are, Pessoa, et al. [5] ran each of the 135 problem instances for up to 24 hours using CPLEX (Version 11). Pessoa, et al. [5] found proven optimal solutions for all 45 KMIN instances, for 25 KMED instances, and for 25 KMAX instances. Researchers typically compare their results to these CPLEX results. The best-known solutions of Pessoa, et al. [5] were updated based on better solutions obtained by Wang, et al. [9] (in blue in the tables). Best known solutions were also updated based on SSIT results (in red in the tables).

Table 6, 7, and 8 compare the results of using SSIT1, SSIT2, and SSIT3 as described previously in this article with the *best* results (over 10 independent runs) reported in Wang, et al. [8]; Wang, et al. [9]; Salehipour [7]; Al-Shihabi [6] and Pessoa, et al. [5] to solve the 45 SKCPs at KMIN, KMED, and KMAX respectively.

To make a simple comparison among the five previously published best performing SKCPs and the three SSIT scenario results for the 135 test instances, the average deviation from the optimum or best-known solution for each of the five solution methods are calculated over all 135 test instances. The average deviations from the optimum/BKS objective function value over all 135 problems for Al-Shihabi [6] was 0.20%, for Pessoa, et al. [5] was 0.23%, for Salehipour [7] was 0.13%, for Wang, et al. [8] was 0.05%, and for Wang, et al. [9] was 0.01%. All three SSIT scenarios had the same (to two decimal places) average deviation from the optimum/BKS objective function value over all 135 problems of 0.03%. All of these solution procedures generated very good results for the 135 test SKCP instances, with SSIT giving the second-best results at only an average 0.03% deviation from the optimums. Detailed statistical analyses of these eight solution methods will be provided in Section 4.4. All three SSIT scenarios provide excellent bounds (*guaranteed* to be on average within a tolerance of 0.136%, 0.137%, and 0.131% of the optimums) for these 135 SKCPs with respectable average execution time of 84, 75, and 67 seconds for SSIT1, SSIT2, and SSIT3 respectively.

Let's look a little closer at the major advantage that SSIT has over the other approximate solutions methods specifically when solving the SKCP. It was just shown that Wang, et al. [9] on average only deviated 0.01% from the optimums for these 135 SKCPs. In contrast, the three SSIT scenarios all performed "poorly" with an average deviation of 0.03% from the optimum for each SSIT scenario. The real strength of SSIT, in the opinion of the authors of this article, is the fact that *without expending excessive* computer time to determine optimums, the SSIT scenario results are on average *absolutely guaranteed* to be within 0.136%, 0.137%, and 0.131% of the optimums for SSIT1, SSIT2, and SSIT3 respectively. Additionally, SSIT obtains these excellent results using general purpose commercial software with default parameter settings—no specialized algorithms or computer codes are needed. Furthermore, as pointed out earlier, leading integer programming software packages such as CPLEX and Gurobi offer a large number of pre-defined problem templates and customer support.

Finally, suppose that an OR practitioner was faced with the need to solve a SKCP for an actual real-world application and had access to CPLEX or Gurobi. The OR practitioner could code the MLQCC algorithm of Wang, et al. [9] and solve his/her problem using MLQCC and obtain a solution that was *probably* very good, but exactly how good would be totally unknown. Alternately, The OR practitioner could easily use SSIT in conjunction with CPLEX or Gurobi. If the decision would have major financial implications for his/her corporation and the SKCP was very large, the OR practitioner might very well be willing to invest several hours of computing time to generate a solution that had a reasonably tight bound on it. If the reader was the OR practitioner, would the reader use the Wang, et al. [9]; Salehipour [7] MLQCC algorithm or the SSIT methodology?

4.4. Statistical Analyses

In this section, with the 135 SKCP instances (45 KMIN instances, 45 KMED instances, and 45 KMAX instances), the three SSIT scenarios (SSIT1, SSIT2, and SSIT3) suggested in this article are compared to the top five SKCP solution methods Wang, et al. [8]; Wang, et al. [9]; Salehipour [7]; Al-Shihabi [6] and Pessoa, et al. [5] in the literature. Tukey's pairwise comparison (after significant differences among the eight methods are detected from the one-way repeated measures ANOVA) is used for the three K values (KMIN, KMED, and KMAX). Tukey [13] the common significant level at 5% is applied in the analyses.

In the tables of the following sub-sections, Wang, et al. [8]; Wang, et al. [9], Saleh, Al-Shihiba, and Pessoa indicate Wang, et al. [8]; Wang, et al. [9]; Salehipour [7]; Al-Shihabi [6] and Pessoa, et al. [5] respectively. Table 9, 10, and 11 summarize the pairwise comparisons among the eight methods in terms of percent deviation from the optimum for KMIN, KMED, and KMAX, respectively. Note that the percent deviation from the optimum is getting smaller in order of A, B, C, and D, and methods that do not share a letter are significantly different.

4.4.1. Comparison – KMIN

Readers can check many results from Table 9 including (1) there is no statistically significant difference between Al_Shihiba and Pessoa and (2) there is no statistically significant difference among Saleh, Wang, et al. [8]; Wang, et al. [9], and the three SSIT scenarios (SSIT1, SSIT2, and SSIT3).

Table-9. Summary – Three SSIT scenarios vs. Top five methods. with KMIN.

Grouping Information Using the Tukey Method and 95% Confidence.

Method	N	Mean	Grouping		
Al_Shihiba	45	0.0020897	A		
Pessoa	45	0.0012286	A	B	
Saleh	45	0.0007770		B	C
SSIT2	45	0.0000000			C
WANG 2019	45	0.0000000			C
WANG 2017	45	0.0000000			C
SSIT3	45	0.0000000			C
SSIT1	45	0.0000000			C

Note: Means that do not share a letter are significantly different.

4.4.2. Comparison – KMED

In Table 10, readers can check many results such as (1) Pessoa shows the worst performance, (2) there is no statistically significant difference among Wang, et al. [8] SSIT1, SSIT2, and SSIT3 and (3) Wang, et al. [9] shows the best performance but the difference between SSIT3 and Wang, et al. [9] is not statistically significant.

Table-10. Summary – Three SSIT scenarios vs. Top five methods with KMED.

Grouping Information Using the Tukey Method and 95% Confidence.

Method	N	Mean	Grouping			
Pessoa	45	0.0037011	A			
Al_Shihiba	45	0.0026969		B		
Saleh	45	0.0022919		B		
WANG 2017	45	0.0008584			C	
SSIT1	45	0.0007062			C	
SSIT2	45	0.0007020			C	
SSIT3	45	0.0006546			C	D
WANG 2019	45	0.0000800				D

Note: Means that do not share a letter are significantly different.

4.4.3. Comparison – KMAX

Readers can check many results from Table 11 including (1) there is no statistically significant difference between Saleh and Wang, et al. [8] and (2) there is no statistically significant difference among the four best performing methods - Wang, et al. [9] SSIT1, SSIT2, and SSIT3.

Table-11. Summary – Three SSIT scenarios vs. Top five methods with KMAX.

Grouping Information Using the Tukey Method and 95% Confidence.

Method	N	Mean	Grouping			
Pessoa	45	0.0018616	A			
Al_Shihiba	45	0.0011707		B		
Saleh	45	0.0008030			C	
WANG 2017	45	0.0006769			C	
SSIT2	45	0.0002701				D
SSIT1	45	0.0002488				D
SSIT3	45	0.0002315				D
WANG 2019	45	0.0001604				D

Note: Means that do not share a letter are significantly different.

In the next section, a test set of 65 SVKCP instances will be defined and solved using SSIT.

5. SVKCP EMPIRICAL RESULTS

5.1. SVKCP Data Sets

Since the SKCP instances commonly used in the literature to test SKCP solution approaches are based on SCPs from Beasley's OR-library, the SVKCPs defined now will also be based on these SCPs as well as the SKCPs previously discussed. The 65 SCPs from Beasley's data sets 4, 5, 6, A, B, C, D, E, F, G, and H will be used to create SVKCPs.

Recall that, for a given SCP, KMAX is equal to the sum of the ones in a row with the minimum number of ones. For each of the 65 SCPs mentioned above, the following 65 SVKCPs will be defined by randomly selecting integer K values for *each* row of the problem from the integers in the interval [2, KMAX]. In other words, instead of a fixed K value for all the rows of the problem (as is the case for SKCPs), the K values vary by row, but are integers randomly taken from the integers in the interval [2, KMAX]. For example, for SCP41, the interval would be from 2 to 11, so each row in the corresponding full interval SVKCP would have a K value of either 2, 3, 4, ...,10, or 11. The actual K values used for each row for the 65 SVKCPs, are available on the cloud at SVKCP_K-values.xlsx.

5.2. SSIT for the SVKCP

To demonstrate that SSIT performs well on other PCs, software, and scenarios, the 65 SVKCPs just defined were solved using CPLEX on a PC with specifications: 16 GB RAM on Windows 10, Intel processor with 2.9 GHz, and 1000 GB hard drive. By default, CPLEX uses a number of threads equal to the number of cores or 32 threads (whichever number is smaller). The operating system manages any contention for processors. The PC used has 4 cores, so the number of threads is 4. For these SVKCPs, only one SSIT scenario is used. Specifically, the SSIT scenario used for the 65 SVKCPs was 0.001 at 300 seconds, 0.003 at 60 seconds, and 0.005 for 60 seconds. This scenario contrasts with the SSIT scenarios used for the 135 SKCPs because for this scenario most of the time is spent at the tightest tolerance. The SSIT solution details for these SVKCPs are provided in Table 12.

From Table 12, one can see that all SSIT solutions for the 65 SVKCPs are guaranteed to be within 0.1% of the optimum and only required an average of 12.2 seconds to solve each SVKCP and 120 out of the 130 SVKCPs (over 92%) required less than 2 seconds of solution time. To demonstrate the power of the SSIT matheuristic, these 65 SVKCPs were solved in CPLEX with a tolerance of T= 0.0001 (the default) and up to one hour of execution time. Of these 65 SVKCPs, 47 terminated within the one-hour time limit and hence had found solutions guaranteed to be within 0.01% of the optimum—optimum for all intent purposes. However, 18 had not terminated after 3600 seconds of execution time.

Table-12. SSIT Results for 65 SVKCPs.

	T=0.001			T=0.001	
Problem	**OBJ FN**	**Time**	**Problem**	**OBJ FN**	**Time**
SVKCP41	11019	0.14	SVKCPB1	65005	0.13
SVKCP42	6773	0.08	SVKCPB2	62609	0.47
SVKCP43	5619	0.09	SVKCPB3	59285	0.22
SVKCP44	6408	0.17	SVKCPB4	59069	0.25
SVKCP45	9860	0.03	SVKCPB5	63159	0.27
SVKCP46	8094	0.08	SVKCPC1	62757	0.61
SVKCP47	8355	0.06	SVKCPC2	62543	0.20
SVKCP48	9829	0.13	SVKCPC3	66477	1.19
SVKCP49	9153	0.17	SVKCPC4	63568	0.83
SVKCP410	6926	0.16	SVKCPC5	57532	0.64
SVKCP51	19615	0.14	SVKCPD1	99270	0.95
SVKCP52	25113	0.16	SVKCPD2	96204	0.66
SVKCP53	16488	0.14	SVKCPD3	97091	0.94
SVKCP54	15731	0.13	SVKCPD4	96351	1.84
SVKCP55	16488	0.09	SVKCPD5	95537	0.42
SVKCP56	14720	0.11	SVKCPNRE1	175154	1.02
SVKCP57	22186	0.13	SVKCPNRE2	174079	1.14
SVKCP58	18163	0.20	SVKCPNRE3	166165	1.56
SVKCP59	16318	0.08	SVKCPNRE4	161895	1.41
SVKCP510	22296	0.05	SVKCPNRE5	162255	1.14
SVKCP61	13185	0.22	SVKCPNRF1	178893	1.70
SVKCP62	11868	0.41	SVKCPNRF2	189989	1.59
SVKCP63	15527	0.67	SVKCPNRF3	189785	1.83
SVKCP64	15040	0.27	SVKCPNRF4	188813	1.13
SVKCP65	15847	0.36	SVKCPNRF5	196674	1.61
SVKCPA1	39046	0.44	SVKCPNRG1	217562	227.14
SVKCPA2	36828	0.34	SVKCPNRG2	215316	2.61
SVKCPA3	37781	0.16	SVKCPNRG3	219089	52.31
SVKCPA4	41455	0.22	SVKCPNRG4	221559	184.17
SVKCPA5	34844	0.39	SVKCPNRG5	227274	281.81
			SVKCPNRH1	317145	3.00
			SVKCPNRH2	320148	3.64
			SVKCPNRH3	319924	3.53
			SVKCPNRH4	315907	3.33
			SVKCPNRH5	320799	3.50

Although **all** SSIT solutions for the 65 SVKCPs were guaranteed within 0.1% of the optimum, how did these solutions compare to the solutions that were found when CPLEX was executed at the default T=0.0001 tolerance? As a simple comparison, the average objective function value for the 65 SVKCPs executed at T=0.0001 was 94042.6 and the average execution time was 1054.9 seconds compared to the SSIT average objective function value of 94084.0 and an average execution time of 12.2 seconds. Hence, after the fact, the SSIT solutions were off from the solutions obtained by CPLEX with T = 0.0001 by 0.04%, but the SSIT execution time was only 1.2% of the CPLEX execution time. Note that, even "before the fact", the SSIT solutions were *guaranteed* within 0.1% of the optimum.

6. SUMMARY AND FUTURE WORK

In this article the simple sequential increasing tolerance (SSIT) matheuristic is used to solve generalizations of the classic set covering problem such that bounded solutions are efficiently generated. This multi-pass matheuristic is used in conjunction with integer programming software (in this case both Gurobi and CPLEX) and employs a sequence of increasing tolerances that are used with the integer programming software. Best solutions found at one tolerance are then input as starting solutions for the next looser tolerance. In addition to SSIT finding bounded solutions quickly, its use of general-purpose integer programming software (such as CPLEX or Gurobi) is a significant benefit to OR practitioners. Specifically, it allows OR practitioners to quickly develop SSIT models using default software parameter values and templates with no need for problem-specific algorithms and corresponding computer code. Based on the particular application, the user has the flexibility to set the number of tolerances as well as their values. Additionally, the user determines the maximum execution time for each tolerance. Furthermore, for industrial systems that use SSIT, the performance of these systems is "automatically" improved when new versions of the optimization software are installed.

Specifically, the SSIT matheheurstic was shown to be highly effective and efficient at solving 135 set K-covering problems (SKCP) that appear in the literature. Three SSIT scenarios were tested and average deviation from the optimum or best-known solution was 0.03% for all three scenarios over all 135 SKCPs with average execution times of 84, 75, and 67 seconds for SSIT1, SSIT2, and SSIT3 respectively. Only one of the five published algorithms had a smaller average deviation and that was Wang, et al. [9] with an average deviation of 0.01%. However, for the 135 problems, statistical analyses demonstrated that the difference between SSIT3 and Wang, et al. [9] was statistically insignificant. Also, SSIT1 and SSIT2 results were statistically as good as Wang, et al. [9] for the KMIN and KMAX problems. SSIT3 and Wang, et al. [9] were slightly better than SSIT1 and SSIT2 for the KMED problems. What is much more important is that SSIT generated solutions that were *guaranteed* to be within a small percentage of the optimum while there are no solution quality guarantees for any of the other SKCP solution methods that appear in the literature. Specifically, over all 135 SKCPs, SSIT found solutions *guaranteed* to be *at most* 0.136%, 0.137%, and 0.131% from the optimum for scenarios SSIT1, SSIT2, and SSIT3 respectively.

This article introduced the set variable K-covering problem to be a SKCP in which the K value varies by row constraints. Based on set covering problems from Beasley's OR-Library, 65 SVKCPs were defined and effectively and efficiently solved using the SSIT matheuristic. For these 65 SVKCPs, SSIT found solutions guaranteed within 0.1% of the optimum for all 65 problems in an average time of 12 seconds.

Finally, since the SSIT matheuristic is a general-purpose strategy for solving combinatorial optimization problems, the authors plan to test the performance of SSIT on solving other difficult-to-solve combinatorial optimization problems using several different commercial integer programming software packages.

Funding: This study received no specific financial support.
Competing Interests: The authors declare that they have no competing interests.
Acknowledgement: All authors contributed equally to the conception and design of the study.

REFERENCES

[1] M. G. Resende, "An optimizer in the telecommunications industry," *SIAM SIAG/Optimization Views-and-News*, vol. 18, pp. 8-19, 2007.

[2] C.-J. Chang, Y.-T. Huang, and K.-M. Chao, "A greedier approach for finding tag SNPs," *Bioinformatics*, vol. 22, pp. 685-691, 2006.Available at: https://doi.org/10.1093/bioinformatics/btk035.

[3] G. Lin and J. Guan, "Solving maximum set k-covering problem by an adaptive binary particle swarm optimization method," *Knowledge-Based Systems*, vol. 142, pp. 95-107, 2018.Available at: https://doi.org/10.1016/j.knosys.2017.11.028.

[4] E. HAE, E.-L. YMA, and A. SM, "A new approximation algorithm for k-set cover problem," *Arabian Journal for Science and Engineering*, vol. 41, pp. 935–940, 2016.Available at: https://doi.org/10.1007/s13369-015-1895-3.

[5] L. S. Pessoa, M. G. Resende, and C. C. Ribeiro, "A hybrid Lagrangean heuristic with GRASP and path-relinking for set k-covering," *Computers & Operations Research*, vol. 40, pp. 3132-3146, 2013.Available at: https://doi.org/10.1016/j.cor.2011.11.018.

[6] S. Al-Shihabi, "A hybrid of max–min ant system and linear programming for the k-covering problem," *Computers & Operations Research*, vol. 76, pp. 1-11, 2016.Available at: https://doi.org/10.1016/j.cor.2016.06.006.

[7] A. Salehipour, *A heuristic algorithm for the set k-cover problem. Communications in Computer and Information Science 1173*. Cham: Springer, 2020.

[8] Y. Wang, M. Yin, D. Ouyang, and L. Zhang, "A novel local search algorithm with configuration checking and scoring mechanism for the set k-covering problem," *International Transactions in Operational Research*, vol. 24, pp. 1463-1485, 2017.Available at: https://doi.org/10.1111/itor.12280.

[9] Y. Wang, C. Li, H. Sun, J. Chen, and M. Yin, "MLQCC: An improved local search algorithm for the set k-covering problem," *International Transactions in Operational Research*, vol. 26, pp. 856-887, 2019.Available at: https://doi.org/10.1111/itor.12614.

[10] B. McNally, "A simple sequential increasing tolerance matheuristic that generates bounded solutions for combinatorial optimization problems," Master's Thesis, Kutztown University of Pennsylvania, 2021.

[11] Y. Lu, B. McNally, E. Shively-Ertas, and F. Vasko, "A simple and efficient technique to generate bounded solutions for the multidimensional Knapsacvk problem: A guide for OR practitioners," *IOnternational Journal of Circuits, Systems, and Signal Processing*, vol. 15, pp. 1650-1656, 2021.Available at: https://doi.org/10.46300/9106.2021.15.178.

[12] A. Dellinger, Y. Lu, B. McNally, M. S. Song, and F. J. Vasko, "A simple and efficient technique to generate bounded solutions for the generalized assignment problem: A guide for OR practitioners," *Research Reports on Computer Science*, vol. 1, pp. 13-34, 2021.

[13] J. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, vol. 5, pp. 99-114, 1949.Available at: https://doi.org/10.2307/3001913.