Available online at http://scik.org

J. Math. Comput. Sci. 11 (2021), No. 3, 3747-3766

https://doi.org/10.28919/jmcs/5676

ISSN: 1927-5307

AN ENHANCED APPROXIMATION ALGORITHM TO FIND MINIMUM REPRESENTATIVE PATTERN SETS

R. PRABAMANIESWARI[1,†,*], D.S. MAHENDRAN[2], T.C. RAJA KUMAR[3]

[1]Research Department of Computer Science, St. Xavier's College, Palayamkottai,

Affiliated to Manonmaniam Sundaranar University, Tirunelveli-627012, Tamil Nadu, India

[2]Aditanar College of Arts & Science, Tiruchendur, Tamil Nadu, India

[3]Department of Computer Science, St. Xavier's College, Palayamkottai, Tamil Nadu, India

Abstract: The problem of selecting as few as the possible number of sets that would cover the whole universal set is called a set cover. The commonly used set cover algorithm GreedySetCover gives a better approximation result. There are various alternative algorithms for GreedySetCover are proposed in the research, depending on the nature of the problem. This paper proposes an enhanced GreedySetCover algorithm such as MaxFrequentGroupGreedy to find the minimum number of representative pattern sets by combining two or more maximum frequent itemsets groups and checking set cover among them. The groups are combined by using the percentage difference formula and set cover is done by removing the itemset which is contained in another itemset in the combined group. The effectiveness of this approach is tested on frequent itemsets which are determined from NCFP-tree and

*Corresponding author

E-mail address: prabacs2006@gmail.com

†Research Scholar, (Reg No. 11991)

Received March 10, 2021

ModifiedRPset algorithms and a reduced number of representative itemsets is obtained. The reduced itemsets approximate all other itemsets.

Keywords: frequent itemsets; greedy algorithm; maximum frequency; representative pattern sets.

2010 AMS Subject Classification: 68W01.

1. INTRODUCTION

One of the key techniques used by large retailers is to identify relationships between the items that people buy. They find out the combinations of items that occur together frequently in transactions. In data mining, discovering frequent itemsets is taken into account important research due to its large applicability in real-world applications. But, the complete set of discovered frequent patterns often contains a lot of redundancy. To overcome this problem, finding a few numbers of representative patterns with an approximation guarantee is suggested in the literature. Several approximation algorithms have been proposed to find approximate solutions for the Minimum Set Cover problem and research is still going on to optimize the solution. The widely used approximate solution for the Set Cover Problem is the Greedy Set Cover algorithm.

Set covering is considered NP-Hard in optimization and search problems and NP-Complete in decision-based problems. In the set cover problem, we are given a universe U, such that $|U|=n$, and sets $S_1, \ldots, S_k \subseteq U$. A set cover is a collection C of some of the sets from$S_1, \ldots, S_k$ whose union is the entire universe U. Formally, C is a set cover if $\bigcup_{S_i \in C} S_i = U$. We would like to minimize |C| i.e) Set Covering is to select a minimum number of subsets in such a way that they contain all the elements from a fixed universe [6]. The study on set covering provides many useful results for the market basket analysis problem.

In this paper, the MaxFrequentGroupGreedy algorithm is proposed to find the minimum number of representative pattern sets based on approximation. It approximates the support count value for reducing the number of frequent itemsets. In the business intelligence world, “market basket analysis” helps retailers better understand – and ultimately serve – their users by predicting their purchasing behaviours. This algorithm helps to make decisions based on the resultant set which

is obtained by using the support count value. It initially finds the maximum frequency count1and groups the itemsets which have the same frequency. This group is taken as g. Then, it finds the next maximum frequency (support count2) and applies the percentage difference formula between them and checks that calculated percentage difference value against the given percentage difference value d. The percentage difference formula is given below:

$$\text{Percentage Difference} = \frac{|\text{count1}-\text{count2}|}{((\text{count1}+\text{count2})/2)} \text{ x } 100 \quad (1)$$

If the difference is less than or equal to d, then it adds that itemset that has frequency count2 to group g. The grouped itemsets are removed from F where F refers to the set of frequent itemsets given as input. The grouping of itemsets is repeated by finding the next maximum frequency count2 each time and applies the percentage difference formula with count1 till the difference is less than or equal to d. Afterwards, it checks each set in the group g whether it belongs to any one of the three categories such as i) subset ii) superset and iii) independent set and adds that set into the resultant set depends on the newly added elements in visited set. This algorithm approximates the frequencies while grouping the itemsets. The subset, superset and independent set are determined based on the size of the itemset only because all itemsets which belong to the same group g are considered as having the same frequency. This algorithm maintains the visited set for keeping the elements/items of the set if the set is selected for adding into the resultant set. Here, the proposed algorithm adds the elements into the visited set instead of deleting elements from the universal set in the GreedySetCover approach if the set is selected for adding into the resultant set. The above process is repeated by checking all the sets in the group g, clearing existing group g and creating a new group from F till F reaches a null set and the size of the visited set is less than the size of the universal set.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 describes the proposed method for generating a reduced number of representative pattern sets with an example. The experimental results are shown in section 4. Finally, section 5 concludes the paper.

2. RELATED WORK

Algorithms for optimization problems typically go through a sequence of steps, with a set of choices at each step. It returns a near-optimal solution. Timothy Chan et al. [19] say that optimization has numerous applications in different areas of studies and industrial applications. Many algorithms are introduced in two aspects such as to achieve a better approximation ratio and to get a minimum number of subsets. Bar-Yehuda and Even presented a linear time approximation algorithm [2] for the weighted set-covering problem. Chvatal et al. [3] proved a greedy-type algorithm to approximate the SCP with a performance guarantee log |S|. Petr Slavic [12] showed that the approximation ratio is in fact ln(m) - ln(ln(m)) + (1), where m is the size of the universe. Fabrizio Grandoni et al. proposed an algorithm [4] based on the interleaving of the standard greedy algorithm that selects the min-cost set which covers at least one uncovered element. Fatema Akhter [5] proposed a heuristic approach to solve the problem using a modified hill-climbing algorithm. Monjurul Alom et al. proved [11] the better results. They proposed a scanning of every subset S from the solution against the union of the other subsets of the solution to determine whether all the elements S covers are already covered by the other sets. If this is true, then S is removed from the solution. Stefan Spasovski et al. proposed [18] Optimization of the Polynomial Greedy Solution for the Set Covering Problem. They mentioned the derived GreedySetCover algorithm from Bar-Yehuda and Even. They modified the existing greedy algorithm to find if the element /elements belong to only one set. If it is true, then it will be added to the solution set. It is considered as the preprocessing step and the remaining procedure is the same as the GREEDYSETCOVER algorithm. They obtained optimal results opposite to the GREEDYSETCOVER algorithm in the best case. Rafael Hassin and Asaf Levin proposed a modification of the GREEDYSETCOVER algorithm, called greedy algorithm with withdrawals, SETCOVERWITH WITHDRAWALS [15]. This algorithm subtracts any subset S from the solution if it is replaced with other subsets that contain the elements covered by S. They concluded that withdrawal operation was crucial to obtain a better approximation ratio and proved the result. Anupam Gupta et al. [1] presented generic techniques for a dynamic set cover problems related to time bounds and limited resources. Habib Mostafaei et al. [8] focused on the

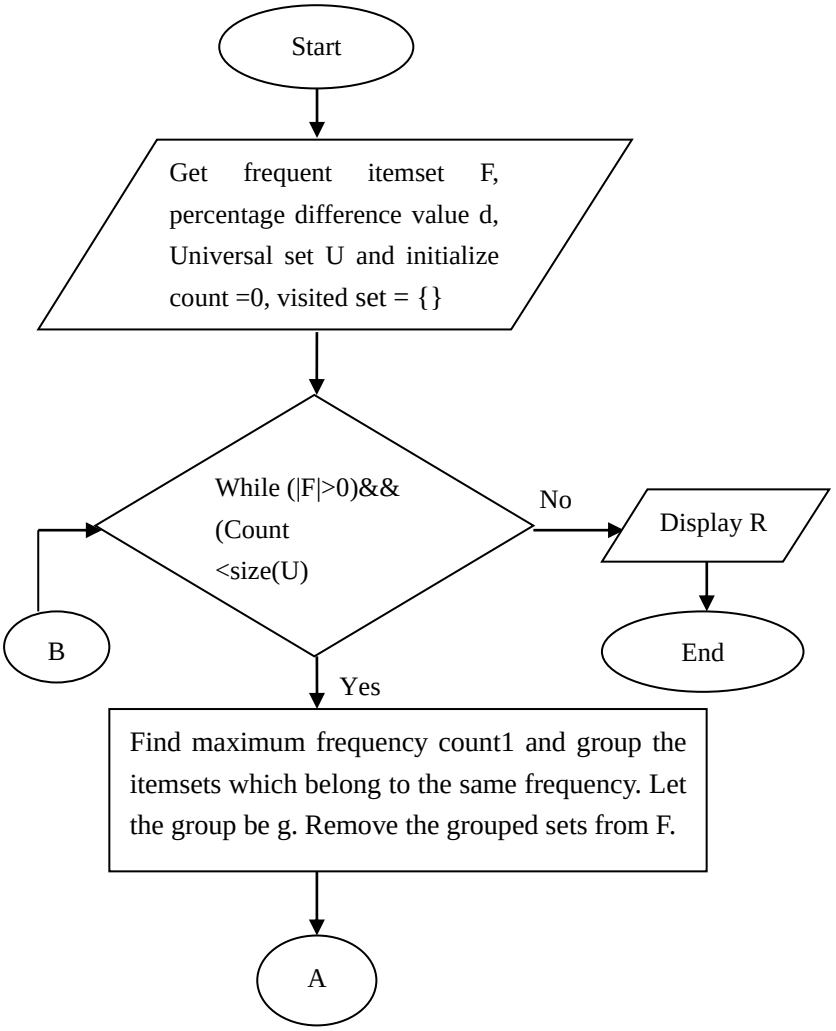
problem of partial coverage and presented an algorithm PCLA for minimizing the number of sensors to activate for covering the desired portion of the region of interest preserving the connectivity among sensors. An improved deterministic distributed algorithm for hypergraph maximal matching and improved algorithms for edge-coloring are discussed in [10]. R. Ramdani et al. proposed an approach [16] to improve segmentation based on word embedding. In their research, greedy splitting was improved by applying the window approach and the greedy process is minimized by defining the number of sentences or words that would be examined. An improvement plan for the efficiency of the Energy Storage System (ESS) and energy use is proposed in [17]. It suggests that the use of sodium-ion batteries will overcome the disadvantages of lithium-ion batteries, which are dominant in the current market. The greedy algorithm and the Floyd–Warshall algorithm were proposed as a method of scheduling energy use while considering the elements that could affect communication output and energy use. The simulation results showed that the greedy algorithm was more efficient. Graham Cormode et al. provide a new algorithm [7] that finds a solution that is probably close to that of greedy. Ioannis Tsamardinos et al. proposed a Parallel Forward–Backward with Pruning (PFBP) algorithm [9] for feature selection (FS) for Big Data of high dimensionality. PFBP partitions the data matrix both in terms of rows as well as columns. It provides asymptotic guarantees of optimality for data distributions faithfully representable by a causal network (Bayesian network or maximal ancestral graph). In this paper, our study focuses to approximate the support count value for grouping the itemsets and to determine superset and independent sets to get the minimum number of itemsets (pattern sets). It uses a similar GreedySetCover approach for selecting and adding an itemset into the resultant set based on the size of a set.

3. PROPOSED ALGORITHM

Our proposed algorithm MaxFrequentGroupGreedy is similar to the GreedySetCover algorithm. This algorithm initially finds the maximum frequency count1and groups the itemsets which have the same frequency (support count1). This group is named g. The grouped itemsets are removed from F. Then, it finds the next maximum frequency (support count2) and determines the

percentage difference between both frequencies. It would be less than or equal to the given percentage difference value d. If it is true, it adds the itemsets with frequency count2 into g. The percentage difference between the two frequencies is calculated using the formula (1). The number of groups that is to be combined to group g is decided based on the percentage difference between the first maximum frequency count1and the next successively finding maximum frequency count2 i.e., each time, the frequency count2 is calculated and it is compared with the initial frequency count1 till the difference between them is less than or equal to d. The grouped itemsets are removed from F. Afterwards, it checks each set in the group g against all other sets and determines whether it belongs to any one of the three categories such as i) subset ii) superset and iii) independent set of any other set. This is determined based on the size of sets only because all the sets in group g have approximately the same frequencies. If the sizes (the number of items/elements) of the itemsets are different, then the proposed algorithm selects the set from the group g by determining the set which contains another set and adds that set into the resultant set R. It discards the set from the group if it is contained in another set. Suppose, if the set to be checked is not a subset /superset of any other set, then it decides that it automatically belongs to the third category i.e., independent set. The independent sets are not only having different sizes but also having the same size but different elements. Therefore, this independent set category is again checked for the same size after checking all the categories for a different size. It maintains the visited set for keeping the elements/items of the set if the set is selected for adding into the resultant set. Each time, it checks either the superset or the independent set with the visited set before adding into the resultant set R. The visited set is updated by adding those elements of the superset/independent set which does not already exist in the visited set i.e., the duplicate elements are removed from the visited set. If the number of elements of that set does not exist in the existing visited set is non zero, then that set will be added into the resultant set R. Otherwise, it will not be added. The proposed algorithm applies the greedy approach while selecting and adding the itemset into the resultant set R. The GreedySetCover approach removes the elements of the set from the universal set if the set is selected for adding into the solution set. But, the proposed algorithm adds the elements into the visited set instead of deleting elements from the

universal set in the greedy approach if the set is selected for adding into the resultant set. It keeps the visited set for checking and adding the superset and independent set into resultant set R. It also keeps the size of the visited set for checking the process whether it is completed or not. The above-said process is repeated till all the sets in the group g are processed. Then, it repeats the whole process with a new maximum frequency count1 and group g till all itemsets in F are processed and the size of the visited set is less than the size of the universal set. This algorithm gives better results while approximating support count values for grouping the frequent itemsets. This paper compares the proposed approach MaxFrequentGroupGreedy with Greedy approaches such as set coverage and weighted set coverage. The overview of our proposed approach is given in Fig. 1.



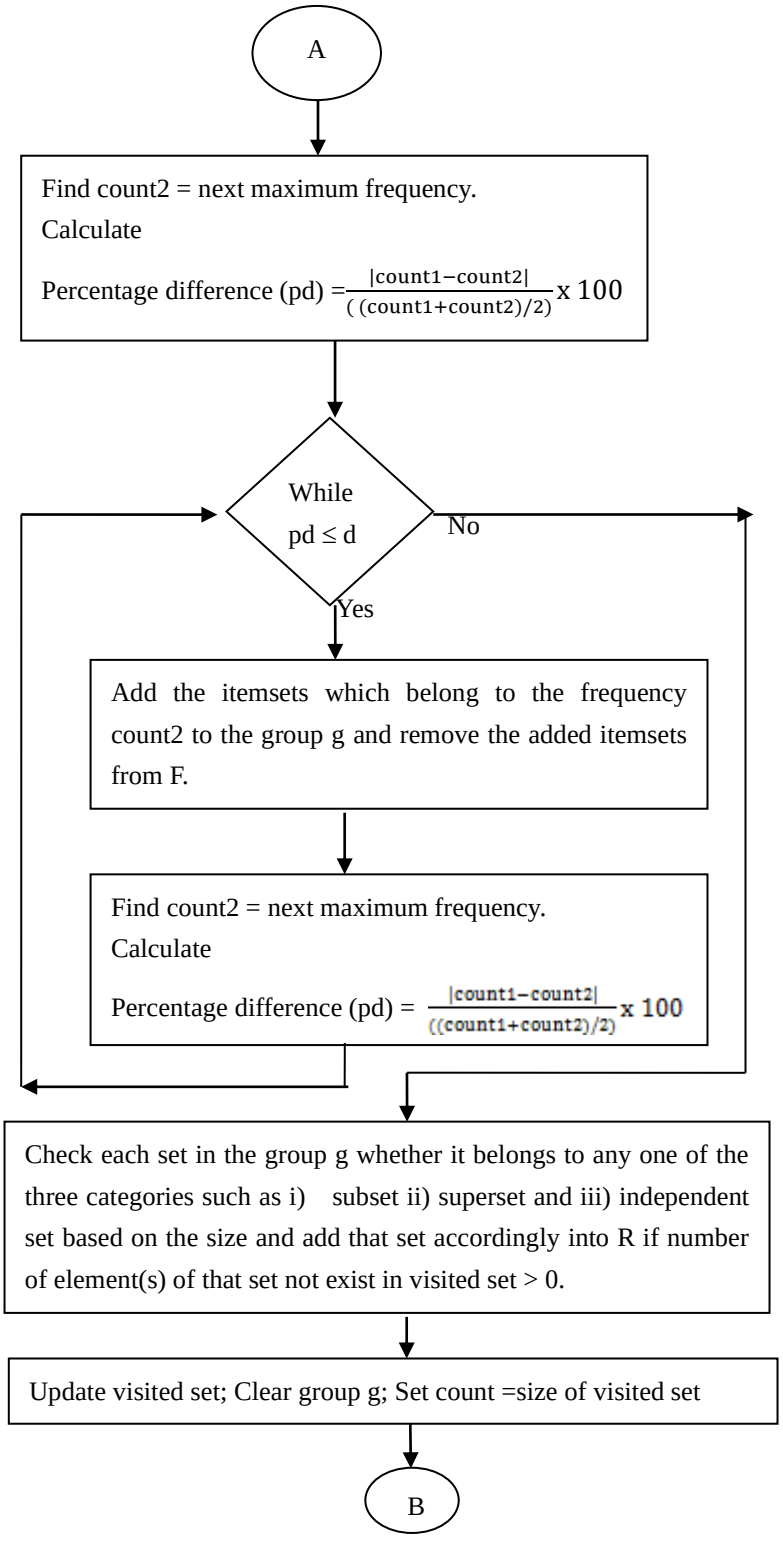


Fig. 1. Overview of MaxFrequentGroupGreedy

The proposed algorithm is given below:

Algorithm: MaxFrequentGroupGreedy

Input: Frequent itemset F={{s_1,c_1},{s_2.c_2},{s_3,c_3}……..{s_n,c_n}}, percentage difference value d , visited set = {},Universal set U and count =0

Output:

Representative Pattern Sets R

Description:

1. Create a new group g for each iteration based on d and do the following

while ((F !=ϕ) && count < size(U))

begin

//choose first maximum frequency (support count)

Let count1 = Maximum Frequency c ∈ F

Let n = | {s_i} | if c = c_i where i= 1to size (F)

for i = 1 to n

Add { s_i } to the group g

Remove { s_i } from F

end for //next i

//choose next maximum frequency (support count)

Let count2 = Maximum Frequency c ∈ F

Calculate pd = $\frac{|count1-count2|}{((count1+count2)/2)}$ x 100

while (pd <=d)

begin

Let n = | {s_i} | if c = c_i where i= 1to size (F)

for i = 1 to n

Add{ s_i }to the group g

Remove { s_i }from F

end for //next i

Let count2 = Maximum Frequency c ∈ F

Calculate pd = $\frac{|count1-count2|}{((count1+count2)/2)}$ x 100

end while

//Check each set whether it is a subset/superset/independent set

for i=0 to size(g)

num1 = $|s_i|$; status1 =0; status2 =0;

for j=i to size(g)

num2 = $|s_j|$

if (num1 != num2)

if ($s_i \subseteq s_j$) // s_i - subset

Continue to next iteration for i

else if ($s_j \subseteq s_i$) // s_i - superset

status1 = 1

end if

end if

else if (num1 = num2) //independent set (same size)

status2 =1

end if

end for // next j

number of elements added=0

if (((status1=0)&&(status2=0)) || **(**(status1=0)&&(status2=1)) || **(**status1=1) **)**

// independent set different, same size, superset

// Update visited set and R

Add each element of s_i to visited set if it already does not exist

number of elements added=number of newly added elements to visited set

if (number of elements added > 0)

Add s_i to R

end if

end if

end for // **next i**

Let count =size (visited set)

Set g =ϕ // clear group g

end while

2. display Representative Pattern Sets R
3. end

3.1 Example

Consider the Transactional Database given in Table 1. It has seven transactions, that is |D|=7. Assume min-sup=40%. The set of frequent itemsets which are determined by using the NCFP-tree algorithm [11] is given in Table 2. The NCFP-tree stores frequent itemsets in a compact form.

Table 1 Transaction Database

TID	TRANSACTION
1	a, c, e, f, m, p
2	a, b, f, m, p
3	a, b, d, f, g
4	d, e, f, h, p
5	a, c, d, m, v
6	a, c, h, m, s
7	a, f, m, p, u

Table 2 Frequent itemset (NCFP-tree)

Frequent Itemsets(Compact Form) (min_sup = 40%)
cma:3
d:3
pfma:3 pf:4
fma:3 fa:4 f:5
ma:5
a:6

Here, the proposed algorithm considers the percentage frequency difference as 0.3. Initially, the maximum support count itemsets a (a:6), ma (ma:5) and f (f:5) are combined based on the percentage frequency difference. The itemset a is contained in ma. Therefore, a is skipped. Then, ma and f are added into the resultant set because they are independent of each other. Now, the process is repeated by grouping the remaining itemsets such as cma (cma:3), d, (d:3), pfma(pfma:3), pf(pf:4), fma(fma:4) and fa(fa:4). The itemsets cma, d and pfma are added into the resultant set and the itemsets pf, fma and fa are not added because their elements are already included in the visited set. The resultant set obtained is {{ma:5}, {f:5}, {cma:3}, {d:3},

{pfma:3}}. But, the greedy approaches give the resultant set {{cma: 3}, {d: 3}, {pfma: 3}}.

Here, our approach gives importance to approximating the support count value because, in the market-basket analysis problem, support count is very much important for finding frequent itemsets. Our proposed method gives the same result as the greedy method if percentage frequency difference d is considered as 0.7. Therefore, our proposed method aims not only to reduce the frequent itemsets but also gives importance to consider the support count value. It gives a better result in that aspect.

4. EXPERIMENTAL RESULTS

The experiments are carried out on the computer with the configuration such as Intel(R) Core(TM) i3CPU, 3 GB RAM, 2.53 GHz Speed and Windows 7 Operating System. The approaches MaxFrequentGroupGreedy, GreedySetCover and Greedy Weighted Set Cover [3] are implemented in java. The experiments are evaluated on three datasets such as mushroom dataset, retail dataset and internet usage data dataset. The datasets are handled in two ways such as i) finding frequent itemsets from the dataset and applying the resultant frequent itemsets and ii) generating a weight randomly for each transaction of the dataset and applying the resultant transactional dataset.

The mushroom dataset contains the characteristics of various species of mushrooms. It has 119 items and 8124 transactions. The minimum, maximum and average length of its transaction is 23. The retail dataset contains the retail market basket data from an anonymous Belgian retail store. It has 16,470 items and 88,162 transactions. The maximum length of its transaction is 77 and the average length of its transaction is10. Both datasets are taken from the FIMI data repository page [20]. The internet usage data dataset available from UCI's Machine Learning Repository [21] contains general demographic information on internet users in 1997. This dataset has 10,104 instances and 72 attributes, and the data types of this dataset were categorical or integer.

4.1 Performance on the frequent itemsets

The experiment is performed on frequent itemsets which are obtained from the mushroom dataset and retail dataset. The algorithms NCFP-tree [13] and ModifiedRPset [14] are used to find frequent itemsets. In ModifiedRPset, the patterns which are obtained before applying the greedy algorithm are taken and used. The determined frequent itemsets from these algorithms are applied in MaxFrequentGroupGreedy, GreedySetCover and Greedy Weighted Set Cover to get a minimum number of frequent itemsets. The experiment considers both execution time and number of representative pattern sets for finding the performance of the algorithms.

The number of representative pattern sets obtained from MaxFrequentGroupGreedy (while considering percentage frequency differences d= 10%, d=50% and not considering frequency difference d= 0%), GreedySetCover and Greedy Weighted Set Cover is given in Table 3 and 4. In Table 3, the three approaches use the frequent itemsets which are obtained from NCFP-tree (min-supp varies from 0.2 to 0.8) and ModifiedRPset (min-supp=0.4) algorithms. Here, the mushroom dataset is applied in both algorithms NCFP-tree and ModifiedRPset to get frequent itemsets. The retail dataset is used in Table 4. The min-supp is varied from 0.02 to 0.1. Here, the NCFP-tree algorithm is used for getting frequent itemsets and it is given as input for three approaches.

Table 3 Representative Pattern Sets (Mushroom Dataset)

S.No	Min Supp	Number of frequent itemsets from NCFP-tree	MaxFrequent Group Greedy			Greedy Set Cover	Greedy Weighted Set Cover
			d=50%	d=10%	d=0 %		
1.	0.2	6089	18	32	22	10	22
2.	0.4	565	7	13	6	5	5
3.	0.6	51	7	13	6	3	5
4.	0.8	23	1	2	1	1	1
5.	0.4	Modified RPset (65)	14	14	14	14	14

Table 4 Representative Pattern Sets (Retail Dataset)

S.No	Min supp	Number of frequent itemsets from NCFP -tree	MaxFrequent Group Greedy			Greedy Set Cover	Greedy Weighted Set Cover
			d=50%	d=10%	d=0 %		
1.	0.02	4903	3	5	3	6	4
2.	0.04	116	1	1	1	1	1
3.	0.06	26	1	1	1	1	1
4.	0.08	26	1	1	1	1	1
5.	0.1	26	1	1	1	1	1

The proposed algorithm gives the same number of representative pattern sets as the Greedy Weighted Set Cover algorithm when not considering the percentage frequency difference (d=0%). But, it gives more number of representative pattern sets while grouping the itemsets based on percentage frequency difference when applying NCFP-tree algorithm and it gives the same number of representative pattern sets in the case of applying ModifiedRPset algorithm. It gives more or less the same number of resultant itemsets as greedy approaches which are given in Table 4. The proposed algorithm divides the given itemsets into many groups depends on the percentage difference and it gives the resultant itemsets from each group. Therefore, it gives more number of resultant itemsets if it has a lesser percentage frequency difference. This concept of the proposed algorithm helps to predict the purchase behavior of the customer and also helps to support for market basket analysis problem. But, the GreedySetCover algorithms process all given itemsets as a single group.

It is observed that the proposed algorithm gives the number of pattern sets that is closer to GreedySetCover algorithms if no percentage difference is considered (d=0%) i.e., it behaves in the same manner as GreedySetCover algorithms when the input is considered as a single group.

The execution time of MaxFrequentGroupGreedy is compared with GreedySetCover and Greedy

Weighted Set Cover algorithms. Figure 2 and Figure 3 show the running time of the three algorithms. They use frequent itemsets from the NCFP-tree algorithm. The min-supp is varied from 0.2 to 0.8 for the mushroom dataset and it is varied from 0.02 to 0.1 for the retail dataset. Here, the proposed algorithm applies the frequent itemsets when d= 0% (single group), d= 10% and d= 50%.

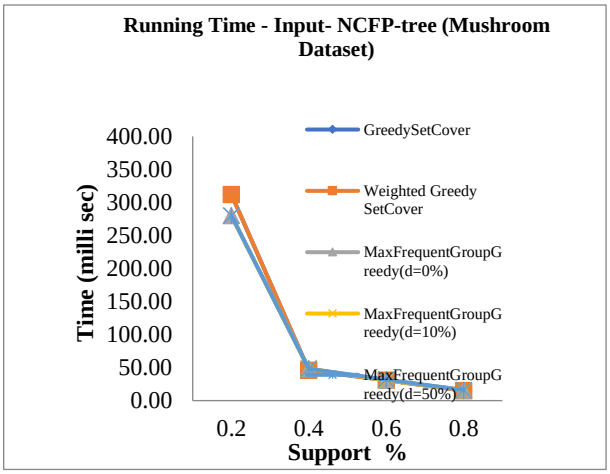


Fig. 2. Running Time using Mushroom Dataset

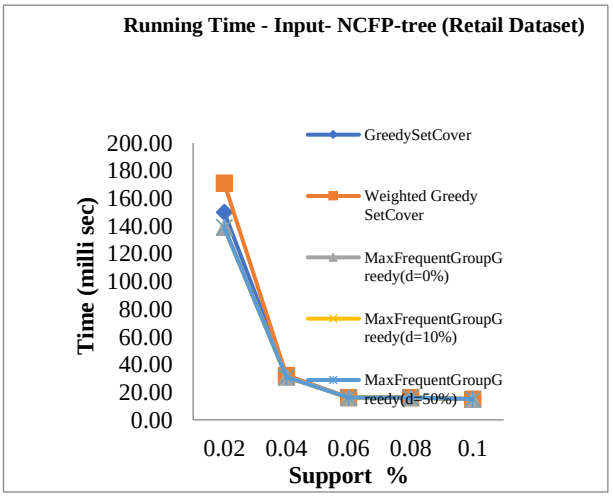


Fig. 3. Running Time using Retail Dataset

From the figures, we observed that when min-supp is increased, all the three methods give the same execution time. But, if min-supp is lesser, the proposed method MaxFrequentGroupGreedy gives slightly lesser execution time compared to GreedySetCover and Greedy Weighted Set Cover algorithms. The proposed algorithm MaxFrequent GroupGreedy gives more or less the same execution time for all three cases (d =0%, d= 10% and d =50%). It reduces the execution time by a factor of 10% comparing to Greedy Weighted Set Cover when min-supp=0.2 while applying the mushroom dataset. Similarly, it gives the same execution time in the case of the retail dataset when min-supp is increased and it reduces the execution time approximately by a factor of 18% comparing to Greedy Weighted Set Cover when min-supp=0.02. The acceptability of a percentage error depends on the application. Here, the importance is given for approximately grouping frequent itemsets using support count value to find the number of frequent itemsets regardless of the time taken by the algorithm.

4.2 Performance on the transactional dataset

To handle large datasets, two datasets such as the internet usage data dataset and the mushroom dataset are directly applied without finding frequent itemsets. The internet usage data dataset has frequency values but, the mushroom dataset does not have frequency values. Therefore, a weight is assigned to each transaction of the mushroom dataset and is assumed as the frequency for the transaction. Generally, any dataset can be taken and the weight can be randomly generated depends on the size of the dataset. For example, the weight for each transaction of the mushroom dataset is randomly generated from 1 to 8124. The datasets are utilized in two ways such as by partitioning the dataset and by considering the whole dataset (without partitioning). Table 5 shows the performance of the approaches Greedy Weighted Set Cover, MaxFrequentGroupGreedy(d=0%) and MaxFrequentGroup Greedy (d=50%) without partitioning the dataset and Table 6 shows the performance of these algorithms with partitioning the dataset. When following the partitioning approach, initially the partition size is fixed. The partitions are successively processed iteratively and the result obtained from the previous iteration is combined with the input for the next iteration. Therefore, each time the number of transactions to be retrieved from the dataset is calculated as subtracting the number of previous iteration's resultant transactions from the fixed partition size. The resultant tables are given as follows:

Table 5 Performance of the algorithms for large datasets (No Partition)

Dataset	Algorithm	Running Time (milli sec)	Number of Resultant Set
Mushroom	Greedy Weighted Set Cover	3620	35
	MaxFrequentGroupGreedy (d=0%)	1680	36
	MaxFrequentGroupGreedy (d=50%)	1571	39
Internet Usage Data	Greedy Weighted Set Cover	10857	36
	MaxFrequentGroupGreedy (d=0%)	16392	41
	MaxFrequentGroupGreedy (d=50%)	16197	41

Table 6 Performance of the algorithms for large datasets (Partition size = 4k)

Dataset	Algorithm	Running Time (milli sec)	Number of Resultant Set
Mushroom	Greedy Weighted Set Cover	2637	31
	MaxFrequentGroupGreedy (d=0%)	1323	36
	MaxFrequentGroupGreedy (d=50%)	1169	39
Internet Usage Data	Greedy Weighted Set Cover	9887	36
	MaxFrequentGroupGreedy (d=0%)	15102	41
	MaxFrequentGroupGreedy (d=50%)	16004	41

From the tables, we observed that our proposed algorithm gives more or less the same number of resultant sets as the Greedy Weighted Set Cover algorithm for both datasets. The proposed algorithm reduces the execution time by a factor of 55% approximately (in the case of the mushroom dataset) and it gives at most approximately a 62% increase in time (in the case of Internet usage Data Dataset) comparing to the Greedy Weighted Set Cover algorithm. It is known that the minimum representative pattern sets are determined after finding frequent itemsets from the transactional database. Therefore, the proposed algorithm would give better results if frequent itemsets are given as input.

5. CONCLUSION

In this paper, we followed a generalized version of the set covering called the GreedySetCover method and modified the concept to select the set for finding minimum representative pattern sets. The Greedy algorithm may provide an efficient solution that is close to optimal. But, there is no general template on how to apply the greedy method to a given problem. Therefore, we applied the greedy method while selecting the itemsets based on the categories from the approximately grouped itemsets (grouping percentage frequency difference d) and produced a minimum number of itemsets. Our algorithm gives the mostly same result (reduced number of sets) as existing algorithms if no percentage frequency difference (single group i.e) d=0%) is considered and it gives better results if percentage frequency difference d is considered. It also supports finding representative pattern sets for large datasets. As part of our future work, we can apply the proposed algorithm in a parallel processing environment to improve the overall performance.

CONFLICT OF INTERESTS

The author(s) declare that there is no conflict of interests.

REFERENCES

[1] A. Gupta, R. Krishnaswamy, A. Kumar, D. Panigrahi, Online and dynamic algorithms for set cover, in: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, ACM, Montreal Canada, 2017: pp. 537–550.

[2] R. Bar-Yehuda, S. Even, A linear-time approximation algorithm for the weighted vertex cover problem, J. Algorithms. 2 (1981), 198–203.

[3] V. Chvatal, A greedy heuristic for the set-covering problem, Math. Oper. Res. 4 (1979), 233–235.

[4] F. Grandoni, A. Gupta, S. Leonardi, P. Miettinen, P. Sankowski, M. Singh, Set covering with our eyes closed, in: 2008 49th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Philadelphia, PA, USA, 2008: pp. 347–356.

[5] F. Akhter, A heuristic approach for minimum set cover problem, Int. J. Adv. Res. Artif. Intell. 4 (2015), 40-45.

[6] G. Gens, E. Levner, Complexity of approximation algorithms for combinatorial problems: a survey, ACM SIGACT News. 12 (1980), 52–65.

[7] G. Cormode, H. Karloff, A. Wirth, Set cover algorithms for very large datasets, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management - CIKM '10, ACM Press, Toronto, ON, Canada, 2010: p. 479.

[8] H. Mostafaei, A. Montieri, V. Persico, A. Pescapé, A sleep scheduling approach based on learning automata for WSN partialcoverage, J. Netw. Computer Appl. 80 (2017), 67–78.

[9] I. Tsamardinos, G. Borboudakis, P. Katsogridakis, P. Pratikakis, V. Christophides, A greedy feature selection algorithm for big data of high dimensionality, Mach. Learn. 108 (2019), 149–202.

[10] M. Ghaffari, D.G. Harris, F. Kuhn, On Derandomizing Local Distributed Algorithms, in: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, Paris, 2018: pp. 662–673.

[11] B.M.M. Alom, S. Das, M.A. Rouf, Performance evaluation of vertex cover and set cover problem using optimal algorithm, DUET J. 1 (2011), 8-13.

[12] P. Slavík, A tight analysis of the greedy algorithm for set cover, J. Algorithms. 25 (1997), 237–254.

[13] R. Prabamanieswari, NCFP-tree: a non-recursive approach to CFP-tree using single conditional database, Int. J. Res. Appl. Sci. Eng. Technol. 5 (2017), 386–393.

[14] R. Prabamanieswari, D.S. Mahendran, T.C. Raja Kumar, A modified algorithm for finding representative pattern sets, Int. J. Eng. Res. Computer Sci. Eng. 5 (2018), 201-205.

[15] R. Hassin, A. Levin, A better-than-greedy approximation algorithm for the minimum set cover problem, SIAM J. Comput. 35 (2005), 189-200.

[16] R. Ramdani, A.F. Huda, M. Arif Bijaksana, Text Segmentation Based on Word Embedding on Indonesian Quran Translation by Greedy with Window Approaching, in: 2019 7th International Conference on Information and Communication Technology (ICoICT), IEEE, Kuala Lumpur, Malaysia, 2019: pp. 1–5.

[17] S.-M. Je, J.-H. Huh, An Optimized Algorithm and Test Bed for Improvement of Efficiency of ESS and Energy Use, Electronics. 7 (2018), 388.

[18] S. Spasovski, A.M. Bogdanova, Optimization of the polynomial greedy solution for the set covering problem, in: The 10th Conference for Informatics and Information Technology (CIIT 2013), 175-177.

[19] T.M. Chan, E. Grant, J. Könemann, M. Sharpe, Weighted Capacitated, Priority, and Geometric Set Cover via Improved Quasi-Uniform Sampling, in: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2012: pp. 1576–1585.

[20] Workshop on frequent itemset mining implementations (FIMI'04), http://fimi.cs.helsinki.fi (2004).

[21] https://archive.ics.uci.edu/ml/datasets/Internet+Usage+Data