Missing Value Estimation for Mixed-Attribute Datasets

G. Jayasree[1] & M. Sarada[2]
[1]PG Student, Department of MCA, St. Ann's College of Engineering & Technology, Chirala
[2]Assistant Professor, Department of MCA, St. Ann's College of Engineering & Technology, Chirala

Received: January 29, 2019 Accepted: March 03, 2019

ABSTRACT: *The growing interest in data storage has made the data size to be exponentially increased, hampering the process of knowledge discovery from these large volumes of high-dimensional and heterogeneous data. In recent years, many efficient algorithms for mining data associations have been proposed, facing up time and main memory requirements. Nevertheless, this mining process could still become hard when the number of items and records is extremely high. In this project, the goal is not to propose new efficient algorithms but a new data structure that could be used by a variety of existing algorithms without modifying its original schema. Thus, our aim is to speed up the association rule mining process regardless the algorithm used to this end, enabling the performance of efficient implementations to be enhanced. The structure simplifies, reorganizes and speeds up the data access by sorting data by means of a shuffling strategy based on the hamming distance, which achieve similar values to be closer and considering both an inverted index mapping and a run length encoding compression. In the experimental study, I explore the bounds of the algorithms' performance by using a wide number of data sets that comprise either thousands or millions of both items and records. The results demonstrate the utility of the proposed data structure in enhancing the algorithms' run time orders of magnitude and substantially reducing both the auxiliary and the main memory requirements.*

I. Introduction

Feature selection, also called attributes education, is a common problem in pattern recognition, data mining and machine learning. In recent years, both the number and dimensionality of items in data sets have grown dramatically. For examples, tens, hundreds and even thousand so attributes are stored in many real world application databases. It is well known that attribute irrelevant to recognition tasks may deteriorate the performance of learning girths. In other words, storing and processing all attributes (both relevant and irrelevant) could be computationally very expensive and impractical. To address this issue, as pointed outing, some attributes can be omitted, which will not seriously affect their salting classification (recognition) accuracy.

Therefore, the omission of some attributes could be not only tolerable but also even desirable relative to the computational costs involved In feature selection, there are two general strategies, namely wrappers and filters The former employ sale arming algorithm.

Feature selection based on rough set theory starts from a data table, which is also called an information system and contains data about objects of interest that are characterized by a finite set of attributes. According to whether or not there are missing data (null values), information systems are classified into two categories: complete and incomplete. In general, by anon complete information system, I mean a system with missing data (null values). For an in complete information system, if condition attributes and decision attributes are distinguished from each other, then it is called an in complete decision table. Feature select informing complete data usually starts from incomplete decision tables. In the last two decades, many techniques for attributes education have been develop d in rough set theory especially £ in order to brain all attribute redacts of a given data set, Sopron proposed discernibility matrix method [10]. However, these feature selection algorithms are usually time-consuming to process large-scale data. Aiming get efficient feature selection, many heuristic attributes education algorithm shave been developed in rough set theory. Each of the seal go- rhythms preserve sap articular property of a given information system. To accomplish attribute reduction from incomplete decision tables, similar to the discernibility matrix proposed by Sopron, Kryszkie wicz ga vea generalized discernibility matrix to obtain all attributes reducts of an incomplete decision table. To efficiently obtain an attributes edict, several heuristic attribution reduction approach save been presented [8].For convenience of our further development, were view several representative heuristic attribute reduction go rithms in the context of in complete data here. Applying the idea of positive- region reduction, Yang and Shuproposed aheuristic feature selection algorithm in complete decision tables, which

keeps the positive region of target decision unchanged [9]. Kumar et al. [11] discussed the cloud storage and convention techniques. In this paper proposes a system for secure delicate information partaking in cloud, including secure information conveyance, stockpiling, use, and devastation on a semi-confided in cloud environment. They exhibit Kerberos convention over the system and a client procedure insurance technique in view of a virtual machine screen, which offers help for the acknowledgment of framework capacities. The proposed procedure framework is secure and effective. Liang et al. defined new information entropy to measure the uncertainty of an incomplete information system [12] and applied the corresponding conditional entropy to reduce redundant features. Qian and Liang presented the combination entropy for measuring the uncertainty of an incomplete information system and used its conditional entropy to obtain a feature subset. As Shannon's information entropy was introduced to search reducts in classical rough set model [14], an extension of its conditional entropy also can be used to calculate a relative attribute reduct of an incomplete decision table. However, the above algorithms are still computationally time-consuming to deal with large-scale data sets.

II. Related work

A. Multiobjective Optimization

A MOP consists of minimizing or maximizing an objective function vector under some constraints. The general form of
a MOP is as follows [2]:
Min $f(x) = \_f1(x), f2(x), ...,fM(x)\_T$
$gj(x) \geq 0\ j = 1, \ldots, P$
$hk(x) = 0\ k = 1, \ldots, Q$
$xLi \leq xi \leq xUi\ i = 1, \ldots, n$ (1)
where M is the number of objective functions, P is the number of inequality constraints, Q is the number of equality constraints, and xLi and xU
i correspond to the lower and upper bounds of the variable xi. A solution satisfying the $(P + Q)$ constraints in addition to the bound constraints defined in (1) is said feasible and the set of all feasible solutions defines the feasible search space denoted by X. In this formulation, I consider a minimization MOP, since maximization can be easily turned to minimization based on the duality principle. The resolution of a MOP yields a set of trade-off solutions, called Pareto optimal solutions or no dominated solutions, and the image of this set in the objective space is called the Pareto-optimal front. Hence, the resolution of a MOP consists in approximating the whole Pareto front. In the following, I give some background definitions related to multi objective optimization.

Definition 1: (Pareto optimality) A solution $x*$ is Pareto optimal if there does not exist any $x \in X$ such that $fm(x) < fm(x*)$ for all $m \in I$ where $I = \{1, 2, \ldots, M\}$.
The definition of Pareto optimality states that $x*$ is Pareto optimal, if no feasible vector x exists which would improve some objective without causing a simultaneous worsening in atleast another one. Other important definitions associated with Pareto optimality are essentially the following.

Definition 2: (Pareto dominance) A solution
$u = (u1, u2, \ldots, un)$ is said to dominate another solution
$v = (v1, v2, \ldots, vn)$ denoted by $f(u) \_ f(v)$ if and only if $f(u)$
is partially less than $f(v)$. In other words, $\forall m \in \{1, \ldots, M\}$,
I have $fm(u) \leq fm(v)$ and $\exists m \in \{1, \ldots, M\}$ where
$fm(u) < fm(v)$.

Definition 3: (Pareto optimal set) For a MOP $f(x)$, the
Pareto optimal set is $P* = \{x \in X \neg \exists x \in X, f(x) \_ f(x)\}$.
In the classical multiobjective optimization literature, this set is called the efficient set.
Definition 4: (Pareto optimal front). For a given MOP $f(x)$ and its Pareto optimal set $P*$, the Pareto-optimal front is
$PF* = \{f(x), x \in P*\}$.
MOPs have received considerable attention in operations research [26]. Traditional methods to solve MOPs transform the MOP to a SOP via an aggregation method. Several aggregative methods were proposed in the specialized literature. I cite, for example, the weighted-sum method, the ε-constraint method, the reference direction method, the light beam search approach, and so forth (see [3] for an exhaustive review). The main problem with these methods is that they require several parameters to tune subjectively by the decision

making which is not an easy task. Additionally, these methods depend on the geometrical features of the objective functions and they can provide only one solution in each simulation run. Due to their population-based nature, over the two last decades, MOEAs have been demonstrated to be effective black-box tools to handle MOPs (see [28] fora detailed recent survey).

B. Brief Summary of Existing NDSAs

Since one of the main contributions of this project is the proposal of a new quasi-linear average time complexity NDSA,I give in this section a brief summary of existing work in the nondominated area. The goal of any NDSA is to classify a population of solutions into nondominated fronts where:

1) two solutions belonging to the same front are nondominated with each other and 2) each solution belonging to front

Fk+1 is dominated by at least one solution from *Fk*. Solutions having better front ranks are preferred over solution having worse ones. The number of existing NDSAs in the literature is not large. These algorithms could be classified into two main families.

1) Pareto Dominance Decomposition-Based Algorithms:

The basic idea is to find the Pareto rank of each solution progressively by considering initially the first objective, then the second one, and so on. Since,I did not use all objectives simultaneously but rather sequentially to find the Pareto ranks,I can say that these algorithms decompose the Pareto dominance relation. The two main works in this family are and They both independently extend Kung *et al.*'s divide and conquer-based vector sorting algorithm which identifies only the first front. Based on Kung's study, this type of algorithms has a time complexity of $N(log(N))^{M-1}$ (where M is the number of objectives and N is the population size). It is important to note that Jensen's algorithm puts duplicates solutions in the same front which represents the major drawback for this algorithm.

2) Pareto Dominance Evaluation-Based Algorithms:

These algorithms finds the Pareto ranks based on the overall evaluation of the Pareto dominance between pairs of solutions (i.e., by considering all objectives simultaneously). Algorithms falling in this family could be categorized into two subfamilies.

1) *Linear Data Structure-Based Algorithms:* In this family,

I find mainly Deb *et al.*'s [9] NDSA where for each of the N solutions, two quantities are computed: 1) ni corresponding to the number of solutions dominating solution i and 2) si corresponding to set of solutions dominated by solution i. The calculation of these two entities requires $O(MN2)$ comparisons. After that, solution having $ni = 0$ are identified and saved in a list named $F1$. The latter is called the current front. Now, for each solution i in the current front, each member k from its set si is visited and its nk count is reduced by one. In doing so, if for any member k the count nk becomes zero, I put it in a separate list H. Once all members of the current front have been checked, I declare the solutions in the list $F1$ as solutions of the first front. I then continue this process using the newly identified front H as our current front. Each such iteration requires $O(N)$ computations. This process continues till all fronts are identified. Since at most there can be N fronts, the worst case complexity of this loop is $O(N2)$.

Hence, the overall time complexity of Deb *et al.*'s [9] NDSA is $O(MN2) + O(N2) = O(MN2)$. 2) *Nonlinear Data Structure-Based Algorithms:* The basic idea is to exploit nonlinear data structures coupled with the divide-and-conquer strategy in order to reduce the number required comparisons for identifying the different fronts. I find three recently proposed algorithms

in this family: 1) Fang *et al.*'s [37] NDSA; 2) the deductive sort; and 3) the climbing sort. The two latter are proposed by McClymont and Keedwell

Fang *et al.* NDSA uses a tree-like data structure, called dominance tree, in order to save the dominance information among solutions with the aim to reduce the number of repetitive (redundant) comparisons. On the one hand, a solution of a child node in the dominance tree cannot dominate any of its parent's siblings. On the other hand, the solution of a node may not necessarily dominate its siblings' children. The authors demonstrated that the best case of their algorithm corresponds to the situation where each front contains a unique solution; however, the worst one corresponds to the case where all solutions are nondominated with each others (they form a single front). The worst case complexity of Fang *et al.*'s [37] algorithm is demonstrated to be $O(N2)$.

The best case one is shown to be $O(MNlog(N))$. The average case is demonstrated to be $O((1/p)MNlog(N))$ where p is the probability that either of two solutions dominate the other. Since the p decreases during the MOEA run and with the increase of the number of objectives, the general time complexity of this algorithm remains near $O(N2)$. The deductive sort and the climbing one are based on a graph-like data structure called

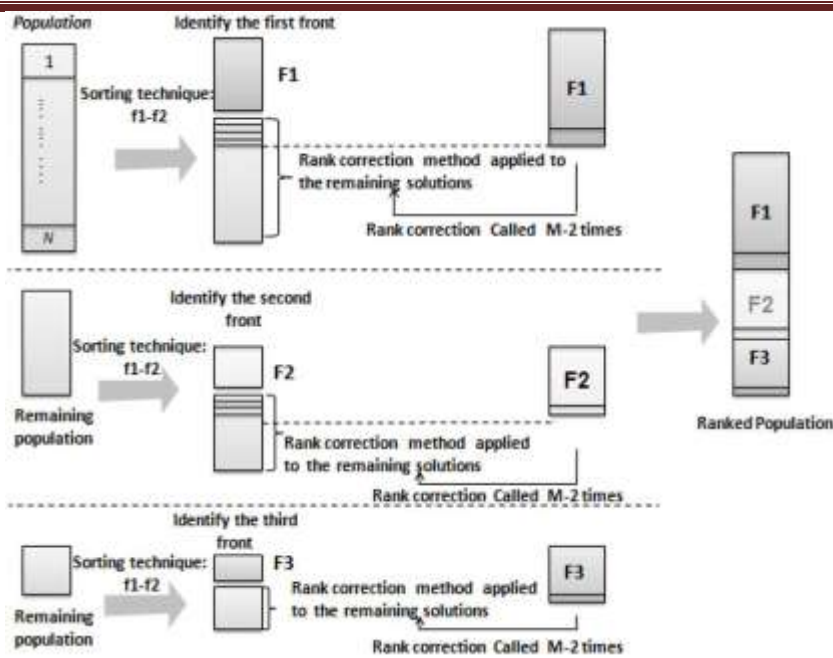
dominance graph. The latter is used to save the direct dominance, indirect dominance, and the mutual nondominance relationships between the different solutions. In this way, the algorithms using such data structures avoid not only redundant comparisons but also useless ones. The deductive sorting and the climbing one have the same worst case and best case as for Fang *et al.*'s [37] NDSA. Although, the number of comparisons is demonstrated theoretically and experimentally to be reduced when compared to Deb *et al.*'s [9] NDSA, the overall time complexity for each of these algorithms remains nonlinear with an order of $O(N2)$.

III. Background

Q-NDSA

1) Q-NDSA for the Bi-Objective Case: The bi-objective case of Q-NDSA is inspired by Jensen [34], Yukish [35], and Fang *et al.* [37]. The basic idea of Q-NDSA for the bi-objective case is as follows. Firstly, all population members are sorted based on the first objective $f1$. In this way, solutions preceding a particular solution u may dominate it. However, solutions following u cannot dominate it. Consequently, the first solution (having the minimal $f1$ value) is assigned a Pareto rank of 1, since it is not dominated with respect to all population members. Once the first nondominated solution is identified, the next step is to find the rest of nondominated solutions. In order to achieve this goal, I stock $f2(u)$ in a witness variable W and I visit the second solution v according to the sorting based on $f1$. If $(f2(v) < W)$ then v is assigned a rank of 1 since it is not dominated by u, and W takes $f2(v)$. This process is repeated for every solution of the rest of the population to find all members of the first nondominated level in the population. Once the first front $F1$ is identified, I discard it temporary from the population P and I reiterate the overall process with the aim to find next front $F2$, and so on. The algorithm repeats this process until the classification of all population embers into nonmdominated fronts. This fact can considerably reduce the number of comparisons and then can improve the computational efficiency of any Pareto dominance evaluation-based algorit-hms. Fig. 3 shows an example of execution of the bi-objective sorting. In what follows, I study the computational cost of our Q-NDSA for the bi-objective case from a time viewpoint: the best case of our algorithm corresponds to the case where all population members are nondominated with each others, Running example of Q-NDSA for the bi-objective case. and therefore they form a single front. For this case, I require $Nlog(N)$ computations to sort the population using quick-sort and $(N-1)$ comparisons with the witness variable W, in order to conclude that all members are nondominated. Thus, the overall time complexity for the best case is $O(Nlog(N) + (N-1)) \sim O(Nlog(N))$. The worst case corresponds to the situation where each front is composed by a single solution. In this way, I have N fronts. For this case, I require $Nlog(N)$ computations to sort the population using quick-sort and several comparisons to find the fronts. Therefore, in each iteration, one solution will be discarded from the population. The first front requires $(N-1)$ comparisons. The second one requires $(N-2)$, the third one necessitates $(N-3)$, and so on. The last front requires 0 comparison. The overall number of comparisons for finding all fronts is $Z = (N-1) + (N-2) + \cdots + 0$. I remark that Z corresponds to an arithmetic suite with a common difference of 1 and N terms. Consequently, $Z = (N)[(N-1)+0)]/2 \sim N2$.Hence, the overall computational cost for the worst case is $O(Nlog(N) + Z) \sim O(N2)$. The average case corresponds to obtaining number of fronts (NF) fronts where $NF <<< N$ (NF is too small compared to N). Consequently, I need as well $Nlog(N)$ computations for quick-sort and Y comparisons for the detection of the fronts. Assuming q to be the average size of a front, $Y = ((N-1) - q) + ((N-1) - 2q) + \cdots + 0$. Thus, the overall complexity for the average case is $O(Nlog(N) + Y) \sim O(Nlog(N) + N) \sim O(Nlog(N))$. In summary, the Q-NDSA for the bi-objective case has the following properties: 1) a quasi-linear time complexity of $Nlog(N)$ for the best case; 2) a quasi-linear time complexity of $Nlog(N)$ for the average case; and 3) a quadratic time complexity of $N2$ for the worst case.

2) Q-NDSA for the M-Objective Case: The basic idea of the Q-NDSA for the M-objective case is as follows. Once nondominated solutions based on the two first objectives are identified, I apply a rank correction strategy in order to find the rest of nondominated solutions for the M-objective case (see Fig. 4). In fact, solutions having a rank of 1 based on $f1$ and $f2$ remain always nondominated. However, solutions dominated based on $f1$ and $f2$ may become nondominated when



Algorithm 1 Q-NDSA Pseudocode
Input: Solution Set *P*
Output: Ranked *P*
01: **Begin**
02: *P* ← QuickSort (*P*, 1) /* Sort *P* based on the first objective *f*1*/
03: *Rank*← 1
04: F = *φ*
05: **While** (*P* _= *φ*) **do**
06: /*Step 1: Identifying the actual best front from (truncated) *P**/
07: /**Step 1.1: Identifying the first non-dominated solutions**/
08: *W*← *s*1.ObjValue(2) /* *S*1 is the current solution having the minimal *f*1value in truncated *P* */
09: **For** each *si* _*P* **do**
10: **If** (*si*.ObjValue *(2) <W*) **Then** /**Si* is not dominated by *S*1*/
11: *W*←*si.*ObjValue(2)
12: *si.Rank* = *Rank*
13: *F*←Union (*F*, *si*) /* Store the solutions belonging to the same front in a temporary variable *F**/
14: *P*←Remove (*si*, *P*) /* Remove *si* from (truncated) *P**/
15: **End If**
16: **End for**
17: /**Step 1.2: Identifying the rest of non-dominated solutions via rank correction**/
18: **If** (*M* >2) **Then** /*The correction step is required only when *M*>2*/
19: *m* ← 3
20: **while (**(*m* ≤ *M*) *and* **(***p* _= 0**)) do**
21: **RankCorrection (***F, F.Rank, P,m***)**
22: *m* ← *m* + 1
23: **End while**
24: **End If**
25: /*Step 2: Increment the actual Pareto rank to identify the next front*/
26: *Rank* ← *Rank* + 1
27: **End while**
28: **End**

IV. Experimental evaluations

In this section, I validate the performance of NCRO algorithm. Our experiments can be divided into two parts. The first one is devoted to compare NCRO against four well cited

MOEAs having different fitness assignment schemes. The algorithms are as follows. 1) NSGA-II which is based on Pareto-based ranking. 2) IBEA which uses indicator-based selection. 3) AGE which operates with a formal notion of approximation improved during the iterative process.
4) MOEA/D which decomposes the MOP into *N* subproblems (I use the Tchebycheff version of MOEA/D). The second one is dedicated to CPU time analysis in order to assess the efficiency of our approach from a computational time viewpoint. I note that all algorithms are coded in Java programming language and all simulations are performed on the same machine (Intel Core i7-4500 CPU 1.8 GHz, 8 GB RAM).
ARM algorithms, like Apriori-Inverse, the aforementioned property is meaningless, since the goal is to obtain any item set that satisfies at least one instance. Hence, infrequent rule mining algorithms need the whole search space to look for relations, so the computational time required just for mining a single rule is much higher than in other type of algorithms. Finally, it should be noted that there is no single run time for each specific number of rules or item sets, and the run time depends on the data set (number of both instances and attributes). For example, 100 rules could be discovered by a specific algorithm for a specific data set, whereas the same number of rules could be also discovered for a different data set having different features (either number of attributes or instances). Thus, the resulting set of values is a scatter plot that represents the relationship between the number of rules or item sets and the run time. In order to compare the results,I represent all the values as a linear regression, which models a set of *n* points in such a way that makes the sum of squared residuals of the model as small as possible. In this project,I have first run the Apriori algorithm considering its original data representation. Second,I have run the same algorithm with the proposed data structure. Note that the algorithm schema is the same, the only difference lies in the form in which data are accessed. Fig. 11 illustrates the linear regression (in a logarithmic scale) obtained from the results. The plot illustrates the run time required by the Apriori algorithm when both the original and the proposed data representations are considered. The proposed structure enables enhancing the performance of exhaustive search algorithms for mining frequent association rules. The difference in run time between the executing of Apriori with the original data structure and with the proposed one is even higher with the increasing number of rules output.

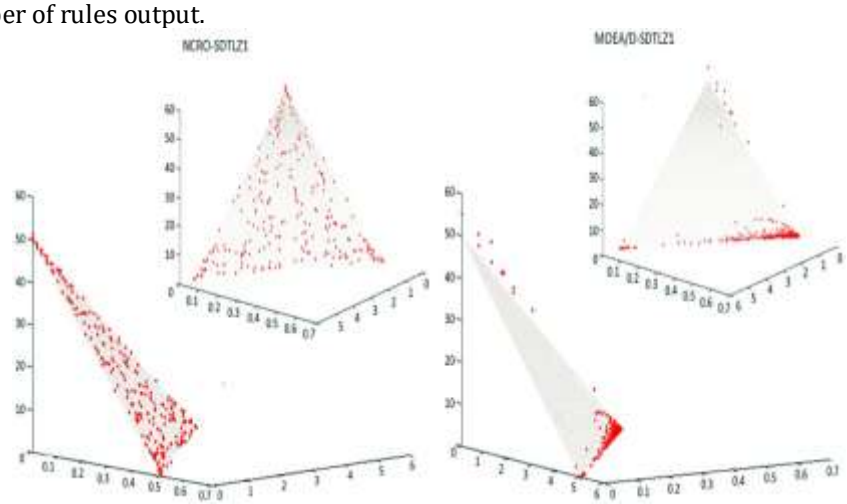


Fig. 6. Nondominated solutions with MOEA/D and NCRO on SDTLZ1.

V. Conclusion

In this project, I proposed a novel data structure specifically designed to used by ARM algorithms.ARM algorithms to deal with much larger volumes of data reducing the main of this approach to implement the algorithm without any modifications so it is a great advantage since the performance capabilities of many existing algorithms could be improved domain sizes of attributes in a data set are not so large. Luckily, this is often the case; even numerical attributes even for large data sets typically have a low number of distinct values meaning that there is a room for efficient indexing and effective index compression.
The shuffling stat orgy is the new storing capacity of the concept

REFERENCES

1. K. Deb, Black Multi-Objective Optimization Using Evolutionary Algorithms. New York, NY, USA: Wiley, 2001.
2. C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, Black Evolutionary Algorithms for Solving Multi-Objective Problems. Berlin, Germany: Springer, 2007.
3. K. Miettinen, Nonlinear Multiobjective Optimization. Dordrecht, The Netherlands: Kluwer Academic, 1999.

4. C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in Proc. Int. Conf. Genet. Algorithm (ICGA), San Francisco, CA, USA, 1993, pp. 416–423.
5. K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I: Solving problems with box constraints," IEEE Trans. Evol. Comput., vol. 18, no. 4, pp. 577–601, Aug. 2014.
6. K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part II: Solving problems with box constraints," IEEE Trans. Evol. Comput., vol. 18, no. 4, pp. 602–622, Aug. 2014.
7. J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in Proc. 1st IEEE Conf. Evol. Comput., Orlando, FL, USA, 1994, pp. 82–87.
8. S. Nidamarthi and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," Evol. Comput., vol. 2, no. 3, pp. 221–248, 1994.
9. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, Apr. 2002.
10. E. Ziztler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," IEEE Trans. Evol. Comput., vol. 3, no. 4, pp. 257–271, Nov. 1999.
11. Maddali M.V.M. Kumar, V. Suresh Babu, "An Efficient and Secure Data Storage Operations in Mobile Cloud Computing", International Journal of Scientific Research in Science, Engineering and Technology, Vol. – 04; Issue - 01; ISSN: 2394-4099; pp 1385-1390, 2018.
12. E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," Evol. Methods Design Optim. Control (EUROGEN), Barcelona, Spain, 2001, pp. 95–100.
13. D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multi-objective optimisation," in Proc. Parallel Probl. Solving Nat. (PPSN VI), Paris, France, 2000, pp. 839–848.
14. E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in Proc. Parallel Probl. Solving Nat. (PPSN VIII), Birmingham, U.K., 2004, pp. 832–842.
15. A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolumebased multiobjective optimization: Theoretical foundations and practical implications," Theor. Comput. Sci., vol. 425, no. 1, pp. 75–103, 2012.
16. D. H. Phan and J. Suzuki, "R2-IBEA: R2 indicator based evolutionary algorithm for multiobjective optimization," in Proc. IEEE