A Novel Case base Indexing Model based on Power Set Tree

Khaled El-Bahnasy
Information Systems
Ain Shams University
Khalifa El-Maamon st,
Abbasiya sq., Cairo
Egypt

Kareem Mohamed Naguib
Computer Science
Ain Shams University
Khalifa El-Maamon st,
Abbasiya sq., Cairo
Egypt

Mostafa Aref
Computer Science
Ain Shams University
Khalifa El-Maamon st,
Abbasiya sq., Cairo
Egypt

ABSTRACT
CBR has been successfully applied to the areas of planning, diagnosis, law and decision making among others. It uses useful prior cases to solve the new problems. CBR must accurately retrieve similar prior cases for getting a good performance. Throughout this thesis The Novel Case Base Indexing Model based on Power Set Tree has been introduced. A custom solution designed and built to find the unique combinations for each case in a Case Base. Then use these unique combinations to build the Case Base Index. Finally, a better algorithm has been built to balance the resources consumptions and harness them to serve the purpose of finding the unique combinations for large cases that has more than 38 finding.

Keywords
Case Based Reasoning – Pre-processing - Classification – Power Set – Tree – Vectors

1. INTRODUCTION
The mission of generating a case base index based on Power set tree can be divided into three major phases which will be discussed in details later on throughout this chapter. Most of systems and techniques relied on rough set to get uniqueness; this model will combine big data and knowledge management with Case based reasoning. Big data can affect any domain it is used in. Old cases are repeated consequently and they almost use the same solution with little modifications. Abundant techniques have been used to deal with big data and how knowledge extracted from it. Effective data and knowledge manipulation are corner stone in all situations.

In business, Enterprise firms like IBM, Microsoft or SAP realized the importance of data to themselves and their customers. They are investing hundred millions of dollars every year in big data researches and improve their software performance regarding big data manipulation and knowledge extraction in order to support their customers' decisions in the current fast-changing business environments.

In this paper we are going to explain in details with case studies the novel model and how can it affect both the performance and accuracy of finding unique features among set of features.

1.1 The Novel Case Base Indexing Model
In this section the overview of Case Based Indexing Model will be discussed, **Fig 1** illustrates the CBR Life Cycle and which stages will be enhanced & improved using this model (surrounded by black rectangle). This model will mainly focus on fast cases retrieval and apply solutions to new case.

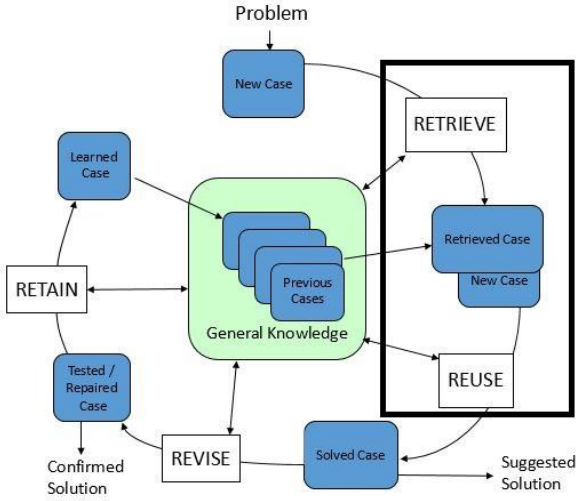

Fig 1: CBR Life Cycle

This model generates a case base index. The case base index consists of unique combinations for each case in the original case base. These unique combinations are added to the case base index and refer to the original cases. This model uses the Power Set methodology to get all unique combinations and prove how power set tree is a complete solution. The generated index can be used for different purposes such as reasoning and building decision trees.

The Novel Case Base Indexing Model Main Phases:-

1- Original Case Based is managed to get the case base index, this process is the most crucial process among all other processes.
2- The most similar cases to a new case are retrieved in short time, using the Case Base Index. The similarity weight for each retrieved case is calculated. Sort retrieved cases descending.
3- Try the proposed solutions, generate the power set for the new case, add it to Original Case base and add unique combinations to Case Base Index.

Fig 2 illustrates the processes of generating the Case Base Index from original one. The generated Case Base Index will be used later on, to find the most similar cases to new case.

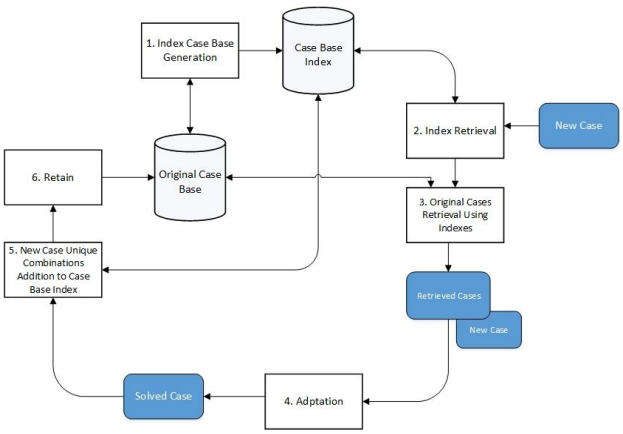

Fig 2: Indexing Case Base Model View

The Case Base Indexing model consists of 6 Module:

1- Case Base Index Generation: - The power set tree will be generated for each case individually, then find the unique findings combinations and add them to the Case Base Index.

2- Index Retrieval: - The most similar cases indexes will be retrieved from Index Case Base, measure the weight of similarity for each index retrieved and sort them descending.

3- Original Cases Retrieval: The original cases will be retrieved from Original Case Base and pass these cases to the next module "Adaptation".

4- Adaptation: - Transforms the retrieved solution into an appropriate solution for the current case using Substitution adaptation technique.

5- New Case Unique Combinations Addition to Case Base Index: - The power set tree will be generated for the new case, to find the unique findings combinations and add them as index in the Case Base Index.

6- Retain: - The new case will be added to the original case base.

The Novel Case Base Indexing Model proves how uniqueness is important and how can it help in resolving problems with higher accuracy than any other techniques. Power set is a magical complete solution for finding unique features in several domains, but power set complexity is growing exponentially. Power Set generation provides 2^n**-1** combinations**.** Due to this exponential complexity, power set generation becomes very complex. To generate power set tree you will need to provision resources that can handle this high complexity. Although Power set is complete solution as it generates all possible combinations, its complexity still an impediment to use it in powerful manner. In [1], they introduced a Tree Structure called Power Set Tree (PST), which is an ordered tree that represent power set and each possible reduct is mapped to a node in the PS-tree using Deleting Feature technique. The novel case base indexing model based on reversing their technique and generate the power set tree but by Adding Feature as shown in **Fig 3.** Power Set Tree (PST) is used to leverage the benefit of pruning feature which will decrease the number of comparisons and visited nodes.

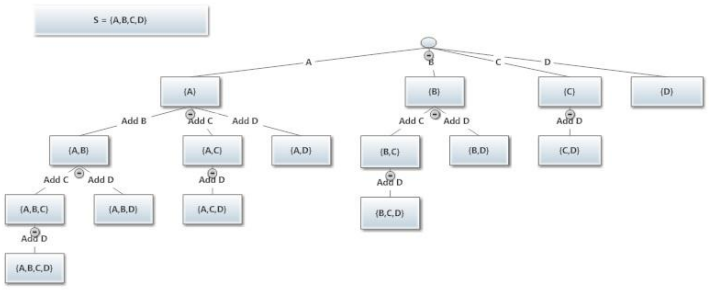

Fig 3: Power Set Tree by Adding Feature for S = {A, B, C, D}

2. LITERATURE SURVEY FOR CLASSIFICATION & DECISION TREE TECHNIQUES

The classification task can be seen as a supervised technique where each instance belongs to a class, which is indicated by the value of a special goal attribute or simply the class attribute [2].

The goal attribute can take on categorical values, each of them corresponding to a class.

In a paper "A Rough Set approach to feature selection based on Power Set Tree" [3], they discussed the motivation of feature selection in data mining and machine learning is to reduce the dimensionality of feature space, improve the predictive accuracy of a classification algorithm, and improve the visualization and the comprehensibility of the induced concepts. In this Paper, they introduced a Tree Structure called Power Set Tree (PST), which is an ordered tree that represent power set and each possible reduct is mapped to a node in the PS-tree. They gave two kinds of PS-tree-based rules for pruning unpromising parts of the search space. Two novel feature selection algorithms based on PS-tree are also given. One is a complete algorithm which can guarantee to find the minimal reduct. The other is a heuristic algorithm based on PS-tree. The performance of the first algorithm is compared with that of the strong equivalence method. The performance of the second algorithm is compared with that of traditional hill-climbing algorithms and stochastic algorithms. Trees provide us an efficient way to solve many problems. The power set tree (PS-tree) is a tree structure to represent the power set in an order fashion. Since the PS-tree completely enumerates the subsets of a power set using a particular order, it can represent the search space of a particular feature selection problem. **Fig 6** illustrates the Power Set Tree (PST) for <a,b,c,d>.

The PST size is growing exponential $2^n - 1$ where **n** is the number of elements in set. And Due to its exponential size it is impossible to completely explore it, so they used **Pruning Rules** to eliminate a branch of the search tree from consideration without examining the nodes in the Branch.

In another paper "Effects of data set features on the performances of classification algorithms" [4], they evaluated scenarios that examine which data set characteristics most affect the classification algorithms' performance. It is still a complex issue to determine which algorithm is how strong or how weak in relation to which data set. In this research they have experimentally examined how data set characteristics affect algorithm performance, both in terms of accuracy and in elapsed time. The classification algorithm is widely applied from natural science to business applications such as customer

relationship management (CRM) software, finance, marketing segmentation, location-based services, and more. Even in an application domain, it is important to select a classification algorithm that is optimal for adjusting to the customer's current context, because any change in the structure or data set content can potentially affect the algorithm's classification performance. Working with big data sets is growing today as an important issue for business intelligence. Big data is defined as a large data set where legacy data analysis tools cannot effectively gather, store, search, or analyze primarily because the volume of data generated exceeds the capability of analysis tools and data storage [5]. To sufficiently process big data, a system has to be equipped with certain capabilities. First, the system must gather and process in a timely manner the large volumes of data that flow from various sources. Big data is also called the 3Vs: Volume, has a large data set; Velocity, must be processed quickly, and Variety, sourced from very diverse data sources. This renders current analysis tools nearly ineffective [6]. According to The Economist, 150 Exabyte of data were created in 2005; in 2010 that number was expected to rise to 1200 EBs. Big data includes text and multimedia gained from various inputs and sensors, thus, justifying the need to develop a special algorithm or toolkit like Python to structuralize the unstructured data [7]. Data visualization at the macro level also has been suggested to support big data analysis in inspecting the correlation among data. For example Maltego, which visualizes LinkedIn™ data, is an excellent tool to mine social network information from various and unstructured data sets. Other tools proposed for unstructured data processing include Google's BigQuery, Amazon's AWS (Amazon Web Service) and NoSQL. However, none of them support classification as data analysis.

Recently, real time analysis of data sets, derived from multiple sensors, has been increasingly critical. This intelligence helps make business applications more successful by providing wider knowledge detection, perspective sharing, and agile decision making. Such data sets are characterized by size, diversity of sources and data formats, and the frequency of their updates. These are seldom considered in conventional data mining algorithms. Despite its importance, few studies have investigated what makes the performance of data mining algorithms under such situations increase or decrease. In this paper, focusing on classification algorithms, they examined which characteristics of a data set influence the performance classification algorithms. As a result, classification algorithms show different performance about different kind of data structures, content and context. This implies that context-aware selection of classification algorithms will be meaningful in selecting optimal algorithms.

3. POWER SET GENERATION TECHNIQUES

Power Set generation has different techniques and methodologies. In The Novel Case Base Indexing Model, power set generation is considered as the most crucial stage among the whole model. If Power Set Generation fails, the whole model will be collapsed. Hence, Power Set Generation has to be accurate, fast and guaranteed to ends up. In this section, 3 different techniques will be presented in details. The first algorithm discusses Power Set generation using Vector, The rest techniques will be based on Power Set Tree. These techniques illustrate the difference between different methodologies, the strengths and weaknesses of each methodology. Before start to proceed through the Power Set Generation Algorithm using Vector, some of terminologies and symbols have to be shown first.

S = Set, **R** = Rule, **F** = Finding, **D** = Disorder, **UKB** = Unique Knowledge Base, **OKB** = Original Knowledge Base

Subsumsion = Set is entailed from another smaller set, eg. **S1** = {A, B}, **S2** = {A, B, C}, **S2** is Subsumsion from **S1**.

In the first experiment of implementing Power Set to get all unique combinations. Vector is used as data structure, the algorithm shown in **Fig 3** illustrates how to get unique combinations using power set and vector.

Unique Combinations Algorithm using Power Set & Vector:

Vector Algorithm:

1- Sort all rules ascending with respect to number of findings per each rule.
2- For Each Rule in the OKB.
2.1 Read Rule "R".
2.2 Generate Power Set for its Findings "F".
2.3 Loop on each Combination of Findings.
2.3.1 If (Combination is Unique).
2.3.1.1 IsSubSumsion (Finding "F").
2.3.1.2 If Yes, Go to Step 2.3.
2.3.1.3 If No, Add Finding "F" to the UKB.
3- Go to Step 2.

Fig 3: Unique Combinations Algorithm

Power Set Generation using Vector Algorithm shown in **Fig 3** can be considered as the brute-force technique to generate power set. Possible combinations are generated consequently and saved to Vector. After generating all possible combinations, vector values are to be checked for unique combinations. If any combination found as unique, it must be checked if it is subsumsion from another unique combination or not. Finally, add the unique and not subsumsion combinations to the Case Base Index.

Power Set Generation using Vector has its pros & cons. **Table 1** illustrates these pros & cons.

Table 1. Power Set Generation using Vector Pros & Cons

Pros	Cons
• Easy to Implement	• Cannot be parallelized. • Number of Comparisons is equal to number of power set generated reductions ($2^n - 1$). • Causes an Out of Memory with Sets that have more than 20 finding. • Unbalanced utilization of CPU. • 2 checks required for each combination (Uniqueness Check, Is Subsumsion check)

The following experiment showing how to apply this algorithm on Knowledge Base to get Unique Combinations: Suppose the Original Knowledge Base (OKB) contains the following rules:

R1: F3, F4, F6, F7 → **D1**
R2: F1, F4 → **D1**
R3: F6, F8, F9 → **D2**
R4: F5, F6, F7 → **D3**
R5: F2, F8 → **D3**
R6: F2, F3, F5 → **D3**
R7: F3, F7, F10 → **D4**
R8: F1, F7, F10 → **D5**

Reference to Power Set Generation with Vector Algorithm, all Rules (R) have to be sorted ascending with respect to number of Findings (F) in each rule.

After Sorting the Original Knowledge Base (OKB,) the rules would be as the following:

R2: F1, F4 → **D1**
R5: F2, F8 → **D3**
R3: F6, F8, F9 → **D2**
R4: F5, F6, F7 → **D3**
R6: F2, F3, F5 → **D3**
R7: F3, F7, F10 → **D4**
R8: F1, F7, F10 → **D5**
R1: F3, F4, F6, F7 → **D1**

Initially the Unique Knowledge Base (UKB) is empty. By applying the algorithm mentioned in **Fig 3**, Power Set will be generated for each rule, all combinations will then added to the Vector. After adding all possible combinations to the Vector, each combination will be checked for uniqueness and subsumsion. If any combination found as unique and is not subsumsion, this combination will be added to the Unique Knowledge Base (UKB).

Finally the Unique Knowledge Base (UKB) contains the following rules:

UR1: F4 → **D1**
UR2: F2 → **D2**
UR3: F9 → **D2**
UR4: F6, F8 → **D2**
UR5: F5 → **D3**
UR6: F3, F10 → **D4**
UR7: F1, F7 → **D5**
UR8: F1, F10 → **D5**
UR9: F3, F6 → **D1**

As shown in the previous sample, number of comparisons is equal to the number of combinations generated (2n – 1). 2 checks had to be performed for each finding, one for uniqueness and another one for IsSubsumsion check. For this case the total number of comparisons = 56 Comparison.

From Real Case with different KBs, it was found that by using Core 2 Duo CPU and 4 GB RAM, the algorithm cannot work on more than Rule with 20 findings (2^{20} – 1). Performing all these computations consumed all available memory and CPU. Therefore, the data structure should be changed and generate a Power Set Tree then perform a breadth first search for each level. If any node found as unique, delete it from the tree and all its children. This means that the IsSubsumsion check will not be performed anymore, number of comparisons will be reduced too.

The algorithm shown in **Fig 4** describes how Power Set Tree can be used to get unique combinations, with less number of comparisons than used in vector and benefit from tree pruning techniques. The whole Power Set Tree reducts will be built first, and then the breadth first search applied for each level. Power Set Tree (PST) generation ordering is Lexicographic Ordering. Building the Power Set Tree (PST) with this methodology and ordering causes some limitations related to memory consumption and unbalanced utilization of CPU. If Power Set Tree (PST) will be generated to a case with more than 25 finding, memory will be over-filled with tree nodes. That will cause an Out of Memory Exception and system will totally crash.

Unique Combinations Algorithm using Power Set Tree:

1. **Sort all rules ascending with respect to number of symptoms per each rule.**
2. **For Each Rule in the OKB.**
   - 2.1 Build the Power Set Tree.
   - 2.2 Perform a Breadth First Search.
     - 2.2.1 If any node found as unique
       - 2.2.1.1 If this node has only one finding.
         - 2.2.1.1.1
   - 2.3 Add Finding to the UKB.
   - 2.4 Delete this node from the tree.

Fig 4: Unique Combinations Algorithm using Power Set Tree

The Unique Combinations Algorithm using Power Set Tree is to be applied for the same Original Knowledge Base mentioned in the previous experiment, to show the difference and benefits of using Power Set Tree (PST) than Vector. The same Unified Knowledge Base (UKB) generated from Power Set Tree Algorithm but with better performance. The main and clear difference between Vector and Power Set Tree was in the number of comparisons. Using power set tree, algorithm did not have to visit or perform some comparisons, although these comparisons might be unique, but it is subsumsion from another unique finding. Power Set Tree (PST) leveraged the capability to know subsumsion combinations without even checking them.

Vector used **56** comparison to generate the Unified Knowledge Base (UKB), but Power Set Tree (PST) used only **31** comparison to get the same result. This difference would greatly affect performance with large and real data sets.

Table 2 illustrates the main Pros & Cons of Power Set Tree algorithm.

Unique Combinations Algorithm using Power Set gave us the ability to find unique combinations for rules with up to 27 finding with the same infrastructure used in Vector Algorithm. These rules would generate Tree of (2^{27} – 1) node. Because of this huge number of nodes the memory is filled if the whole tree generated at once. This means that we need to duplicate the memory with each +1 increase in the rule finding, as Power Set generation complexity is growing exponentially. These results still unsatisfactory, the main goal of this research is to generate the power set tree for more than 2^{38}. Further enhancements should be done to this algorithm and generate PST level by level in order to avoid out of memory exception.

Finally, the conclusion of these two experiments is that Vector implementations could not exceed the rule with more than 20

finding which means 2^{20} probability with unbalanced consumption of resources, but the Power Set Tree (PST) technique can reach 2^{27} with better performance and better resources consumption.

Table 2. Power Set Generation using Power Set Tree Pros & Cons

Pros	Cons
• Tree Pruning Feature. • No Subsumsion Check.	• Cannot be parallelized. • Causes an Out of Memory Exception with Set that have more than 27 finding. • Unbalanced utilization of CPU.

4. THE NOVEL CASE BASE INDEXING MODEL IMPLEMENTATION

The implementation of the Novel Case Base Indexing model shown before is going to be discussed in this section. The model has been implemented using C# programming language, .Net Framework 4.0, TPL, LINQ query language and XML as the Case Base Files. No external off-shelf tools have been used. The whole model has been implemented using C#, LINQ and XML. Language-Integrated Query (LINQ) is a set of features have introduced in Visual Studio since 2008 that extends powerful query capabilities to the language syntax of C# and Visual Basic. LINQ introduces standard, easily-learned patterns for querying and updating data, and the technology can be extended to support potentially any kind of data store. Visual Studio includes LINQ provider assemblies that enable the use of LINQ with .NET Framework collections, SQL Server databases, ADO.NET Datasets, and XML documents. Throughout the implementation, a partial parallelization in generating the Case Base Index Stage using the Task Parallel Library (TPL) has been implemented. The Task Parallel Library (TPL) is a set of public types and APIs in .Net Framework. The purpose of the TPL is to make developers more productive by simplifying the process of adding parallelism and concurrency to applications. The TPL scales the degree of concurrency dynamically to most efficiently use all the processors that are available. In addition, the TPL handles the partitioning of the work, the scheduling of threads on the ThreadPool, cancellation support, state management, and other low-level details. By using TPL. Starting with the .NET Framework 4, the TPL is the preferred way to write multithreaded and parallel code. However, not all code is suitable for parallelization; for example, if a loop performs only a small amount of work on each iteration, or it doesn't run for many iterations, then the overhead of parallelization can cause the code to run more slowly. Furthermore, parallelization like any multithreaded code adds complexity to program execution.

4.1 Real Case Bases

This section presents the results of The Novel Case Base Indexing Model. This model has been applied to 5 real case bases with different complexities, these cases will be divided into two parts (Plants, Animals). Plants case bases are varying from small to medium case bases complexity. The Animals case base is big case base. Throughout this chapter the Reasoning tool will also be presented, it can be attached to the original system. The Reasoning Tool can be used in diseases diagnosis with high efficiency. This tool can add a great value to the whole model as it can be used to test the resulted unique combinations, by inserting each unique combination and make sure that this combination retrieve only one disorder. **Table 3** showing all cases that will be used to prove the efficiency of The Novel Case Base Indexing Model. These cases are real cases received from National Agriculture Research Center**.**

Table 3. Original Case Bases Properties

	Plants				Animals
Case Base	Rice	Strawberry	Cucumber	Wheat	Animal Diseases
# of Cases	33	69	130	334	3575
# of Disorders (Classes)	18	31	44	41	489
# of Findings (Tuples)	93	255	511	1659	46023
Maximum Number of Tuples per Case	5	7	8	8	37

Table 3 describes the properties of each case base. Each case base complexity is classified based on the number of cases, number of classes and the maximum number of tuples per case. These cases are saved in XML files with the format shown in **Fig 5**. Case is consists of 2 attributes (Disorder, Name), and variable number of tuples depending on Case Complexity. Each Tuple consists of 3 attributes CPT stands for Concept, Prop stands for Property and Val stands for Value.

```
<Case Disorder="Iron_def" Name=" Iron_def ">
        <Tuple Val="رملية" Prop="Type" Cpt="Soil"/>
        <Tuple Val="أخضر" Prop="Vein_color" Cpt="Leaves_obs"/>
        <Tuple Val="الأوراق العليا" Prop="Color_Position" Cpt="Leaves_obs"/>
        <Tuple Val="اصفر" Prop="Color" Cpt="Leaves_obs"/>
</Case>
```

Fig 5: Sample of Case Representation

The Implemented system is managed to handle only the format mentioned in **Fig 5**. In the next phases we are planning to build Formatter system to convert from some popular formats to this format. Language Integrated Query (LINQ to XML) has been used to connect with XML files, and gain full control over XML file. Task Parallel Library (TPL) has been used to implement partial parallelization on class level. The Case Base is segmented according to its classes. Each class has different cases to describe it, these cases are passed to a thread to work on them sequentially.

4.2 Outputs

As a result of finding the unique combinations for each disorder using the Power Set Tree (PST). The resulted files will be used later as the case base index. **Table 4** showing the

properties of each file. Cases Representation in the case base index would be the same as the original case base to facilitate the matching process.

Table 4. Case Base Index Properties

	Plants				Animals
Case Base	Rice	Strawberry	Cucumber	Wheat	Animal Diseases
# of Cases	60	99	185	140	21606
# of Disorders (Classes)	18	29	42	37	433
# of Findings (Tuples)	76	143	286	204	58092
Maximum Number of Tuples per Case	2	3	2	3	5

As shown in **Table 3, Table 4** the total number of cases has been increased, due to the number of unique combinations generated from each case. Each case might has several number of unique combinations. These unique combinations saved in the Case Base Index with reference to the Original Cases. However, although the number of cases has been increased but the complexity of each case is reduced. For Instance, in the Animal Diseases Case Base, the maximum number of tuples per case was 37, but in the Case Base Index it is only 5. Thus the number of comparisons will be reduced and that would greatly affect the performance of retrieving and comparing cases processes. **Table 3, Table 4** demonstrates the difference between number of disorders in the Original Case Base, and the Case Base Index.

In **Table 4**, number of disorders (classes) in the original case base is decreased in the case base index. This phenomenon means that some disorders were sumbsume from another disorders which considered a not accurate cases and must be neglected to not affect the reasoning or matching with the new cases processes, this would be discussed thoroughly in the Analysis part. Moreover, **Table 4** showing that maximum number of tuples per case has been decreased too. This methodology of generating the case base index is effective with large case bases like the Animal Diseases case base. Later on the percentage of data refinement can be used to expect to which level we have to build the Power Set Tree (PST) for any further cases. For Example. In The Animal Diseases Case Base, the tree has been built according to the maximum number of tuples, if the case has 37 tuple, the PST will be built with 37 levels with 2^{37} Probability. However, the greatest number of tuples in the Case Base Index was 5, which means that from level 6 to level 37 no unique combinations found and all these comparisons were useless. If the tree had been built to only level 5, a substantial saving of memory space and CPU consumption could be done. **Table 3, Table 4** showing the difference between number of tuples in original and index case base. These differences in number of disorders or tuples reveals that there are some cases that subset from another larger cases.

For Instance from Animal Diseases Case Base, **Table 5** shows a case from Animal Diseases Case Base. After refinement and generation of Case Base Index, the same case would be represented as shown in **Table 6**. One can clearly see that case complexity has been decreased. The original case has 37 Tuple with number of combinations equals to $2^{37} - 1$, but the refined case has only 3 tuples. The refined case can be used in several situations and for different purposes like reasoning or diagnosis.

Table 5. Original Case from Animal Diseases Case Base

Disorder	Property	Value
Anthrax (acute form)	complaint	abnormal coloured urine
	vulva_size	bloody
	vulva_size	swollen
	milk_character	pink to red
	milk_production	decreased
	sub_maxillary_region	edema
	rumination	stopping
	rumen_contractility	sluggish
	briskit	edema
	temperature	high fever
	pulse_rate	increased
	heart	tachycardia
	respiratory_rate	increased (polypnea)
	oral_mucosa	hyperemic
	nasal_discharge_color	bloody
	nostrils	discharge
	muzzle	dry
	eye_conjunctiva	congested
	eye_discharge_nature	bloody
	eye_discharge	yes
	ears_touch	hot
	swelling_touch	warm
	swelling_tenderness	painful
	swelling_consistency	doughy
	swelling_site	briskit
	swelling_site	limbs
	swelling_site	scrotum
	swelling_site	chest
	swelling_site	abdomen
	swelling	yes
	urine_color	red
	feces_contents	bloody
	feces_color	black
	defecation	diarrhea
	water_drinking	ano-dipsia
	appetite	anorexia

After finding the unique combinations for the case represented in **Table 5**, the case will be presented in the case base index as shown in **Table 6**.

Table 6. Refined Case from Animal Diseases Case Base

Disorder	Property	Value
Anthrax (acute form)	rumination	stopping
	muzzle	dry
	swelling	yes

4.3 Analysis

This section will discuss and explain the output of the Case Base Indexing Model. From the results generated in the Case Base Index, we found some phenomena like shown in **Table 3, Table 4.**

1. Disorders in the original case bases are greater than disorders in the case base index.
2. The total number of cases in the case base index is greater than the Original Cases.
3. Number of Tuples in the Case Base Index is less than the number of tuples in the original case base.

These phenomena will be discussed and explained separately for Plants & Animals Case Bases.

Case bases output **(Table 7)**. **Table 7** illustrates the difference between Original Case Base and the Case Base Index. Throughout this section an explanation will be introduced. Why these phenomena appeared and the benefits of them.

Table 7. Optimization Ratio for Original Case Bases

Case Base Name		Number of cases	Number of findings	Number of Disorders	Average Findings/Case	Optimization Ratio
Cucumber	Original CB	130	511	44	3.93	61%
	Index CB	188	286	42	1.52	
Rice	Original CB	33	93	18	2.81	55%
	Refined CB	60	76	18	1.26	
Strawberry	Original CB	69	255	31	3.69	60%
	Refined CB	99	143	29	1.44	
Wheat	Original CB	334	1659	41	4.96	70%
	Refined CB	140	204	37	1.45	
Animals Disorders	Original CB	3575	46023	489	12.87	79%
	Index CB	21606	58092	433	2.68	

As shown in **Fig 3, 4, 5,** disorders in the case base index are less than total number of disorders in the original case base. This happens because some disorders are subsumed from other disorders.

These cases could be considered as in-accurate cases or need more refinement to refer properly to the disorder. This also means that the technique has a self-cleaning feature which ignores in-accurate cases for better results. The generated cases in-accurate cases can be referred back to the expert in order to correct or eliminate them from the case base.

The second finding is that the total number of cases are much greater than that cases in the original case base. This happens because one case in the original case base can generate more than one unique combination. The more the case has tuples the more probability of generating large number of unique cases. Although these unique cases might appear as large number of cases, but substantially their complexities are less than the original cases. The Unique cases in case base index has less number of tuples which simplify the comparison process in case of any new case. Moreover, these unique cases can be used to build one tree for the whole case base . Moreover, these cases can be perfectly used in reasoning and diagnosis and would be rapidly retrieve results instead of searching and comparing to large number of cases in the original case base.

The third finding is that number of tuples in the Case Base Index is less than number of tuples in the Original Case Base which means that there were some tuples never found as unique and never selected in any unique combination. These tuples can be considered a part of the original cases base cleansing part. The tuples that never found as unique are useless and only increase the complexity of each case.

5. CONCLUSION & FUTURE WORK

In this paper we have discussed an indexing method to improve the performance of indexing and retrieving in the data warehousing. Throughout this thesis The Novel Case Base Indexing Model based on Power Set Tree has been introduced. A fully customized solution has been designed and built to find the unique combinations to each case in a Case Base, and use these unique combinations to build the Case Base Index. We get over a lot of unbalanced consumption of resources, finally, we have built a better algorithm to balance the resources consumptions and harness them to serve the main purpose of this research in finding the unique combinations for large cases that has more than 38 finding. The main strengths of this model that it is applicable for any domain. The Generated Case Base Index can be used for many purposes beyond only being a Case Base Index. After the completion of this thesis. Moreover, A complete solution has been implemented to build the Case Base into our format along with reasoning tool to justify any results and a statistics solution to measure the main difference between the original case base and the case base index.

6. REFERENCES

[1] Yumin Chen, Duoqian Miao, Ruizhi Wangb, Keshou Wua , A rough set approach to feature selection based on power set tree, Knowledge-Based Systems, 275–281, 2011.

[2] Sunita Beniwal, Jitender Arora, Classification and Feature Selection Techniques in Data Mining, Department of Information Technology, Maharishi Markandeshwar University, Mullana,, August 2012, Ambala-133203, India

[3] Yumin Chen, Duoqian Miao, Ruizhi Wangb, Keshou Wua , A rough set approach to feature selection based on power set tree, Knowledge-Based Systems, 24 (2011) 275–281.

[4] Ohbyung Kwon, Jae Mun Sim, Effects of data set features on the performances of classification algorithms, Expert Systems with Applications, 40 (2013) 1847–1857

[5] Manyika, J., Chi, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C.. Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute, 2011

[6] Madden, S. From databases to big data. IEEE Internet Computing, pp. 4–6. Maltego (2012), Paterva. <http://www.paterva.com>, access date: May, 2012.

[7] Bradbury, D. Data mining with LinkedIn. Computer Fraud and Security, 2011(10), 5–8.

[8] Cao, L., Domain-driven data mining: challenges and prospects. IEEE Transactions on Knowledge and Data Engineering, 22(6), 755–769, 2010

[9] Yuan-Hai Shao, Wei-Jie Chen, Wen-Biao Huang, Zhi-Min Yang, Nai-Yang Deng, The best separating decision tree twin support vector machine for multi-class classification, Procedia Computer Science 17 1032 – 1038, 2013.

[10] Ohbyung Kwon, Jae Mun Sim, Effects of data set features on the performances of classification algorithms, Expert Systems with Applications, 40,1847–1857, 2013.