Big Data Analytics for Industrial Process Control

Abdul Rauf Khan
Department of Electronic Systems
Aalborg University, Denmark

Henrik Schiøler
Department of Electronic Systems
Aalborg University, Denmark

Murat Kulahci
Dept Applied Mathematics and Computer Science
Technical university of Denmark and
Luleå University of Technology Sweden

Torben Knudsen
Department of Electronic Systems
Aalborg University, Denmark

Abstract

Today, in modern factories, each step in manufacturing produces a bulk of valuable as well as highly precise information. This provides a great opportunity for understanding the hidden statistical dependencies in the process. Systematic analysis and utilization of advanced analytical methods can lead towards more informed decisions. In this article we discuss some of the challenges related to big data analysis in manufacturing and relevant solutions to some of these challenges.

KEYWORDS; Big data analytics for industrial process, Cloud Computing, Data Mining, Genetic Algorithm.

I. Introduction

We are currently going through some exciting times of technological advances in many fields. These advances across the fields are leveraging each other, causing profound shifts across all industries [1] and forging new types of markets. More interestingly, disruptive technologies through disruptive business models are enabling this shift with unprecedented pace. Experts are seeing this as the outset of the fourth industrial revolution: the digital revolution [2]. The concept of digital revolution is closely related to the onset of the information age and advances in Information and Communication Technology (ICT) [3], [4]. In this article we will briefly discuss some of the recent advances in Information and Communication Technology and their potential applications in industrial manufacturing. More specifically, this article focuses on the technologies related to information processing and data analytics in industrial manufacturing.

As mentioned before, technological improvements in different fields are acting as an amplifying factor for each other. Recent developments in the fields of computer science, machine learning and data mining constitute a quintessential example of this phenomenon. The foundation of data mining is entrenched in the mathematical/statistical techniques that are developed within the fields of artificial intelligence and machine learning. But most of these advanced methods are computationally intensive which calls for the use of all available computational resources. In fact, the igniting factor of the whole domain of knowledge discovery and data mining is the advancements in computer science. Indisputably, present processing capabilities are unsurpassed but still acquisition of sufficient computational resources is a demanding factor in data mining, as the size of available data is increasing exponentially.

978-1-5090-6505-9/17/$31.00 ©2017 European Union

I. Advances in Computing technology

Today, we enjoy unprecedentedly faster computer networks. Beside its other benefits this improvement in network communication technology has turned into the solution to some of the computational challenges in big data analysis, through distributed computing or more precisely cloud computing. Buyya [5] considers computing clouds as a "next-generation data centres with nodes "virtualized" through VM's (virtual machines)". He notes "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers".

In general, while dealing with large data sets, we mainly encounter two key computational challenges:

- Memory limitations

- Computational limitations

Although our main focus in this article is on the cloud computing paradigm in distributed computing, it will be worthwhile to discuss some of the other widely used paradigms as well, in order to distinguish cloud computing from others.

Cluster Computing: The seminal work by Pfister [6] defines clusters of computer as "A cluster is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource".

Grid Computing: Buyya [7] defines Grid computing as "A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed 'autonomous' resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements".

By the above definitions we can consider cloud computing paradigm as combination of Cluster and Grid computing with some extra features of its own [5]. Whether it is the Cloud computing, Grid computing or Cluster computing, the critical element that enables these frameworks to work efficiently is the software and the ability to communicate between computational nodes as well as the strategy for managing the whole process. Apache Hadoop [8] is one of the most widely used distributed computing framework throughout the industry. It provides a complete framework to deal with both communication and management issues for distributed computing. Apache Hadoop is based on four modules, Hadoop Common (common utility to support Hadoop modules), Hadoop Distributed file systems (HDFS)[9], Hadoop Yarn (job scaling framework) and Hadoop MapReduce (for parallel precessing). A recent development in the field is, Apache spark [10]. Apache Spark not only provides high level API's (Application programming interface)to the widely used programming languages (Java, Python and R), but it also has an advantage over the traditional MapReduce as it is orginated from the idea of distributed shared memory. The resilient distributed data (RDD)[11] structure is a manifestation of the concept of shared memory, which provides an advantage to Apache Spark over Apache Hadoop MapReduce or HDFS (Hadoop distributed file systems) [12]. The discussion on which option is better and in which case, is beyond the scope of this article. Therefore without going into this discussion, we will use cloud computing and Apache spark implementation for computational purposes. The key reasons for this choice is, in the author's point view; cloud computing is the most viable option for most of the small and medium businesses size (SME's) as it can be more cost-effective than the fixed investment on acquiring computational capabilities and expertise.

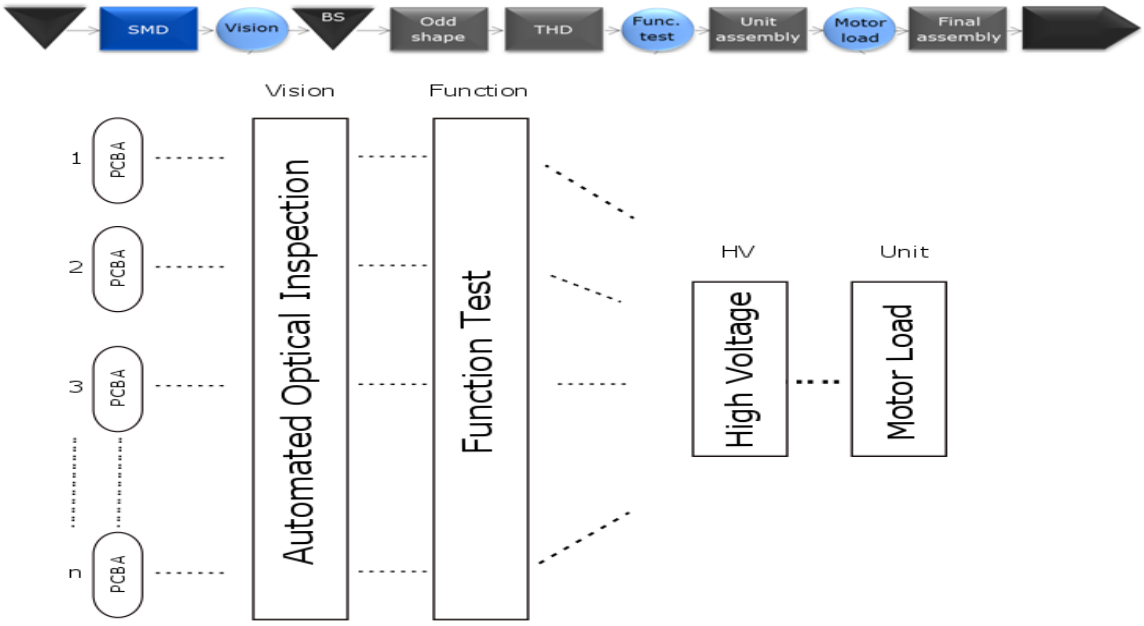
II. Advances in Categorical Data Analytics

Tree based methods (Classfication tree algorithms) are renowned for their ability to handle categorical data. The key advantage with these algorithms (CART, CHAID, C4.5, CRUISE, CTREE, etc) is their "easy to interpret" property at the expense of the relatively poor prediction accuracy of a single tree algorithm. Ensemble methods are mostly used to improve prediction accuracy of tree algorithms. In fact there are some studies claiming that on average the single tree method is 10% less accurate than the ensemble tree algorithms [13]. But the price of this improvement is low interpretability. On the other hand, other well known classification algorithms such as support vector machines (SVM)[14] and ANN (Artifical neural Networks) are originally designed for numeric (continuous) data and in principal both belongs to the so called category of black box methodologies implying low interpretability. Recently some studies attempted to fill this gap by extending exsisting methods such as nearest neighbour classification method [15] kernel based linear classification method [16] and Genetic Algorithm for categorical classfication [17]. This study is an extenssion of [17] with slightly modified fitness function. The aim in this study is to first present a cloud implementation of [17] and then consider the potential gains in terms of computational time by deploying parallel computing options and additional resources for computation.

II. Industrial Case Study

In this article the same multi-stage manufacturing process (electronics component manufacturing) with an extended data set is under consideration as presented in [17]. The extended data set is approximately 10 times bigger than the previous one, therefore the results will be different so as the computational challenges. As it is shown in Figure 1 the process under consideration is multi-stage. The core objective of this analysis is to predict the performance ("OK" or "NOK") of an item at the last stage (Unit) by using the information from previous stages (Vision, Function and HV). In this particular case it may potentially save considerable resources in terms of time and space, since the last (unit) test stage is significantly more expensive than the previous stages.[17].

Figure 1: *Manufacturing Process*



Test	Pass	Repair	Fail
AOI (Vision)	2213905	606315	0
FT (Function)	2059030	21813	40115
H.V (High Volt)	724244	2012	241
M.Load (Unit)	709287	21800	569

Table I: *Frequency counts w.r.t test results*

As shown in Table I the dataset contains information for over 6 million (6,358,975) parts. Preprocessing of this big data set is itself a challenge. A standard PC with 8GB RAM is not enough for basic data handling functions such as join, filter and subset due to the limited memory. To resolve this issue we used Apache spark [10]. More specifically the SparkR [18] library is used for the preprocessing of the data. Another important clarification about the preprocessing of the raw data is that it is done in house instead of cloud due to the confidentiality reasons. The data mining challenge is to predict bad performing units (rejected) from good performing (accepted) units at the last stage of an industrial manufacturing process by using the categorical outputs (OK or Not OK) from previous stages. Our data mining strategy is based on the comparison of performance probabilities and we use genetic algorithm to predict the bad performing units. The remainder of this article is divided into three parts, in the next section we will present the data mining strategy and the core algorithm for discovering the required knowledge. After the methodology section, we discuss the cloud implementation of the algorithm and knowledge gathered by the whole process in the results section. The last section is devoted to discussion on the results and gains by using cloud computing.

I. Data Mining Strategy

Conceptually, quality tests are designed with an aim to detect faults, i.e., each quality test divides population of objects (or units) into two groups "OK" and "Not OK". Traditionally this information is mainly used for control charts and construction of summary statistics. In [17] we presented another use of this categorical information i.e., classification of bad performing items. Assume a multi-stage process with "N" quality control stages as presented in Table I. Moreover, the status of each quality control stage is a categorical variable with two levels: OK and Not OK (NOK). This implies that there exists 2^N distinct combinations of the status of QC stages.

Table II: *Output of Quality control stages*

Combination	QC-1	QC-2	QC-3	. . .	QC-N
1	OK	OK	NOK	. . .	OK
2	NOK	OK	OK	. . .	NOK
3	NOK	NOK	OK	. . .	OK
. . .	. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .
2^N	NOK	OK	OK	. . .	OK

Table III: *Problem Formulation*

	Class		
Status	C1	C2	
OK	$X_{OK,C1}$	$X_{OK,C2}$	$X_{OK.}$
Not OK	$X_{NOK,C1}$	$X_{NOK,C2}$	$X_{\overline{OK}.}$
Total	$X_{.C1}$ (1.0)	$X_{.C2}$ (1.0)	$X_{..}$

The core idea is to partition these distinct combinations of QC test-status, into two classes (C1 and C2), with an objective that the conditional probability of "NOK" in one class (C1) is significantly smaller than the other class (C2). Let NOK_i denote a Not OK predicate of test stage i, then the data mining goal is to find the best values $C1$ and $C2$ such that

$$P(\text{NOK}_i|C1) << P(\text{NOK}_i|C2) \tag{1}$$

The objective formulated in (1) is both ambiguous and relies on the knowledge of unknown probabilities. Therefore we need to transform the objective (1) into an objective function of available statistics. Consider the contingency given in Table III for a generic choice of $(C1, C2)$

Table III comprises, for a generic choice of C_1 and C_2, total count $X..$, counts X_{OK_N} and X_{NOK_N}, marginal counts $X_{(OK_N,C_i)}$ and $X_{(NOK_N,C_i)}$. We choose to let our objective function Γ be measurable in (a function of) the statistics available in Table III. An immediate choice would be to let Γ depend only on Maximum Likelihood (ML) estimates of the conditional probabilities, i.e., $\hat{P}_{NOK_N|C_i} = \frac{X_{NOK_N,C_i}}{X_{.,C_i}}$ and $\Gamma = \hat{P}_{NOK_N|C_2} - \hat{P}_{NOK_N|C_1}$ However, this would give a high chance, that maximizing Γ would not necessarily fulfil objective (1), e.g. if either C_1 or C_2 hold very few items. Instead, we consider Neyman-Pearson (NP) confidence intervals $[L_i, U_i]$ for the ML estimates $\hat{P}_{NOK_N|C_i}$, i.e., we define

$$\Gamma = \frac{(L_2 - U_1)}{(L_1 + U_1) + (L_2 + U_2)} \tag{2}$$

With this definition we ensure, that maximizing Γ leads to statistically maximally separated estimates and in turn a high probability that objective (1) is fulfilled.

II. Genetic Algorithm

For each selection of C_1 and C_2 we collect the available statistics as in Table II and compute the objective function Γ. For fixed C_i and total counts we assume counts X_{OK_N,C_i}, and X_{NOK_N,C_i} to be conditionally binomially distributed given $X_{.,C_i}$. With this assumption we can compute exact NP confidence intervals $[L_i, U_i]$. The genetic algorithm follows the overall structure as given in [19], i.e.

1. Initialization or selection of population
2. Reproduction step
3. Crossover step
4. Mutation step

The corresponding procedure is given as

```
input  : X_{OK.}, X_{\overline{OK}.}
output: C_1, C_2
Initialize "M" and "m"; Size of population and
  chromosomes;
Select number of iterations iter = i ;
for iter ← 1 to value(i) do
    foreach element M Population do
        Calculate the Fitness score (Γ) and sort;
        Crossover between high ranked chromosomes;
        Randomly change the value of the string position;
    end
end
```

Algorithm 1: Genetic Algorithm

The genetic algorithm with Γ as fitness function does search through the space of $(2^{N-1})^{N-1}$ possible classifications. An immediate question at this point is, whether the problem is in nature truly combinatorial. That is, whether a heuristic optimization procedure is needed or there is a deterministic polynomial algorithm maximizing Γ over the space of possible classifications. We will consider this issue in our future research.

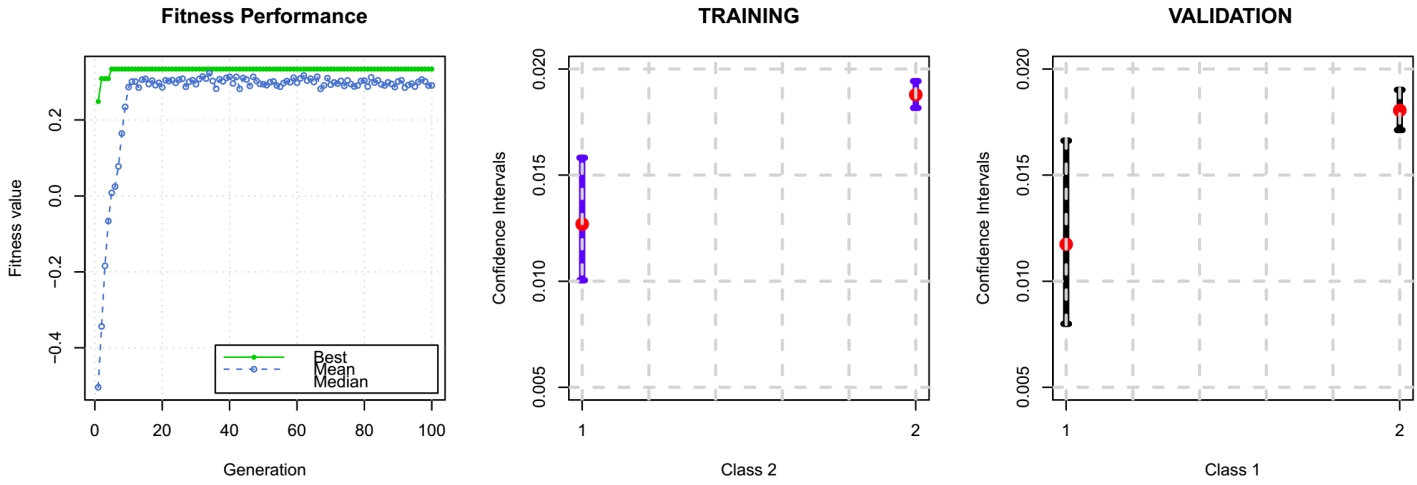
III. Results

The computationally intensive part of the proposed methodology is the genetic algorithm which therefore is implemented on cloud. The product under consideration is a combination of two PCBA's (Printed Circuit Board). This implies that, before the final stage we have 2-readings of vision (AOI), 2-readings of Function Test (FT) and one reading of High voltage test (HV) (maximum of $5^5 = 3125$ distinct combinations with "OK" or "NOK" in previous five stages). Hence, the classification problem involves the 2^{3125} possibilities.

I. Data Mining Algorithm Results

Throughout the analysis we fixed the crossover probability and mutation chance to 0.8 and 0.2 respectively. Secondly linear rank selection criterion is used to select the best individuals from the population. The detailed discussion on different selection criteria and other parameters can

Figure 2: *Classification of the real data through Genetic Algorithm (95% confidence intervals)*



be found in [20]. In the current dataset only 36 distinct combinations of the total 3125 are found. A contingency table for optimal C1 (9 out of 36 combinations) and C2 (27 out of 36 combinations) values for the training data is given in Table IV where ML estimates of class conditional probabilities are given in parentheses. From the results presented in Figure 2, we observe the statistically significant difference between the conditional probabilities of "NOK" at the final stage given categories C1 and C2. Moreover, consistency between training and validation tests (Figure 2 and Table V) demonstrates validity of our proposed methodology to solve a classification problem by constructing two (statistically) distinct classes. But the use of evolutionary principle to solve this problem also points towards a rather demanding issue of scalability and computational efficiency in big data analysis. Remainder of this article sheds some light on this issue by discussing computation intensive element of our methodology, namely genetic algorithm.

Typically, convergence to the global solution in genetic algorithm depends upon population size, mutation ratio and crossover probability. Increase of population size has direct impact on computation time and resources, therefore variable population sizes (with fix mutation and crossover probabilities) can be used to examine performance of our proposed methodology to solve computationally heavy classification tasks.

	Class		
Status	C1	C2	
OK	179045	2675	181720
	(1- 0.01844)	(1-0.02833)	
NOK	3364	78	3442
	(0.01844)	(0.02833)	
Total	182409	2753	185162
	(1.0)	(1.0)	

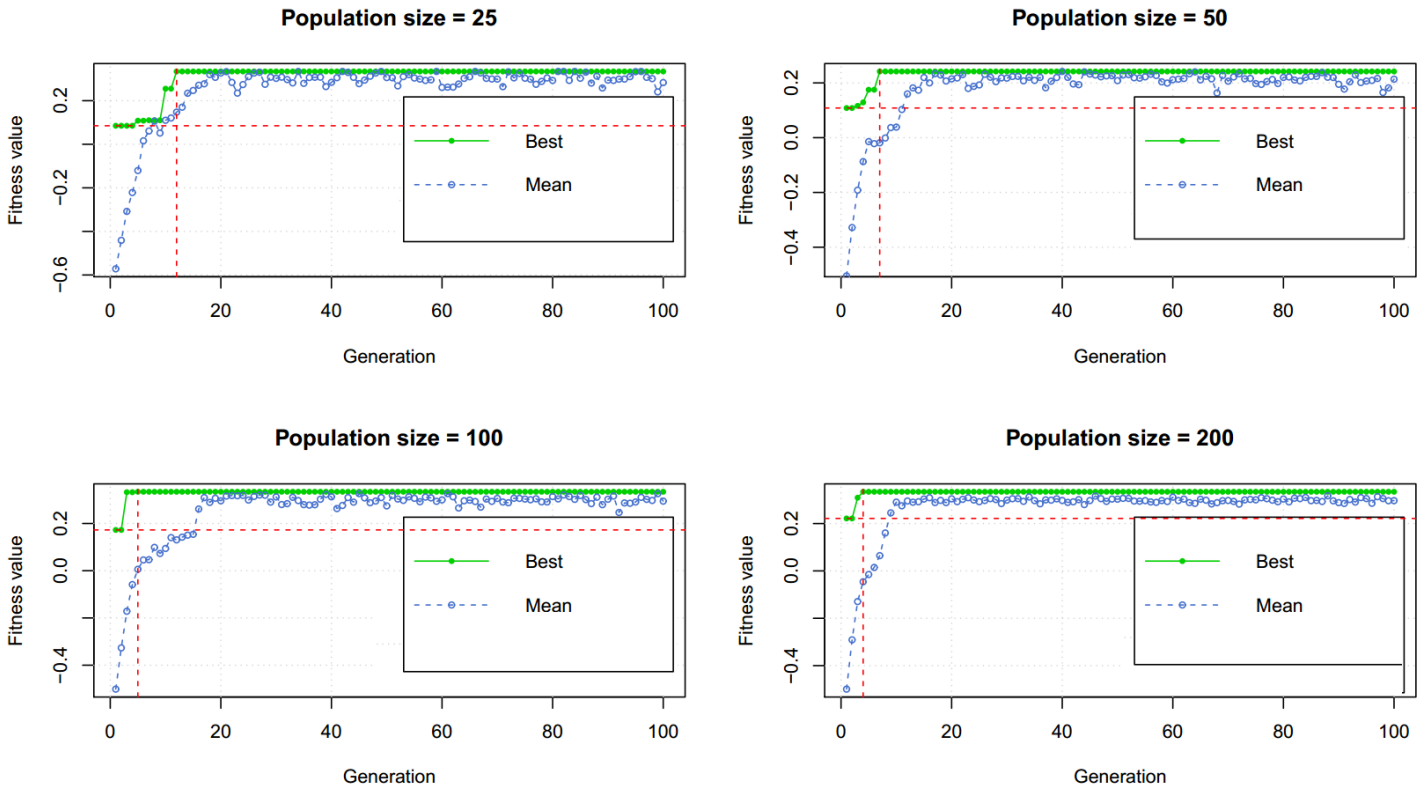
Table IV: *Contingency table for optimal choice of categories C1 and C2 (Training Data)*

	Class		
Status	C1	C2	
OK	76771	1544	78315
	(1- 0.01769)	(1-0.02525)	
NOK	1383	40	1423
	(0.01769)	(0.02525)	
Total	78154	1584	79738
	(1.0)	(1.0)	

Table V: *Contingency table for optimal choice of categories C1 and C2 (Testing/Validation Data)*

As a first step we will present a relationship between population size and algorithm convergence (Figure 3) and then in the next section we will discuss other computational aspects of our methodology namely cloud implementation. For all four population sizes (25, 50, 100 and 200) presented in Figure 3 algorithm converges to the best solution rather quickly. Even in this short span (initialization to convergence), careful examination of all four plots suggests that increase in population size implies early convergence as both length and width of the top left red window is decreasing with increasing population size. Secondly we

Figure 3: *Comparison of Different Population Sizes and Convergence*



obtain more stable results by increasing the population size as indicated with the blue mean curve in the plots.

II. Cloud implementation

As mentioned in the previous section, the genetic algorithm is implemented on cloud. More specifically, Amazon web service for cloud computing (AWS EC2) with "m4.4xlarge" instance is used for computational purposes. In order to maximize the utilization of the computing resources in m4.4xlarge instance we have implemented master-slave parallelization strategy for genetic algorithm (i.e., master saves the population and distributes to slaves for fitness computation). Ideally, the potential gain by adding additional resources should be directly proportional to the number of resources added to solve the computational challenge, but this is not completely true. In computer science Amdahl's law [21] provides a quantification and explanation of this phenomenon. By Amdahl's law, theoretical speed up (S) depends on the proportion of parallel part of the program (p) and number of additional resources to solve a computational challenge (n):

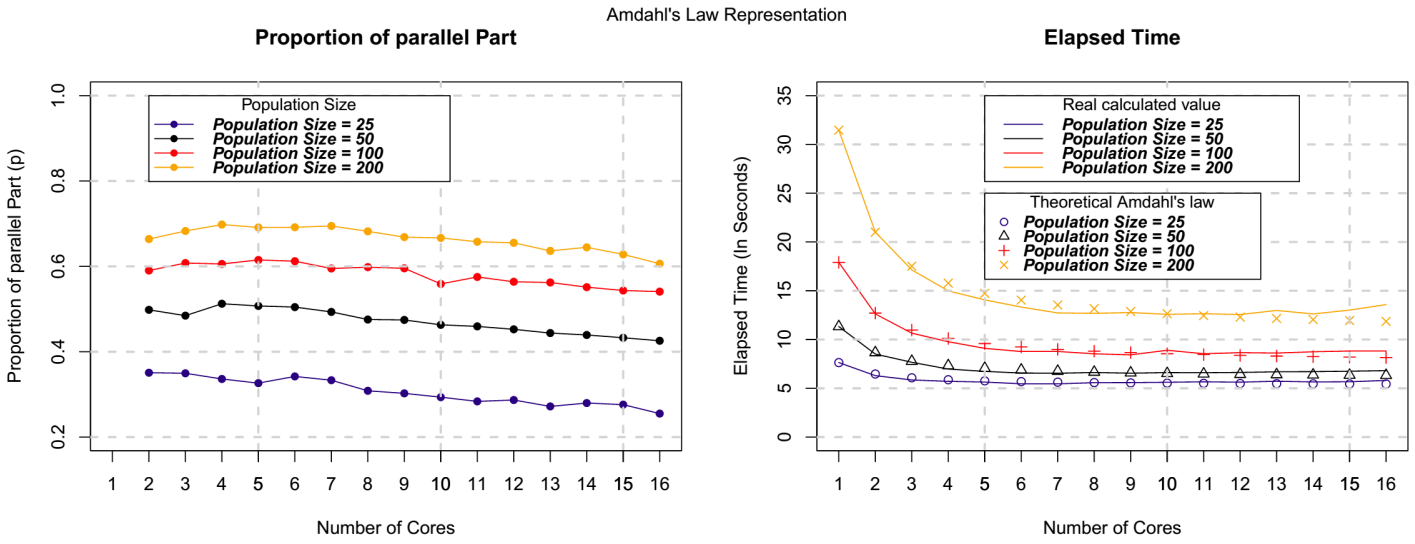
$$S = \frac{1}{(1-p) + \frac{p}{n}} \tag{3}$$

In Figure 4 we present Amdahl's law representation of Elapsed Time (Total time to complete the computational job) along with quantities S and p. From (3), the quantity p can be calculated as

$$p = \frac{n(1-s)}{s(1-n)} \tag{4}$$

In reflection to the results presented in the Figure 4, we can conclude that in almost all four cases marginal gain in speed up (or reduction of elapsed time) is approaching to zero or completely diminishing after increasing number of cores. In other words computational speed up cannot only be gained by simply adding more resources. In fact in all four cases 5 cores are enough to get maximum speed up. Relatively constant nature of time curves (for Speed-up and elapsed time) by adding more cores (after 5-cores) points out the superfluous use of the resources. Another important aspect of this plot is varying p values for the all four population sizes as presented in Figure 4. It is observed that large population size leads to a large value of p. More specifically, parallelization or adding additional resources are more suitable in the case of big population size. Therefore it is worthwhile to implement parallel computing only in case

Figure 4: *Comparison between parallel and non-parallel implementations*



of relatively big and complex computational challenges.

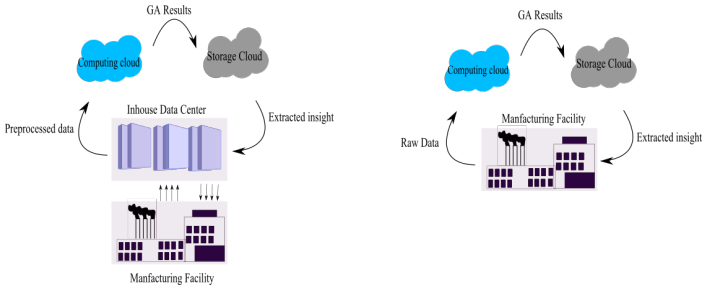
IV. Discussion

In this article we consider both the data mining methodology and the computational resources needed to analyse large amounts of manufacturing data. For the former, we show that our proposed methodology works well to solve the categorical and combinatorial classification problem. For the latter, we present the cloud implementation of the proposed data mining methodology.

The proposed data mining strategy is based on Genetic Algorithm which is iterative in nature and well suited for distributed computing. By implementing the parallel computing strategy, we optimized the current computational resources. Moreover this implementation made the code scalable. That is, by adding additional hardware resources we managed to reduce computation time. Achieving scalability is considered a key programming objective in the field of big data analysis.

In the end, two different strategies for cloud computing in manufacturing are presented in Figure 5. Our proposed approach at the moment depicted on the left hand side of Figure 5, where traditional data center with the process data is connected to cloud computing options for the analysis. The idea of replacing the traditional data centres with clouds has real potential as it is presented on the right side of Figure 5. Especially, this computing model is quite suitable for small industries where accumulating resources and expertise for complex data centres will always be a challenge. Moreover, on-line information processing or real

Figure 5: *Vision for the future factory and real time information processing*



time data analysis is indeed a crucial concern in a smart factory. Yet it also involves other issues such as data security which is beyond the scope of this article.

References

[1] K. Schwab, "The fourth industrial revolution," in *Geneva: World Economic Forum*, 2016.

[2] C. Freeman and F. Louçã, *As time goes by: from the industrial revolutions to the information revolution*. Oxford University Press, 2001.

[3] M. Castells, "The information age, volumes 1–3: Economy, society and culture," 1999.

[4] E. Brynjolfsson and A. McAfee, *Race against the machine: How the digital revolution is accelerating innovation, driving productivity, and irreversibly transforming employment and the economy*. Brynjolfsson and McAfee, 2012.

[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[6] G. F. Pfister, *In search of clusters*. Prentice-Hall, Inc., 1998.

[7] R. Buyya, "High performance cluster computing," *New Jersey: FâĂŹrentice*, 1999.

[8] M. Bhandarkar, "Mapreduce programming with apache hadoop," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pp. 1–1, IEEE, 2010.
[9] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pp. 1–10, IEEE, 2010.
[10] Spark, "http://spark.apache.org/docs/latest/," 2014.
[11] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 2–2, USENIX Association, 2012.
[12] S. Gopalani and R. Arora, "Comparing apache spark and map reduce with performance analysis using k-means," *International Journal of Computer Applications*, vol. 113, no. 1, 2015.
[13] W.-Y. Loh, "Fifty years of classification and regression trees," *International Statistical Review*, vol. 82, no. 3, pp. 329–348, 2014.
[14] N. Cristianini and B. Scholkopf, "Support vector machines and kernel methods: the new generation of learning machines," *Ai Magazine*, vol. 23, no. 3, p. 31, 2002.
[15] L. Chen and G. Guo, "Nearest neighbor classification of categorical data by attributes weighting," *Expert Systems with Applications*, vol. 42, no. 6, pp. 3142–3149, 2015.
[16] L. Chen, Y. Ye, G. Guo, and J. Zhu, "Kernel-based linear classification on categorical data," *Soft Computing*, pp. 1–13, 2015.
[17] A. R. Khan, H. Schioler, T. Knudsen, and M. Kulahci, "Statistical data mining for efficient quality control in manufacturing," in *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, pp. 1–4, IEEE, 2015.
[18] SparkR, "https://spark.apache.org/docs/latest/sparkr.html," 2016.
[19] D. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion wesley*, vol. 1989, 1989.
[20] R. Sivaraj and T. Ravichandran, "A review of selection methods in genetic algorithm," *International journal of engineering science and technology*, vol. 1, no. 3, pp. 3792–3797, 2011.
[21] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, pp. 483–485, ACM, 1967.