Árpád GELLÉRT

Prediction-Based Modeling and Estimation in Advanced Computing Systems

HABILITATION THESIS

in Computers and Information Technology

2023

Table of Contents

Summary

This habilitation thesis is a synopsis of the most important research achievements after the PhD thesis in prediction-based modeling and estimation in topics like computer architecture, smart factories, smart buildings, image processing and web mining. All the presented methods have in common a large exploitation of advanced prediction techniques and forecasting models.

In computer architecture, different value prediction schemes have been developed and evaluated with the goal of increasing instruction-level parallelism and the overall processing performance and decreasing at the same time the energy consumption of superscalar, multithreaded and multicore microprocessors. Value prediction is a speculative technique, which anticipates the results of high-latency instructions and unlocks subsequent dependent instructions by speculatively executing them earlier. The prediction must be accurate, since any misprediction is treated by a recovery of the correct processor context, which consumes additional cycles. This work brings original contributions in developing and evaluating advanced value prediction methods applied selectively on critical Load instructions. We present counter-based and perceptron-based prediction schemes integrated into simulators like M-SIM or Sniper. The high number of parameters of the simulated microprocessors generates a huge simulation complexity in the design space exploration process. Therefore, the multi-objective optimization is realized by heuristic search through genetic algorithms.

My most recent research topic is aiming to improve the smart factory ecosystem with contributions in predicting and modeling assembly processes. The main goal is the integration of a prediction module into assembly assistance systems able to support factory workers in their manufacturing activities by providing choices for the next assembly step. First, a two-level contextual predictor was used to suggest the next assembly steps which consists in a state register (the first level) which indexes a table storing pairs of state-patterns and their associated next states (the second level). Another variant of the two-level contextual predictor extended each assembly state with an automaton which could be in stable or in unstable substate. Unfortunately, this scheme provided insignificant improvement, and because it used supplementary information and additional steps in the prediction process, it was considered less efficient than the scheme without automata. The Markov model is another context-based two-level predictor which can store multiple next states, together with their number of occurrences for each pattern. The length of the context determines the order of the model. The state with the highest number of occurrences is extracted from the prediction table entry selected with the left-shift state register and is then provided as the predicted one. Such a predictor can provide multiple next assembly choices with different probabilities, but for time-critical decisions it can be configured to return the most probable state. The prediction by partial matching algorithm was also evaluated as an assembly step predictor. It combines different order Markov predictors. The model of order R first tries to predict with the Markov chain of order R. If the Markov chain can predict, its prediction is returned. Otherwise, if the current Markov chain cannot predict, the order is decremented until a prediction can be done or all the models were evaluated without success and, in that case, no prediction can be provided. Because the current context cannot always be found in the prediction table, an enhanced prediction scheme was considered, which explores the neighboring characteristics of the user if the actual characteristics do not have an exact match. Neighboring context exploration involves changing one characteristic of the worker at a time followed by prediction trial. The step that was predicted the most from the neighboring states will be considered the next assembly step.

In the smart buildings research area, a main contribution consists in smart energy management systems designed for households equipped with photovoltaics and energy storage systems making automated decisions based on forecasted electricity production and

consumption. At any time, the system can combine the own produced electricity with the stored electricity and with electricity from the grid. The interest is to adjust and synchronize through prediction the electricity consumption and production increasing self-consumption and reducing the intake from the power grid. Thus, the total annual operating cost is lower and, as additional benefit, the losses in the distribution networks are reduced. Based on the predictions, the energy management system may decide to activate some household appliances when cheap electricity is available and to delay their activation when only high-cost electricity is available. Markov chains, stride predictors, a Long Short-Term Memory and hybrid models were developed and evaluated on the available datasets.

In the image processing topic, a context-based denoising method was developed for grayscale images affected by impulse noise. The proposed algorithm is using Markov chains to replace the detected noise with the intensity having the highest number of occurrences in similar contexts. The context of a noisy pixel consists in its neighbor pixels and is searched in a larger but limited surrounding area. We have analyzed different search methods and different context shapes. The experimental results obtained on the test images have shown that the most efficient model applies the search in form of “*” of contexts having the form of “+”. Beside the better denoising performance obtained on all the noise levels, the computational time has been also significantly improved with respect to the context-based filter which applies full search of full context. We have also compared this Markov filter with other denoising techniques existing in the literature, most of them being significantly outperformed. Another original contribution is a context-based inpainting method which is using Markov chains to repair pixel colors from images affected by external factors (defects) or to replace pixel colors belonging to image areas covered by objects or texts. First, the user must select the target area and then the developed inpainting algorithm is replacing each pixel intensity from the target area based on the surrounding unaffected context information. The restoration process is applied from the exterior to the interior within the selected target area. For the replacement of a certain pixel intensity, we explored a limited surrounding image area to identify the intensity occurring with the highest probability in similar contexts. Since we use context information, the proposed inpainting technique can very well rebuild the image details.

In the web mining research area, several prediction-based web-prefetchers have been studied. The prediction by partial matching algorithm was evaluated, as well as a dynamic decision tree with different order Markov predictors as components. The predicted web object is prefetched into the cache to make it available for possible next accesses. The experiments performed on a dataset from the Boston University show that the optimal method is the dynamic decision tree which uses as features the current link, the link type and the predictions provided by the Markov chains of orders 1-4. This optimal predictor outperformed all the Markov models applied separately, but also the prediction by partial matching.

Rezumat

Această teză de abilitare este o sinteză a celor mai importante realizări de după teza de doctorat în modelare și estimare bazată pe predicție în domenii precum arhitecturi de calcul, fabrici inteligente, clădiri inteligente, procesarea imaginilor și web mining. Toate metodele prezentate au în comun o largă exploatare a tehnicilor avansate de predicție și a modelelor de prognoză.

În domeniul arhitecturilor de calcul, s-au dezvoltat și evaluat diferite scheme de predicție a valorilor având ca scop creșterea paralelismului la nivelul instrucțiunilor și a performanței de procesare și în același timp scăderea consumului de energie în cazul microprocesoarelor superscalare, multifir sau multicore. Predicția valorilor este o tehnică speculativă aplicată pentru anticiparea rezultatelor instrucțiunilor cu latență de execuție ridicată și deblocarea instrucțiunilor dependente prin execuție speculativă. Predicția trebuie realizată cu acuratețe pentru că fiecare predicție greșită trebuie tratată printr-un mecanism de recuperare a contextului corect, ceea ce introduce timp de execuție suplimentar. Această lucrare aduce contribuții originale în dezvoltarea și evaluarea unor metode avansate de predicție a valorilor aplicate selectiv pe instrucțiuni Load critice. Sunt prezentate scheme de predicție bazate pe contoare respectiv perceptroni integrate în simulatoare ca M-SIM sau Sniper. Numărul mare al parametrilor microprocesoarelor simulate generează o complexitate de simulare ridicată în procesul de explorare a spațiului de proiectare. De aceea, optimizarea multi-obiectiv se realizează aplicând căutarea euristică prin algoritmi genetici.

Tema de cercetare în care m-am implicat cel mai recent propune îmbunătățirea ecosistemului de fabrică inteligentă prin contribuții în predicția și modelarea proceselor de asamblare a produselor. Scopul principal este integrarea unui modul de predicție în stații de asamblare capabile să ghideze muncitorii oferind opțiuni cu privire la următorul pas de asamblare. Ca o primă încercare, pentru furnizarea următorului pas de asamblare s-a folosit un predictor contextual pe două nivele compus dintr-un registru de stare (primul nivel) care indexează o tabelă ce stochează perechi de context împreună cu starea următoare asociată (al doilea nivel). O altă variantă a predictorului contextual pe două nivele extinde fiecare stare de asamblare cu un automat care poate fi în substare stabilă sau instabilă. Din păcate, această schemă a adus îmbunătățiri nesemnificative și, având în vedere că folosește informații suplimentare și pași adiționali în procesul de predicție, s-a dovedit mai puțin eficientă decât schema inițială fără automat. Modelul Markov este un alt predictor contextual pe două nivele care poate stoca stări multiple împreună cu frecvențele lor de apariție pentru fiecare context. Dimensiunea contextului folosit stabilește ordinul modelului. Starea cu frecvența de apariție cea mai ridicată se extrage din intrarea tabelei de predicție selectată cu registrul de stare și este furnizată apoi ca stare predicționată. Un astfel de predictor poate furniza mai multe opțiuni cu probabilități diferite pentru următorul pas de asamblare, dar pentru sistemele în timp real se poate configura să returneze starea cea mai probabilă. A urmat evaluarea algoritmului de predicție bazat pe potrivire parțială pentru furnizarea următorului pas de asamblare. Acesta combină mai multe modele Markov de ordin diferit. Prima încercare de predicție a modelului de ordin R se realizează cu lanțul Markov de ordin R. Dacă acesta poate predicționa, i se returnează predicția. Altfel, dacă lanțul Markov curent nu poate predicționa, ordinul este decrementat până când predicția este realizabilă sau niciun model n-a reușit să predicționeze, caz în care procesul se încheie fără predicție. Deoarece contextul curent nu poate fi întotdeauna găsit în tabela de predicție, am introdus o schemă de predicție îmbunătățită, care explorează caracteristicile vecine ale utilizatorului dacă caracteristicile reale nu se potrivesc exact. Explorarea contextelor vecine implică schimbarea pe rând a câte unei caracteristici a muncitorului urmată de încercarea

generării predicției. Starea predicționată majoritar din contexte vecine va fi considerată predicția finală.

În domeniul clădirilor inteligente, principala contribuție constă într-un sistem inteligent de gestiune a energiei electrice destinat caselor echipate cu panouri fotovoltaice și sisteme de stocare a energiei care ia automat decizii pe baza estimării producției și a consumului de energie electrică. În orice moment sistemul poate combina după necesități energia electrică produsă local cu energia electrică stocată respectiv cu energie din rețeaua electrică. Rolul sistemului de gestiune este ajustarea și sincronizarea consumului și a producției de energie electrică, crescând raportul de autoconsum și reducând presiunea pe rețeaua electrică. Astfel, scad costurile anuale și, ca beneficiu suplimentar, scad și pierderile din rețelele de distribuție. Pe baza predicțiilor, sistemul de gestiune a energiei electrice poate activa electrocasnice când este disponibilă electricitate ieftină și poate întârzia activarea lor când e disponibilă doar electricitate scumpă. Au fost dezvoltate modele Markov, predictoare incrementale, rețele neuronale recurente de tip LSTM și modele hibride și toate au fost evaluate pe seturile de date disponibile.

În ceea ce privește procesarea imaginilor, am dezvoltat o metodă contextuală de eliminare a zgomotului de tip impuls din imaginile în nivele de gri afectate. Algoritmul propus folosește lanțuri Markov pentru înlocuirea zgomotului detectat cu intensitatea care a apărut cel mai frecvent în contexte similare. Contextul unui pixel afectat de zgomot constă în intensitățile pixelilor din strânsa vecinătate și este căutat într-o vecinătate mai largă dar limitată. Am analizat diferite metode de căutare și diferite forme de context. Rezultatele experimentale obținute pe imagini de test au arătat că modelul cel mai eficient aplică o căutare în formă de „*” a contextelor în formă de „+”. Pe lângă performanța de filtrare îmbunătățită pe toate nivelele de zgomot, s-a redus substanțial și durata de procesare față de modelul de filtrare contextuală cu căutare completă a contextelor de formă pătratică. Am comparat acest filtru Markov cu alte tehnici de filtrare existente în literatura de specialitate, majoritatea lor fiind net depășite. O altă contribuție originală constă în metode de reconstrucție contextuală a zonelor de imagine afectate de factori externi (defecte) sau acoperite de obiecte sau text. Într-o primă fază, utilizatorul trebuie să selecteze zona dorită, iar apoi algoritmul dezvoltat înlocuiește intensitatea fiecărui pixel din zona marcată pe baza informațiilor contextuale neafectate din jur. Procesul de restaurare se aplică din exterior spre interior în cadrul zonei selectate. Pentru înlocuirea intensității unui anumit pixel, se explorează o zonă limitată din jur în vederea identificării intensității care apare cel mai frecvent în contexte similare. Prin folosirea informațiilor contextuale, tehnica propusă poate reconstrui foarte bine detaliile din imagini.

În domeniul web mining, am studiat diverse metode de preîncărcare a obiectelor web pe bază de predicție. S-a evaluat algoritmul de predicție bazat pe potrivire parțială, precum și arbori de decizie cu diferite componente Markov. Obiectul web predicționat se preîncarcă în cache pentru a-l face disponibil în caz de accesare. Experimentele efectuate pe setul de date colectat de Universitatea din Boston arată că metoda optimă este cea bazată pe arbore de decizie care folosește ca trăsături link-ul curent, tipul link-ului și predicțiile generate de lanțurile Markov de ordin 1-4. Această metodă optimă de predicție a depășit toate modelele Markov aplicate individual, dar și algoritmul de predicție bazat pe potrivire parțială.

I. ACHIEVEMENTS

1. Introduction

This habilitation thesis provides a summary of my research related to prediction methods in different applications developed since the defense of my PhD thesis in November 2008. This chapter presents the context and motivation of my works, summarizes the scientific contributions as well as their scientific and didactic impact.

1.1. Context and Motivation

The general prediction mechanism consists in anticipating future contexts based on current and previous context information. Prediction can be applied in computing systems if it brings a certain gain in performance, energy consumption, or resource management in general, through the availability of some data in advance. Prediction can be also used in modeling different processes or signals.

In some of the computing systems, like microprocessors, inexactness is not accepted and, therefore, if the speculation fails, the correct context might be recovered. Such a correct state recovery has its timing costs. Obviously, the misprediction rate can be kept on low levels by using the most accurate predictor. In other computing systems, like assembly assistance systems, smart building applications, image restoration applications, web prefetching systems, etc., inexactness is acceptable and, thus, a recovery is not necessary in the case of failed speculation, but for the highest benefits the most accurate predictor is preferred.

In both situations, the predictor might be updated run-time after mispredictions in order to dynamically improve the prediction accuracy. The quality of a prediction model is highly dependent on the quality of the available data. Especially the choice of the features to base the prediction on is important. This habilitation thesis presents several prediction methods adapted and configured for different applications.

1.2. Research Roadmap

Most of my research activities were focused on analyzing, developing and adapting prediction methods for different applications. Thus, all the researches presented in this habilitation thesis have in common the prediction.

As an extension of the research from my PhD thesis, chapter 2 presents different hardware prediction schemes applied in microprocessors. Power became a very important design constraint in nowadays processors and a lot of research targets toward analysis and optimization of power consumption. In this context, we analyzed the efficiency of selectively anticipating the results of some long-latency instructions within superscalar and Simultaneous Multithreaded (SMT) architectures. The goal is to prove that our selective value prediction technique beside the IPC improvement also reduces energy consumption. We show that the performance lost by reducing L1 cache capacity can be covered by our selective load value prediction technique. The experimental results, performed on the SPEC 2000 benchmarks, show that reducing the L1 data cache space by quartering its size and using a selective value prediction produces a significant IPC speedup and a considerable energy reduction against the baseline microarchitecture. Next, we have included into our analysis the L2 unified cache. The considered parameters increased too much the design space and, therefore, we performed an automatic design space exploration using a special developed software tool by varying several architectural parameters through heuristic search. The experimental results show that our automatic design space exploration (DSE) provides significantly better configurations than our previous manual DSE approach, considering the proposed multi-objective approach to find the optimal configurations in terms of CPI (Cycles per Instruction) and energy consumption.

The results on engineering a context-aware assistive system for manual assembly tasks are presented in chapter 3. The assembly station employs context-based predictors to suggest the next steps during the manufacturing process and is based on data collected from experiments with trainees in assembling a tablet. The predictors are trained with correct assembly styles extracted from the collected data and assessed against the whole dataset. Thus, we found the predictor that best matches the assembly preferences. We analyzed two-level context-based predictors, Markov predictors and the Prediction by Partial Matching (PPM) algorithm in terms of prediction rate, coverage and prediction accuracy. We have also improved the Markov predictor with a padding mechanism which improved the coverage and, thus, there is a significantly higher number of assembly steps which are correctly correlated with the real intentions of the workers.

Humans typically act in a certain habitual pattern, which can be exploited in different anticipative systems within a smart building. Chapter 4 presents different prediction techniques applied in intelligent energy management systems, as components of smart buildings. Ubiquitous systems use context information to adapt appliance behavior to human needs. Even more convenience is reached if the appliance foresees the user’s desires and acts proactively. The electrical power sector must undergo a thorough metamorphosis to achieve the ambitious targets in greenhouse gas reduction set forth in the Paris Agreement of 2015. Reducing uncertainty about demand and, in case of renewable electricity generation, supply is important for the determination of spot electricity prices. In this context, we evaluate a context-based technique to anticipate the electricity production and consumption in buildings. We focus on a household with photovoltaics and energy storage system. We analyze the efficiency of Markov chains, stride predictors and also their combination into a hybrid predictor in modelling the evolution of electricity production and consumption. All these methods anticipate electric power based on previous values. The main goal is to determine the best method and its optimal configuration which can be integrated into an intelligent energy management system. The role of such a system is to adjust and synchronize through prediction the electricity consumption and production in order to increase self-consumption, reducing thus the pressure over the power grid.

In chapter 5, the focus is on two image restoration techniques: impulse noise filtering and inpainting. Digital images are often degraded by impulse noise during acquisition from sensors or transmission through a faulty communication channel. Salt-and-pepper is a typical impulse noise composed of minimum and maximum valued pixels within the affected image. The main objective of salt-and-pepper denoising methods is the preservation of unaffected pixels while restoring the missing information. We introduced a new image denoising method relying on prediction through Markov chains. The algorithm replaces the noisy pixel’s intensity with the predicted one: the value occurring with the highest frequency in the same context as the replaceable pixel’s intensity. We have analyzed different search methods and different context shapes. Since it is a context-based technique, it preserves the details in the filtered images better than other methods. After the success of the Markov chains in image denoising, we analyzed the possibility to apply them for image inpainting, thus to restore the colors of objects from images affected by some external factors (like scratches or wipes) or partially covered by other objects. Damages or unwanted objects can be removed from an image by replacing the intensity of each pixel from such an area, based on the surrounding unaffected context information. The restoration process is applied from the exterior to the interior, using for replacement colors occurring with the highest probability in similar contexts. Since we use context information, the proposed inpainting technique can successfully rebuild details in images.

Different prediction-based web prefetching techniques are analyzed in chapter 6. Prefetching anticipates the next accesses and loads the corresponding web objects into the cache. If the user accesses a web object which is available in the cache, the browser can load it without any delays. Prediction-based prefetching can be very effective in long browsing sessions by minimizing the access latencies. One of the evaluated method is PPM. Since the number of visited web pages can be high, tree-based and table-based implementations can be inefficient from the representation point of view. Therefore, we present an efficient way to implement PPM as simple searches in the observation sequence. Thus, we can use high number of states in long web page access histories and higher order Markov chains at low complexity. The time-evaluations show that the proposed PPM implementation is significantly more efficient than previous implementations. We have also enhanced the predictor with a confidence mechanism, implemented as saturating counters, which classifies dynamically web pages as predictable or unpredictable. Predictions are generated selectively only from web pages classified as predictable, improving thus the accuracy. The other evaluated method is a hybrid one consisting in a dynamic decision tree and different order Markov predictors as components. The predictions generated by the Markov chain components are used as features within the dynamic decision tree. We use a decision tree to select the most predictive features from the considered feature set and based on those selected features we generate predictions. In our application the feature set includes the current link, the type of the current link as well as the predictions of different order Markov chains.

Finally, chapter 7 presents the main further work plans. I will continue the research in all the five directions: anticipative techniques in multicore architectures (ideas defined in [179] and [16]), adaptive assembly assistance systems, smart energy management systems, context-based image restoration and prediction-based hybrid web prefetching.

1.3. Impact of Contributions

Starting from my teaching assistant position at Lucian Blaga University of Sibiu in 2003, I had a sustained research activity by publishing 7 books, 40 papers in conference proceedings and journals indexed in ISI Web of Knowledge and 29 papers in conference proceedings and journals indexed in other international databases.

My main research topic, derived from my PhD thesis and still active today, was in the computer architecture domain, the research activities being deployed in prof. Lucian Vintan’s research center. The experience and the knowledge accumulated from that period allowed me to diversify my research areas and interests, covering today also topics like smart factories, smart buildings, image processing and web mining, with significant results in publications indexed ISI Web of Knowledge. I was member in the research teams of 8 research projects. I was manager or responsible from Lucian Blaga University of Sibiu of other 8 research projects. In 2010 I received the “AD AUGUSTA PER ANGUSTA” prize awarded by Lucian Blaga University of Sibiu for excellence in scientific research. The recognition of my publications is reflected by more than 600 citations and a Hirsch index of 13 in Google Scholar. The Hirsch index is 8 in Scopus and 7 in ISI Web of Science. As another recognition of my research activity, I was invited to be part of the program committee of international conferences, to serve as scientific reviewer for international journals and also to be member of the editorial board of Journal of Digital Information Management, Journal of E-Technology, International Journal of Advanced Statistics and IT&C for Economics and Life Sciences, and Future Internet (special issue: Computer Vision, Deep Learning and Machine Learning with applications). I was also member of the organizing committee of international conferences and workshops. Currently, I am an active member of the Advanced Computer Architecture and Processing Systems (ACAPS) research center from Lucian Blaga University of Sibiu and an affiliate member of the European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC).

Most of the work was possible due to my research mobilities at Barcelona Supercomputing Center in 2006 and at Polytechnic University of Milan in 2009, but especially due to the very fruitful collaborations with prof. Theo Ungerer (University of Augsburg), prof. Colin Egan (University of Hertfordshire), prof. Cristina Silvano (Polytechic University of Milan), prof. Ugo Fiore (Parthenope University), prof. Francesco Palmieri (University of Salerno), my colleagues prof. Lucian Vintan, prof. Adrian Florea, prof. Remus Brad and prof. Constantin-Bala Zamfirescu from the Computer Science and Electrical Engineering Department and my students who provided so many times their technical support. The results presented in this habilitation thesis were obtained after my PhD thesis, mostly as associate professor (since 2015).

2. Speculative Computer Architectures

A load value predictor is a hardware architectural enhancement which speculates over the results of load instructions to speed-up the execution of the subsequent instructions. Lipasti first proposed the Load Value Prediction (LVP) concept and, particularly, he developed a non-selective LVP in 1996 but we have not used his LVP structure due to its huge complexity. Our proposed architectural enhancement, the Selective Load Value Prediction (SLVP), differs from a classic value predictor due to an improved selection scheme that allows activating the predictor only when a miss occurs in the first level of cache. We use a simple direct-mapped table which requires less additional hardware enabling reduced energy consumption than traditional approaches (Lipasti's first proposal).

Taking into account that low-power computing became an important design objective for mobile, battery-operated devices and even in high-performance microprocessors, the goal of this chapter is to explore several configurations of architectures with selective load value prediction and to determine the most efficient one in terms of Instruction Per Cycle (IPC) and energy consumption. A very important question is: would the selective load value prediction mechanism allow us to decrease the data cache size and implicitly the energy consumption, without loosing performance? We'll answer this question by performing a design space exploration regarding the size of the L1 data cache in order to find the optimal configuration, which keeps high performance at low energy consumption. We also compare different superscalar and simultaneous multithreaded (SMT) architectures (enhanced with our selective load value predictor, Lipasti's non-selective load value predictor, Jouppi's victim cache, etc.) and show that our anticipative technique outperforms the others.

2.1. Superscalar and SMT Microarchitectures with Load Value Predictor

In this section we present a manual design space exploration of superscalar and SMT architectures with SLVP, published in [68]. In [67], we have analyzed the efficiency of selectively anticipating the results of some long-latency instructions within superscalar and SMT architectures. Particularly we have focused on multiply (MUL), Division (DIV) and critical Load instructions (with miss in L1 data cache [58]). We integrated into the M-SIM simulator [185] a Dynamic Instruction Reuse (DIR) scheme for the MUL/DIV instructions and a LVP for the critical Load instructions. Our improved superscalar architecture achieved an average IPC speedup of 3.5% on the integer SPEC 2000 benchmarks, of 23.6% on the floating-point benchmarks, and an improvement in energy-delay product of 6.2% and 34.5%, respectively. Our evaluations have also shown higher IPC and lower relative energy consumption (energy-delay product) on all the evaluated SMT configurations (1, 2, 3 and 6 threads).

Since our previous results show that most of the IPC speedup was generated by the load value predictor we further focalize only on this speculative technique. In [68], we performed a manual design space exploration regarding the size of the L1 data cache in order to find the optimal configuration, which keeps high performance at low energy consumption. We have shown that the performance lost by reducing the L1 cache capacity can be covered by our SLVP technique. The experimental results, performed on the SPEC 2000 benchmarks, have expressed that reducing the L1 data cache space by quartering its size and using SLVP produces an improvement of the IPC and energy consumption in both the superscalar and SMT architectures against the corresponding baseline architectures.

2.1.1. Related Work

Lipasti et al. [134] firstly introduced value locality as the third facet of the statistical locality concepts used in computer engineering. They defined the value locality as “the likelihood of the recurrence of a previously-seen value within a storage location inside a computer system”. Measurements using SPEC95 benchmarks show that value locality on Load instructions is about 50% using a history of one (producing the same value as the previous one) and 80% using a history of 16 previous instances. Based on the dynamic correlation between Load instruction addresses and the values the Loads produce, Lipasti et al. proposed a new data-speculative micro-architectural technique, the LVP, that can effectively exploit value locality. LVP is useful only if it can be done accurately since incorrect predictions can lead to increased structural hazards and longer Load latency. By classifying Load instructions separately (unpredictable, predictable and constants) based on their dynamic behavior, it can be extracted the full advantage of each case. The cost of mispredictions can be avoided by detecting the unpredictable Loads. On the other hand, identifying highly predictable Loads reduces the cost of memory access. An important difference between our value prediction approach and Lipasti’s is that we selectively predict only critical Loads. Thus, we attenuate the mispredictions cost and reduce the hardware cost of the speculative micro-architecture. Moreover, since less hardware is required, there is also less power consumption.

Mutlu et al. presented in [152] a new hardware technique named *address-value delta (AVD) prediction*, able to parallelize dependent cache misses. They observed that some Load instructions exhibit stable relationships between their effective addresses and data values, due to the regularity of allocating structures in the memory by the program, which is sometimes accompanied by the regularity in the program’s input data. In order to exploit these regular memory allocation patterns, the authors proposed an AVD structure that maintains the Load instructions having a stable address-value difference (delta). Each entry of the AVD table consists in the following fields: *Tag* (the upper bits of the Load’s PC), *AVD* (the address-value delta corresponding to the last occurrence of that Load) and *Conf* (a saturating counter that records the confidence of AVD). The *Conf* field is used to avoid predictions for Loads with an unstable AVD. If a Load instruction having a stable AVD occurs with a cache miss, its data value is predicted by subtracting the stable delta from its effective address. This prediction enables the pre-execution of dependent instructions, including Loads with cache miss. The experimental results show that integrating a 16-entry AVD predictor into a *runahead* processor improves the average execution time by 14.3%, but only for pointer-intensive applications.

In [23], the authors proposed a hardware-based method, called *early load*, in order to hide the load-to-use latency (the latency that instructions wait for their operands produced by Load instructions) with little additional hardware costs. The key idea is to make use of the time that instructions are waiting in the instruction queue to load the data early, before the Loads are effectively executed, by pre-decoding instructions during the *fetch* stage. Thus, instead of using previous instances (values) of the current Load instruction Chang et al. are using an earlier executed instance (value) of the current Load instance. In this way, the chance to be a correct value seems to increase. They use a small table, called Early Load Queue (ELQ) that records Load instructions and the early loaded data. The proposed scheme allows Load instructions to load data from memory before the execution stage. Obviously, a detection method assures the correctness of the early operation before the Load enters into the execution stage. If the corresponding ELQ entry is valid in the Load’s dispatch stage, the execution of the Load instruction is completely avoided and all the dependent instructions get the data from the ELQ. Unfortunately, this method does not work for out-of-order speculative architectures whereas our technique does. Also, it works only for very small instruction queues. The experimental results showed that this scheme can achieve a performance improvement of 11.64% on the *Dhrystone* benchmark and 4.97% on the *MiBench* benchmark suite.

In [107], the author proposes a new load latency tolerant design that is both energy efficient, and applicable to both in-order and out-of-order cores, based on slice re-execution as an alternative use of multi-threading support, efficient schemes for register and memory state management, using a chained store buffer for efficient store-to-load forwarding, and using pruning mechanisms to reduce re-execution overheads. The main idea of load latency tolerance is to virtually scale the critical execution structures (issue queue, physical register file). Load latency tolerance designs remove from these window resources all the instructions dependent on Loads with miss in caches in order to allow younger instructions to enter the pipeline and execute. When a miss returns, the instructions which depend on it are re-injected into the pipeline – re-acquiring issue queue entries and physical registers – and re-executed.

2.1.2. Selective Load Value Prediction

We integrated into our architectures a simple last value predictor used only for Loads with miss in the L1 data cache (selective approach). In this way, the implemented structure is more efficiently used; the collisions number will be lower against the approach that predicts all Load instructions, having tables of the same size. The information about Load instructions is maintained in a direct mapped Load Value Prediction Table (LVPT). The LVPT is accessed during the execution stage, only if the current Load instruction involves a miss in the L1 data cache. The structure of the LVPT is presented in Figure 1.

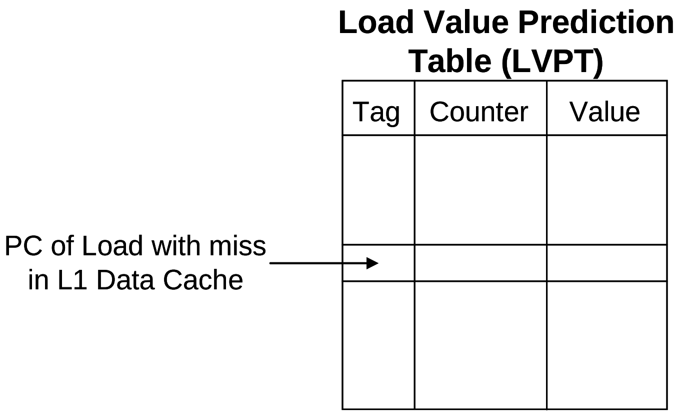

Load Value Prediction Table (LVPT)
Tag
Counter
Value
PC of Load with miss in L1 Data Cache

Figure 1. The last value predictor's architecture

Each LVPT entry has the following fields: *Tag* (the higher part of the PC), *Counter* (a 2-bit saturating confidence counter with two *unpredictable* and two *predictable* states), and *Value* (the Load instruction's result). In the case of a hit in the LVPT, the corresponding *Counter* is evaluated. If the confidence counter is in an unpredictable state, the Load is executed without prediction. Otherwise, the *Value* from the selected LVPT entry is speculatively forwarded to the dependent instructions. In the commit stage, when the real value is available, in the case of misprediction, a recovery is necessary to squash speculative results and selectively re-execute the dependent instructions with the correct values (see Figure 2). We considered in our simulations a recovery taking 7 cycles in the misprediction case.

During the *commit* stage, every critical Load updates the LVPT: only the *Counter* field in the case of correct prediction or the *Value* and the *Counter* fields in the case of misprediction. In the case of miss in the LVPT, the *Tag* and the *Value* are inserted into the selected entry, and the *Counter* is reset.

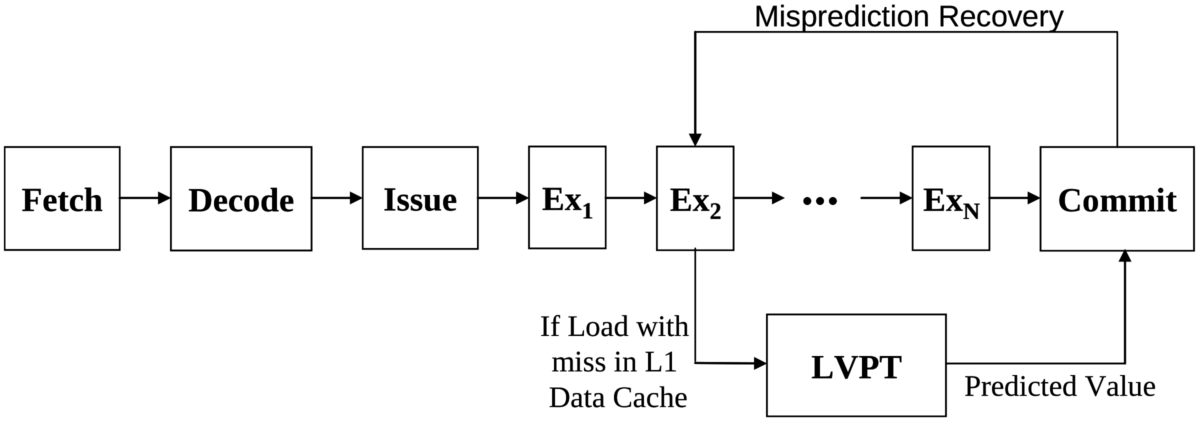


Figure 2. Pipeline with Selective Load Value Predictor

In our selective approach we predict only Loads with miss in the L1 data cache. Therefore, the prediction latency consists in the L1 data cache tagging and the LVPT access latency. We obtained using Cacti [186] an L1 data cache tagging of 2 cycles and a LVPT access latency of one cycle. Thus, the LVPT prediction latency, in our selective approach, is 3 cycles. In contrast, the original non-selective approach accesses the LVPT for all loads (without waiting for the L1 data cache tagging) and, therefore, the prediction latency is one cycle.

2.1.3. Simulation Methodology

We developed a cycle-accurate execution driven simulator derived from the M-SIM simulator [185] supporting the unmodified, statically linked Alpha AXP binaries as well as the power estimation as supplied by the Wattch framework [15]. M-SIM extends the SimpleScalar toolset [19] with accurate models of the pipeline structures, including explicit register renaming, and support for the concurrent execution of multiple threads. We modified M-SIM to incorporate our SLVP model in order to measure the relative IPC speedup and relative energy consumption reduction when the results of long-latency (critical) Loads are anticipated.

In the SMT mode [59], some processor structures (i.e. issue queue, physical register files, functional units, caches) are shared among the threads, and others (rename tables, ROBs, Load/Store Queues, branch predictors) are private to each thread. Our LVPT structure is private for each thread.

All the simulation results are generated on the SPEC 2000 benchmarks (http://www.spec.org) and are reported on 1 billion dynamic instructions, skipping the first 300 million instructions. For the superscalar architecture we evaluated six floating-point benchmarks (*applu*, *equake*, *galgel*, *lucas*, *mesa*, *mgrid*) and seven integer benchmarks: computation intensive (*bzip*, *gcc*, *gzip*) and memory intensive (*mcf*, *parser*, *twolf*, *vpr*). In SMT mode, the M-SIM simulator concurrently runs multiple benchmarks as different threads. We used the benchmark pairs {*bzip, gcc*}, {*gzip, parser*}, {*twolf, vpr*}, {*applu, equake*}, {*galgel, lucas*}, {*mesa, mgrid*} for our SMT with 2 threads. Processing more than 2 threads involves lower IPC speedups, because the shared functional units are better exploited due to the higher thread-level parallelism and therefore the LVPT structure becomes less important [67].

Next, Table 1 presents the parameters of some important units of the simulated microarchitecture whose configuration can influence both the processing performance and the energy consumption.

	Execution unit	**Number of units**	**Operation latency**
Execution Latencies	intALU	4	1
	intMULT / intDIV	1	3 / 20
	fpALU	4	2
	fpMULT / fpDIV	1	4 / 12
Superscalarity	Fetch /Decode /Issue /Commit width = 4		
Branch predictor	bimodal predictor with 2048 entries		
LVPT	1024 entries, Access lat. = 1 cycle, Prediction lat. = L1 cache tagging (2 cycles) + Access lat. (1 cycle) = 3 cycles		
Caches and Memory	**Memory unit**		**Access Latency**
	2-way associative L1 data cache, 64 KB		2 cycles
	2-way associative L1 instruction cache, 64 KB		2 cycles
	8-way associative unified L2 data cache, 4 MB		14 cycles
	Memory		270 cycles
Resources	**Register File:** [32 INT / 32 FP]*8		
	Reorder Buffer (ROB): 128 entries		
	Load/Store Queue (LSQ): 48 entries		

Table 1. Parameters of the simulated architecture

Our measurements are generated using an 80 nm CMOS technology and 1.2 GHz frequency. We used the following formula for the relative IPC speedup calculation:

$$IPC_{speedup} = \frac{IPC_{improved} - IPC_{base}}{IPC_{base}} \cdot 100 \text{ [\%]} \tag{1}$$

where IPC_{base} and $IPC_{improved}$ are the instructions executed per cycle with the baseline and improved architectures, respectively.

The detailed power modeling methodology, used in the simulator, is presented in [15]. The dynamic power consumption in CMOS microprocessors is defined as:

$$P = C \cdot V_{dd}^{2} \cdot a \cdot f \tag{2}$$

where C is the capacitance, generated using Cacti [186], V_{dd} is the supply voltage, and f is the clock frequency. V_{dd} and f depend on the assumed process technology. The activity factor a indicates how often clock ticks lead to switching activity on average. The power consumption of the modeled units highly depends on the internal capacitances of the circuits. From the capacitance point of view, there are three categories of architectural structures: array structures, content-associate memories, and complex logic blocks. The first two categories are used to model the caches, branch predictors, the reorder buffer, the register renaming table, and the register file, while the last category is used to model functional units.

For the power consumption evaluation, we used the *aggressive non-ideal conditional clocking* model [24] which scales linearly the power of active units with their usage and assumes 10% power dissipation in the case of unused units. The instantaneous average power consumption (P_{Mean}) for a certain benchmark is computed with the following relation:

$$P_{Mean} = \frac{\int_0^T P(t) \cdot dt}{T} \qquad (3)$$

where T is the total simulation time in cycles and P is given in relation (2). The energy consumption is given by:

$$E = P_{Mean} \cdot T \qquad (4)$$

where P_{Mean} is computed with relation (3) and T is the simulation time in cycles. The average energy (weighted mean) is given by the following formula in $[W \cdot cycles]$:

$$E_{Mean} = \frac{\sum_{i=1}^{N} E_i \cdot C_i}{\sum_{i=1}^{N} C_i} \qquad (5)$$

where N is the number of benchmarks, E_i is the total or per unit energy computed for benchmark i and C_i is the total number of cycles executed within benchmark i. The relative energy reduction percentage is given by:

$$E_{reduction} = \frac{E_{base} - E_{improved}}{E_{base}} \cdot 100 \text{ [\%]} \qquad (6)$$

where, E_{base} and $E_{improved}$ are the energy consumptions of the baseline and our improved architectures, respectively. Thus, a positive value of $E_{reduction}$ means an improvement of the relative energy consumption.

2.1.4. Experimental Results

We evaluated the IPC speedup, power dissipation and energy consumption with M-Sim [185] on SPEC 2000 benchmarks. We considered the following configurations:

- Baseline architecture (2-way assoc. 64KB L1 data cache, 2-way assoc. 64KB instruction cache, 8-way assoc. 4MB unified L2 cache);
- Baseline architecture with selective LVPT (direct mapped, 1024 entries, 15 KB);
- Baseline architecture with non-selective LVPT [134], for comparison;
- Baseline architecture but with only 1/2 L1 data cache (2-way assoc. 32KB);
- Baseline architecture but with only 1/2 L1 data cache & selective LVPT;
- Baseline architecture but with only 1/4 L1 data cache (2-way assoc. 16KB);
- Baseline architecture but with only 1/4 L1 data cache & selective LVPT.

Figure 3 presents the IPC speedups over the baseline architecture. It shows that by halving or quartering the cache, the IPC speedup is still significant with the LVPT. The IPC is just slightly lower in the case of using a quarter of the L1 data cache because its misses are covered by the L2 data cache. Since Lipasti's original prediction scheme presented in [134] is too complicated (being composed of three tables), it's obviously outperformed. Thus, we compare the simple last value prediction table used selectively [67] and non-selectively (predicting every load instruction).

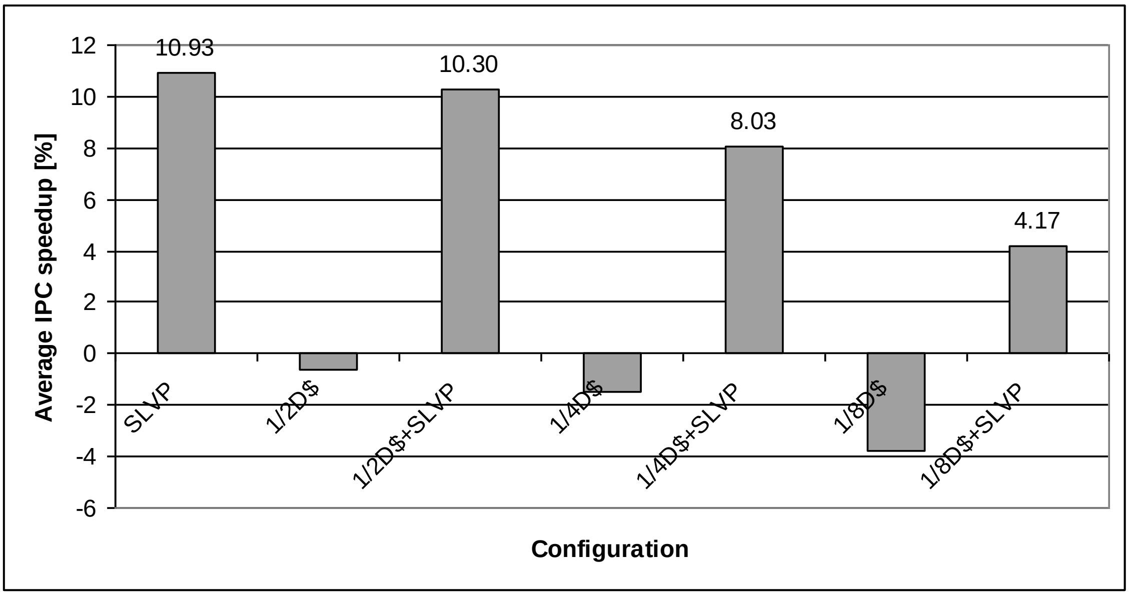


Figure 3. Average IPC speedups over the baseline superscalar architecture

	Selective LVPT (critical loads)		Non-selective LVPT (statistics extracted for critical loads)	
Benchmark	Predictions	Correct predictions	Predictions	Correct predictions
applu	22782796	21551807	17230347	16402766
bzip2	879219	736996	973495	773384
equake	5188	4394	1340	1225
galgel	159536411	159425807	159432043	159371494
gcc	7125184	6563152	6739856	6132672
gzip	4630937	3828240	4144950	3519523
lucas	17425768	17404806	17418002	17400437
mcf	47456	45347	1028	1023
mesa	1678431	1559547	1514038	1512910
mgrid	13079513	12934598	4814996	4778241
parser	5919387	5251232	5284201	4606584
twolf	5990146	4549855	4657853	3601658
vpr	10383135	8823526	7446884	5975711
Total	249483571	242679307	229659033	224077628

Table 2. Predictability with selective vs. non-selective LVPT

The IPC is higher on our selective approach because, as Table 2 shows, the number of predictions of critical loads is significantly higher than on the non-selective approach. The reason is that a higher number of Loads are fighting for the LVPT in the non-selective approach, producing interferences, in the detriment of critical loads. Considering the poor results obtained with the non-selective model, we renounced to present the corresponding results.

Figure 4 illustrates the energy reductions over the baseline superscalar architecture. As we can observe, the architecture using the quarter of the L1 data cache enhanced with selective load value prediction is the most efficient evaluated configuration. Besides the positive influence of LVPT to energy reduction, quartering the L1 data cache size substantially contributes to save energy.

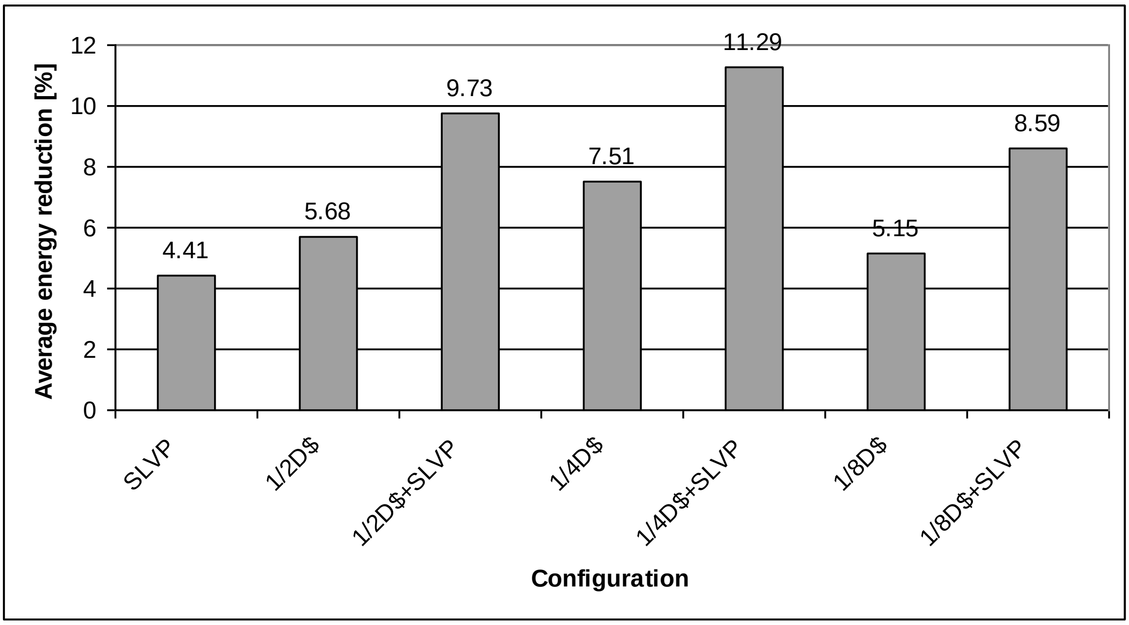


Figure 4. Average energy reductions over the baseline superscalar architecture

We also evaluated a SMT architecture in the same configurations as those presented above. Figure 5 presents the IPC speedup, whereas Figure 6 depicts the energy reduction over the baseline SMT architecture.

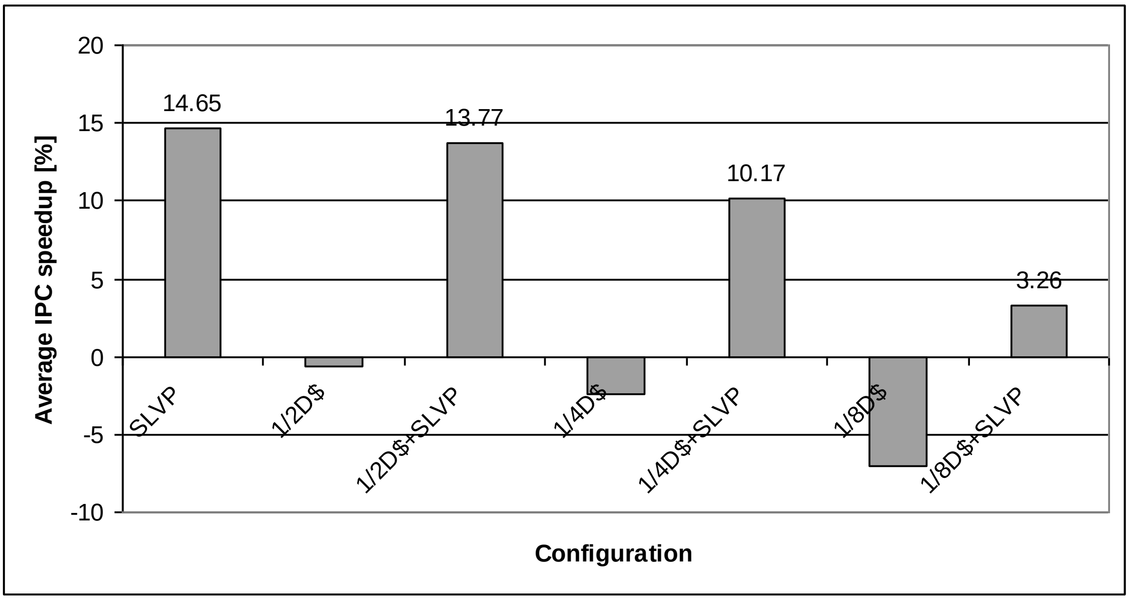


Figure 5. Average IPC speedups over the baseline SMT architecture

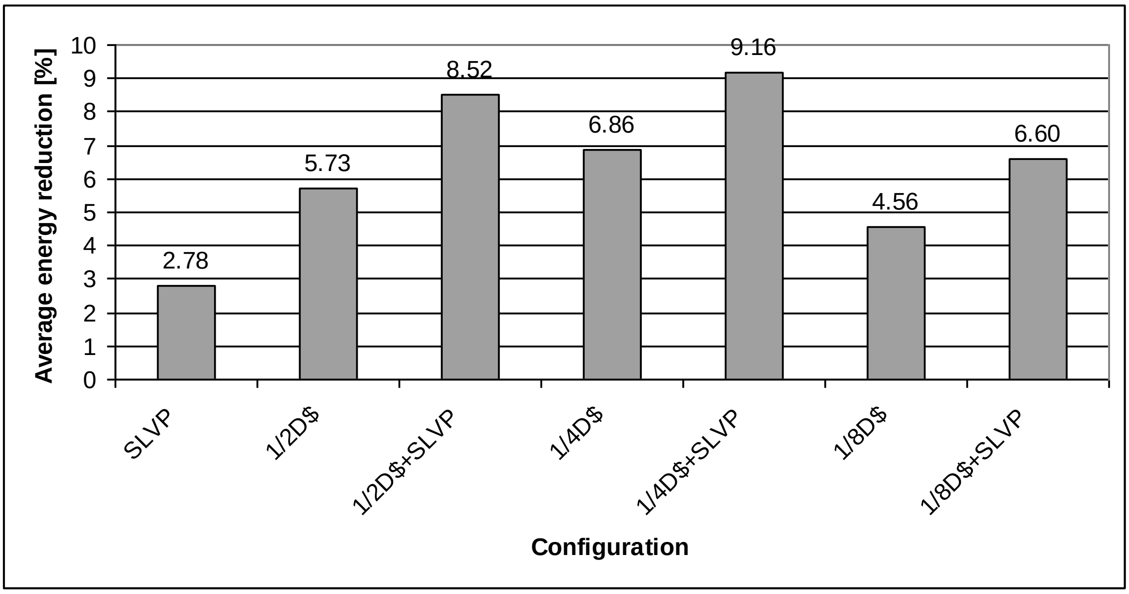


Figure 6. Average energy reductions over the baseline SMT architecture

As Figure 6 shows, the most efficient SMT configuration uses the half of the L1 data cache and our selective load value prediction technique. Quartering the L1 data cache does not further reduce the energy consumption as in the superscalar architecture and the IPC speedup is also lower.

2.1.5. Summary

Power consumption and processing performance represent two design constraints of any computing system ranging from embedded system to desktop computers. In this section we focused on the SLVP method by evaluating different architectural configurations in order to determine the most efficient one in terms of IPC and energy consumption. The experimental results, performed on the SPEC 2000 benchmarks, show a significant speedup and reduced energy consumption when we use a selective last value prediction table. Reducing the L1 data cache capacity by quartering its size and using a selective LVPT we obtained a relative IPC speedup at 8.03% and a relative energy reduction of 11.29% against the baseline architecture. These results encourage us to decrease the L1 data cache capacity in order to reduce the power consumption without loosing performance.

For the SMT architecture, halving the L1 data cache size and using a selective LVPT is the most efficient configuration, with a relative IPC speedup of 13.77% and a relative energy reduction of 12.48%. Quartering the L1 data cache of a SMT architecture with two threads does not further reduce the relative energy consumption as in the superscalar architecture.

As further work, we intend to study the thermal dissipation effects of our proposed architectural techniques and to try different dynamic energy and thermal management strategies: deactivating areas of L1 data cache, dynamic voltage or frequency scaling in function of the temperature, migrating computation in reserve modules, global clock gating, etc. The main idea would be to reduce power dissipation in the hotspots. Also, understanding and quantifying value prediction benefits in a multicore architecture might be a very important challenge.

2.2. Automatic Optimization of Superscalar Architectures with Load Value Predictor

This section presents based on [69] an automatic design space exploration of an architecture containing a SLVP scheme suitable for energy-aware superscalar processors. In this section we extend the manual design space exploration of a SLVP-based superscalar architecture to the L2 unified cache. Our goal is to find optimal configurations in terms of CPI (Cycles per Instruction) and energy consumption. After this manual design space exploration performed by varying only 2 parameters we intend to increase the number of the varied parameters to 19. By varying 19 architectural parameters, as we proposed, the design space is over 2.5 millions of billions configurations which obviously means that only heuristic search can be considered. Therefore, we propose different methods of automatic design space exploration based on our developed Framework for Automatic Design Space Exploration (FADSE) tool which allow us to evaluate around 2500 configurations of the above-mentioned huge design space, while still finding good solutions! We implemented a domain ontology consisting of some micro-architectural restrictions and expert knowledge expressed through fuzzy rules, in order to accelerate the design space exploration. By performing a multi-objective automatic design space exploration of the same architecture (using the FADSE tool [20]), but varying 19 architectural parameters, the obtained configurations are significantly better than the manually obtained ones.

2.2.1. Related Work

M3Explorer [187] is a DSE framework that includes many design space exploration algorithms. M3Explorer can use response surface models to accelerate the design space exploration. Another DSE tool is in a form of a website: archexplorer.org [37]. The users can upload their component on the website where it is integrated into a computer system simulator. The design is compared against other designs introduced by other users. The users do not have any control on the algorithm being used. NASA [117] is also a similar tool. It allows the user to easily integrate his/her DSE algorithm and offers the possibility to connect to any simulator (features offered also by FADSE). Magellan [121] is a DSE tool which is bounded to a certain simulator (SMTSIM). Magellan can perform only single objective DSEs.

In [114], the FADSE tool was used to explore the vast design space of the Grid Alu Processor (GAP) and its post-link optimizer called GAPtimize, both developed at Augsburg University. It has shown that FADSE is able to thoroughly explore the design space for both GAP and GAPtimize and it can find an approximation of the Pareto frontier [20] consisting of near-optimal individuals in moderate time. For the GAP, FADSE can find, due to the approximation of the complexity, efficient configurations.

In [174], the authors present an interesting DSE methodology for FPGAs. In [198], the authors are focusing on developing a DSE method for mapping tasks to the microarchitectural resources of an MPSoC. They exploit specific domain knowledge related to task allocation through a genetic algorithm. In contrast to our work, their domain knowledge is not focused on finding the best parameter values for a mono-core/multi-core hardware architecture from a multi-objective point of view. Also, their implemented domain knowledge is not considering hardware constraints, hierarchical parameters and design rules expressed in fuzzy logic as in our work.

In [201] the authors present their developed performance modeling and simulation framework called ABSOLUT which reduces the complexity of the exploration by applying abstract virtual system models. The authors used the ABSOLUT tool in different applications. The average difference between the simulation results and the measurements performed on real platforms was quite acceptable (12%).

In [171] the authors present a DSE tool for an FPGA soft processor. The proposed framework uses regression trees and allows a fast DSE in an early design stage. In [30], the authors present an electronic system-level methodology for FPGA devices. They define a system-oriented computation model to capture the structure of parallel applications. The authors also present interesting early hardware cost estimation techniques which can speed up the DSE process. The methodology is adequately validated through a case-study on parallel JPEG encoding.

To our knowledge FADSE is the single DSE tool that allows the user to introduce domain knowledge through fuzzy rules, written in a human-readable form, in order to accelerate the design space exploration.

2.2.2. Simulation Methodology

All the experimental results presented further were obtained using the SPEC 2000 benchmarks on 500 million dynamic instructions, skipping the first 300 million instructions. We evaluated six floating-point benchmarks (*applu*, *equake*, *galgel*, *lucas*, *mesa*, *mgrid*) and six integer benchmarks: computation intensive (*bzip*, *gcc*, *gzip*) and memory intensive (*mcf*, *twolf*, *vpr*). Our measurements are generated using an 80 nm CMOS technology and 1.2 GHz frequency.

The target architecture is a superscalar Alpha AXP 21264 processor augmented with a direct mapped selective load value predictor of 1024 entries, access latency of 1 cycle and prediction latency of 3 cycles [68]. It has a Register File of [32 int / 32 fp]*8, a Reorder Buffer (ROB) of 128 entries and a Load/Store Queue (LSQ) of 48 entries. First-level caches are 64 KB, 2-way associative, with a 1-cycle latency. The second-level unified cache is 4 MB, 8-way associative and 6-cycle latency. The main memory has a latency of 100 cycles.

The energy reduction metric was presented in 2.1.3 (see formula (6)). For the performance metrics we chose CPI (and not IPC) because we want to minimize all the objectives for the clarity of the Pareto graphs. For the relative CPI reduction we used the following formula:

$$CPI_{reduction} = \frac{CPI_{base} - CPI_{improved}}{CPI_{base}} \cdot 100 \text{ [\%]} \qquad (7)$$

where CPI_{base} and $CPI_{improved}$ are cycles per instructions with the baseline and improved architectures, respectively. A positive value of $CPI_{reduction}$ means a performance improvement related to the baseline architecture.

2.2.3. Manual Design Space Exploration

A method to increase the cache performance is to reduce the penalty in case of miss using multilevel caches. Our simulated architecture uses two level exclusive caches. This allows smaller L2 data caches involving less power consumption. The evictions are performed based on the Least Recently Used (LRU) algorithm for both cache levels.

The first goal of our research consists in performing a design space exploration regarding the sizes of the L1 data cache and the unified (instruction & data) L2 cache in superscalar architectures augmented with SLVP structures. Thus, we will double, halve, quarter and eighth the L2 cache and we will halve, quarter and eighth the L1 data cache, considering as reference the architecture presented in Section 3. We note with *m*UL2_*n*DL1 a configuration using m*4 MB 8-way associative unified L2 cache (m=2, 1, 1/2, 1/4, 1/8) and n*64 KB 2-way associative L1 data cache (n=1, 1/2, 1/4, 1/8).

Figure 7 presents the relative CPI and energy reduction – computed based on formulas (7) and (6), respectively – of different configurations with SLVP of 1024 entries reported to the baseline configuration without SLVP. It can be observed that the SLVP helps maintaining a better CPI and energy consumption when the cache sizes are reduced. The CPI reduction with the help of the SLVP is positive up to using halve of UL2 and eighth of DL1. Starting with quartering UL2, the CPI reduction is negative and therefore no performance improvement is achieved.

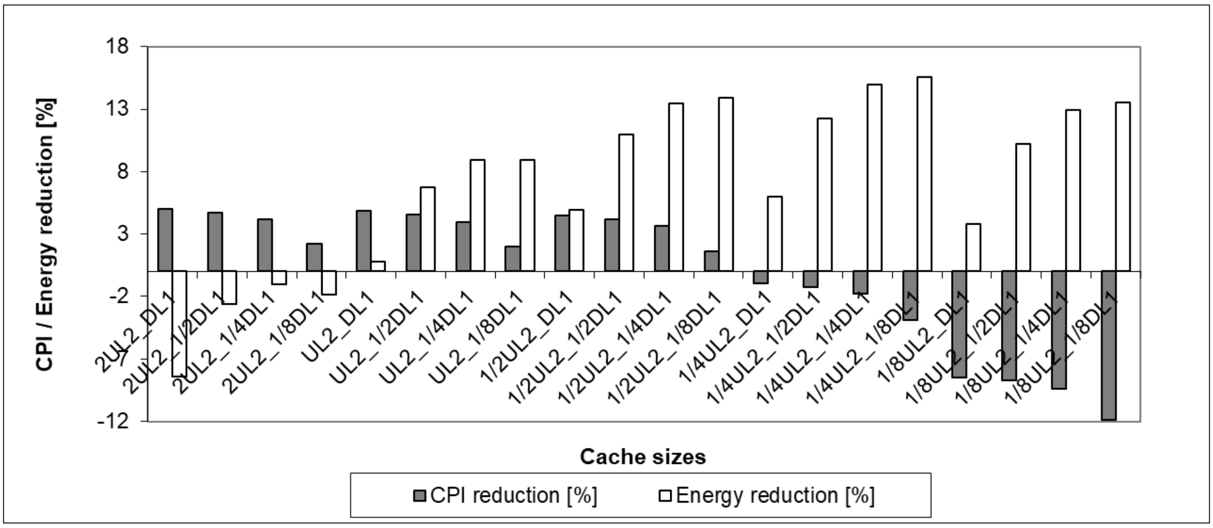


Figure 7. Relative CPI and energy reduction reported to UL2_DL1 without SLVP as baseline

The energy reduction is lower in the case of reducing the L2 cache to 1/8 than in the case of quartering it. The energy consumption has a static and a dynamic component. The 1/8UL2 cache implies a higher miss rate than the 1/4UL2 and therefore higher dynamic power consumption, due to the higher number of off-chip accesses. Thus, even if the static power consumption of 1/8UL2 is lower than of 1/4UL2 the energy consumption is higher due to the higher dynamic power consumption. Therefore, using only the quarter of the L2 cache (2 MB) and the eighth of the L1 data cache (8 KB) is optimal from the energy consumption viewpoint.

As a preliminary conclusion, after the manual design space exploration the best configuration regarding CPI is 2UL2_DL1 whereas the best configuration in terms of energy consumption is 1/4UL2_1/8DL1. There are also some optimal configurations from both CPI and energy viewpoints: 1/2UL2_1/2DL1 and 1/2UL2_1/4DL1. These results obtained through manual DSE encourage us to explore a larger design space by automatic DSE because the best and the optimal configurations are different and there are also other parameters which can be varied.

2.2.4. Automatic Design Space Exploration

In the previous section we varied only the cache sizes through our manual design space exploration. Beside the caches there are several parameters that can highly influence our two objectives: CPI and energy consumption. We selected 19 important architectural parameters to be varied during our automatic design space exploration, with the lower and upper limits given in Table 3. By varying these 19 architectural parameters the design space grows over $2.5*10^{15}$ (2.5 millions of billions) configurations which obviously means that only heuristic search can be considered. Therefore, we propose different methods of automatic design space exploration based on our developed FADSE tool that contains also a NSGA-II genetic algorithm implementation.

Parameter		Lower limit	Upper limit
DL1 / IL1 cache	Sets	2	32768
	Block size (bytes)	8	256
	Associativity	1	8
UL2 cache	Sets	256	2097152
	Block size (bytes)	64	256
	Associativity	2	16
SLVP (entries)		16	8192
Decode / Issue / Commit width		2	32
ROB / LSQ / IQ size (entries)		32	1024
Number of physical register sets (int / fp)		2/2	8/8
Int / fp ALU		2	8
Int / fp MUL/DIV		1	8

Table 3. Parameter limits

To perform design space exploration, we have developed a tool called FADSE. It includes many state-of-the-art evolutionary algorithms through the included jMetal [43] library. FADSE can be connected to almost any existing simulator. The parameters are described through an extensible XML interface. FADSE allows parallel evaluation (included algorithms had to be modified to allow this).

FADSE is a client-server application. The number of clients can be dynamically changed. Clients can be stopped or started while the DSE process runs. Since the DSE process can take a lot of time (weeks), reliability of the DSE tool is a major concern. FADSE is able to cope with failing clients, failing networks or even power loss of the entire system. It is able to recover from these situations by detecting the problems and resubmitting the simulations to other clients. In the case of power loss, it can restart the DSE process by making use of the integrated checkpointing mechanism. It contains a database which allows reusing already simulated individuals. This leads to a reduction of the time required to perform an exploration process. FADSE includes many metrics to evaluate the DSE process or to compare different algorithms. Some of the implemented metrics are: hypervolume, coverage, two set difference hypervolume, etc.

We have chosen for our automatic DSE the NSGA-II genetic algorithm. NSGA-II is a multi-objective genetic algorithm developed by Deb et al. [35]. NSGA-II is not a distributed algorithm by default. To accelerate the DSE process we have changed the algorithm and now the individuals are evaluated in parallel. This is possible because the values of the objectives of an individual are required only after all the individuals are evaluated. So, an entire population can be evaluated in parallel and a single synchronization point has to be established at the end of a generation. We configured the NSGA-II algorithm as described below.

As stop condition we will observe the hypervolume progress. If there is no progress for at least X generations we consider that the algorithm has converged. To measure the progress, we will use the following formula:

$$\text{Progress} = \sum_{i=1}^{X} (H_k - H_{k-i}) \tag{8}$$

where H_k is the hypervolume of the current generation k, $X \leq k$. When this sum is smaller than a specified threshold θ the algorithm is stopped.

The population size is 100 – as recommended in [35]. We use the bit flip mutation with a mutation probability of $1/nparam$ [35] (where *nparam* is the number of varied parameters). In our situation the mutation probability is set to 0.05 (19 parameters).

We apply single point crossover, with the probability of crossover set to 0.9 (as specified in [35]). We use binary tournament selection (described in [35]). Used metrics:

– Hypervolume: in a maximization problem the hypervolume is the volume enclosed between the Pareto front approximation and the axes. In a minimization problem a point must be selected (called hypervolume reference point). The hypervolume reference point is selected at the coordinates provided by maximum values of the objectives.
– Other metrics: number of generated individuals, comparisons between the obtained Pareto fronts approximation.

2.2.4.1. Run without prior information

First, we start FADSE with an initial randomly generated population, without prior information. We search for the optimal SLVP-based superscalar configurations considering the same two objectives, CPI and energy consumption, as in the previous manual design space exploration. We vary the parameters presented in Table 3 with the hope to find better configurations than our manually obtained "optimal" configurations. To avoid extremes which can generate unfeasible configurations, we used the following constraints:

- UL2 > DL1 + IL1
- UL2_bsize ≥ DL1_bsize
- UL2_bsize ≥ IL1_bsize

where *UL2_bsize*, *DL1_bsize* and *IL1_bsize* are the block sizes for the unified L2 cache, L1 data cache and L1 instruction cache, respectively. Additionally, we limited the cache sizes by using the following hard constraints (borders):

- DL1: 16 KB - 1 MB
- IL1: 16 KB - 1 MB
- UL2: 1 MB - 8 MB

Unfortunately, the constraints used within the initial run does not allow FADSE to efficiently explore the borders and, therefore, the configurations were not better than those obtained manually (see Figure 8) from the energy point of view. Consequently, we relaxed the minimum cache capacities as follows:

- DL1: 4 KB - 1 MB
- IL1: 8 KB - 1 MB
- UL2: 256 KB - 8 MB

As Figure 8 shows, with relaxed borders FADSE provides significantly better configurations than our previous manual design space exploration. With these constraints the design space is reduced to 3% of the initial space, meaning $7.7*10^{13}$ configurations. Considering both objectives, the better results are influenced by the following parameters: less DL1 sets, less IL1 sets, higher decode/issue/commit width, higher ROB size, higher IQ size, higher number of MUL/DIV and higher SLVP size. Big structures will increase the energy consumption thus we do need FADSE to find the relations between different parameters. We can observe again that SLVP helps maintaining better CPI and energy consumption with reduced cache sizes.

Since the exploration with relaxed borders was superior to the initial constraints, in the next experiments we used only the relaxed borders.

2.2.4.2. Run with manually obtained "optimal" configurations

The second step in our experiment consists in starting FADSE with an initial randomly generated population but containing also our manually obtained "optimal" configurations and their vicinity (with the goal to find better ones). We selected from Figure 7 the best configuration in CPI, 2UL2_DL1, the best configuration in terms of energy consumption, 1/4UL2_1/8DL1, and other two configurations which are optimal from both CPI and energy viewpoints: 1/2UL2_1/2DL1 and 1/2UL2_1/4DL1. We also considered the vicinities of these four configurations by varying the SLVP size, L1 data cache size and L2 unified cache size one step up and down. Thus, we started FADSE again with randomly generated population but containing also our 24 selected configurations: the “optimal” manual configurations and their vicinities (some of them are overlapped). Figure 8 shows the obtained Pareto fronts after 25 generations by the first three runs (initial run, run with relaxed borders and run with initial good configurations) compared with the manually obtained configurations.

In terms of CPI all the runs find much better solutions than the manually obtained configurations. The run with relaxed borders clearly finds better configurations than the ones obtained through manual exploration and also better than the ones found during the initial run (restrictive constraints). The obtained solutions are distributed evenly along the Pareto approximated front.

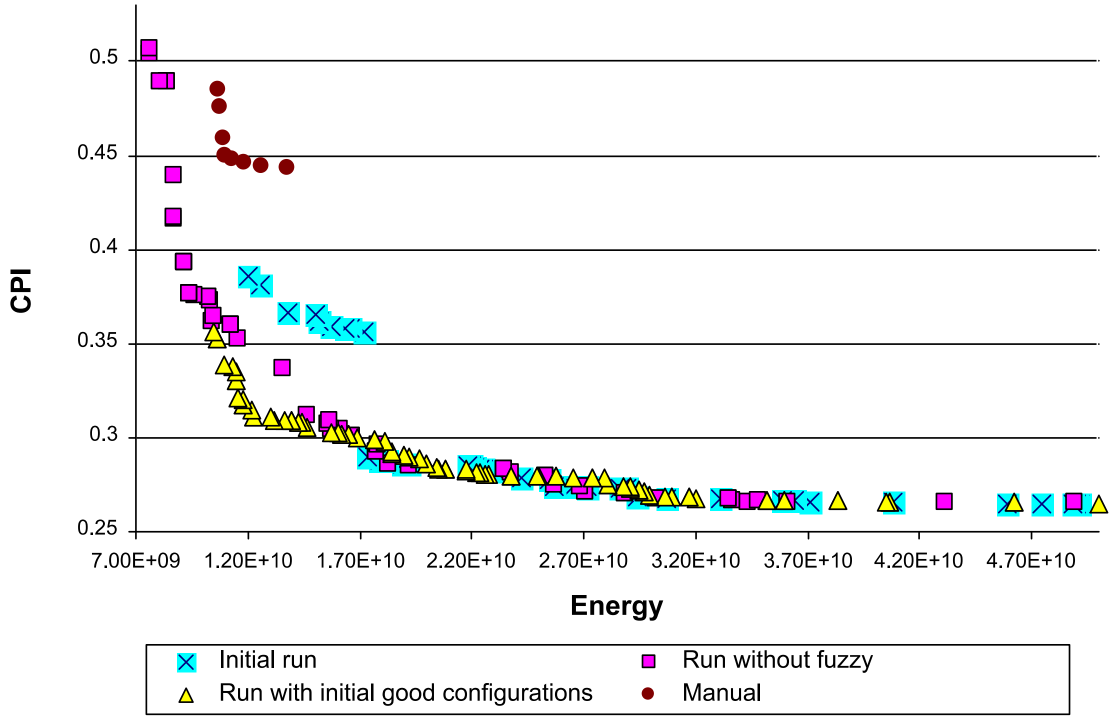


Figure 8. Pareto fronts’ comparison

Inserting good configurations into the initial population provides also good results but it is not able to explore the area with very low energies. It obtains better results than the run with relaxed borders in the vicinity of energy 1.20E+10 $[W \cdot cycles]$. Low energy configurations are not found probably because the initial configurations were better (and we have observed this on our analysis of the Pareto front approximation evolution over the generations) than all the other individuals inserted randomly in the population. So, all these good individuals survived until the next generation and most of the offspring were generated from them, thus loosing diversity. The mutation operator with a probability of 0.05 of changing one parameter has a small chance to influence significantly the produced offspring, leading to a reduction of diversity.

2.2.4.3. Run with knowledge expressed through fuzzy rules

We are using fuzzy rules to allow the designer to express knowledge. The information provided by these fuzzy rules is then used during the search process to guide the DSE algorithm. For this purpose, we have included the jFuzzyLogic library in FADSE. A user can define rules in a standard FCL file (IEC 61131 part 7). In [69], we have developed and implemented the following rules derived from our experience in computer architecture design:

- IF Number_Of_Physical_Register_Sets IS *small/big* THEN Decode/Issue/Commit_Width IS *small/big*
- IF SLVP_size IS *small/big* THEN L1_Data_Cache IS *big/small*

We have selected the Mamdani-type fuzzy systems [143]. These imply the following steps that need to be carried to extract information: fuzzification of the input variables, evaluating the rules, aggregating the outputs and then defuzzification. The fuzzification was done using trapezoidal functions. For the evaluation the *min* function was used for "and" and *max* for "or". For inference, the Mamdani implication has used (*min*). The rule aggregation was performed using the Mamdani aggregation (*max*). This system was selected because of its popularity.

Two different mutation operators were used. Both are based on the bit flip mutation. To preserve diversity, the information provided by the fuzzy rules is not always taken into consideration. To obtain this, a probability of applying the fuzzy information (called fuzzy probability) is used. The only difference between the implemented methods is how the probability to apply the information provided by the fuzzy rules is computed.

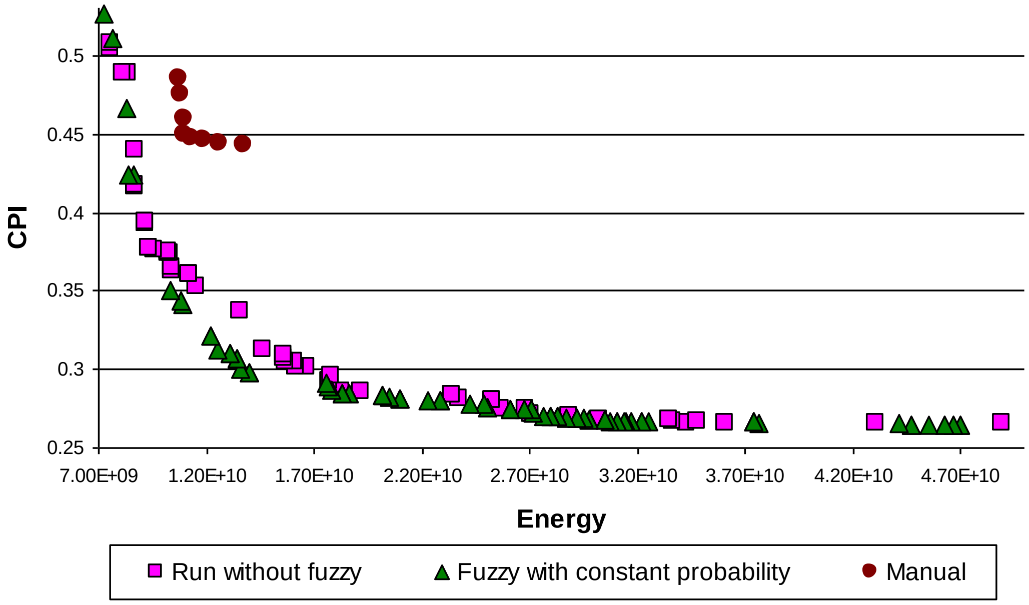


Figure 9. Pareto front comparisons between the run with fuzzy rules and the run with relaxed borders

In the simple implementation, this fuzzy probability is constant during the run of the algorithm and it is set to be equal with the probability of mutation (*mutation_prob*). If the fuzzy rule is not applied, the algorithm switches to the classical bit flip mutation for the current parameter. The second implementation uses a Gaussian probability so there is a higher chance to apply the fuzzy rules for the first generations. As the DSE process runs, the probability to apply the fuzzy rules decreases to a value close to *mutation_prob*. We have selected the parameters of the Gaussian function such that at generation 5 the function is close to 0. The Gaussian function is then translated so that the minimum is close to *mutation_prob*. The final form of the function is shown below:

$$f(x)_{final} = (1 - mutation\_prob) \cdot e^{\frac{-(x)^2}{2\cdot(150)^2}} + mutation\_prob \qquad (9)$$

where x increases with one for each individual generated by the algorithm. The result of this function is further multiplied by 0.8 and by the membership value [228] to obtain a maximum value less than 1. Figure 9 presents the results obtained with fuzzy information compared with the previous results.

We have selected the run with relaxed borders (called "run without fuzzy") as the reference run since it has found solutions all along the approximated Pareto front. In Figure 9, the run with fuzzy information (constant probability to apply the fuzzy rules) and the run without fuzzy information are compared. It can be easily observed that the run with fuzzy information obtains very good results. Figure 9 also shows that the run with fuzzy rules finds better results in the vicinity of energy 1.20E+10 $[W \cdot cycles]$ than the run without fuzzy information. We have also compared the run with fuzzy rules with the one with initial good configurations and we observed that in fact the later obtains a few individuals which are slightly better.

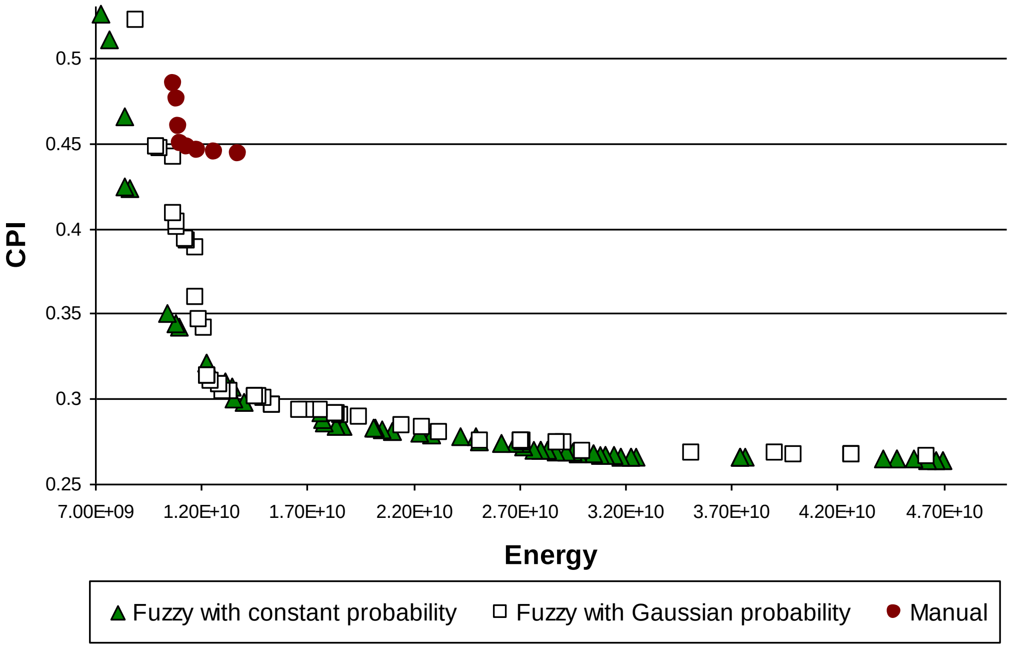


Figure 10. Pareto fronts comparison between the runs with fuzzy rules

Figure 10 performs a comparison between the results obtained with fuzzy information but with different methods of calculating the probability to use the information provided by them. The run with constant probability finds better individuals in the area with low energy. Having an almost 80% probability to apply the rules during the first generations might lead to a loss in diversity of the individuals on the parameters influenced by the rules. This fact might explain the poorer results.

Figure 11 gives us two types of information: about the convergence of the algorithms and about the quality of results. It can be observed that the algorithms tend to stop the rapid evolution after 15 generations (initial run is an exception). The algorithms were run until generation 25 due to time constraints.

After a comparison of the hypervolume values we can conclude that: the initial run with restricted borders obtained the worse results. Relaxing the borders considerably improved the quality of results even though the size of the design space has become larger by a factor of around 2 (from $3.8*10^{13}$ to $7.7*10^{13}$). Having good configurations inserted in the initial population can lead to very good results but it starts to perform worse after generation 14, rising a bit after generation 19 and then dropping dramatically after generation 23. Observing the obtained Pareto front approximation and its evolution we have concluded that the algorithm tends to focus on a smaller area of the space, falling into local minima. This can be explained by the lack of diversity of the initial configurations, since all the individuals inserted differ on only two parameters from a total of 19.

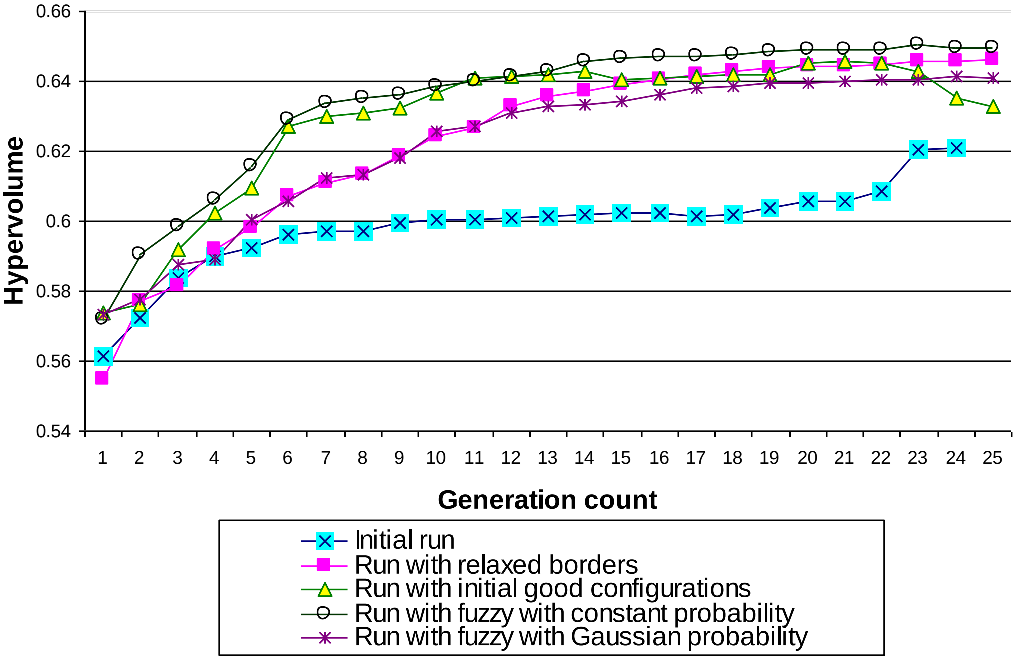


Figure 11. Hypervolume comparison

The run with a constant probability of accepting the results from the fuzzy rules provided the best results. The run with a Gaussian probability of applying the information provided by the fuzzy rules had a similar behavior at the beginning with the run using relaxed borders. After generation 12 the results are slightly worse. We can conclude that imposing a high probability of the rules will reduce diversity, especially with a small number of rules. In our previous work more rules were used and the membership functions had many intervals (associated linguistic terms) [115]. In this situation the runs with Gaussian probability provided better results.

It can be observed that using some extra knowledge (initial configurations or fuzzy rules) makes the algorithm start from a better initial population (see the hypervolume values at generation 1) and, as a consequence, the algorithm's convergence speed is higher.

The hypervolume corresponding to the run with a constant probability of applying the fuzzy rules at generation 15, is reached by the run with relaxed borders only at generation 24. This is a great improvement. In our experiments, running one generation on 96 cores belonging to an Intel Xeon powered HPC system, with cores running at 2GHz, takes around one day. Running with fuzzy rules we achieved the same results 9 days earlier (36% faster) than without fuzzy rules. Additionally, after the same amount of time (25 generations) the hypervolume reached by the run with fuzzy rules is never reached by the simple run. If more qualitative information would have been provided through the fuzzy rules, we do expect even better improvements. A single experiment takes around 25 days. Running all the five experiments took over 4 months of simulation on a HPC system using 96 cores.

All the runs evaluate roughly the same amount of individuals: around 2200 from the 2500 individuals sent for evaluation; the rest are reused from the database (12% reuse degree). This means that the produced offspring are almost all of them new/different individuals. This behavior is caused by the extremely large design space. In previous explorations on different simulators (smaller design space – 10^6) a reuse degree of around 60% was observed [114].

After analyzing the best results obtained during the automatic DSE process we extracted the parameter values for optimal configurations from either high performance or low energy viewpoints. The results are presented in Table 4. As it can be observed, some parameters like Decode width and SLVP size must have high values in order to obtain both high performance and low energy.

Parameter		**For high performance**	**For low energy**
DL1 cache	Sets	2048	32
	Block size (bytes)	256	64
	Associativity	2	2
IL1 cache	Sets	1024	32
	Block size (bytes)	16	256
	Associativity	8	1
UL2 cache	Sets	8192	256
	Block size (bytes)	256	256
	Associativity	4	16
SLVP (entries)		4096-8192	4096-8192
Decode / Issue / Commit width		32 / 16 / 32	16-32 / 4-8 / 16-32
ROB / LSQ / IQ size (entries)		1024 / 512-1024 / 128	256 / 32-64 / 64
Number of physical register sets (int / fp)		8 / 8	2 / 2
Int / fp ALU		8 / 8	8 / 2
Int / fp MUL/DIV		8 / 8	8 / 8

Table 4. Parameter values for optimal configurations

2.2.5. Summary

We have observed that the SLVP helps maintaining a better CPI and energy consumption when the cache sizes are reduced. Therefore, the optimal configurations, obtained by both the manual and automatic design space exploration of our SLVP-based superscalar architecture, have lower cache sizes than the baseline architecture without SLVP.

FADSE is able to find good configurations by evaluating a very small percentage of the total search space. We reduced the number of evaluated configurations to only 2500, representing $3*10^{-11}$% of the huge constrained design space of 77 thousands of billions configurations. The experimental results show that our automatic design space exploration provides significantly better configurations than our previous manual design space exploration.

Starting FADSE with initial good configurations can accelerate the DSE process. It is recommended that the configurations differ on multiple parameters so that diversity is preserved. In our situation the configurations differed only on three parameters thus leading to a loss of diversity and finally it could not explore the entire Pareto front.

Using fuzzy rules can considerably accelerate the DSE process (9 days earlier to reach the same result in our situation). Also the obtained results are better than the ones obtained with no prior information after the same amount of time. In this concrete optimization process, the constant probability of applying the fuzzy rules lead to better results. In our previous work – where the number of rules was higher and had more linguistic terms associated to the membership functions – we have obtained better results by modulating the probability with a Gaussian function during the generations. With a larger number of rules, the individuals are mutated into a more diverse population. Thus, forcing the rules to be applied often does not lead to a loss of diversity in the population.

We plan to repeat these experiments on SLVP-based SMT and multi-core architectures. Other further work possibilities are to access the SLVP only in the case of miss in both the L1 and L2 data caches, to index the SLVP table with the memory address instead of the instruction address, to exploit an N-value locality instead of 1-value locality as we are currently exploiting, to evaluate set-associative SLVP configurations and yet another one to design and implement an adaptive dynamic run-time thermal manager (temporarily deactivating the SLVP unit, voltage scaling, frequency scaling, migrating computation, etc.). We also plan to explore, as the third objective, the die size needed by the memory architecture.

2.3. Multicore Architectures with Load Value Predictor

In the last years the research in computer architecture was focused on multicore and manycore systems which mainly exploit the parallelism from concurrent programs. Some of the techniques applied to increase the performance in superscalar processors can have benefits in multicore systems, too. One such technique is load value prediction which speculates the results of loads to unlock subsequent dependent instructions.

In this section, based on our previous work published in [71] and [72], we have enhanced the Sniper state-of-the-art multicore simulator with load value prediction capabilities. For doing this, we have integrated into Sniper private (per-core) LVPTs. The value prediction is selectively applied only on load instructions with miss in the first level of data cache (DL1). In this way, by focusing only on the high latency loads, a small and fast LVPT is enough to take all the benefits of load value prediction in terms of performance and energy consumption. Thus, we have used the LVPT structure to apply the SLVP technique.

We have investigated a simpler counter-based method and a more complex perceptron-based one. Perceptrons have been successfully applied in [118], [119] and [120] for efficient dynamic branch prediction within two-level adaptive schemes that are using fast per branch single-cell perceptrons instead of two-bit saturating counters. The branch address is hashed to select the corresponding perceptron's weights, which are used to generate a prediction based on the global / local branch history, however, as it was shown in [94], [95], [93], [202], [203], [204], [56], [57], [178] and [157], such prediction information is not always relevant in the case of some hard-to-predict branches. The perceptron, one of the simplest neural networks, is a natural choice for branch prediction because it can be efficiently implemented in hardware. In this section, we adapted the perceptron-based branch prediction scheme presented in [118] and [119] for load value prediction [72], in order to ameliorate the so-called issue bottleneck (data-flow bottleneck). For each load instruction stored in the LVPT, the corresponding perceptron's weights are stored, too. Actually, the perceptron's hardware structure, representing a binary adder (Wallace-Tree of 3 to 2 Carry-Save, having a logarithmic depth), is global. Based on the previous behaviors of a certain load instruction, the corresponding perceptron classifier (a unique hardware structure together with the load instruction's corresponding weights) determines if that load instruction is predictable or not. Only if a load instruction is predictable, the predicted value is used to speculatively unlock the subsequent dependent instructions' processing. Thus, if the prediction is correct, the dependent instructions are earlier executed, which contributes to a global speedup of the multicore system. On the other hand, if the prediction is wrong, a recovery process is necessary, in order to re-establish the correct architectural state of the corresponding core (thread). Therefore, high prediction accuracies are necessary, which can be obtained with efficient predictors applied only on the load instructions which are dynamically classified as predictable. Thus, with good prediction accuracies on a significant number of dynamic loads, this technique can provide a speedup. If the speedup is sufficiently high, it can also reduce the energy consumption, as we already have shown in the related papers [68], [69] and [71].

We have applied a manual design space exploration of the proposed parameterized speculative architecture on the Splash-2 parallel benchmarks. Our objectives are to analyze the prediction accuracy of the SLVP, the speedup and the energy consumption of multicore microarchitectures enhanced with SLVP, as well as to investigate the correlation between the obtained speedups and the number of critical loads (with miss in the DL1 cache). We believe that the importance of our work could be significant, taken into account that, as far as we know, nobody investigated the benefits of implementing value prediction in multicore systems.

2.3.1. Related Work

Value prediction was first introduced by Lipasti et al. in [134] and it was further intensively investigated in monocore processors. Monocore architectures enhanced with direct mapped SLVP tables have been presented in our previous works [67], [68], [69]. In [70], we have parameterized the SLVP table in a monocore microarchitecture, allowing to access it with instruction or data address, to use multiple values per entry, set-associative table organization, selective access on miss in the L1 data cache or in the L2 unified cache. The evaluations performed with M-SIM on the SPEC 2000 benchmarks have shown that a set-associative SLVP table is more effective than a direct mapped one and using multiple values per SLVP entry is better. Value prediction focused on the logical registers of superscalar architectures, as a low-power alternative to the instruction-centric value prediction paradigm, have been successfully applied by us in [205], [206] and [207]. The Sniper state-of-the-art multicore simulator and the Splash-2 parallel benchmark suite have been successfully used for microarchitectural evaluations in [54]. Further, we will limit our presentation only to some of the most recent valuable papers focused on this issue.

In [146], the authors have investigated load value estimation in applications that accept inexactness. Thus, rollbacks are eliminated, since re-execution in the case of misprediction is not necessary. They show that without rollbacks, the load value estimation is still presenting low error in the application’s output. Spatio-value approximation is also exploited through the so-called Bunker Cache in [147]. In contrast, our method targets all the types of applications by not accepting any inexactness of the speculated data values.

In [164], the authors proposed a confidence estimation mechanism for value prediction in monocore architectures. They used 3-bit confidence counters and they predicted only on saturated counter (so on highest value). They incremented the counters on correct prediction and reset them (to zero) on misprediction. The authors reported prediction accuracies between 95% - 99%. They have also introduced the VTAGE context-based value predictor (derived from the previous ITTAGE predictor) which uses global branch history and path information to predict values. In [166], the authors proposed a block-based value prediction scheme which associates the predicted values with fetch blocks. They have also presented the Differential VTAGE predictor (D-VTAGE), which uses stride-based value prediction. The obtained average speedup was 11.2%. In contrast with our work, which investigates value prediction in a multicore system, all these authors were focused on monocore processors.

In [165], Perais and Seznec proposed the Early Out-of-order Late Execution (EOLE) monocore architecture, which delays the value prediction validation within the pipeline until the commit stage. Thus, the authors avoid selective replay and enforce complete pipeline squash on misprediction, significantly simplifying the hardware design. Additionally, the authors further reduced the design’s complexity by dynamically classifying instructions into early execution, out-of-order execution and late execution instructions. The instructions having immediate or predicted operands are executed early and in-order, in the front end. On the other hand, predicted instructions and high confidence branches are executed late and in-order in a pre-commit processing stage. Both the early and late execution instructions avoid the out-of-order engine, reducing thus the pressure on this unit which led to lower energy consumption. The same classification of the instructions into the early execution, out-of-order execution and late execution categories is applied in [167], too. In contrast with this briefly presented work, limited to monocore approaches, we applied the value prediction on critical load instructions (with miss in the Data L1 Cache) processed within a complex multicore architecture.

In [48], Endo et al. have evaluated the potential of value prediction in the context of the EOLE microarchitecture and the D-VTAGE value predictor, considering different compilation options. The authors observed a large benefit from load value prediction, especially in the case of unoptimized codes. In [160], Orosa et al. proposed a load value prediction mechanism which predicts the load address first. The predicted load address is then used to index a Value Table (VT) in order to predict the load’s value. For a better coverage, the authors improved the hit rate of the VT with an adaptive algorithm which is prefetching future predicted addresses into the VT.

In [144], a very interesting original work, the authors have shown that value prediction can violate the sequential consistency in multithreaded and multicore architectures. The problem can occur when value prediction is applied on codes manipulating pointer-based shared variables. As a concrete instructive example, the authors are considering two distinct threads: one thread is inserting to the front of a list (writer), while the other thread is reading the first element of the list (reader). No further synchronization is necessary between these two threads. The reader or writer may execute its code first, or the instructions may occur in an interleaved manner. If the reader thread is executed first (with load value prediction), the predicted value V1 of its L1 load instruction is speculatively used as an address for another subsequent load instruction L2. An initially wrong prediction (V1) might be erroneously seen as being correct at the time of verification, reading thus a possible wrong value V2 with L2 instruction, since another thread or core (the writer) has already modified the values V1 and V2, between (L1) prediction and verification (late validation). Thus, the authors have firstly shown that predicting the value of an instruction with later validation is not sufficient in a multicore architecture. Despite the fact that this process might be counterintuitive, however it can appear. To solve such possible consistency problems of shared variables, we applied in the simulator the value-based detection solution proposed in [144]. According to this approach, all the load instructions executed with directly or transitively predicted address are re-executed when the address becomes non-speculative. If the corresponding values match, the sequential consistency was not violated and the execution can continue. If the values do not match, all the subsequent data-dependent instructions are re-executed with the right values, restoring in this way the sequential consistency. We applied this selective re-issue mechanism in our work, too.

2.3.2. Counter-Based Selective Load Value Prediction in Multicore Architectures

The proposed SLVP technique [71] requires per-core LVPTs within the multicore microarchitecture. The role of the LVPT is to exploit load value locality through value prediction. It keeps the last data values of the critical load instructions with the hope that they will have the same outputs on eventual next dynamic executions, based on the value locality statistical principle. On next occurrences, if the attached predictability confidence is sufficiently high, the stored values can be speculatively used by the subsequent dependent instructions, increasing thus the performance. Based on the structure of a Nehalem core presented in [54], we provide in Figure 12 how a dual core architecture can integrate SLVP. Obviously, this organization can be extended to any number of cores. We have evaluated configurations consisting in 1, 2, 4, 8 and 16 cores. As Figure 12 illustrates, the LVPT is private together with the IL1 cache, DL1 cache, L2 unified cache, branch predictor (BP), reordering, paging, and execution units; only the L3 cache is shared by the cores. The generic structure of the LVPT is presented in Figure 13.

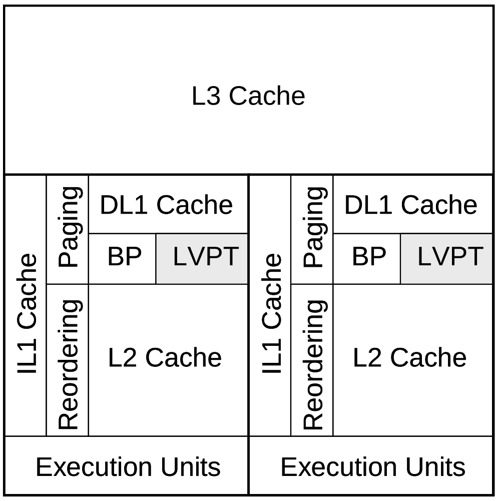


Figure 12. Dual Core Microarchitecture with SLVP

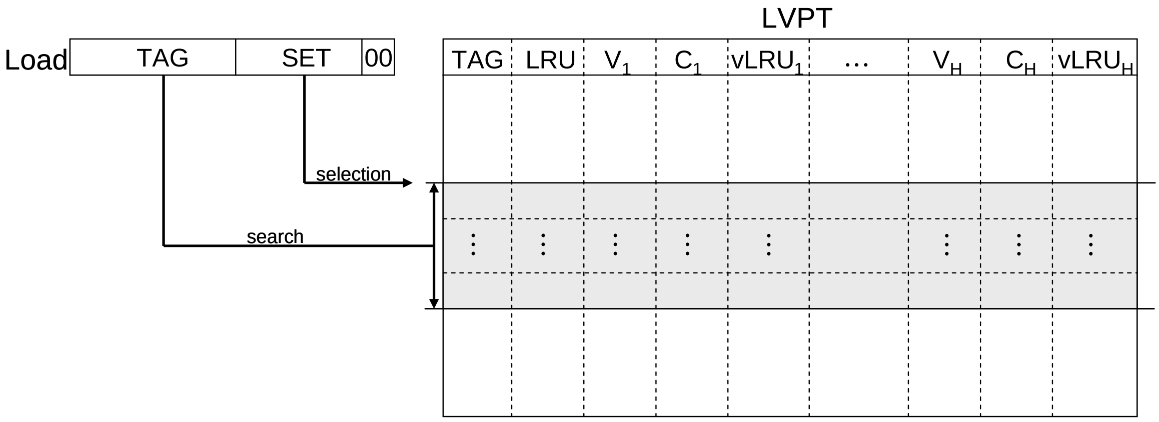


Figure 13. The LVPT structure

Each LVPT entry has a TAG field consisting in the most significant bits of the load instruction's address, a LRU field necessary for the decisions regarding the replacements within the set-associative table, a history of the last distinct values, each such value V having associated a confidence automaton C and a vLRU field. The role of the vLRU fields is to decide which one of the H stored values must be replaced when a new one occurs. The vLRU is set to the maximum for the correct value and is decremented for the other values.

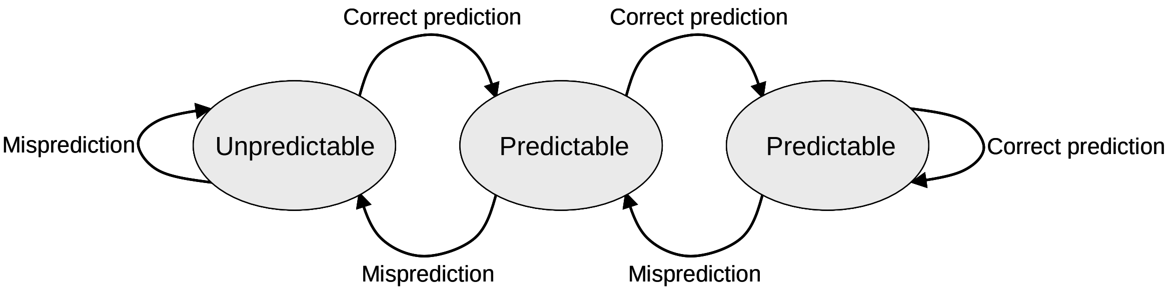


Figure 14. Confidence automaton

The role of the confidence automaton is to dynamically classify each load value kept in the LVPT into unpredictable or predictable. We denote the confidence automata U/P, where U is the number of unpredictable states and P is the number of predictable states. A 3-state 1/2 confidence automaton is depicted in Figure 14. We have implemented the confidence automaton as a saturating counter which is incremented on correct predictions and decremented on mispredictions. The initial state is the unpredictable one.

When a load instruction with miss in the DL1 cache occurs, the SET field (consisting in the least significant bits of the load instruction's address, excluding the last two bits which are always 0) is used to select the corresponding set from the LVPT. After that, the TAG is searched within the TAG field from the selected set. If it is not found, the load instruction is normally executed and, after this, it is inserted into the selected set of the LVPT by replacing the least recently used load instruction from that set (the entry with the lowest LRU). In that case, all the confidences are reset, the first vLRU is set to the maximum value and all the other vLRUs are reset. If the TAG is found, the highest confidence C is identified. If that confidence is in unpredictable state, the load is normally executed, without value prediction. If the confidence is in predictable state, its associated value V is predicted as being the result of the load. This predicted value is forwarded to the dependent instructions from the reservation stations that will be speculatively executed. After the normal execution of the load, the value of the real result is used to update the LVPT entry. If it is a new value, it will replace the least recently used value of that entry (the value having the lowest vLRU) and all the vLRUs and confidences are correspondingly updated. Otherwise, only the vLRUs and the confidences are updated. In the case of misprediction, a recovery process is also necessary and the dependent instructions executed with wrong values are squashed and re-executed with the correct values (selective re-issue).

2.3.3. Perceptron-Based Selective Load Value Prediction

In this section, we improve our SLVP scheme implemented in a multicore architecture [71] by classifying the critical load instructions into predictable or unpredictable through simple one-cell perceptrons instead of saturating counters [72]. The main goal is to increase the predictability of the SLVP unit. We adapted the perceptron-based branch prediction presented in [118] and [119] for load value prediction. The whole load value prediction process is presented in Figure 15. The LVPT has a TAG field consisting in the most significant part of the load instruction's address stored in the Program Counter (PC), a LRU field necessary for the decisions regarding the replacements within the set-associative table, a Locality History Register (LHR) containing the last behaviors of the load instruction, the set of perceptron weights W, a history of the load instruction's last distinct produced values, each such value V having associated a confidence automaton C and a vLRU field. The role of the vLRU fields is to decide which one of the last H stored values must be replaced when a new one occurs.

Further we present the selective load value prediction mechanism, the perceptron-based procedure of dynamically classifying load instructions into the predictable or unpredictable classes, as well as the update mechanism.

If the currently executed load instruction is involving a miss in the Data L1 Cache, the LVPT is accessed during its pipelined issue stage. The LVPT set is selected using the least significant part of the PC and then an associative search is performed for the TAG within the selected set. In the case of miss in the LVPT, the load instruction is considered unpredictable and it is normally executed through the instructions' pipeline. If there is a hit in the LVPT, the highest confidence is evaluated to select the corresponding value. The LHR and the W fields are used for the perceptron's forward stage, in order to determine if the load instruction is predictable or not. In the unpredictable case, the load is normally executed, without prediction and speculation. Otherwise, the value selected from the LVPT is speculatively forwarded to the in-flight Read After Write (RAW) dependent instructions and they are executed in a speculative manner, potentially reducing the processing time. In the commit stage, the predicted value is compared with the real value. In the case of misprediction, a recovery process is necessary in order to squash the wrong speculative results and selectively re-execute the RAW dependent instructions with the correct produced values.

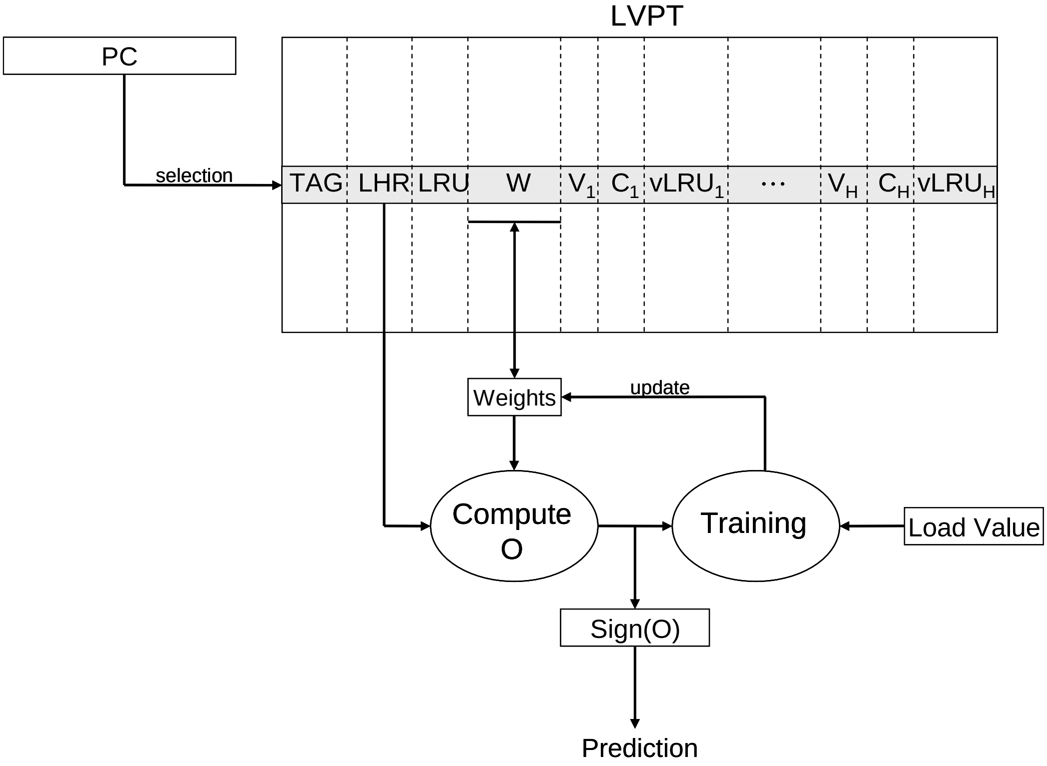


Figure 15. The whole load value prediction process

During the commit stage, every critical load updates the corresponding SLVP entry: the confidences, the vLRU fields and, additionally, the data value, in the case of a new produced value. The confidence automaton is incremented for the correct value and decremented for the wrong values. If the actual produced value is not belonging to the stored history, it overwrites the value having the lowest vLRU. The vLRU fields are set to their maximum value for the correctly predicted value and decremented for the others. The LHR is correspondingly updated, too. The backward step is also applied, if necessary, to adjust the weights according to the simplified implemented gradient descent learning rule.

In the case of miss in the LVPT, the entry with the lowest LRU from the set is selected. The new TAG is inserted into the selected entry, the V_1 field is updated with the produced data value and all the confidences from that entry are reset. The first vLRU is set to the maximum, whereas all the other vLRUs are reset. Finally, the LRU of the selected entry is set to its maximum value and the LRUs corresponding to the other entries from the corresponding set are decremented. The LHR bits are reset, excepting LHR_0 which is kept on 1. The weights W are reset, too.

The LHR contains the last *n* behaviors (L_i, $1 \leq i \leq n$) of a certain load instruction: $L_i=0$ when the corresponding Load's real output value (V_R) is not in the V set, or $L_i=1$ when the produced output value belongs to the V set. The V set from a certain LVPT entry is containing the last H distinct values produced by the previous dynamic instances of the corresponding load instruction. The LHR is logical right-shifted after the commit of each load instruction whose behavior (0 or 1) is inserted as the most significant bit (MSB) into the LHR. We keep LHR_0 always on logical 1 ($L_0=1$) for the bias weight w_0. The detailed perceptron-based load value prediction scheme is depicted in Figure 16.

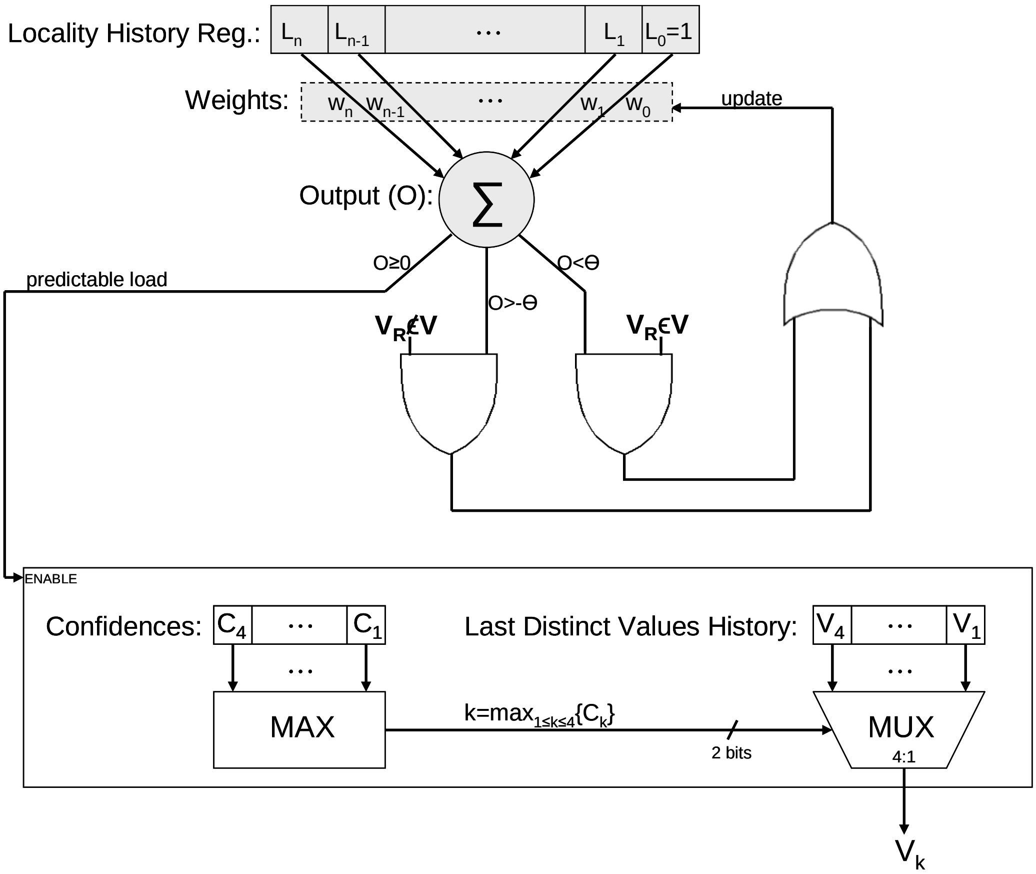


Figure 16. Perceptron-Based Load Value Prediction Scheme (H=4)

The MAX circuit works as follows: if MAX (C_1, C_2, C_3, C_4) = C_K then the MAX circuit generates the binary value of *k*, codified on 2 bits. Based on this value, the 4:1 multiplexer (MUX) will select the predicted value V_k, k=1÷4.

The weights are signed integers, represented on one byte, which cannot exceed the value of Θ. The threshold Θ is also used to decide if the training stage is necessary. As it was shown in [118], the value of Θ is “exactly” $\lfloor 1.93n + 14 \rfloor$.

2.3.3.1. The Forward Stage

The forward stage consists in calculating the perceptron’s output sign which is used to decide if the corresponding load instruction is predictable or not. The output O is given by the following formula (as in [120] for branch prediction):

$$O = w_0 + \sum_{i=1}^{n} \begin{cases} w_i, \text{ if } L_i = 1 \\ -w_i, \text{ if } L_i = 0 \end{cases} \tag{10}$$

where w_i represent the perceptron’s corresponding weights (w_0 being the bias weight). The formula (10) is equivalent with the following one:

$$O = w_0 + \sum_{i=1}^{n} w_i \cdot (-1)^{L_i+1} \tag{11}$$

Implementing formula (10) in hardware is not very complicated. According to [119], the above sum can be obtained using a Wallace-tree of 3-to-2 carry-save adder, which reduces the process of adding n bytes to the problem of adding just two bytes. The final two signed integers are added with a carry-look-ahead adder. Taking into account that the Wallace-tree and the carry-look-ahead adder have logarithmic depths, the computation is relatively quick.

Only the sign bit of the result is needed to decide if the load instruction is predictable or not. A prediction is generated only in the case of a positive output.

2.3.3.2. The Backward Stage

The backward (learning) stage is applied if the output is in contradiction with the fact that the real value V_R belongs or not to the V set, or if the magnitude of the output is less or equal with the threshold Θ. The pseudocode of the backward algorithm is given below:

```
If ($O<0$ and $V_R \in V$) or ($O \geq 0$ and $V_R \notin V$) or $|O| \leq \Theta$ then
   If $V_R \in V$ then
      If $w_0 < \Theta$ then
         $w_0 := w_0+1$
      Endif
   Else
      If $w_0 > (-\Theta)$ then
         $w_0 := w_0-1$
      Endif
   Endelse
   For i:=1 to n in parallel do
      If ($V_R \in V$ and $L_i=1$) or ($V_R \notin V$ and $L_i=0$) then
         If $w_i < \Theta$ then
            $w_i := w_i+1$
         Endif
      Endif
      Else
         If $w_i > (-\Theta)$ then
            $w_i := w_i-1$
         Endif
      Endelse
   Endfor
Endif
```

According to the single-cell perceptron’s stochastic gradient descent learning method, the simplified implemented learning algorithm increments the bias weight w_0 if V_R belongs to V and decrements it otherwise. Furthermore, it increments the weight w_i if L_i agrees with the fact that V_R belongs or not to V and it decrements that weight in the case of disagreement. Thus, the weights with mostly agreement become positive with large magnitude and those with mostly disagreement become negative with large magnitude. In these cases, there is a high correlation between the output and the weight. The correlation between the output and a certain weight is weak if its magnitude is close to 0.

2.3.3. Simulation Methodology

In this study we have used the Sniper 6.1 multicore simulator [22] and the large datasets of the Splash-2 suite of parallel benchmarks [221] which are characterized as follows: Barnes simulates in three dimensions a system of bodies; Cholesky operates matrix factorization; Fmm simulates in two dimensions a system of bodies; Lu factors a dense matrix; Ocean studies large-scale ocean movements; Raytrace renders a three-dimensional scene using ray tracing; Water evaluates forces and potentials in a system of water molecules.

The simulations have been run on a computer with Intel Core 2 CPU at 2.4 GHz and a DRAM of 2 GB, under Fedora 22 (kernel 4.0.8-300). The simulated microarchitecture is Intel Nehalem and can include between 1 and 16 cores configured to run at 2.66 GHz. The baseline configuration is presented in Table 5.

DL1 / IL1 cache	Size	16 KB
	Block size	64 B
	Associativity	4
	Latency	3 cycles
L2 cache	Size	256 KB
	Block size	64 B
	Associativity	8
	Latency	9 cycles
L3 cache	Size	8192 KB
	Block size	64 B
	Associativity	16
	Latency	35 cycles
Memory	Latency	175 cycles
SLVP	Latency	1 cycle
	Recovery	7 cycles

Table 5. Parameters of the simulated architecture interacting with the SLVP

The latencies have been determined using Membench in [22] and configured in gainestown.cfg. The cache sizes and associativity degrees have been configured in nehalem.cfg. The L3 cache is shared among cores.

In order to integrate our SLVP into the Sniper simulator, it was necessary to apply the following setups:

Configuration	File
[general] issue_memops_at_functional = false	*gainestown.cfg*
[perf_model/core] type = interval	*nehalem.cfg*
[perf_model/dram/queue_model] enabled = false	*base.cfg*
[network/emesh_hop_by_hop/queue_model] enabled = false	*base.cfg*
[network/bus/queue_model] enabled = false	*base.cfg*

Table 6. Sniper configuration

The counters necessary for statistics like the number of loads, the number of critical loads, the number of load predictions and the number of correct load predictions have been defined in performance_model.h, registered as output metrics in performance_model.cc and written to the sim.out file within gen_simout.py. The predictor's structure and functions have been implemented in micro_op_performance_model.cc and called in the handleInstruction function from the same source file. We presented these technical details because they are not documented.

The performance improvement with SLVP can be roughly evaluated based on a simplified idealistic analytical model. The speedup of multicore architectures can be theoretically estimated according to Amdahl's low:

$$S(N) = \frac{1}{s + \frac{(1\text{-}s)}{N}} \tag{12}$$

where N is the number of cores, $s \in [0,1]$ is the algorithm's inherently sequential computation fraction. It is considered that the remaining (1-s) computation is completely parallelizable.

The theoretical speedup of multicore architectures with value prediction can be determined by generalizing Amdahl's low as in the following formula:

$$S_{VP}(N) = \frac{1}{s \cdot (1\text{-}p) + \frac{(1\text{-}s)}{N}} \tag{13}$$

where $p \in [0,1]$ is the average value prediction accuracy. We have ideally considered that $(s \cdot p)$ fraction of computation is performed instantaneously, due the correct value prediction. Thus, the speedup of a monocore architecture (N=1) is:

$$S_{VP}(1) = \frac{1}{1\text{-}s \cdot p} \tag{14}$$

Considering s=1 (thus, the whole program is sequential), we obtain:

$$S_{VP}(1) = \frac{1}{1\text{-}p} \tag{15}$$

This last pure theoretical formula is identical with the simplified formula obtained in [63], based on an idealized stochastic mathematical model, showing thus the consistency of our developed theoretical model.

Since the Instructions Per Cycle (IPC) metric of a certain core can be computed as the number of instructions executed by that core divided to the number of CPU cycles, we can define the IPC of a multicore architecture, on a certain benchmark, as the total number of instructions executed by all the cores divided to the longest cycle time among cores:

$$IPC = \frac{\sum_{i=1}^{N} I_i}{C} \tag{16}$$

where I_i is the number of instructions executed by the core i, N represents the number of cores and C is the execution time, in cycles. The speedup of a multicore architecture extended with value prediction (VP), with respect to the baseline multicore architecture (B), is:

$$S_{VP} = \frac{IPC_{VP}}{IPC_B} \tag{17}$$

Since the total number of dynamic instructions executed by these two compared architectures is approximately the same (the slight differences can be neglected), the speedup can be computed as:

$$S_{VP} = \frac{C_B}{C_{VP}} \tag{18}$$

Finally, the relative speedup of a multicore architecture with value prediction can be computed as:

$$RS_{VP} = \frac{C_B - C_{VP}}{C_B} \cdot 100[\%] \tag{19}$$

The energy consumption, expressed in J, can be determined using the following formula:

$$E = \frac{P \cdot C}{f_{CPU}} \tag{20}$$

where P represents the total power consumption measured in W (provided, in our case, by the McPAT simulator), f_{CPU} represents the frequency of the simulated microprocessor in Hz and C represents the total number of execution cycles. The relative energy reduction is given by the following formula:

$$E_{reduction} = \frac{E_B - E_{VP}}{E_B} \cdot 100\ [\%] \tag{21}$$

where, E_B and E_{VP} are the energy consumptions of the baseline and our improved architectures, respectively. Obviously, a positive value of $E_{reduction}$ means an improved energy consumption.

We will also use Pearson's correlation in order to determine what is influencing the speedup in a SLVP-based multicore architecture. Thus, we will analyze the correlation between the relative speedup and the DL1 miss rate, the prediction rate and the prediction accuracy, respectively. The prediction rate is the percentage of predicted critical loads divided by the total number of critical loads. The prediction accuracy is the percentage of correctly predicted critical loads divided by the number of predicted critical loads.

2.3.4. Experimental Results

2.3.4.1. Counter-Based Selective Load Value Prediction

We started our evaluations by analyzing the LVPT's parameters. For doing this, we used a dual core architecture (N=2) with the configuration presented in Table 5.

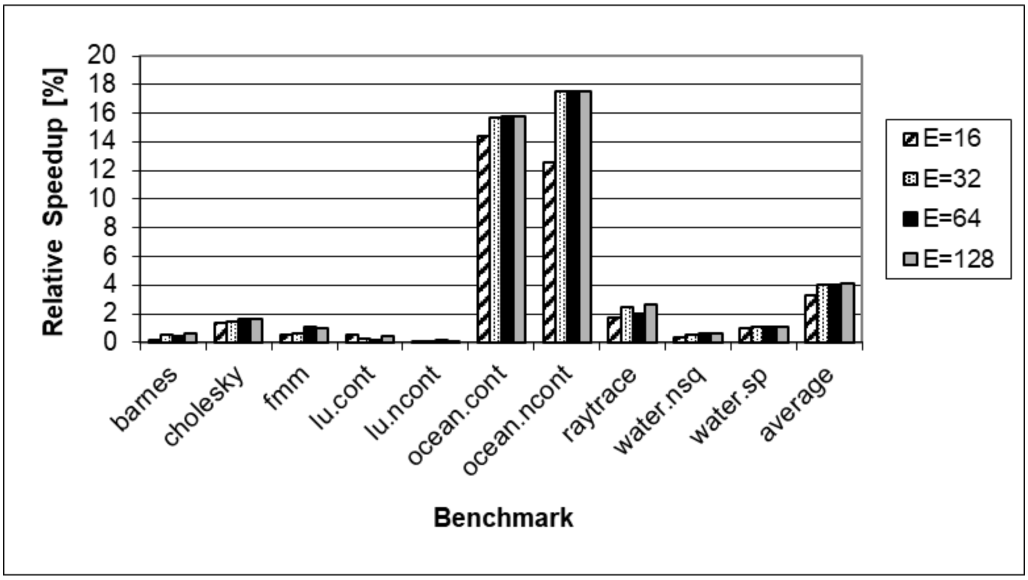


Figure 17. Varying the number of LVPT entries

For the LVPT we fixed the associativity to 2, the history (the number of stored values per entry, denoted H) to 2, we have also considered an 1/2 confidence automaton and we have varied the number of LVPT entries (E). This initial configuration was chosen after some apriori laborious simulations. Figure 17 illustrates the relative speedup obtained over the baseline architecture using different LVPT sizes. The relative speedup has been computed using formula (19). As Figure 17 shows, the optimal number of LVPT entries is 32. Over this size the performance improvement is insignificant. On the most of the benchmarks the relative speedup was less than 2%, but on the ocean.cont and ocean.ncont benchmarks it was over 15% (see Table 7 and the explanation presented below it). Thus, the average relative speedup, obtained with the optimal LVPT size (E=32) was 4.01%.

The next analyzed parameter is the LVPT associativity (A). Thus, we have fixed the size of the LVPT to the optimal 32 entries obtained above, the load values' history to 2, we have used 1/2 confidence automata and we have varied the LVPT associativity degree. Here some comments are necessary. Of course, that this "hill-climbing" optimization method is not ideal, thus it cannot find the global optimum. On the other hand, the multi-objective automatic optimization problem is a NP-hard problem in this case, as we have already shown in [29]. Such complex optimization problems are solved using heuristic algorithms, in an approximate manner. It would be the main aim of another dedicated work. Thus, in this section we will assume a manual optimization method. Figure 18 depicts the relative speedup over the baseline architecture using different LVPT associativity degrees. As the evaluations show, the optimal associativity is 2.

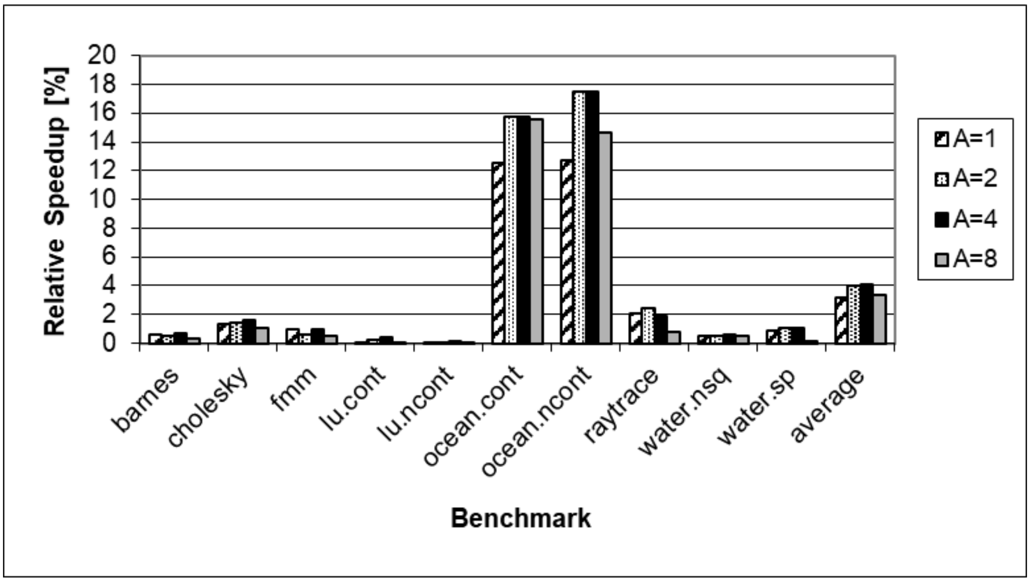


Figure 18. Varying the LVPT associativity

Next, we have evaluated the impact of the value history's length over the relative speedup, using a LVPT with 32 entries, an associativity degree of 2 and also 1/2 confidence automata. As Figure 19 shows, the performance is quite invariant to history length's changes. Therefore, we have chosen to continue the experiments with just one stored value per LVPT entry (H=1).

We have continued our evaluations by analyzing different confidence automata, fixing the LVPT to 32 entries, associativity degree of 2, and values' history of 1, as we previously obtained. Figure 20 illustrates that the optimal confidence automaton is a three-states one, using one predictable and two unpredictable states (denoted 1/2).

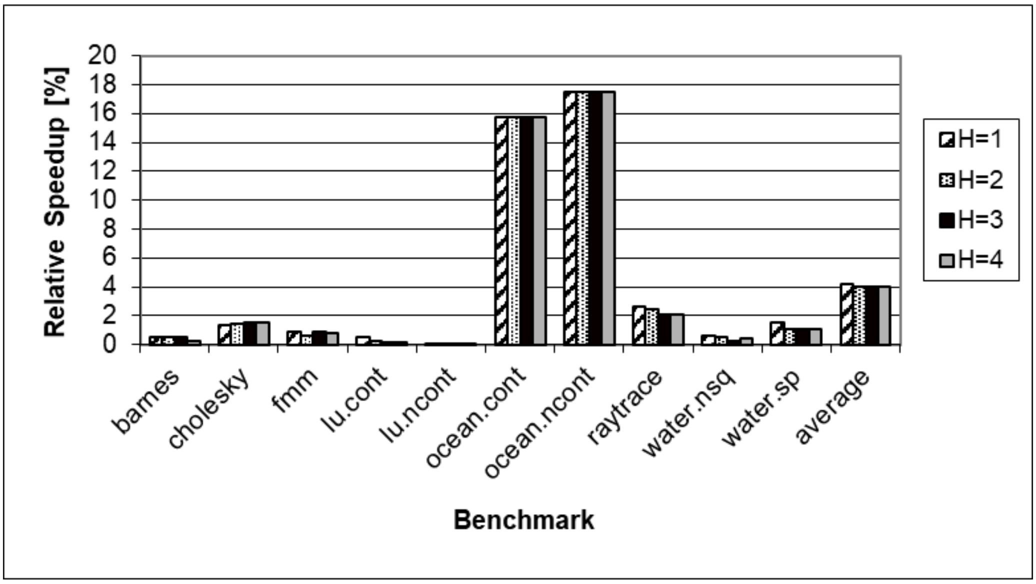


Figure 19. Varying the LVPT history

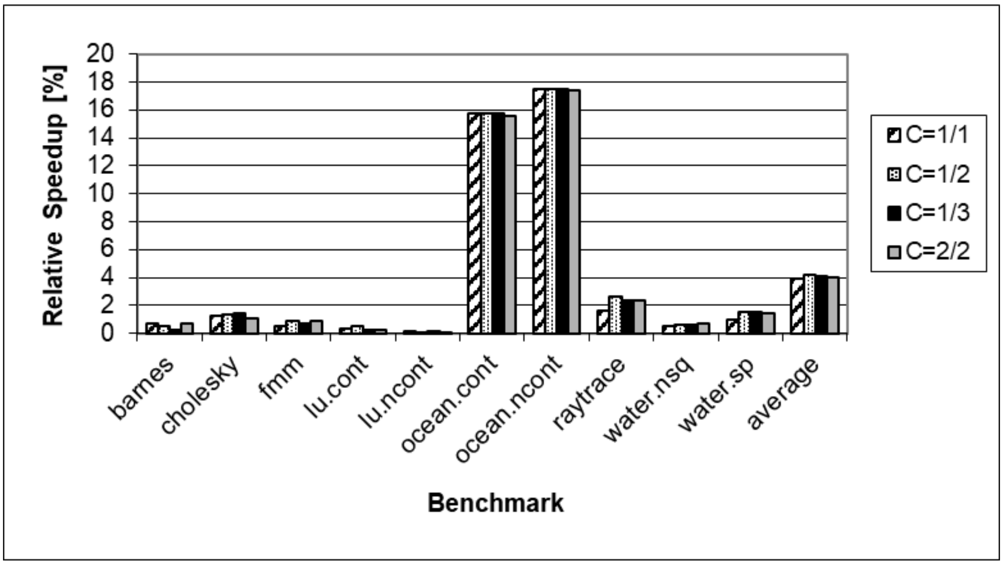


Figure 20. Varying the confidence automata

We have also evaluated the relative speedup by predicting all the critical loads without a classification into predictable and unpredictable. The performance in that case was significantly lower. Moreover, on most of the benchmarks we have observed significant performance degradation even related to the baseline architecture. Consequently, the role of the confidence automata is essential for better prediction accuracies (and thus fewer recoveries) which constitute the basis for the speedup of such speculative architectures.

Benchmarks	Loads Miss DL1 [%]	Prediction Rate [%]	Prediction Accuracy [%]	Relative Speedup [%]
barnes	4.98	3.03	82.07	0.51
cholesky	3.59	12.40	75.30	1.38
fmm	1.69	25.01	89.87	0.88
lu.cont	0.99	0.24	84.32	0.5
lu.ncont	12.57	0.13	99.79	0.09
ocean.cont	22.92	22.21	99.12	15.72
ocean.ncont	22.12	23.85	99.09	17.53
raytrace	6.01	28.30	87.00	2.59
water.nsq	1.39	22.53	88.25	0.64
water.sp	2.10	31.17	66.88	1.55
average	7.84	16.89	87.17	4.14

Table 7. Analyzing the quasi-optimal LVPT in a dual core architecture with 16 KB DL1 cache

Further we present a detailed analysis of the current quasi-optimal LVPT configuration (E=32, A=2, H=1, C=1/2) in a dual core architecture with 16 KB DL1 cache. Table 7 shows the DL1 miss rate, the prediction rate, the prediction accuracy and the relative speedup for each benchmark. As Table 7 shows, we have calculated a strong positive correlation between the relative speedup and the percentage of loads with miss in the DL1 cache and the load value prediction rate, respectively, and there is a lower correlation with the prediction accuracy. More precisely, the Pearson correlation coefficient between the relative speedup and the percentage of loads with miss in the DL1 cache is 0.9. The same correlation coefficient between the relative speedup and the load value prediction rate is 0.9, and it became only 0.54 by correlating the relative speedup with the load value prediction accuracy. Thus, our SLVP unit is the most efficient in architectures and benchmarks that involve high percentages of loads with miss in the DL1 cache (thus high DL1 miss rates) and high load value prediction rates. That is why we have obtained good relative speedups on some benchmarks (like ocean.cont and ocean.ncont), but low speedups on other benchmarks (like lu.ncont). In other words, high prediction rates involve a high number of dynamic loads on shared variables. This high number means that there is a significant communication between the threads belonging to a certain task (Splash-2 benchmark in our case). This explanation is validated by the fact that particularly the two Ocean benchmarks contain a lot of communication compared with the other benchmarks that are more computation-intensive, without so many intrinsic communication processes. Table 7 also shows some very low prediction rates which, in our opinion, might be the consequence of low load value localities.

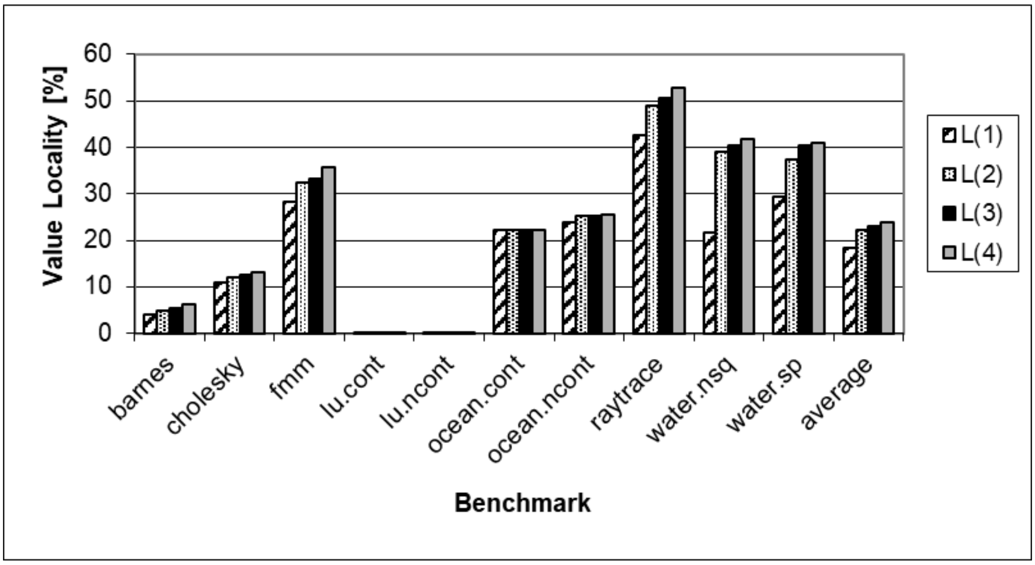


Figure 21. The value localities of the critical loads from the Splash-2 benchmarks

Lipasti et al. [134] have first introduced the value locality concept as “the likelihood of the recurrence of a previously-seen value within a storage location”. Thus, the value locality represents the upper value predictability limit. Significant value localities involve significant prediction accuracies, and vice-versa. Measurements using the SPEC’95 benchmarks have shown that value locality on load instructions is about 50% using a history of one and 80% using a history of 16 previous instances. We have measured the value locality of the critical load instructions from the Splash-2 parallel benchmarks. In Figure 21, L(H) denotes the value locality of the critical loads considering their last H distinct data values. As Figure 21 illustrates, the average value locality of the critical loads from the Splash-2 benchmarks is only 18.31% (less than the 50% from SPEC’95) considering one (the last) data value. The average value locality is very low with higher history lengths, too: 22.21% with 2, 23.04% with 3 and 23.86% with 4 data values, respectively. The value locality is less than 0.2% on the Lu benchmarks. This low value locality is an intrinsic characteristic of the critical loads from the Splash-2 parallel benchmarks and explains the low prediction rates presented in Table 7. The constant locality of the Ocean benchmarks, whose prediction rates were the highest, explains why their relative speedups are constant along different history lengths. Thus, the very low prediction rates presented in Table 7 are consequences of the low value localities of critical loads.

Next, we have analyzed the influence of the DL1 cache size over the relative speedup. Thus, we have varied this parameter in both the baseline and the SLVP-based dual core architectures. We have considered the previously established LVPT parameters: 32 entries, associativity degree of 2, value history of 1 and 1/2 confidence automata. As Figure 22 depicts, the average relative speedup is the same, about 4%, on all the evaluated DL1 cache sizes. Therefore, we will apply the next evaluations with the initial DL1 cache size of 16 KB.

Next, we have evaluated the power consumption, the energy consumption (based on (20)) for the baseline and the SLVP-based (E=32, A=2, H=1, C=1/2) architectures, both with DL1=16 KB and N=2 cores. We have also determined the relative energy reduction using formula (21), as a measure of energy efficiency (see Table 8). The power consumption is slightly higher with SLVP due to the new LVPT unit, but the energy consumption is lower (by 1.25% at average) due to the lower processing time. Next, we have evaluated the relative speedup by varying the number of cores simultaneously in the baseline and in the SLVP-based architectures.

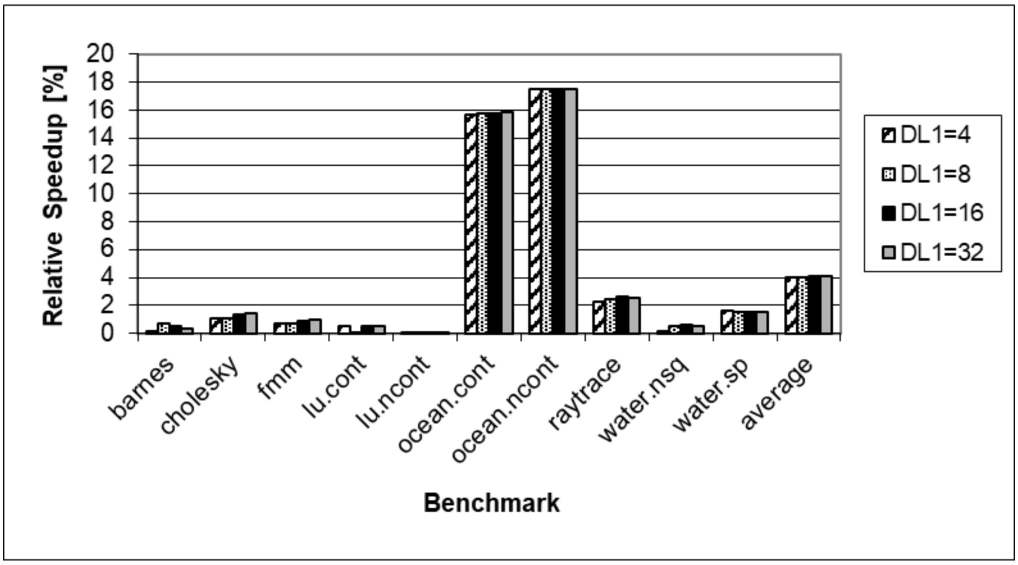


Figure 22. Varying the DL1 cache size

Benchmarks	Baseline power [W]	SLVP-based power [W]	Baseline energy consumption [J]	SLVP-based energy consumption [J]	Relative energy reduction [%]
barnes	25.4	25.8	20.22	20.43	-1.06
cholesky	40.54	41.29	2.62	2.63	-0.45
fmm	42.54	43.07	17.29	17.35	-0.35
lu.cont	47.09	47.63	7.73	7.78	-0.64
lu.ncont	43.23	43.56	7.81	7.86	-0.67
ocean.cont	24.8	27.1	63.18	58.18	7.90
ocean.ncont	25.9	28.76	65.42	59.91	8.42
raytrace	24.91	25.49	10.34	10.31	0.33
water.nsq	33.95	34.38	25.78	25.94	-0.62
water.sp	35.1	35.78	7.55	7.58	-0.36
average	34.35	35.29	22.79	21.80	1.25

Table 8. Power and energy consumption evaluation

Figure 23 presents the relative speedup averaged on all the benchmarks. The highest performance gains were obtained for less cores. As the number of cores grows, the relative speedup is lower. We have also evaluated the IPC of the SLVP-based architectures by varying the number of cores. Figure 24 presents the IPC averaged on all the benchmarks. As the number of cores increases the IPC grows only slightly non-linearly.

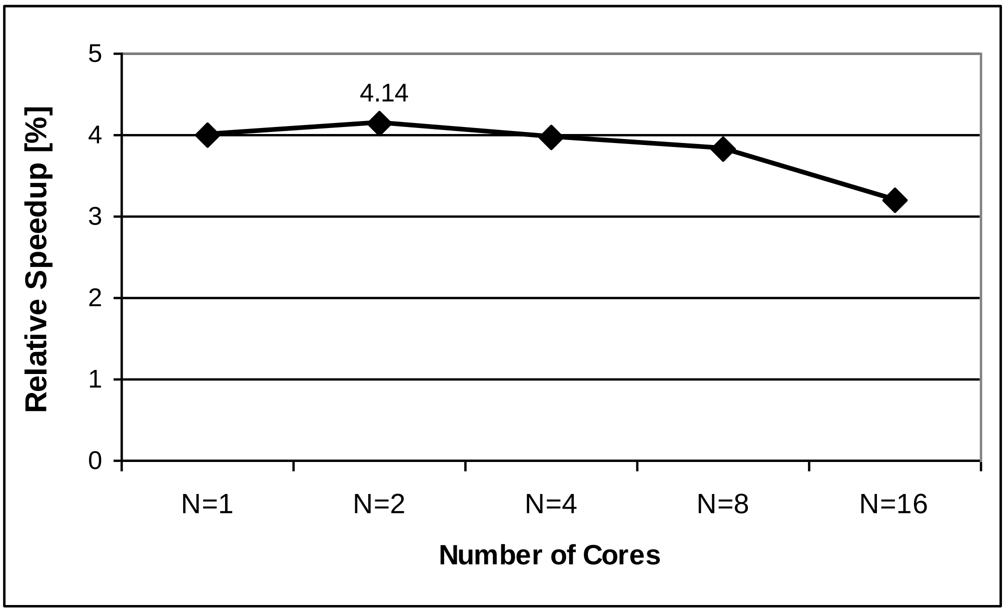


Figure 23. Varying the number of cores

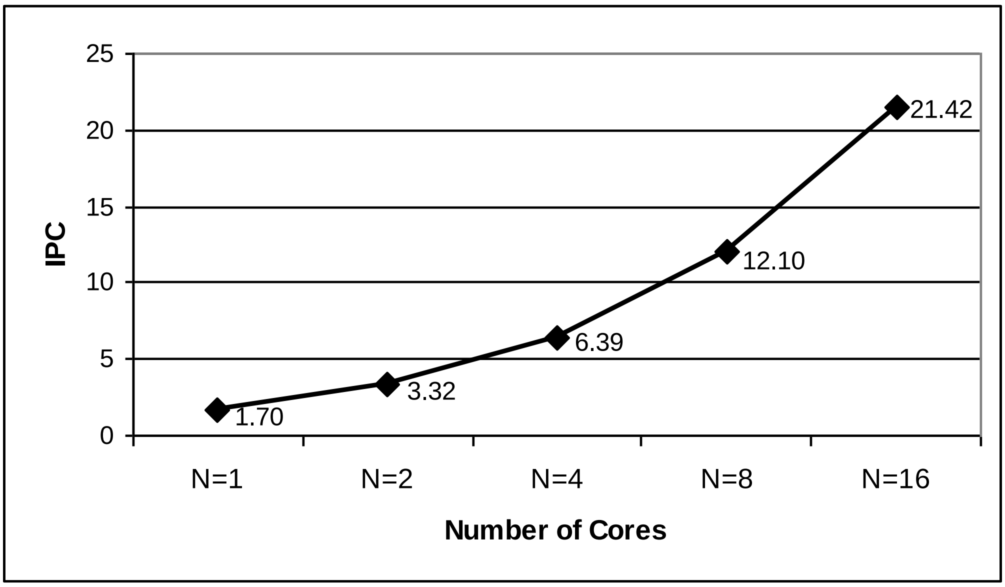


Figure 24. The IPC of SLVP-based architectures

2.3.4.2. Perceptron-Based Selective Load Value Prediction

We considered as a good starting point for our evaluations the optimal configuration obtained in the previous paragraph. Thus, we evaluated a dual-core microarchitecture with a level 1 data cache of 16 KB, and a 2-way associative LVPT of 32 entries, storing just one value per entry. We use confidence automatons belonging to the [0, 3] interval attached to the load's values, whose role is just to select for prediction the value with the highest locality. We have varied the size of the LHR, which is equal with the number of weights (W), and we compared these perceptron-based LVPT configurations with our previous counter-based method. The relative speedups obtained over the baseline architecture are depicted in Figure 25.

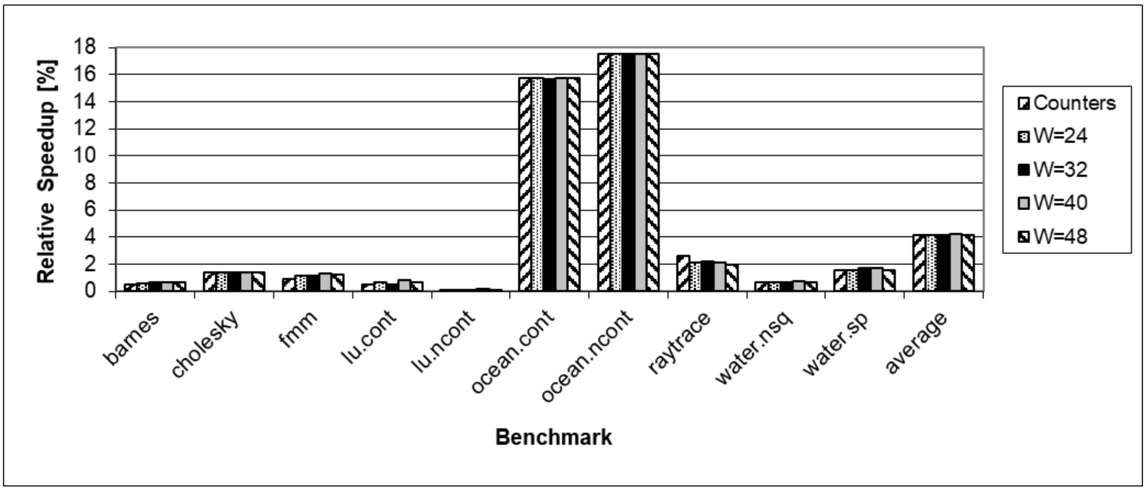


Figure 25. Comparing the relative speedups of the counter-based and perceptron-based methods

As Figure 25 shows, there are only insignificant differences between different LHR sizes. The results are in concordance with those obtained in [71]: on most of the benchmarks, the relative speedup is less than 2%, but on the *ocean* benchmarks it is over 15%. Figure 26 focuses on the averages among benchmarks.

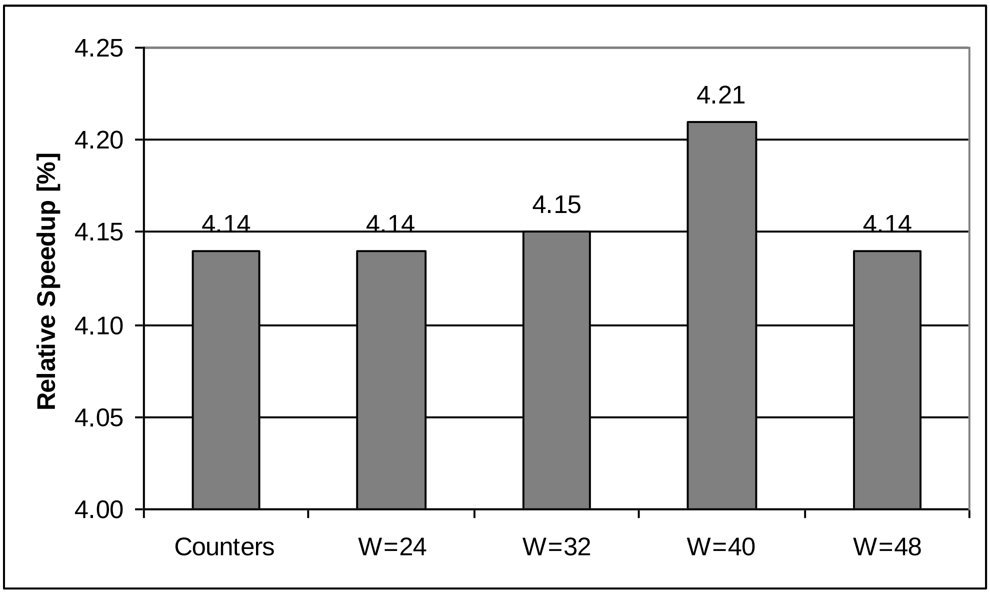


Figure 26. The average relative speedups of the counter-based and perceptron-based methods

As Figure 26 shows, the best configuration is using 40 weights, involving thus an LHR of 40 bits (load instruction behaviors). With that configuration, the average relative speedup was 4.21% which is slightly better than the 4.14% obtained previously with the counter-based method. Next, we have analyzed this best perceptron-based method in comparison with the previous counter-based method. Table 9 presents the coverage, computed as the number of correctly predicted loads divided to the number of critical loads, and also the prediction accuracy, computed as the number of correctly predicted loads divided to the number of predicted loads. The results show that both indicators are better for the proposed perceptron-based method, meaning that a higher number of loads are predicted and the prediction is provided with a higher accuracy.

Splash-2 Benchmarks	Counter-Based Method		Perceptron-Based Method (W=40)	
	Coverage [%]	Prediction Accuracy [%]	Coverage [%]	Prediction Accuracy [%]
barnes	2.49	82.07	2.55	86.85
cholesky	9.34	75.30	9.70	72.73
Fmm	22.48	89.87	24.60	86.94
lu.cont	0.20	84.32	0.15	86.63
lu.ncont	0.13	99.79	0.18	99.13
ocean.cont	22.02	99.12	22.09	99.24
ocean.ncont	23.63	99.09	23.70	99.36
raytrace	24.62	87.00	27.06	73.17
water.nsq	19.88	88.25	20.63	96.22
water.sp	20.85	66.88	26.46	91.12
average	14.56	87.17	15.71	89.14

Table 9. Analyzing the optimal configuration

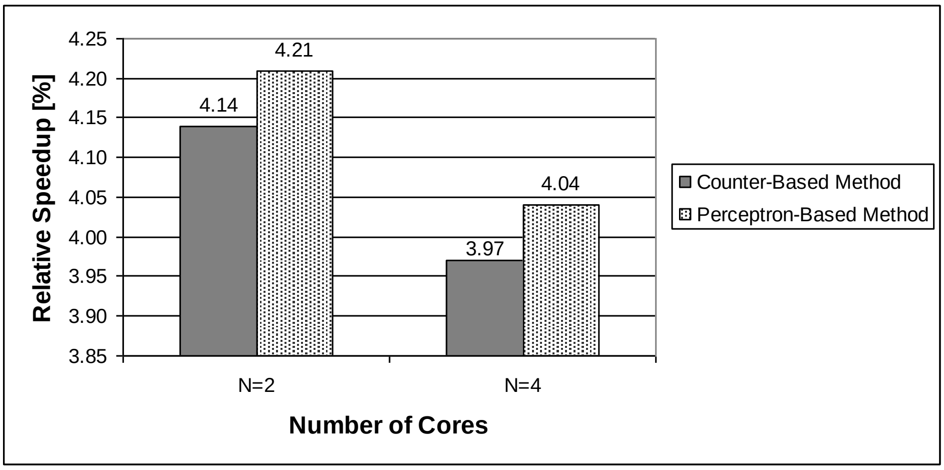


Figure 27. The average relative speedups of the counter-based and the optimal perceptron-based methods considering different core numbers

Finally, we have compared the optimal perceptron-based configuration (W=40) with the counter-based one, considering dual-core and quad-core architectures. Figure 27 presents the relative speedups obtained with respect to the corresponding baseline architectures, averaged on all the benchmarks. The relative speedup is lower as the number of cores grows, but the performance gain is at the same level.

2.3.5. Summary

In this section we have extended the Sniper multicore simulator with counter-based SLVP capabilities. Since the technique is selectively applied only on critical load instructions, a small and fast LVPT is enough to exploit the benefits in terms of performance and energy consumption. As far as we know, we are the first researchers who integrated the SLVP technique into a speculative multicore architecture and evaluated the speedup and the energy consumption reduction on native concurrent applications.

By applying a manual design space exploration of the proposed speculative multicore architecture on the Splash-2 parallel benchmarks, we concluded that the optimal SLVP-based multicore configuration is using a DL1 cache of 16 KB and a 2-way associative LVPT (A=2) with 32 entries (E=32), confidence automata having one unpredictable and two predictable states (C=1/2), and only the last value (H=1). The average relative speedup was about 4%, with a maximum of 17.53%. We have varied the number of cores and the highest performance gains were obtained for fewer cores within the simulated architecture. We have also observed a strong positive correlation between the relative speedup and the percentage of loads with miss in the DL1 cache and the load value prediction rate, respectively. Therefore, the SLVP technique is the most efficient in microarchitectures and benchmarks that involve high percentages of loads with miss in the DL1 cache and high load value prediction rates. The power consumption was slightly higher with SLVP due to the new LVPT unit, but the energy consumption was lower by 1.25% due to the lower processing time.

We have also enhanced the SLVP mechanism with hardware perceptron-based classification of load instructions as predictable or unpredictable. Thus, we have inserted into the LVPT's hardware structure two new fields, the locality history register and the set of weights, which are used to determine if the corresponding load instruction is predictable or not. The goal of the load value predictor is to anticipate the values of critical load instructions and to unlock in a speculative manner the subsequent RAW dependent instructions' execution. Since each misprediction implies a recovery process, we are interested in obtaining a high prediction accuracy. Therefore, we predict only the results of critical load instructions identified as predictable by our developed perceptron. In the experimental process, we have varied the size of the perceptrons between 24 and 48 bits. The results have shown that the best configuration implies 40 weights within each perceptron. With the dual-core configuration, consisting in a level 1 data cache of 16 KB and a 2-way associative LVPT with 32 entries and one value per entry, we have obtained an average relative speedup of 4.21% over the baseline architecture (the same configuration, but without LVPT), with a maximum speedup of about 17% (on the *ocean.ncount* benchmark).

As a further work, we propose to generalise the implemented value prediction method to Intel Nehalem's all long-latency machine instructions (multiplication, division, square-root, etc.), through some dedicated hardware value predictors. In this way, the obtained speedups would be, for sure, much more significant. Also, we intend to further apply an automatic multi-objective optimization method for our speculative multicore system, based on our already developed complex and effective software optimization tools [208] and [60]. Obviously, this automatic design space exploration will provide a better optimal solution, but, due to the enormous design space, not certainly the ideal one.

3. Prediction-Based Assembly Assistance Systems

European jobs involving manual work represent the largest category in the manufacturing sector [28]. Thus, adequate training for manual work plays an important role to enable efficient production processes in the global market. The development of training tools based on information technology will provide an effective solution for a company's competitivity.

Many factories avoid the full automation due to either the flexibility of the human operators or lower production costs. Nowadays, machine-assisted human-centered manufacturing is the common approach. On the other hand, in the absence of full automation, modern factories might adapt to the workers profile for a cost-effective and resource-efficient production. The training stage of the workers can be more efficient if assembly assistance systems are used instead of human trainers. A certain level of information assistance can help to structure, guide and control manufacturing processes [2]. Smart assembly assistance systems can improve the overall performance, and simultaneously reduce the skill requirements of the workers [127]. The automation might be designed and implemented in an adaptive manner, since the adaptability of the automation can mitigate some of the costs of human-machine interaction, such as unbalanced mental workload [103]. The dynamic configuration possibility of automation levels is another requirement [181].

In this chapter, based on our work published in [84], [85], [86] and [87], we describe an automated assembly assistant system which supports the human workers by suggesting the next possible manufacturing steps and by detecting eventual wrong steps. We are interested in an accurate prediction technique which can best model the behavior of the workers. The most efficient predictor will be physically integrated into our assembly assistance system. The final adaptive human-centered training station will be able to receive real-time information about the worker and use it to suggest the manufacturing steps. Our training station will allow the following functionalities: capability of recognizing through sensors the product components and human features and actions, ability of learning patterns and correlate human operator contexts with the assembly states of a certain product, possibility of assisting the trainee in correct product assembly (either by recommending the next step or by detecting wrong steps), capability of connecting the relevant data systems for an easy training set-up. The proposed context-aware assistive system can replace the trainers during the training stage of unexperienced or new workers, but it can be useful even for experimented workers. We chose as target product a customizable modular tablet, but the system can be easily adapted to any other product.

3.1. Assistive Manufacturing System using Two-Level Context-Based Predictors

Assembly assistance systems designed for monotonous manufacturing tasks should not over-challenge or under-challenge the worker, it might adapt to the needs of the human operator in real-time [61] and to the constraints of the task [194]. Moreover, it must consider the skills and a possible functional decrease of the human worker's capabilities [168]. Thus, interactive and context-aware instructions during the assembly processes become more and more important [62].

In this section we present an adaptive assembly assistance system able to dynamically adapt the production process to the worker's actual condition, his/her general characteristics, preferences and behaviors in assembling products. This human-oriented assembly assistance system is using a two-level context-based predictor to recommend the next assembly step based on the current state of both: the semi-product, and the worker. From the experiments performed with 68 trainees, we extracted the relevant correlations between certain human characteristics and assembling behaviors. The obtained dataset was subsequently used to train and evaluate different predictors.

3.1.1. Related Work

Simple and hardware-efficient predictors have been successfully used for branch prediction in the microprocessors' domain. A good description of these prediction structures was provided in [25]. The authors have shown that the two-level predictors are simplifications of PPM, being in fact Markov predictors. Advanced branch prediction schemes were presented in [55]. Some of the branch predictors have been adapted and successfully used in ubiquitous systems, like person movement prediction in smart office buildings [169]. In our opinion, these simple and efficient prediction schemes are appropriate for our needs and, therefore, we adapt them in this work to be usable for assembly assistance. Their detailed description is provided in the next section.

In [170], a comprehensive view of a cyber-physical system is provided, by showing that even simple automated systems have the key features of an anthropocentric cyber-physical system: integrality, sociability, locality, irreversibility, adaptivity and autonomy. The interaction-based architectural design of a manual assembly module is detailed. The augmented cognition feature is addressed through an integrative engineering approach in designing cyber-physical production systems. Their proposed reference architecture can be applied as a template for the architecture of cyber-physical production systems.

Worker and assembly context adaption in assistance stations are researched also from the worker's cognitive perspective. In [46], the authors analyzed the feasibility to use biodata from sensors within assembly assistance systems for manual operations that can adapt to the workers' cognitive state. Moreover, in [61] an assembly assistant system is presented revealing the challenges and importance of cognitive-feedback by using activity recognition for context-awareness in order to keep an appropriate cognitive load (i.e., challenge level). An interesting review of the research challenges in the product assembly domain is presented in [196]. It synthesizes the integration, design and collaboration issues of recent anthropocentric approaches.

3.1.2. Prediction-Based Assembly Assistance System

The assembly assistance system depicted in Figure 28 is a prototype designed to enable execution of a broad-spectrum of training scenarios for manual operations, from the simplest ones (e.g., predefined instructions and training flow) to complex ones (e.g., adaptive instructions based on context-awareness and real-time data from biosensors).

From a hardware perspective, the assembly assistance system consists of the following components: lower frame with embedded electrical motors for easy table height adjustment, large dimension touch screen integrated in the table to display instructions including also microphone and speakers to support voice interaction, upper frame enabling flexible mounting of sensors or bins for instruments or components, sensor (e.g., Kinect 360, Kinect Azure) to detect body segment's movement as well as facial expression, and sensor (e.g., 3D camera) to detect objects and hand movement. Additionally, wearable biosensors to estimate human states or emotions can be used to adapt the training experience.

In order to be able to recommend the next step of a manual assembly process, our system must recognize the current context consisting in the previously assembled components, but it might also correlate with the current features of the human operator. Since in this work we focus on a customizable modular tablet, we must codify all its possible states. The tablet consists in eight components (see Figure 29): a mainboard, a screen and six modules which can be speakers, flashlights and power banks.

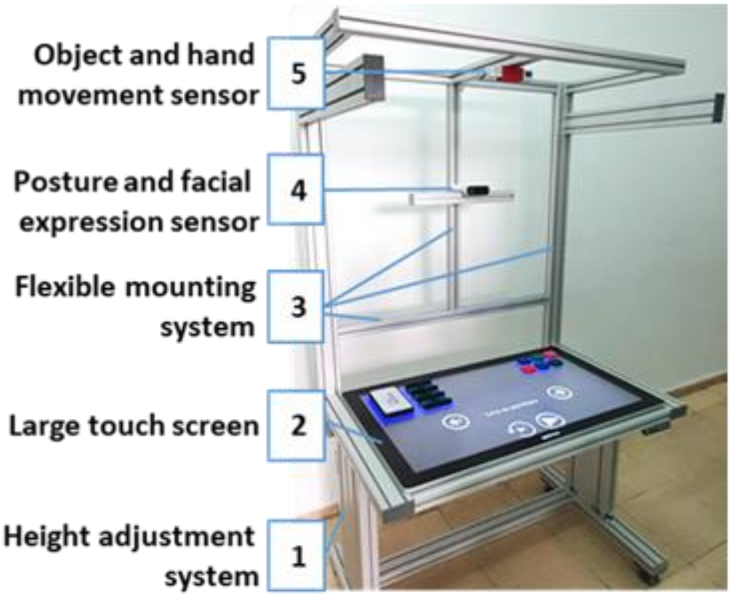


Figure 28. Assembly assistance system set-up example

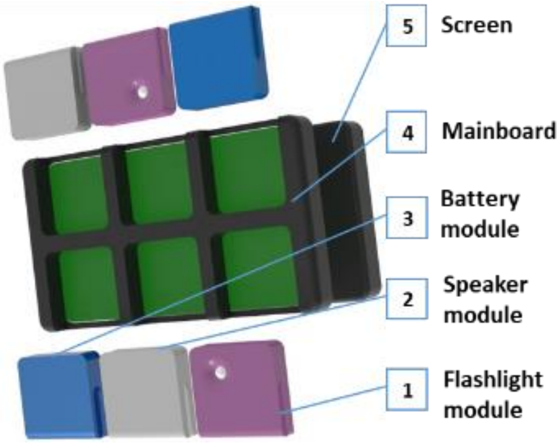


Figure 29. Customizable modular tablet

Thus, seven steps are necessary in a correct assembly process of such a table. We have chosen a binary codification (usable as decimal, too). The mainboard is the main component and the reference for the other assembled components. If a component was placed in the mainboard and it was in its correct slot, it was marked as 1. If no component is attached to the slot or the component is in the wrong slot, it is marked with 0. Because there are seven slots on the mainboard, there is a 7-bit representation of the tablet's assembly progress. The numbers from Figure 30 represent the bits' positions that can be activated or deactivated, 6 being the most significant bit and 0 the least significant bit.

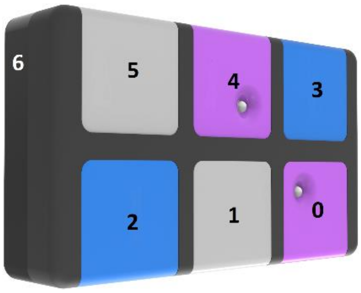


Figure 30. The codification of the customizable modular tablet

Based on the code of a certain state, we can anytime determine which components were already assembled. By comparing the codes of two consecutive states, we can determine which component changed between them.

3.1.3. Two-Level Context Based Predictor

The assembly process codified through the above-described method can now be modeled using a two-level context-based predictor. The first level consists in a left-shift register containing the last R states, R being the order of the predictor. The second level of the predictor is a pattern history table with two columns: the pattern and the state. Each entry of the table contains a context of R consecutive assembly states in the pattern column and the corresponding next state in the state column. The structure of the two-level context-based predictor with simple states is presented in Figure 31.

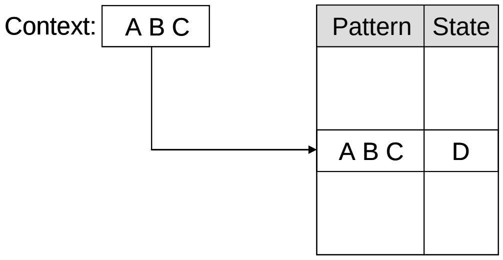


Figure 31. The two-level context-based predictor of order 3 (R=3)

In the example depicted in Figure 31, C is the current state, A and B are the previous two states, thus A, B and C forming the current context, whereas D is the predicted next state. In the learning stage, the predictor is populated with the occurring contexts and their corresponding next states. After the learning stage, the stored data can be used for prediction during the assembly process. As an example, for the sequence ABCDABABC, the prediction with a third order two-level context-based predictor would be D, since after the context ABC we have last seen D in the past. In our application, the states are the 7 bits long decimal codes reflecting the assembly stage of the tablet, as we described above. Thus, a certain bit on 0 means that the corresponding component is not yet mounted, or a wrong component is mounted there, and 1 means correctly mounted component.

The two-level context-based predictor presented in Figure 31 can be enhanced with two-state automata. The automata provide more stability with respect to the variations in the observation sequence. A two-state automaton has a weak state and a strong state. One bit is necessary to store the additional information: a logical 0 if the state is weak or a logical 1 if it is strong. When an observation occurs again and the automaton is in weak state, it will change to strong state. When the same observation occurs again and the automaton is in strong state, it will remain in the strong state. When different observation occurs and the automaton is in weak state, it will store the new observation in a weak state. When different observation occurs and the automaton is in strong state, it will keep the old observation but in a weak state. The structure of the two-level context-based predictor with two-state automata is presented in Figure 32.

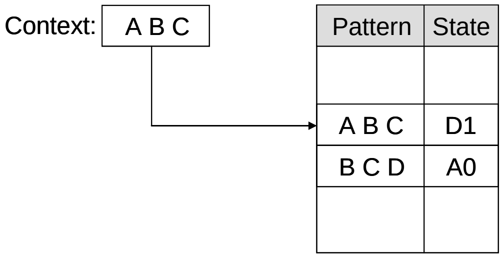


Figure 32. The two-level context-based predictor of order 3 (R=3) with two-state automata

If we consider the observation sequence ABCDBABCDABC, the BCD pattern is followed first by B (the stored weak state is B0) and it is also followed last time by A (B0 is replaced with the weak state A0). The ABC pattern is followed first by D (the stored weak state is D0) and is followed last time by D again (the state is changed to the strong D1). In Figure 32, ABC is again the current context, and the prediction is D. In our application, the states are 8 bits long decimal codes in which the first bit reflects the weakness (0) or the strongness (1) and the rest of the bits are describing the assembly state of the tablet (0 meaning unmounted or wrongly mounted component and 1 meaning correctly mounted component).

3.1.4. Experimental Methodology

The 68 participants (2nd year BSc students) in our experiment, had to freely assemble, without any guidance, the earlier mentioned customizable modular tablet (presented in Figure 29). When the subjects entered in the room, they listened to an audio message stating that the tablet should be assembled using all the components as indicated in the images and only after the audio finished playing, they could start assembling. During the whole process they could only view the images illustrating how the different components of the tablet should be assembled. The recordings were saved and transformed into an encoding that could be utilized in our predictors implemented in C#. Right now, the dataset is composed of correct and incorrect assembly steps and disassembly steps as well. The training and testing dataset had to be extracted. For the predictor's training, only correct assembly steps were selected, while for testing all the unrepeating steps have been selected.

The evaluation metrics are the prediction rate, the coverage, the error detection indicator and the prediction accuracy. The prediction rate is the percentage of predictions with respect to the total number of assembly steps. The coverage is the percentage of correct predictions with respect to the total number of assembly steps. The error detection indicator is the percentage of correctly detected assembly errors from the total number of errors (occurred when the real state and the prediction were different). Finally, the prediction accuracy is the percentage of correct predictions with respect to the total number of predictions.

3.1.5. Experimental Results

First, we evaluated and compared the two-level context-based predictor with simple states (depicted in Figure 31) and the one with two-state automata (presented in Figure 32).

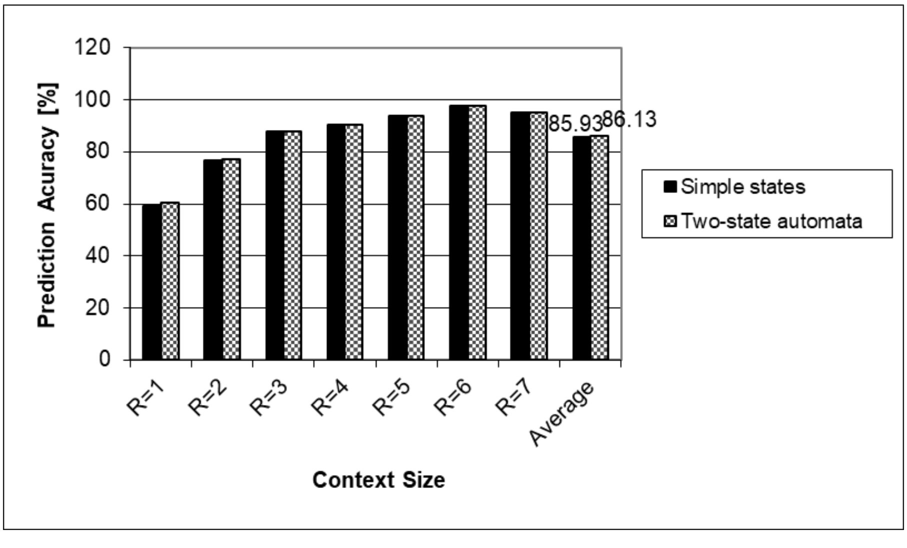


Figure 33. Prediction accuracy of the two-level predictors by considering different context sizes

Figure 33 presents the prediction accuracy obtained with these two predictors considering different context sizes. We can observe that these two compared predictors have about the same accuracy. Only on the context sizes 1 and 2 the predictor with two-state automata is slightly better. On the higher context sizes both predictors provide the same accuracy. We can also observe that the accuracy is increasing up to a context size of 6. The reason of the accuracy decrease starting with the context size of 7 is that such long contexts are rarely matched during the assembly process, which is shown by the extremely low coverage (see Figure 34).

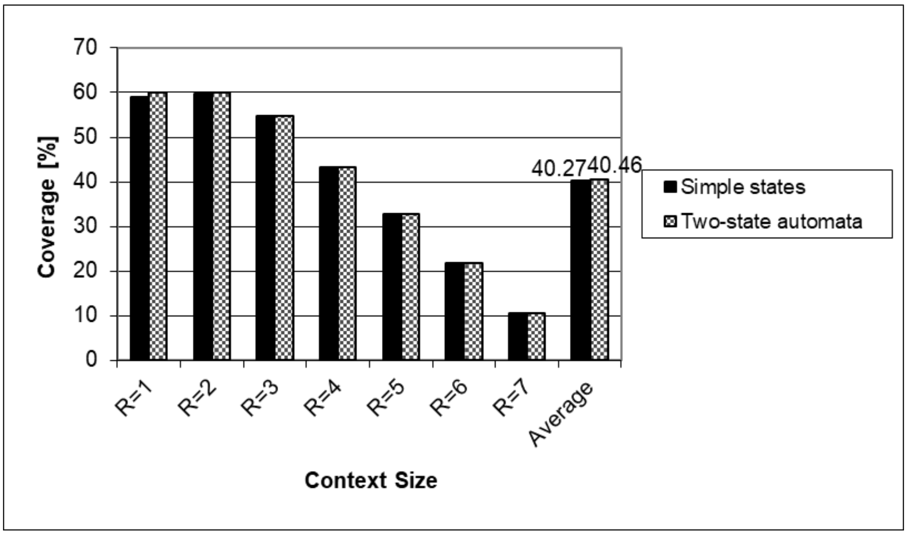


Figure 34. The coverage of the two-level predictors by considering different context sizes

We can see that both predictors have almost the same coverage. As we expected, as higher is the context size, as lower is the predictor’s coverage. The chances are good to find short patterns, but low or very low to find long patterns. A limitation is introduced by the fact that only seven steps are necessary to correctly assemble the tablet. Analyzing Figures 33 and 34, we can conclude that the two-state automata introduced an insignificant improvement. Thus, the simpler prediction scheme proved to be more efficient.

Next, we present the error detection capability of the analyzed predictors. This is an important indicator, since we are going to use such predictors to detect also possible wrong assembly steps beside the useful indications they can provide during the manufacturing process.

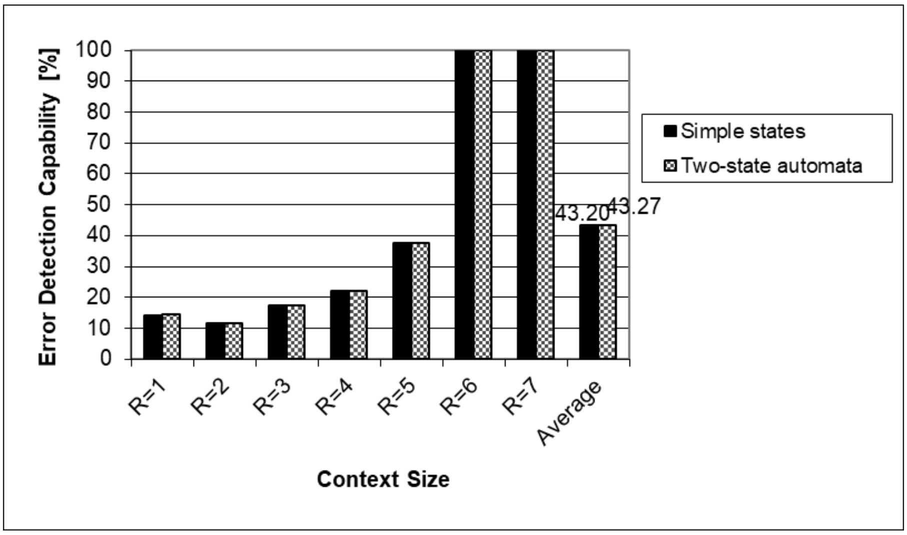


Figure 35. Error detection capability of the two-level predictors with different context sizes

As Figure 35 shows, the error detection capability is quite good for both analyzed predictors. For long context sizes (6 and 7) it reaches even 100%. However, we would like to see such good error detection capabilities for shorter contexts, since they can assure better coverage and higher prediction accuracy.

3.1.6. Summary

Context-aware assistive systems and collaborative robots can support people to manage the increased variability and complexity of products while reducing the number of errors. These systems can potentially provide a diversity of dissimilar opportunities for manufacturing, such as training employees with less experience or even eliminate the need for training, quality assurance, reducing the cognitive complexity associated with assembly tasks, and integrating elderly and disable people into the workplace. Although there are various opportunities to employ advanced technologies such as augmented and virtual reality, artificial vision and biosensors, a user-centered approach is critical to the success of these systems. Beside the specific context of the assembly task and the manufacturing environment, assistive systems should adapt the instructions in real-time based on the workers' psychomotor capabilities. This becomes even more obvious in mixed-initiative human-robot collaboration where the proper allocation of tasks between humans and robots requires great flexibility to achieve optimal joint human-robot performance. It includes models and evaluation criteria for the optimal task-allocation between human and robot in a dynamic environment (i.e., the stochastic nature of the manufacturing process and models of human status and performances).

Pattern recognition in the manufacturing process is an essential enabler to provide the assembly assistance functionalities. Although prediction methods have been used in many domains, to our knowledge there is no research to predict the behavior of human operators in manufacturing assembly tasks. In this section, we have analyzed the possibility of using two-level context-based predictors as assembly assistance of the manufacturing operators. The evaluation results shown that the optimal prediction scheme was the two-level context-based predictor with simple states, which can be either used to assist the trainees during their learning stage or to help the workers by detecting wrong assembly steps. The training stations enhanced with such predictive capabilities allow to replace a fixed and thus static assembly sequence with a dynamic one which is adapted to each human operator, providing flexibility and efficiency. Our approach fits well the widely applied machine-assisted human-centered manufacturing paradigm.

In the next section, we will develop and evaluate some more complex prediction schemes, like Markov chains or the PPM algorithm. We will choose the most efficient prediction method to be integrated into our smart assembly training station. Until now we did not find any significant correlation between the assembly style and the worker's profile. This issue requires further investigations with an extended set of experiments over different user types (i.e., personas).

3.2. Decision Support System with Stochastic Models

Due to the flexibility of the human operators, many factories avoid the full automation and apply machine-assisted human-centered manufacturing solutions. Human trainers can be replaced with smart training stations to train the workers, making thus possible their easier relocation. Moreover, assembly assistance systems can be useful also for experimented workers, especially in detecting wrong assembly steps, but it can decrease monotony, too, by adapting the assembly process to the workers' needs and styles. Thus, particularly under frequent changes in work order, smart assembly assistance systems can highly increase production efficiency and reduce the error rates.

In this section, we propose an adaptive assembly assistant system which is using Markov predictors or PPM to recommend the next assembly step. First, the predictors are trained with correct assembly patterns. After the training stage, the predictors can be used to determine the next assembly step based on the current context consisting in the worker's last assembly steps. We evaluated the predictors and compared them with others on a dataset collected during an experiment performed on 68 inexperienced manufacturing operators. The experimental methodology was presented in 3.1.4. The final goal is to physically integrate the best predictor into our smart assembly station to avoid uncertainty and mental stress for workers by recommending possible steps for assembling the product.

3.2.1. Related Work

Assembly assistance systems can increase manufacturing efficiency and can reduce the skill requirements of the human workers [127]. Such systems should be adaptive in order to mitigate some human-machine interaction costs, like the unbalanced mental workload [103]. The adaptability can be assured by interactive systems able to provide context-aware instructions to the workers during the assembly processes [62]. In [126], the gamification is used to achieve a mental state in which a worker is fully immersed in activity, with energized focus and the belief in the success of that activity.

In [84], we evaluated the first predictors for our context-aware assistive system. The two-level context-based predictor uses a left-shift register in its first level and a pattern history table in the second level storing pairs of assembly patterns and recommended next assembly steps. The predictor is considered of order R if the left-shift register from the first level and the patterns from the second level table are composed of R consecutive assembly states. An improved version uses two-state automata for better stability to possible changes in the assembly sequence. Each next assembly state stored in the pattern history table has associated such an automaton on one bit which can be in weak state (0 logic) or strong state (1 logic). When a certain assembly pattern is followed again by the same assembly state, the associated automaton changes to 1 if it was 0 or remains 1 if it was 1. When a different assembly state occurs and the automaton is 0, it will store the new assembly sate and the automaton remains 0. When different observation occurs and the automaton is 1, it will keep the old assembly state and the automaton switches to 0. The experimental results have shown that this more complex scheme with the automata cannot provide higher prediction accuracy. Another improvement was to store multiple possible next assembly states for each pattern within the pattern history table. This scheme could not increase the prediction accuracy, since the predicted next assembly state was always the last state seen after the current assembly pattern, but it could significantly increase the error detection rate.

In [180], the authors present a human-machine centered assembly station and a case study in a mini-factory laboratory, switching from a manually production to a hybrid assembly system combined with a lightweight robot. The safety risks of the workers have been evaluated, also providing the appropriate measures. The presented mini-factory laboratory is equipped with devices for manual assembly and devices used for automated or hybrid assembly. The manual workstations have flexible plug-in system of tubular frames and tables equipped with electric screwdriver systems and grab containers. Other elements are the lean Kanban flow racks to apply material commission. The laboratory has software systems and several robots usable for automated assembly demonstrations. The case study with manual assembly consists in simulating a manual assembly of pneumatic cylinders by groups of students. As a next goal, the authors extended this case study with human-machine interaction by developing a safety-oriented hybrid workstation for assembling pneumatic cylinders.

In [96], the authors developed a virtual training system for the automotive manufacturing domain. They proposed an automated virtual training technique, integrating the relevant planning data of the IT structures within digital factories. The goal was to increase the production process transparency for the human operator and to allow fast adaptation to new manufacturing requirements. The cost-effective scalable hardware-setup relies on game-based user interaction and, therefore, it is widely accepted by the users. The trainees can acquire relevant knowledge regarding the involved components, as well as assembly positions, modalities and sequences. The learning process consists in three stages: an easy mode for familiarization, a medium mode and finally a hard mode when the appropriate components must be actively chosen. The knowledge is further strengthened through specific games. The automated training content is generated by an interoperable information interface which combines heterogeneous enterprise data from planning processes in a unified information model, used as input by the virtual training module. In [199], the authors analyzed the impact of the virtual reality learning procedures over the workers' training process and concluded that the virtual simulation preceding the real assembly improved the score of correctly assembled modules and shortened the assembly time.

In [138], the authors proposed a semantic service discovery and ad-hoc orchestration system, adaptable to context changes. The processes are generated taking into account the current structure of the production plant and the ability of the field devices to apply semantic discovery and service selection. Their context-based orchestration framework is composed of three layers: service registration, service discovery and selection and service orchestration. The authors developed a structure of modular ontologies used for semantic reasoning in matching and selection processes. The modeled ontologies are then used for the semantic description of the web services provided by field devices. The service with the highest score, weighted among several matching criteria, is selected. The process is then decomposed in atomic processes and the corresponding services are invoked for the resulting composite process in order to control the manufacturing of the current product. In contrast with the above-described works, our assembly assistance system relies on its prediction capability for the next assembly step, and consequently detecting the wrong ones.

Markov models were used as stochastic models and proved to be efficient in several different areas like computational biology [192], web access mining [80], image processing [76], energy management systems [73], etc. In our opinion, this powerful prediction method can be efficiently integrated into smart assembly stations and used to support the manufacturing processes by providing the next assembly step. Therefore, in this section, we extend the prediction scheme presented in [84] with frequencies associated to each possible next assembly state. The pattern history table keeps the evidence of state-frequency pairs for all the assembly contexts encountered in the training stage. Then, during the assembly process, the predictor will recommend the next assembly step as being the most frequently seen state after the current assembly context. Thus, the proposed scheme is in fact a Markov predictor. Context-aware assembly assistance systems have been presented in [46] and [61], but they are not relying on prediction, nor on Markov chains.

3.2.2. Assembly Assistance using Markov Predictor

The manufactured product used in this work is a customizable modular tablet composed of eight components, as it was described in paragraph 3.1.2. Seven steps, in no predefined sequential order, are needed to correctly build up such a tablet. In order to be able to model the assembly process of the tablet using Markov chains, we codified all its possible states as we explained in 3.1.2.

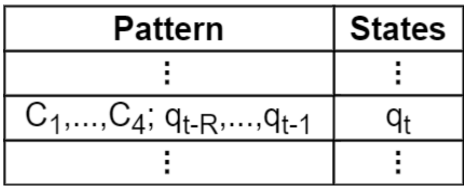
In Markov chains of order 1, the current state depends on the previous state. Such a model can be described by equation (22):

$$P[q_t \,|\, q_{t-1},\ ...,\ q_1] = P[q_t \,|\, q_{t-1}] \tag{22}$$

where q_t is the state at time t. In Markov chains of order R, the current state depends on R previous states. Such a superior order Markov chain is described by equation (23):

$$P[q_t \mid q_{t-1},\ ...,\ q_1] = P[q_t \mid q_{t-1},\ ...,\ q_{t-R}] \qquad (23)$$

In our application, the states are the assembly stages of the tablet. We implemented the Markov predictor as a two-level prediction scheme [85]. Figure 36 presents the structure of a Markov predictor of order R. The patterns in this work are composed of human worker characteristics ($C_1,...,C_4$) and assembly sequences ($q_{t-R},...,q_{t-1}$). The next state can be predicted if the current context is found in the pattern field of the table and then the state q_t with the highest frequency from that entry will be the prediction.



Pattern	States
⋮	⋮
$C_1,...,C_4;\ q_{t-R},...,q_{t-1}$	q_t
⋮	⋮

Figure 36. R^{th} order Markov predictor

The first level is the context built up as a left-shift register containing the previous R states of the tablet, and it is updated after each assembly step. The second level is a pattern history table composed of the Pattern and States fields. Each entry of the pattern history table contains a certain pattern of R consecutive assembly states in the Pattern column and a list of possible next states together with their frequencies in the States column. The frequency associated to a certain state is determined in the training stage as the number of occurrences of that state after the considered pattern. After the training stage, based on this information, the predictor can provide the most probable state in any assembly context. An example with a 3rd order Markov predictor is presented in Figure 37.

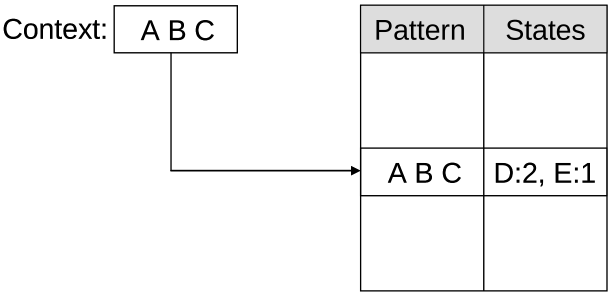


Figure 37. The Markov predictor (R=3)

If we consider the ABCDABCDABCEABC observation sequence, after the pattern ABC we have seen the state D twice and the state E once. These two states are stored together with their frequencies in a list associated to the ABC pattern from the prediction table. If we use the trained predictor to suggest the next state after the pattern ABC, it will provide the state D, because it has the highest frequency. We have implemented the pattern history table as an unlimited dictionary storing key-value pairs. The keys are integer values representing the patterns. The values associated to the keys are lists of next possible states stored as integers together with their frequencies which are also integers.

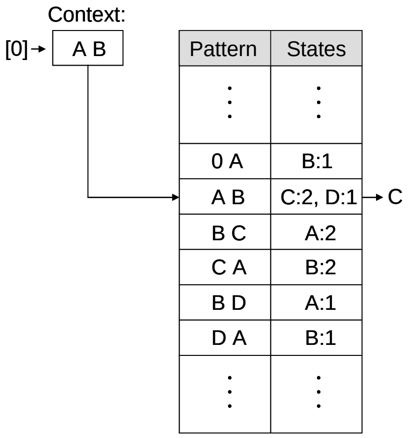
3.2.3. Markov Predictor with Padding

As we explained in the previous section, in superior order Markov chains, the current state depends on a limited number of previous states (see equation (23)). In a Markov chain with padding, equation (23) additionally has the following property:

$$q_{t-R} = \begin{cases} q_{t-R}, \ R < t \\ 0, \ R \geq t \end{cases} \quad (24)$$

meaning that if there are not enough states to build up the pattern, then zeros are used instead of the missing states. This padding mechanism makes possible to use the R^th order predictor even if we do not have yet the entire context of length R. The predictor can be designed as a table having two columns [86]: the first one for the patterns and the second one for the list of possible next states and their frequencies. The prediction table is indexed with a left-shift register containing the context or zeros for the missing context states. The state having the highest frequency within the selected entry is provided by the assembly assistance system as the recommended next manufacturing step. The prediction table was implemented as an unlimited dictionary containing pairs of patterns and their corresponding unlimited lists of possible next states.

An example with a 2nd order Markov predictor is presented in Figure 38. If we consider the state sequence ABCABCABDAB, the predictor is trained with all the extracted pairs consisting in patterns of size 2 and their next state. For the incomplete patterns the padding mechanism is applied.



Pattern	States
⋮	⋮
0 A	B:1
A B	C:2, D:1
B C	A:2
C A	B:2
B D	A:1
D A	B:1
⋮	⋮

Figure 38. Example of 2nd order Markov predictor with padding

Thus, the first pattern 0A is followed once by the state B. The pattern AB is followed twice by C and once by D. In the same manner, the pattern BC is followed twice by A, CA is followed twice by B, BD is followed once by A and DA is followed once by B. Now, at the end of the sequence we are interested in the next state prediction. The current context of size 2 (extracted from the end of the state sequence), in this case AB, is used to index the prediction table. Therefore, the entry having the pattern AB will be selected. From the associated list of states, the state with the highest frequency is provided as the predicted one. Thus, after the exemplified sequence, the system will recommend C as next state. Obviously, in our assembly assistance system, and implicitly in the predictor, the states are represented by values (generated through the codification proposed and used in [84]). Optionally, the predictor can be configured to generate multiple choices for the next assembly step, in descending probability order.

Since in our application the state sequence is very fragmented (a correct manufacturing process takes only 7 steps), the padding mechanism is very useful at the beginning of each new assembly procedure. Thus, through padding, the system will be able to predict more often.

3.2.4. Prediction by partial Matching

As we described in [87], a PPM of order R tries to provide a prediction with the Markov predictor of order R and, if it can do that, its prediction is returned by the PPM. Otherwise, the order of the Markov predictor is iteratively decremented until a prediction can be provided. If the Markov predictor of order 1 cannot issue a prediction, then the PPM itself is not able to do that. The PPM's prediction mechanism is depicted in Figure 39. For the PPM implementation, the Markov predictor with padding (presented in the previous section) is used. The Markov predictor was enhanced to also use the workers' characteristics. Upon instantiation of the PPM algorithm, the order R should be provided as parameter. For each order, starting from R and moving down to 1, a Markov predictor is created. After all the R Markov predictors are created, they are sorted in descending order. When the prediction algorithm is called, it will iterate sequentially through the Markov predictors to find one which has a match for the assembly pattern. If the Markov predictor of order R has no match, then the next Markov predictor from the sorted list is checked. The first Markov model that has knowledge of the assembly pattern is the one that will make the final prediction. If no Markov model can find a matching sequence, then the PPM algorithm is going to return –1, meaning that it cannot make a prediction. As far as we know, PPM has not been used for next assembly step prediction before. The novelty of this method consists in the combination of different order Markov predictors, which ensures that matches for the occurring assembly patterns can be found easier.

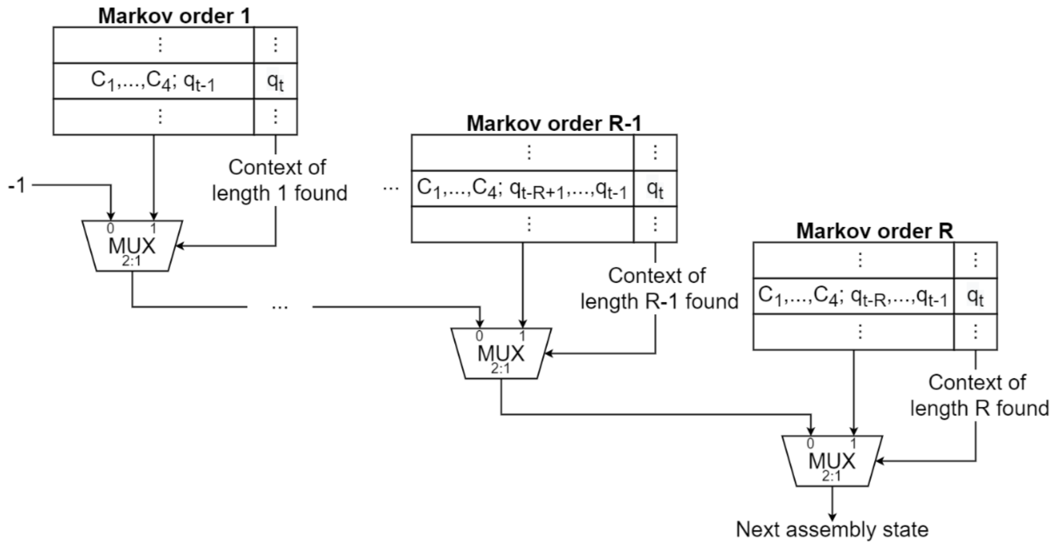

Markov order 1
⋮
$C_1,...,C_4; q_{t-1}$
q_t
Context of length 1 found
-1
0
1
MUX 2:1
...
Markov order R-1
$C_1,...,C_4; q_{t-R+1},...,q_{t-1}$
q_t
Context of length R-1 found
0
1
MUX 2:1
Markov order R
$C_1,...,C_4; q_{t-R},...,q_{t-1}$
q_t
Context of length R found
0
1
MUX 2:1
Next assembly state

Figure 39. The prediction mechanism of the PPM

There are over 5000 possible ways of assembling the tablet and 4 characteristics that can define the behavior of a worker, each one with 2 possible outcomes. Thus, the tablet could have over 80,000 unique assembling possibilities. Because of the high diversity of assembly patterns, the current context cannot always be found in the prediction table of the prediction scheme presented above, which negatively affects the prediction rate. Therefore, an enhanced scheme was considered, which explores neighboring characteristics of the user, whenever the current context (with the actual user characteristics) does not have an exact match. This approach would be practicable even if several additional characteristics were considered.

Neighboring context exploration involves changing one characteristic of the worker at a time. A single trait of the worker is changed sequentially at a time, after which making a prediction is attempted. For example, if the user is a tall male wearing eyeglasses that slept well the previous night, and there is no match for this combination, predictions will be made by varying his characteristics (one at a time), obtaining four different neighboring states. In one of them, the gender is the changed variable, so that a prediction for a tall female wearing eyeglasses that slept well the previous night is made. Afterwards, the variable for height will be changed, then the one related to whether the worker slept well, and lastly the one related to wearing glasses. This approach might yield up to 4 predictions, given that the model has knowledge about the neighbors. The step that was predicted the most from the neighboring states will be considered the next assembly step (majority voting). Ties are resolved by selecting the most frequent prediction, having the lower index in the prediction list. If the model has no match for the assembly state paired with the user's characteristics, nor with neighboring characteristics, then no predictions can be made.

3.2.6. Experimental Results

3.2.6.1. Markov Predictor

Next, we present the evaluation results of the proposed Markov predictor in comparison with the two-level context-based predictor presented in [84]. Other assistive assembly systems are not considered in this comparative evaluation, since none of them are relying on prediction. The evaluations were performed on the dataset obtained through an experiment involving 68 trainees assembling without any guidance the customizable modular tablet presented in the previous section. The experiment was presented in 3.1.4. After we trained the predictors on some extracted correct assembly patterns, we tested them on the collected data. The training set consists in 357 assembly states and the dataset used for evaluation contains 388 states.

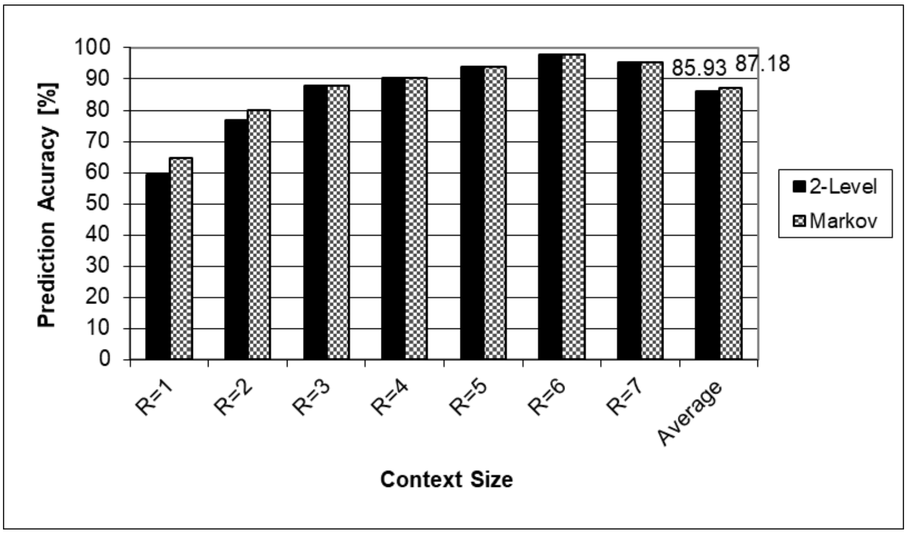


Figure 40. Prediction accuracy

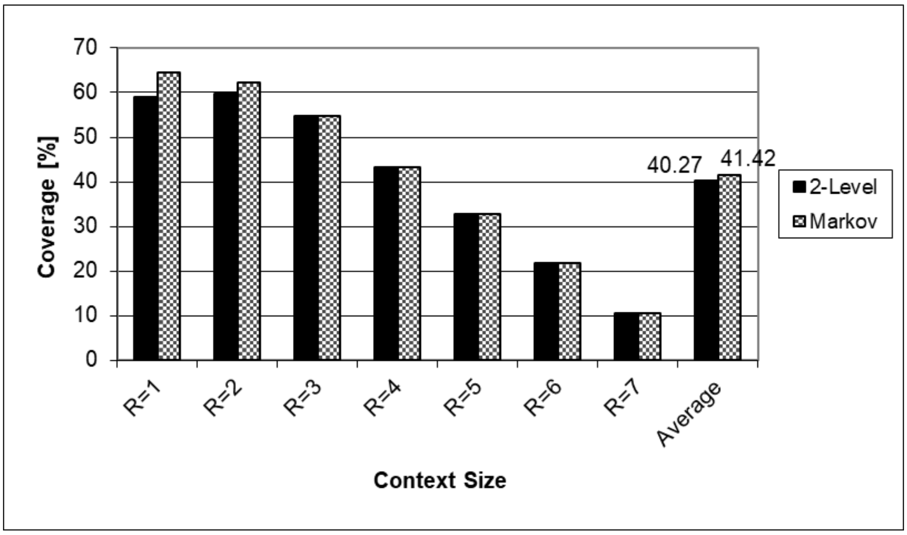


Figure 41. Coverage

Figure 40 presents comparatively the prediction accuracy obtained for different context sizes. The prediction accuracy is the percentage of correct predictions with respect to the number of predictions. We can observe that the Markov predictor is more accurate than the two-level context-based predictor, the highest gain being obtained for a context size of 1. As higher is the order of the models, as lower is the gain for the Markov predictor, which means that the additional frequency information can be avoided if the context information is sufficiently long. However, the higher order models can provide fewer predictions, as we can see in Figure 41 depicting the coverage (percentage of correct predictions with respect to the total assembly steps). Even if the first order models have lower prediction accuracy, they can provide a higher number of correct predictions, by predicting more often.

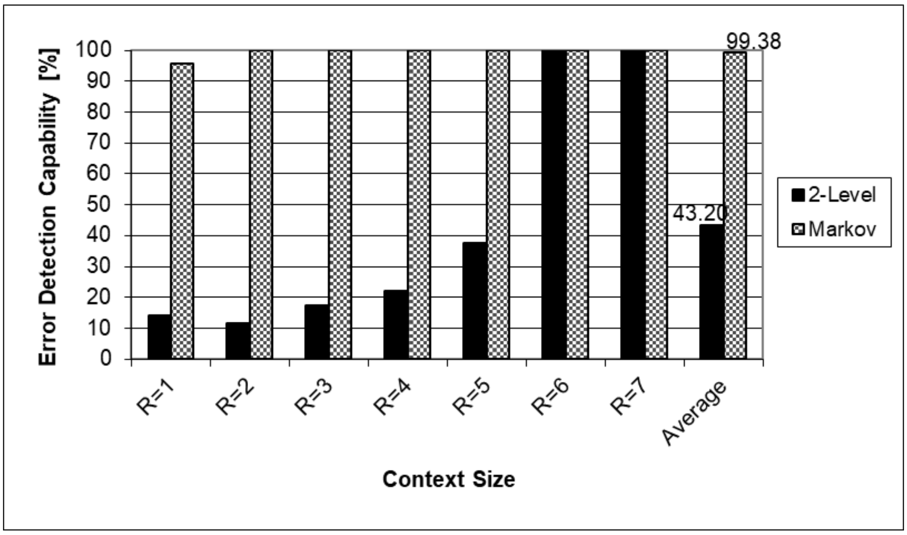


Figure 42. Error detection capability

The highest measured difference between the compared predictors was obtained in their error detection capabilities (see Figure 42), which is over 99% for the Markov predictor and only about 43% in the case of the two-level context-based predictor. This better error detection is possible due to the multiple assembly states stored for each entry of the Markov predictor in contrast to the single state stored for the two-level context-based predictor. Thus, if the Markov predictor is trained with sufficiently high number of different assemblies, it can identify the wrong assembly steps with a higher confidence.

3.2.6.2. Markov Predictor with Padding

The evaluations started by measuring the prediction rate compared to Markov predictors without padding. The order of the predictor was varied between 1 and 7. Higher orders are not justified, considering that a correct assembly process takes seven steps. Obviously, the padding has no sense in the case of the first order predictor, that is why the same results can be observed with both predictors.

Figure 43 depicts that a lower order of the predictor is associated with a higher prediction rate. This is because the shorter contexts searched by the lower order predictors are found more often (almost 100% in the case of a first order predictor). Long contexts are very hard to find, especially the contexts of length 7. But with the padding mechanism applied, even such long contexts are easily found (almost 90%), because the incomplete contexts are filled from the left with zeros, in both training and testing stages. This is a great advantage of the padding mechanism, since it can significantly improve the coverage for the higher order Markov predictors. Figure 44 presents comparatively the coverage of the Markov predictor with and without padding, considering different orders.

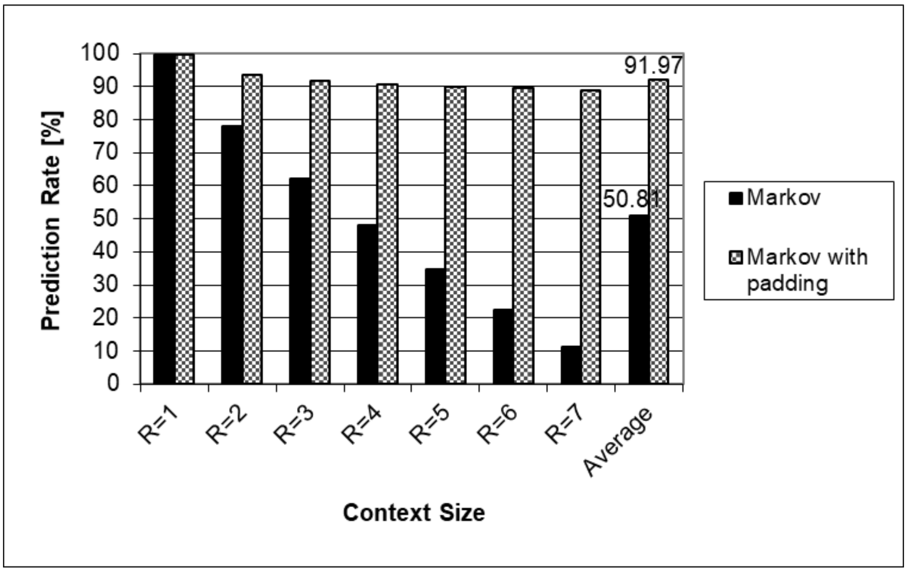


Figure 43. The prediction rate obtained with predictors of different orders

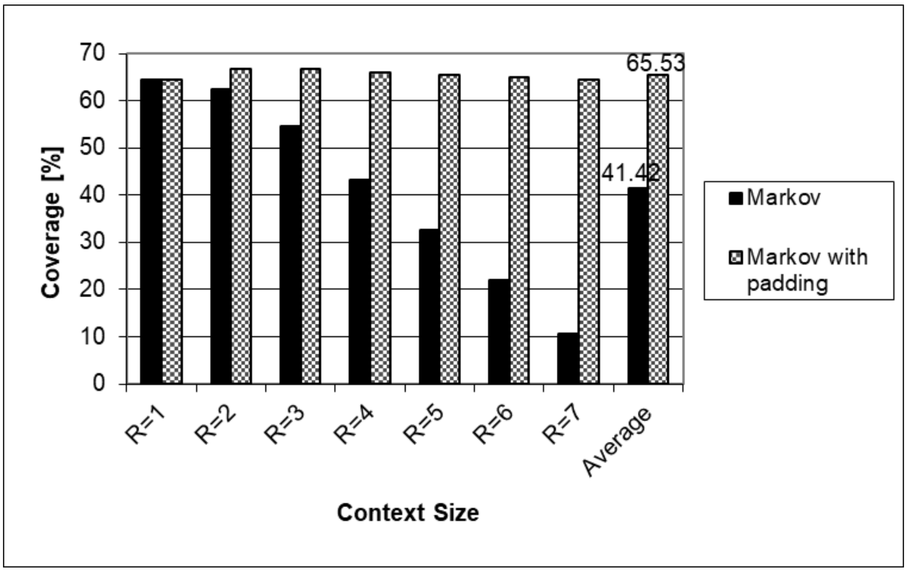


Figure 44. The coverage obtained with predictors of different orders

The benefits of the coverage improvement shown in Figure 44 are remarkable, since with the considerably higher number of correct predictions, the Markov predictor with padding can better model the assembly processes. The padding mechanism improved the coverage from 41.42% to 65.53%, at average, which is considerable. It means that in 65.53% of the cases, the predictor can reproduce exactly the next state. Moreover, the difference between the prediction rate of 91.97% and the coverage of 65.53% consists in other 26.44% of cases when the predictor can give a feasible next assembly step.

A drawback of the padding mechanism is the lower prediction accuracy (see Figure 45). A slight deterioration in accuracy starting with the second order predictor can be observed, where the padding is applied. But the accuracy is not crucial since even a misprediction can be considered in fact a correct next step recommendation. The predictor was trained only with correct assembly steps and, therefore, even if it cannot guess sometimes the intention of a human worker in a certain context, it can provide useful (and correct) next step choices.

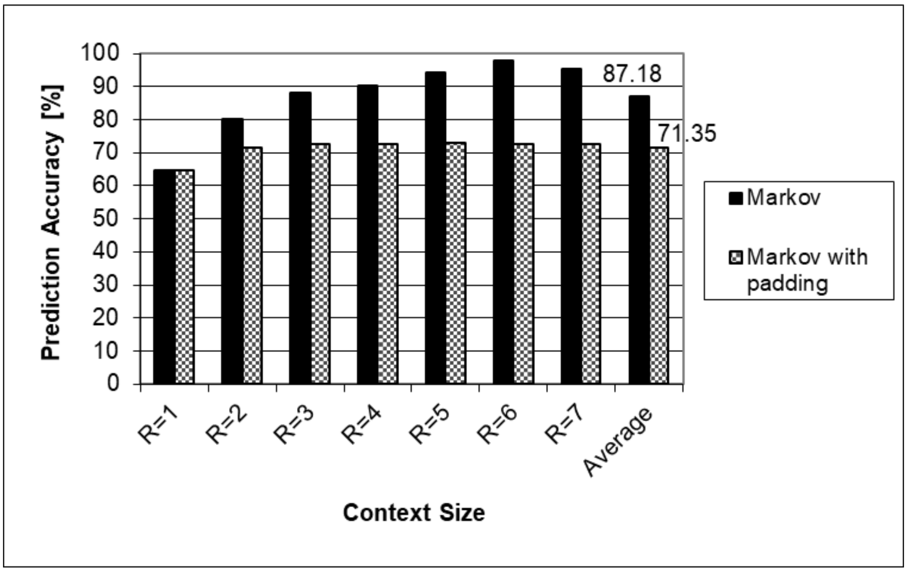


Figure 45. The prediction accuracy obtained with predictors of different orders

Thus, the coverage is the main indicator of our predictor's usefulness. With a coverage of almost 66%, the Markov predictor with padding mechanism is a very good candidate to be integrated into our assembly assistance system in order to recommend the next assembly steps. Both predictors provide the same error detection rate – 95.65% for the first order predictors and 100% for the higher order predictors – and thus, they can be used also to identify eventual wrong assembly steps performed by human workers.

3.2.6.3. Prediction by Partial Matching

This subsection presents the results of the proposed predictor, which will also be compared with other existing prediction methods. The aim is to compare both the capacity of learning the entire dataset and the capacity to adapt to new assembly scenarios. Three metrics have been chosen to evaluate the performance of the prediction algorithms: prediction rate, accuracy and coverage. The prediction rate measures how many times the algorithm was able to make a prediction. The prediction rate is computed in relation to the size of the testing dataset. The accuracy measures how many of the predictions made were correct and the coverage considers the correct predictions in relation to the whole testing dataset.

In the tables presented below, for each metric there are two columns "100/100" and "75/25". These two columns refer to how the algorithms were evaluated: "100/100" indicates that both the training and testing datasets were generated using 100% of the dataset, while "75/25" means that 75% of the dataset was used on the training of the model and the rest of 25% was considered for testing.

PPM	Prediction Rate (%)		Accuracy (%)		Coverage (%)	
Order	100/100	75/25	100/100	75/25	100/100	75/25
1	95.62	54.46	80.05	61.82	76.55	33.66
2	95.62	54.46	82.21	61.82	78.61	33.66
3	95.62	54.46	82.48	61.82	78.87	33.66
4	95.62	54.46	82.48	61.82	78.87	33.66
5	95.62	54.46	82.48	61.82	78.87	33.66
6	95.62	54.46	82.48	61.82	78.87	33.66
7	95.62	54.46	82.48	61.82	78.87	33.66

Table 10. Evaluation of the PPM algorithm on the "Trainees" dataset

Table 10 presents the three metrics for the PPM algorithm of orders 1 to 7, on the "Trainees" dataset. The prediction rate remains constant throughout the orders for both testing methods. For the "100/100" testing method, the accuracy and coverage increase in small amounts up to order 3, where maximum percentages are achieved. Due to higher orders, the sequence can be more tailored to the worker, thus an increase of the accuracy and coverage can be observed. For new data, it seems that the metrics remain constant throughout all the orders.

The PPM predictor was improved by enabling it to search in all the neighboring states (when it is necessary) for a possible assembly. The PPM with neighboring is further denoted PPMN. Table 11 presents the results of the PPMN on the "Trainees" dataset. On known data (with the "100/100" testing method), a small increase can be observed in terms of coverage, a 2% increase in the prediction rate, while the accuracy was slightly lower than that of the PPM without neighboring states. When it comes to new data ("75/25" testing method), the coverage increased by over 10% compared to the PPM without neighboring search and the prediction rate by over 20%. Although there is a slightly lower accuracy, the use of this enhanced algorithm is preferred.

PPMN	**Prediction Rate (%)**		**Accuracy (%)**		**Coverage (%)**	
Order	**100/100**	**75/25**	**100/100**	**75/25**	**100/100**	**75/25**
1	97.68	77.23	78.89	57.69	77.06	44.55
2	97.68	77.23	81	57.69	79.12	44.55
3	97.68	77.23	81.27	57.69	79.38	44.55
4	97.68	77.23	81.27	57.69	79.38	44.55
5	97.68	77.23	81.27	57.69	79.38	44.55
6	97.68	77.23	81.27	57.69	79.38	44.55
7	97.68	77.23	81.27	57.69	79.38	44.55

Table 11. Evaluation of the PPMN on the "Trainees" dataset

After evaluating the two implementations of the PPM algorithm, with and without neighboring states, it can be observed that the optimal order for both algorithms is 3. The optimal configurations of these two algorithms will now be compared with the Markov model with padding enhanced to use human characteristics. For the Markov model with padding, the optimal order is 2. The comparisons are presented on both the "Trainees" and "Workers" datasets. Both the capacity to learn and to adapt to new challenges will be measured.

Figure 46 makes the comparisons in terms of prediction rate. The PPMN has the top prediction on the "Trainees" dataset with the "100/100" testing method. On new data ("75/25" testing), compared to PPM, the PPMN predicts over 10% more often on the "Workers" dataset and over 20% more often on the "Trainees" dataset, with a prediction rate of 91.18% and 77.23%, respectively.

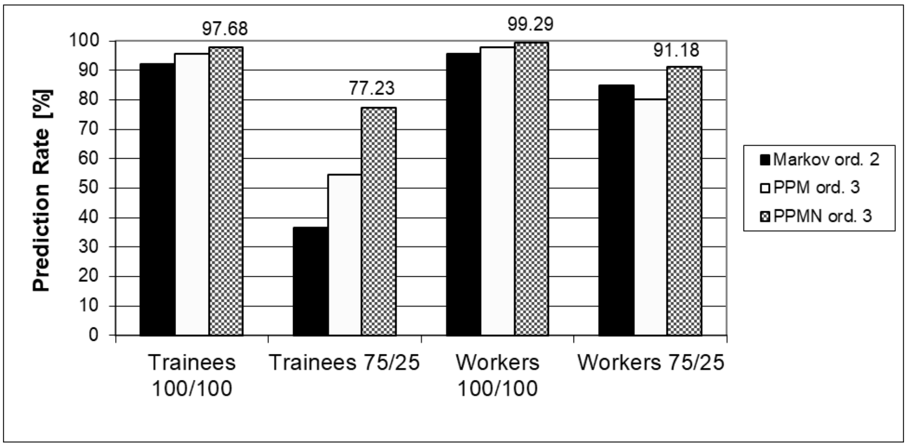


Figure 46. Prediction rate

Both the PPM and PPMN have a similar prediction accuracy across all datasets, with PPM being slightly higher (see Figure 47). On "100/100 testing", the Markov predictor has the highest prediction accuracy.

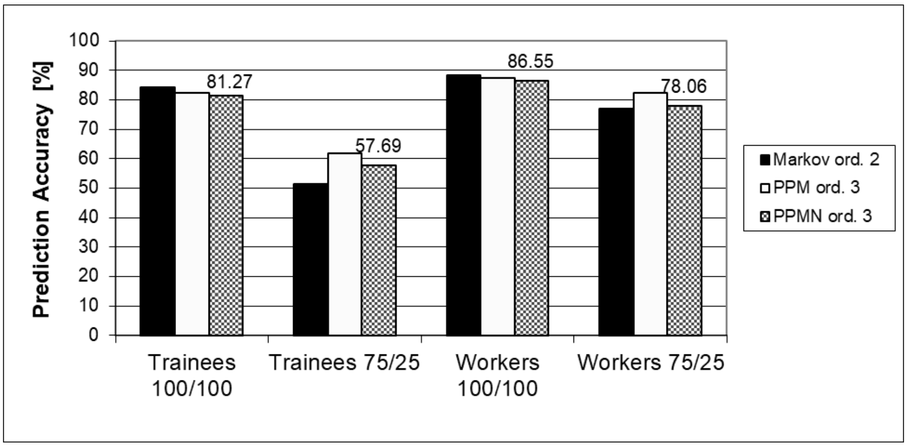


Figure 47. Prediction accuracy

The coverage measures the capacity of these prediction methods to model existing data and to adapt to new data. As Figure 48 depicts, PPMN is the best prediction method to model existing data and has a coverage of 44.55% on the "Trainees" dataset and 71.18% on the Workers dataset, considering the "75/25" testing method. Taking into account that the coverage is an important efficiency indicator, as it expresses the ratio of correct predictions, these results are remarkable. The combination of different order Markov predictors, as well as the exploration of the neighboring states, proved to be a good solution. The PPMN can easier find matching assembly patterns to provide next step prediction.

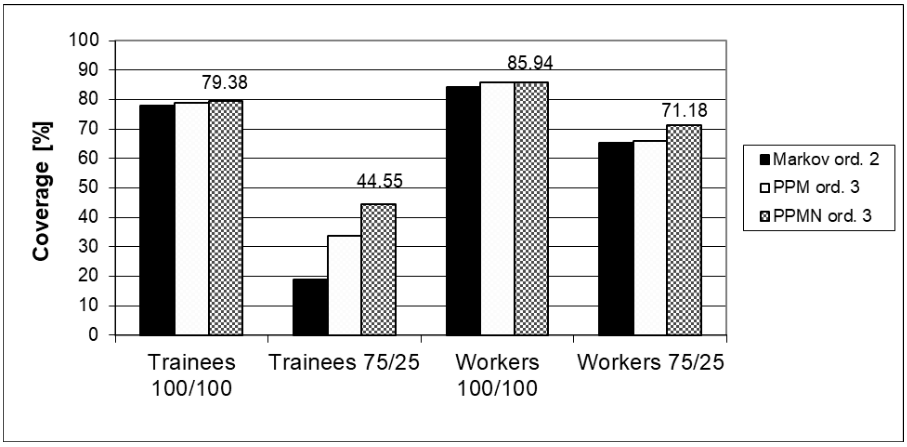


Figure 48. Coverage

3.2.7. Summary

In this section, we analyzed the possibility to provide the next assembly steps by employing Markov predictors in assembly support systems. The experimental results obtained on 68 trainees, have shown increased prediction accuracies for the schemes of orders 1 and 2, with respect to the simpler two-level prediction schemes presented in [84]. With higher order prediction schemes, we have obtained similar prediction accuracies with the existing two-level prediction schemes. Consequently, if the context is sufficiently long, we can avoid the additional frequency data. However, the Markov predictor can keep the error detection capability very close to 100%. Since we obtained the highest coverage with the first order Markov predictor, we consider this configuration as being optimal. Another great advantage of the Markov predictor is that it can provide multiple choices for the next assembly step, in their descending probability order, and consequently better capture the dissimilar psychophysiological profiles of workers. For time-critical decisions, it can be configured to provide only the most probable assembly solution.

Markov chains with padding mechanism were also analyzed as possible prediction component. By completing the missing elements of the contexts with zeros, a significantly higher prediction rate was achieved. Thus, the coverage is substantially increased, even though the prediction accuracy is lower. The experimental results have shown a much higher number of correctly predicted assembly steps, meaning that this improved Markov predictor can better model the behavior of the human workers. The coverage, which is one the most important indicator for our goal, increased from 41.42% to 65.53%. Thus, the proposed predictor proved to be efficient in recommending the next assembly step and in identifying the errors of the workers. Moreover, it can efficiently adapt to each worker’s assembly style, which would be hard for a human trainer. The predictor can be applied for assembling any product, the difference being only the number of states determined by the number of components.

We have also evaluated the PPM algorithm. The experiments have shown that the optimal PPM is of order 3. For a higher prediction rate, a PPM enhanced with a neighbor-states checking mechanism, the PPMN was used. Thus, when the algorithm could not find the current state (consisting of the worker’s characteristics and the sequence of the last assemblies), it also checked the states which were neighbors from the human characteristics point of view and, in case of success, the next assembly step was determined by majority voting among such existing neighbor states. The PPMN has a significantly higher coverage on new data: 44.55% in the case of trainees and 71.18% in the case of factory workers. It also clearly outperforms the Markov models in terms of coverage.

The possibility to anticipate the next assembly state through Hidden Markov Models and Dynamic Bayesian Networks will be analyzed in our future work. Hidden Markov Models are doubly embedded stochastic processes consisting in a hidden stochastic process that relies on a set of observable stochastic processes. Hidden Markov Models were applied with very good results in speech recognition [151], smart buildings [75], computational biology [227], etc. These powerful stochastic models could be further used as prediction methods to provide the next assembly step in manufacturing processes.

4. Anticipative Systems for Smart Buildings

Ubiquitous systems strive for adaptation to user needs by utilizing information about the current context in which a user’s appliance works [210]. A new quality of ubiquitous systems may be reached if context awareness is enhanced by predictions of future contexts based on current and previous context information. Such a prediction enables the system to proactively initiate actions that enhance the convenience of the user or that lead to an improved overall system.

Humans typically act in a certain habitual pattern, however, they sometimes interrupt their behavior pattern and they sometimes completely change the pattern [209]. Our aim is to relieve people of actions that are done habitually without determining a person’s action. The system should learn habits automatically and reverse assumptions if a habit changes. The predictor information should therefore be based on previous behavior patterns and applied to speculate on the future behavior of a person. If the speculation fails, the failing must be recognized, and the predictor must be updated to improve future prediction accuracy.

One of the major societal concerns is the energy consumption and the environmental footprint of consumers (like buildings, city street lighting, IT servers of data centers, etc.). The Environment Protection Agency of USA highlighted that computing systems, in particular data storage centers, consume the same amount of energy as civil aviation (about 2% of the world's total energy in 2010) [42]. Without a considerable energy efficiency improvement, high-end parallel computers will be of questionable economic viability and most of mobile computing, wearables and Internet of Things (IoT) devices will suffer from a lack of autonomy. Energy-saving solutions must follow two directions. On one hand, the total energy consumption must decrease. Electronic devices must be made more efficient from the energetic viewpoint by using less energy-hungry applications on smartphones and in other embedded systems or by diminishing the energy per floating-point operation. On the other hand, it is necessary to research safe alternative energy sources (wind power, renewable sources obtained through passive solar techniques, photovoltaic panels, biomass and hydropower) to replace current energy sources. The current economic approach that basically amounts to a linear industrial model where materials are taken, transformed into products, and then disposed with, is evolving into a circular economy which encourages reduction, reuse, and recycling. The value of resources is thus maintained for as long as possible, resulting in energy savings and in lower greenhouse gas emissions.

The transition from linear to circular economy has received support from the European Commission through a package adopted in 2015. In the same year, the Paris Agreement was signed under the United Nations Framework Convention on Climate Change. The electrical power sector must undergo a thorough metamorphosis to achieve the ambitious targets in greenhouse gas reduction set out in the agreement. Given the growth in trade volumes and diversity of products exhibited by the market of electricity derivatives, electrical energy plays an increasingly important role on the market. Electricity spot prices are driven largely by demand. Reducing the uncertainty relative to non-elasticity of demand (and, in case of renewable electricity, supply) and non-storability of electricity is essential when one is confronted with the arduous task of modelling spot prices [219].

Perhaps one of the most obvious advantages of predicting electricity consumption and production is the spreading of awareness of the negative environmental impact of a high consumption, the importance of balancing between production and consumption, and the benefits from intelligent energy management in buildings. Prediction of electricity patterns with different time granularities (day/week/month/year), isolation of ascending and descending trends, depending on geographic region and the lifestyle of consumers, will help consumers become more aware of their electricity consumption and enable them to develop intervention strategies according to their pattern of consumption. Furthermore, consumers who have direct and frequent access to consumption and production measurements and patterns specific to them, will likely become more ecologically aware about environmental issues.

In a renovated electrical grid, data are collected and used as a basis for decisions, with the aim of improving the efficiency, reliability, and sustainability of electricity production and distribution. The automatization degree of this process plays an important role, as well as the self-monitoring and feedback capabilities offered to customers. In order to have accurate measurements of the operating conditions of the electricity grid, sensors need to be placed throughout it, in particular on production, transmission, and distribution systems, in addition to consumer access points [226]. Processing the collected data, which falls into the realm of big data analytics, helps decision makers to measure the level of energy supply they should guarantee and to estimate reasonable safety margins. In addition, predictions of energy demand can be used to determine innovative, flexible, and dynamic pricing plans that are closely tailored to usage patterns [3]. In this context, the availability of individual predictions complements and completes centralized analysis systems.

The prediction of electricity consumption could be used as a monitoring and diagnostic solution embedded in the self-healing feature of modern smart grid technologies [184]. If some piece of equipment would require an unusual amount of electricity or if there is a defective component inside the network (buildings, factories or city street lighting), our application could send a notification to the maintenance team that will then perform an on-site diagnosis on the identified component, deciding about the appropriate action: further close monitoring, repair, or replacement. Such an approach can reduce costs by preventing component loss, while avoiding unexpected electrical interruptions. Another implication in practice of our developed tool refers to predict electricity demand in order to know in advance when to start ventilation and cooling of electrical systems in data centers, to efficiently manage the networks and target interventions designed to reduce or time-shift peak loads. In general, the usefulness of predictions will be amplified whenever actuators with some latency are present.

In increasingly crowded cities, a detailed and current knowledge of the number of inhabitants becomes useful and necessary for city authorities in planning public transportation and traffic, and accurately managing public transport fleets. Prediction of electricity consumption could provide considerable opportunities to find out household characteristics. Our solution can be used to generate household consumption profiles and link these data to the number of people in buildings in order to implement a smart census process. In an experiment in UK presented in [5], the authors analyze the feasibility of using household energy consumption for a specified period to infer their characteristics as a first step in aggregating them with other population and geographic location metrics. These area level population statistics could represent new insights for enhancing the census taking process with digital trace data.

4.1. Energy Management in Buildings by Contextual and Computational Prediction

Among economic sectors, infrastructure is somewhat special, in that it is normally invisible – unless it breaks down. Transfer of information has traditionally played a marginal role in the electricity grid, most often with the information flow in the form of a monthly bill and usage report. Bringing information to an infrastructure and observing the effects on processes and on the behavior of actors is an opportunity to focus attention on the impact of information in the reorganization of a sector [156]. Data collection, both at the supply side and the demand side, facilitates the balancing of the grid, a critical requirement for an appropriate supply of energy to be maintained. In a smart grid, households could be expected to react accordingly when the electricity grid requires an adjustment, for example when a reduction of intake is desired and achieved through the signaling of instant price increments [156]. The effects of a technology-induced reorganization of the smart grid when users are no longer passive participants in the relationship between them and the infrastructure, but they become active and the information flow becomes bidirectional, can be profound and should be studied with great attention.

Data analysis is one of the nine key factors that characterize data-based value creation [132]. Through the interconnection of sensing and actuating devices, data collected from the smart grid can support decision makers in devising appropriate decisions with respect to the adjustment of the supply level of electricity. In addition, smart grid data analytics can help predict future demand, helping in planning expansions [104]. Such prediction is, though, centralized. If the electricity grid infrastructure is viewed from a socio-political perspective, a reflection on issues regarding power and control comes forward, because infrastructures are essential systems for living and a complete control over them implies potential expedients for enhanced power [156]. The availability of autonomous, independent predictive systems might represent a valid protection mechanism.

Production and distribution of hardware devices is not the sole business model enabled by the mechanism proposed here. The complexity of smart homes and smart grid suggests that it may not be viable for a single actor to build a comprehensive management solution alone. Companies operating in the area will have to establish partnerships among them and such data analysis partnerships are foreseen to be among the most important building blocks in shaping new business models [39]. For example, the analysis of readings from the device proposed in this section could be integrated and augmented with the corresponding information coming from a smart thermostat used to monitor their pattern of use of heating.

Energy systems tend to become increasingly distributed due to the resilience and sustainability needs and the advanced researches in the field of distributed energy resources [110]. The penetration of distributed energy resources is determined by governmental policies, economic incentives and the social pressure on companies and individuals to perceive them green and progressive. The use of distributed energy resources results less transmission losses with respect to the centralized power systems and allows an intelligent management, being thus more efficient and in trend with the smart city concept.

Nowadays, photovoltaics allow decentralized electricity production at a cost lower than that of the power grid. Costs are even lower if energy storage systems are included. Self-consumption, i.e., the consumption of self-produced electricity, can be significantly increased with an intelligent energy management system which is able to streamline electricity production and consumption. The software architecture of such a system was proposed in [52], applying Artificial Neural Networks (ANN) to anticipate the future electricity consumption and production. Based on these predictions, the energy management system can make decisions in order to increase self-consumption, thus reducing the electricity intake from the power grid, finally decreasing the total annual operating cost [183]. The energy management system can decide to activate some electrical appliances when cheap electricity is available and delay their activation when only high-cost electricity is available. Self-consumption will also reduce losses in distribution networks, improving efficiency.

In smart grids, households must react accordingly when the electricity grid requires an adjustment, for example when a reduction of intake is desired and achieved through the signaling of instant price increments [156]. Data analysis is a key factor in data-based value creation [132]. Through the interconnection of sensing and actuating devices, data collected from the smart grid can support decision making with respect to the adjustment of the electricity supply level. Additionally, smart grid data analytics can support the prediction of future demand, helping in planning expansions [104].

This section compares based on [73] different prediction techniques on the data recorded by the FENECON Energy Management System (FEMS) [52]. We evaluate the performance of Markov chains as electricity consumption and production predictors. Markov chains are widely used context-based methods which can provide predictions. Electrical power data cannot be applied directly as input in Markov chains, since that would imply a huge number of states and, thus, the predictor would be inefficient. Therefore, we preprocessed the input data in order to enable the usage of Markov chains with a reasonable number of states for an efficient prediction process. We also analyze the possibility to apply stride predictors which are using the last strides from a sequence of values to predict future values. The previously mentioned preprocessing of the input data is necessary for the stride predictor, too. We will also study the combination of Markov chains and stride predictors for hybrid electricity forecasting. Such an adaptive hybrid prediction would take the advantage of each of its components based on their dynamic behavior. We analyze the accuracy of the proposed predictors on real data with the goal of minimizing the mean absolute error. We will determine the most appropriate configuration for electricity forecasting. We will also compare our optimal prediction method with the already existing neural network approach. Accurately predicting energy consumption could help improve energy management, grid performance, and reduce maintenance costs by pre-planning inspections and replacing equipment susceptible to damage. Moreover, it could improve the social and economic benefits by alerting the population if certain consumption thresholds are overtaken.

4.1.1. Related Work

In [100], the authors emphasized the advantages of big data and cloud computing technology for storing and analyzing massive electricity data, and to explore the pattern of electricity consumption. Several algorithms are analyzed such as Deep Learning, Fuzzy C-Means (FCM) clustering method, including social network-based predictors, suggesting a new perspective for describing the energy consumption behavior of consumers: in the time dimension, user dimension and spatial dimension.

Traditional energy resources (fossil fuels) tend to be exhausted. Without rethinking energy management, critical situations will be reached (researchers estimate in 2040) when energy demand will exceed the world's estimated energy production. To further exacerbate the problem, electrical energy is not uniformly distributed on the planet and does not cover the minimum required for everyone. In 2016 there were over 1 billion people with limited or no access to electricity [217]. Renewable energy solutions like wind power or photovoltaic energy generation represent the key in the fight against climate change. However, the unpredictability of meteorological and climatic parameters (the wind does not constantly beat, the lack of sunny days, rare rain and drought) contrasts with the need for a continuous and stable energy supply, and this represents a drawback of renewable energy systems. Smart solutions are required regarding the locations of wind farms and solar panels, or the introduction of energy storage stations is necessary.

Solar energy is one of the most important renewable energy sources supporting the decarbonization efforts. Over the past twenty years, technology for power generation and storage associated to photovoltaics (PV) has known an impressive development. Due to environmental conditions, fossil fuel depletion, governmental support or just plainly looking for a cleaner energy source, PV were widely adopted. As solar energy depends on many factors and is not constant, prediction is indispensable. There are many studies, some using only solar irradiation while others also considering air temperature, humidity or wind speed. The predictions of energy production contribute to the identification of the right time for plant maintenance and to the establishment of adequate prices for electricity on the day-ahead market.

Moreover, predictions can be made for energy production, but also for energy consumption. Predictions can also be categorized based on their time range: short-term predictions (covering usually a time slot of a few hours), average predictions (ranging from several weeks to one year) and long-term predictions (more than one year). Prediction models have been developed for consumers of different size: individual households, buildings, grids and plants. Most prediction methods aim at determining the minimum and average levels of energy consumption and the economic costs involved. The ultimate goal is to raise awareness, apply corrections to actions already performed, and stimulate customers to increase efficiency, reduce energy consumption, or choose alternative (renewable) resources.

Energy prediction and reduction solutions in Smart Buildings are based on automation of lighting, heating, ventilation and air conditioning (HVAC), security and surveillance, garden irrigation systems [183], on the basis of information inferred from past data or provided by data sources such as environmental sensors, laser beams, weather forecasts, building occupancy profiles, or number of parked cars. An accurate measurement (using Smart Meter instruments) of the energy consumption became a commonly used practice, given the increase in energy prices. However, predicting the electricity load is difficult because demand patterns generally differ among consumers, increasing the uncertainty of prediction. In [64], the authors propose, in a short-term approach, two types of predictors – based on ANNs and Support Vector Machines (SVM) – to forecast the electricity demand of individual households for 24 hours ahead. The benefit to customers is that they will be able to better understand their energy consumption and the afferent costs.

Numerous factors influence and even limit the energy performance of buildings [231]. Among these factors we recall the environmental conditions, the buildings architecture, features and occupancy degree, behavior of inhabitants, and the operating regime of lighting and HVAC subsystems. The large number of input parameters involves a high algorithmic complexity in the prediction of energy consumption. The previously mentioned paper highlights some existing solutions for solving the energy prediction problem using statistical and ANN-based methods, and also further prospects are proposed.

The main difference between the works referred above and our solution is that most of the state-of-the-art solutions are using ANNs, SVMs or time series models, whereas our proposed approach applies context-based methods like Markov predictors, computational (stride) predictors, and the hybridization of these two techniques.

4.1.2. Electricity Prediction

We present Markov chains, stride prediction and their hybridization as methods of forecasting electricity consumption and production. Our goal is to adapt them to be functional in energy management systems with photovoltaics and energy storage. In particular, techniques amenable to efficient implementation in hardware devices are studied. The reason for this decision is that energy management is foreseen to be integrated in lightweight devices such as those used in IoT. The scheme of the electrical installation used in the experiments is depicted in Figure 49.

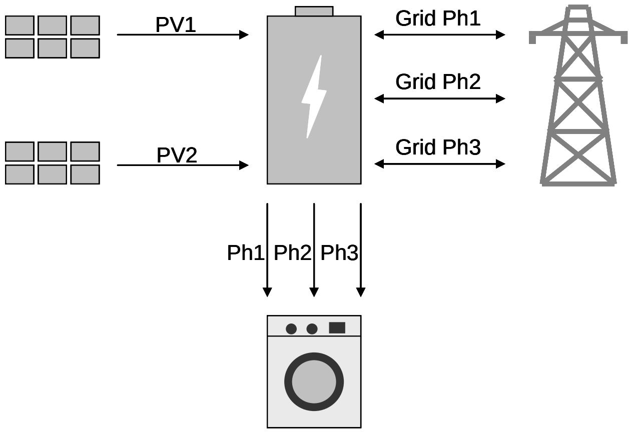


Figure 49. The energy production and storage system

As Figure 49 illustrates, the energy storage system is connected with two photovoltaics of 12.24 kWp (PV1 and PV2) and also with three grid phases (Ph1, Ph2 and Ph3). The energy produced by PV1 and PV2 is kept in the storage system with a capacity of 8.5 kWh. Consumers can take electricity from the storage system or from the grid.

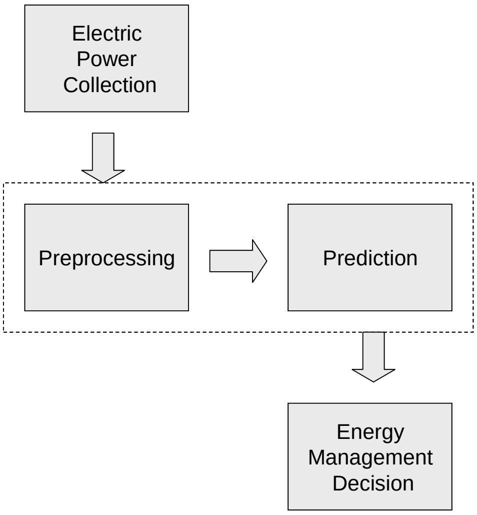


Figure 50. The prediction process in an energy management system

Figure 50 depicts how the prediction process is integrated into an energy management system. In the first stage, data about produced and consumed electricity must be collected. In the next stage, the recorded data must be checked for erroneous values, which must be corrected. Such problems (especially huge values) can occur during data collection. We corrected such identified erroneous data by replacing the wrong value with the previous recorded value. We also encode all the data. After the preprocessing step, the predictor computes the electric power for the next period, based on the data relative to previous periods. These predicted electricity production and consumption values are then used to make decisions by the energy management system to increase self-consumption. Next, we present the prediction methods evaluated in this section. All these methods are returning the next forecasted electric power, or a special value (here, -1) whenever they are unable to deliver a prediction. In such unpredictability cases, we will forecast the next electric power value as being equal to the previous one.

4.1.2.1. Markov Predictors

Observed levels of energy (production or consumption) constitute a sequence where subsequent samples are not independent. Such sequences can be described as being generated by a parametric random process, whose parameters can be learned from a training sample of sequences.

Since the high number of possible distinct electric power values would highly increase the state complexity of the Markov chain, a classical implementation could be inefficient for prediction. Therefore, we implement the Markov chain-based prediction algorithm in an efficient way, as we did in [82] for web access prediction. Instead of predicting the next electric power through trees, graphs or transition tables, we generate the prediction by performing simple searches in the electric power sequence. This way of implementation will make possible to use Markov chains with significantly higher number of states. We will further decrease the state complexity of the Markov chain by preprocessing the input data. All the input electric powers will be encoded to intervals, each interval being represented by an integer value. The output will be determined by decoding the predicted interval (scaling to an approximated electric power value).

In the Markov chains used in this section, states are represented by electric power values in Watts. In a Markov chain of order 1, the current state depends on the history only through the previous value:

$$P[e_t \mid e_{t-1},\ ...,\ e_1] = P[e_t \mid e_{t-1}] \tag{25}$$

where e_t is the electric power at time *t*. In a famous sentence, P. Lévy stated this property as "the past influences the future only through the present". If we generalize, in a Markov chain of order *R*, the current electric power is deduced based on *R* previous electric powers [82]:

$$P[e_t \mid e_{t-1},\ ...,\ e_1] = P[e_t \mid e_{t-1},\ ...,\ e_{t-R}] \tag{26}$$

Markov chains can be used to anticipate the electric power by searching for the current electric power context in the stored electric power history. The predicted power level is the state for which the estimated transition probability from the current state is the highest. Figure 51 presents an example of electricity forecasting with a Markov chain of order 1, using intervals of 10, on a real sequence of nine electric power values extracted from the PV1 dataset.

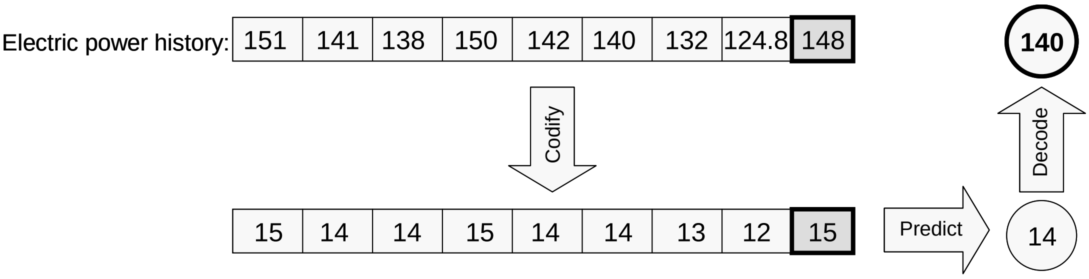


Figure 51. An example of electricity prediction with a Markov chain of order 1

As Figure 51 illustrates, the electric power history (composed of nine values) is codified by division to the interval value (which in this certain case is 10). The codified context is 15. The Markov chain predicts 14, since it occurred with the highest frequency after the context 15. The predicted value 14 is decoded (by multiplying with 10) to the electric power value 140, which is the final prediction. Figure 52 presents the flowchart of the prediction process using a first order Markov chain.

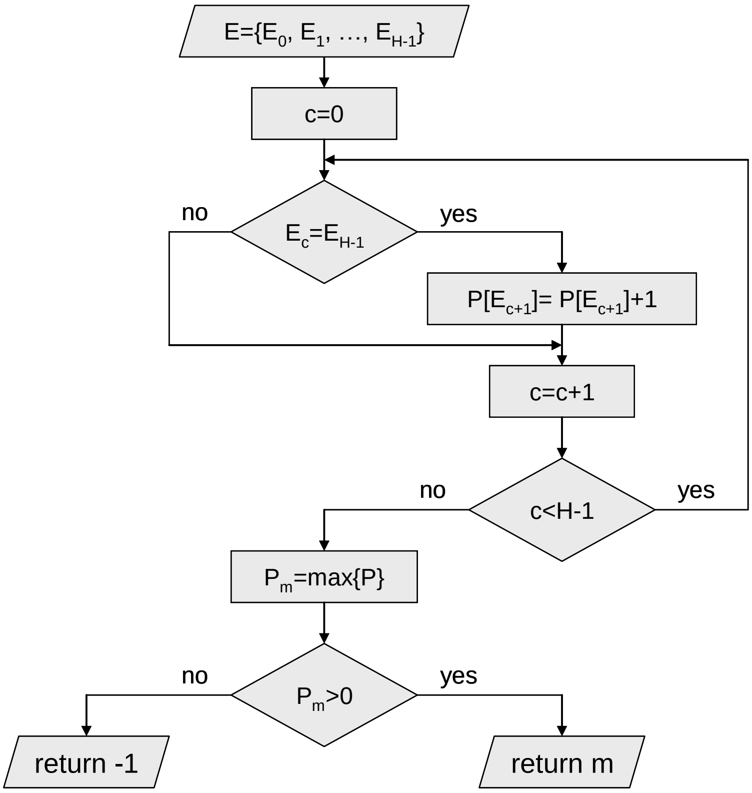


Figure 52. Flowchart of the prediction process with a Markov chain of order 1

The pseudocode of the general Rth order Markov prediction algorithm, used for electricity forecasting, is presented below:

```
 1. MARKOV (E, R)
 2.    for c := 0 to R-1 do
 3.       C[c] := E[H-R+c]
 4.    endfor
 5.    for h := R to H-1 do
 6.       IS_CONTEXT := TRUE
 7.       for c := 0 to R-1 do
 8.          if E[h-R+c] != C[c] then
 9.             IS_CONTEXT := false
10.             break
11.          endif
12.       endfor
13.       if IS_CONTEXT then
14.          P[E[h]] := P[E[h]] + 1
15.       endif
16.    endfor
17.    PREDICTION := 0
18.    MAX := P[0]
19.    for i := 1 to N-1 do
20.       if P[i] > MAX then
21.          MAX := P[i]
22.          PREDICTION := i
23.       endif
24.    endfor
25.    if MAX > 0 then
26.       return PREDICTION
27.    endif
28.    return -1
29. end
```

where R is the order of the Markov chain, E is the electric power sequence, the context C is containing the last R values from the electric power sequence, H is the length of the electric power sequence, P is the probability distribution for N distinct electric power values, PREDICTION is the predicted electric power value, and MAX is the frequency of the predicted electric power value occurring after the context. If the current context is not found in the electric power sequence, which is expressed by returning -1, the Markov chain is unable to deliver a prediction. The Java implementation of such a Markov predictor is presented in [92].

On the lines 2-4 the context C is extracted from the electric power sequence E. After that, on lines 5-16 the context C is searched within the electric power sequence E and each time the context is found, we increase the probability of the electric power value that follows the context. In this way, we compute the probability of some possible electric power values to be the next one in the sequence. On the lines 17-24, we determine the highest probability. On the lines 25-27, if the highest probability is not 0, we return the electric power value corresponding to that probability. Otherwise, if all the probabilities are 0, we return -1 on the line 28, meaning that the algorithm is unable to predict.

In this algorithm, we limit the value of H and thus we use a limited history of recorded electric powers. We will vary the parameter H in the experimental setup.

4.1.2.2. Stride Prediction

The stride predictor is a trend-based computational prediction method which determines the next value as the sum of the immediate previous value and a stride [90]. The stride is the difference between the two most recent values. In our algorithm we chose as condition the equality between the last two strides. Thus, we generate a prediction only if the stride between the last three values was constant. Figure 53 presents an example of prediction with the stride predictor, using intervals of 10, on a real electric power sequence extracted from the PV1 dataset.

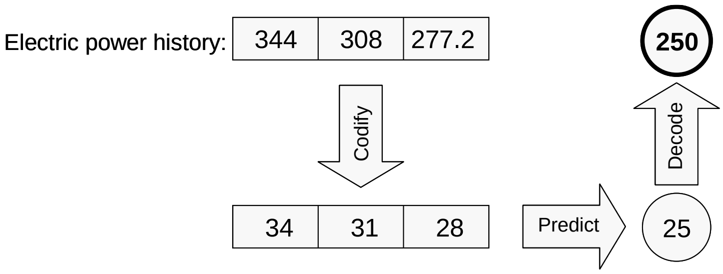


Figure 53. An example of prediction with the stride predictor

As Figure 53 depicts, the electric power history (composed of three values) is codified by division to the interval value (which in this case is 10). After a constant stride of -3 is detected, the prediction is generated by adding the stride -3 to the last value 28. Then, the predicted value 25 is decoded (by multiplying with 10) to the electric power value 250, which is the final prediction in this case. Figure 54 presents the general stride prediction mechanism.

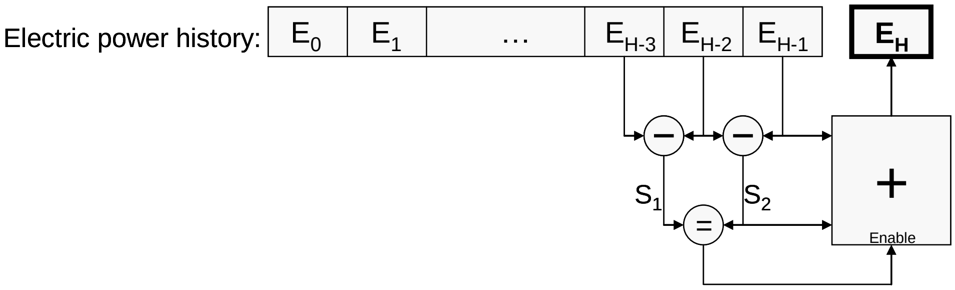


Figure 54. The stride prediction mechanism

As Figure 54 shows, we compute the stride S_1 as the difference between E_{H-3} and E_{H-2} and also the stride S_2 between E_{H-2} and E_{H-1}. Only if S_1 and S_2 are equal, we predict the next electric power value E_H as the sum between the last electric power value E_{H-1} and the stride S_1. Next, we present the pseudocode of the stride prediction algorithm.

```
1. STRIDE ($E_{H-1}$, $E_{H-2}$, $E_{H-3}$)
2.    PREDICTION := -1
3.    $S_1$ := $E_{H-1}$ - $E_{H-2}$
4.    $S_2$ := $E_{H-2}$ - $E_{H-3}$
5.    if $S_1$ = $S_2$ then
6.       PREDICTION := $E_{H-1}$ + $S_2$
7.    endif
8.    return PREDICTION
9. end
```

where *PREDICTION* is the predicted electric power value, which is -1 whenever the stride predictor is unable to predict. On the lines 3-4 the last two strides are computed. On the lines 5-7 the predicted electric power value is determined. This computational predictor should be able to identify and exploit constantly increasing or decreasing electric power sequences.

4.1.2.3. Hybrid Prediction

Since our previous experiments on different problems (branch and value prediction in the microarchitecture domain or webpage prediction) pointed out that a single predictor usually strives in capturing all the various types of predictability patterns that occur in real scenarios, we implemented a hybrid scheme for enabling high prediction accuracy. The hybrid prediction mechanism includes as components the above presented Markov and stride predictors. The hybrid predictor maintains a 4-state saturating counter for each component: C_M for the Markov predictor and C_S for the stride predictor. When a component produces correct predictions, its associated confidence counter is incremented. On the other hand, when a predictor is mispredicting, its counter is decremented. We consider the prediction generated by the component predictor with the largest confidence counter. Thus, the hybrid predictor is dynamically adapting based on the behavior of its components and each time will select the most confident predictor. A low number of states in the saturating counters, assures fast adaptation to possible behavior changes in the electricity consumption or production. The hybrid prediction mechanism is presented in Figure 55.

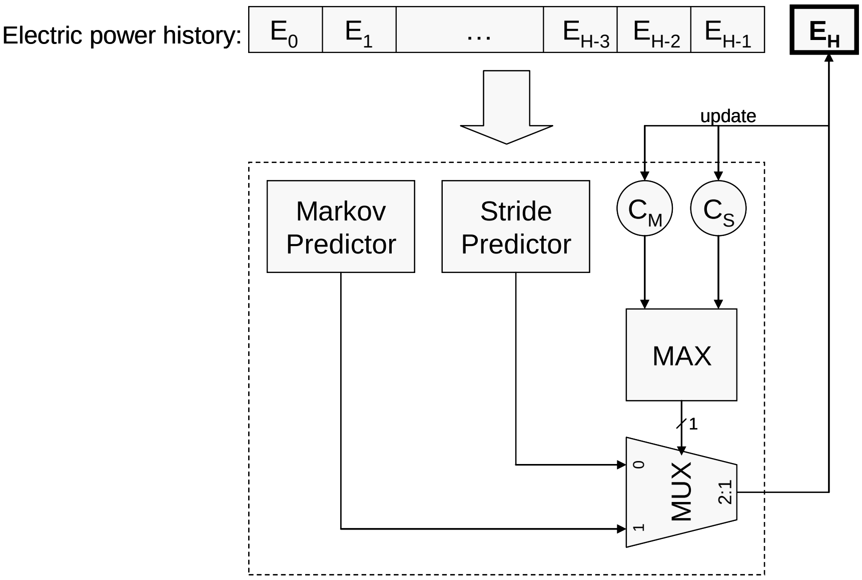


Figure 55. The hybrid prediction mechanism

The MAX unit returns 1 if C_M is greater than C_S and 0 otherwise. Such a hybrid predictor can take the advantage of its components. It can be obviously extended to include more than two predictors.

4.1.3. Experimental Results

The performance of the proposed algorithms will be evaluated from the Mean Absolute Error (MAE) viewpoint, which is computed as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|R_i - F_i\right| \quad (27)$$

where R_i is the observed electric power at time i, F_i is the forecasted electric power at time i and N is the number of recorded electric power values. MAE has been preferred over the Mean

Squared Error because the latter tends to exaggerate the influence of outliers. A manual exploration of the space of hyperparameters has been carried out to determine the best-performing models. Such analysis also provides insights into the dynamics of the data-generating process. We have evaluated the proposed methods on the datasets recorded by FEMS: two datasets with produced electricity (PV1 and PV2) and three datasets with consumed electricity (Ph1, Ph2 and Ph3). The electric power values, expressed in Watts, were collected in 2015 between 1st January and 31st May, with one record per 5 minutes [52].

We started the evaluation with the Markov predictor. Figure 56 shows the MAE for different context sizes (Markov chain orders). We varied R between 1 and 5 and we used an electric power history size of 300 and an interval of 10.

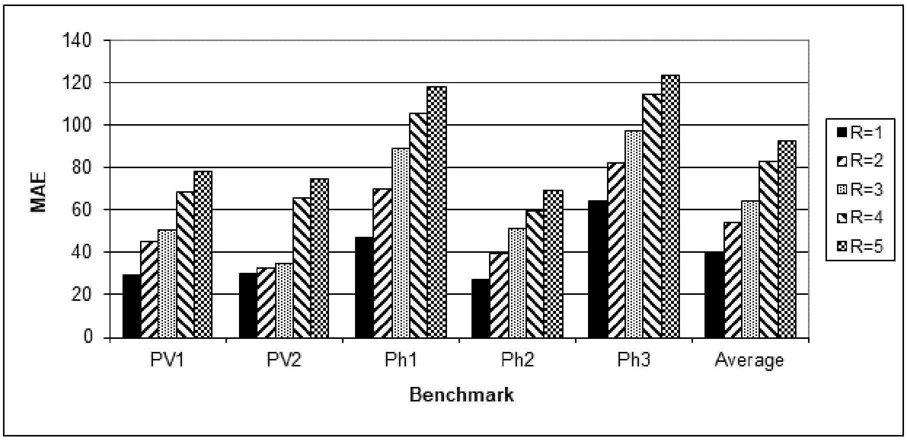


Figure 56. Varying the Markov chain's order

Figure 56 shows that as the higher is the Markov chain's order, the higher is the MAE. Thus, for this type of data, lower order Markov predictors (with shorter context information) are more appropriate.

We continue the evaluations by fixing the order R to 1 and we will now vary the history size. We still keep the (electric power) interval size on 10.

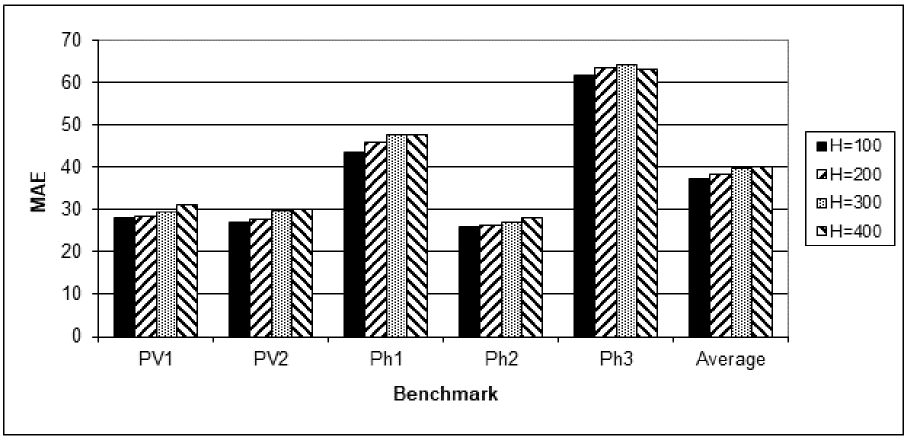


Figure 57. Varying the history size

As Figure 57 illustrates, the lowest MAE was obtained with a history size of 100. With longer histories the MAE is slightly higher. Therefore, we fixed the history size H to 100. Lower values were not evaluated because of the bias-variance trade off.

Next, we have evaluated the interval size I, considering the previously optimized order of 1 and history length of 100.

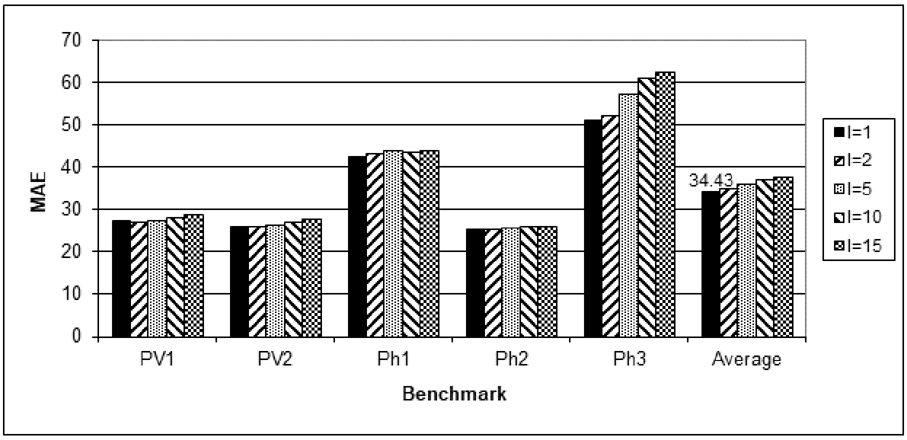


Figure 58. Varying the interval size

As Figure 58 depicts, a lower interval size provides better results. The lowest MAE was obtained with an interval size of 1, meaning that we consider the integer part of the electric power values in the codification process. With the interval size of 1, the decoding step becomes unnecessary. The history length of 100 seems to be sufficiently long to allow an interval size of 1, since the electric power values are mostly found in the electric power sequence. Consequently, the best Markov predictor is of order 1 and it is using a history length of 100 and an interval size of 1.

Finally, we have evaluated the stride predictor in conjunction with the Markov chain. We have also integrated the stride predictor and the best Markov predictor configuration (R=1, H=100, I=1) to a hybrid predictor which will dynamically select its most confident component.

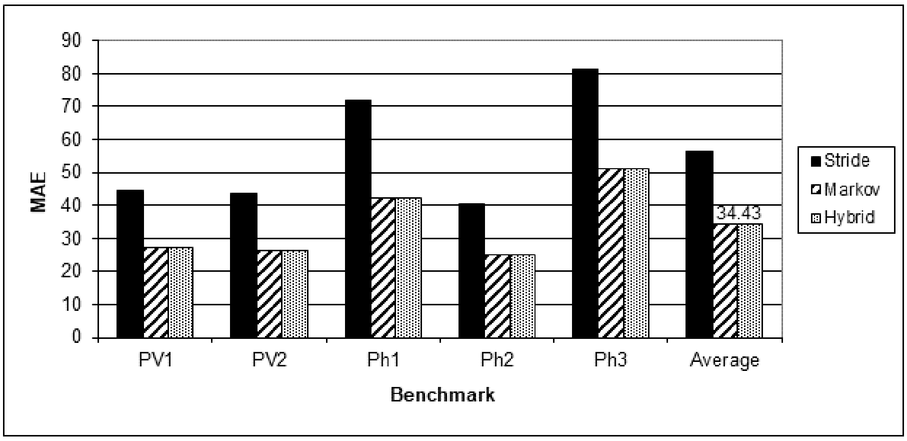


Figure 59. Comparing the Markov, stride and hybrid predictors

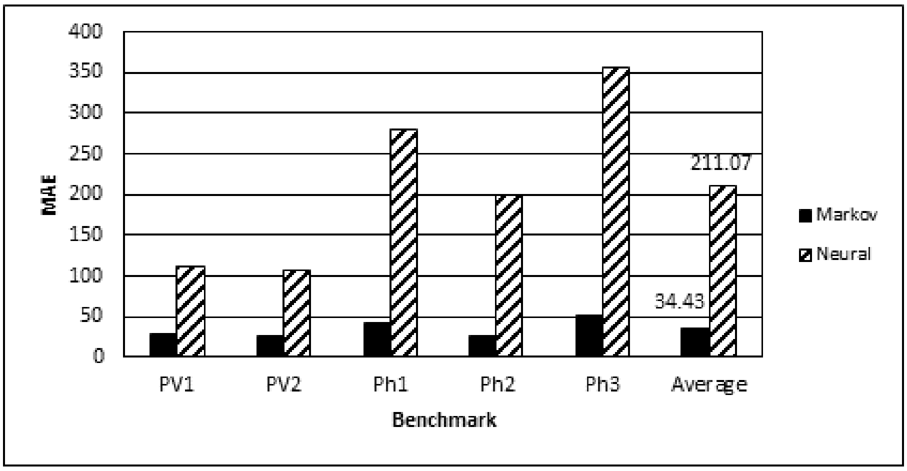


Figure 60. Comparing the Markov and neural predictors

As we can observe in Figure 59, the stride predictor has a considerably higher MAE than the Markov predictor. Due to the weakness of the stride predictor, the more complex hybrid predictor is not better than its Markov predictor component. Moreover, a hybrid predictor would induce higher costs than a single Markov predictor. Therefore, we can conclude that the Markov predictor is the most appropriate for electricity forecasting among the techniques analyzed in this section. In Figure 60 we compare our best method with the neural forecasting technique proposed in [52]. As Figure 60 illustrates, the Markov predictor provided significantly better results than the neural forecasting technique presented in [52]. Next, we will further analyze the Markov predictor by comparing the real electric power values with the Markov chain-based predictions, on three days from the PV1 dataset.

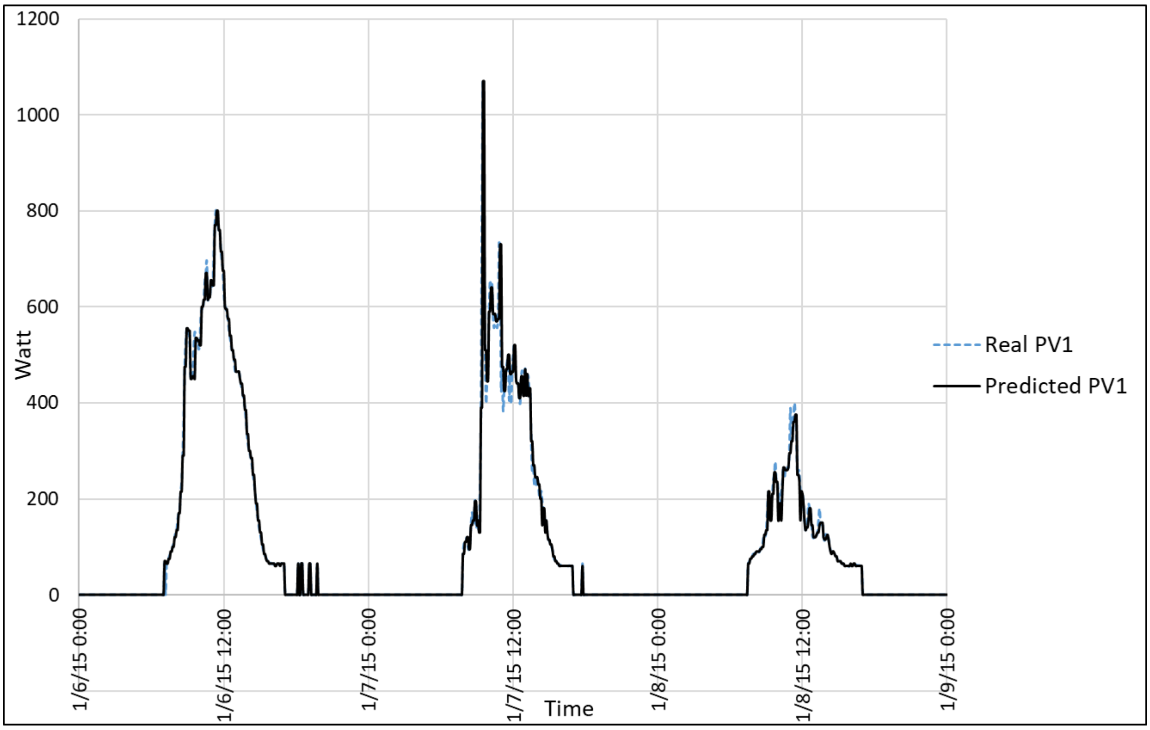


Figure 61. Comparing the real PV1 with the Markov predictions

As Figure 61 shows, the Markov chain can accurately forecast electric power, since the two curves – the real one and the predicted one – are almost completely overlapped. The above presented results validate the Markov predictor as a powerful electricity modelling technique, which can be used for decision making in intelligent energy management systems like FEMS.

4.1.4. Summary

In this section, we proposed a method for intelligent energy management in buildings, aimed at reducing uncertainty about the demand of electricity and its production from renewable sources. Within the framework of a decentralized energy production infrastructure, a network of networks where components are influencing each other, technology must support coordination, communication, and control. Predictions contribute to balance and smoothen the electricity intake from the power grid, with desirable consequences on both the operation of distribution grids and the stability of prices.

Systems that can be deployed as lightweight hardware devices reduce costs and facilitate diffusion and integration with existing infrastructure. We have applied Markov chains, stride prediction and also hybrid prediction to forecast electric power values based on previous values. We have implemented the Markov chains in an efficient manner, avoiding trees, graphs and transition tables so that an adaptation of the mechanism into miniaturized hardware will be not only possible, but very easy. We have further decreased the state complexity of the Markov chain by preprocessing the input data. We evaluated our methods on both produced and consumed electricity recorded by a real energy management system. The mean absolute error measured on the above-mentioned datasets was 34 W. Thus, Markov chains proved their ability to anticipate electricity production and consumption and can be integrated into energy management systems and immediate integration with IoT is contemplated. Moreover, we have shown that our context-based predictor outperforms the ANN-based prediction method – one of the main methods used for prediction of electricity consumption in buildings even if it has disadvantages like slow convergence, fluctuations, and oscillation during training [225].

A possible limitation of our prediction algorithm, that will be treated in a future approach, is the lack of environmental-specific input parameters. Knowing some information about the building's surface, temperature inside and outside of it, humidity, day of the week (workday or not), holiday or not, and weather characteristics like wind speed, may influence the prediction algorithm to increase its accuracy. These data will be fetched from weather stations or from environmental protection agencies. Measurements from locally installed sensors can also be used to automatize and trigger some activities taking into account some outside meteorological conditions like temperature or wind intensity.

In addition, further study will be dedicated to recurrent neural networks and Hidden Markov models used as predictors in energy management systems. Big data and cloud computing favour studying the electricity consumption on long term. Keeping large data sets for analysis leads to more accurate results.

4.2. Electricity Production and Consumption Prediction through Long Short-Term Memory

In this section, we analyze the efficiency of a Long Short-Term Memory (LSTM) (introduced by Hochreiter and Schmidhuber in [108]) in forecasting the electricity consumption and production in a smart house [7].

The ANNs are composed of a multitude of neurons representing simple processing elements that operate in parallel [65]. A great advantage of the artificial neural networks is their capacity to learn based on examples (supervised learning). In order to solve a problem traditionally, we have to elaborate its model, and after that we have to indicate a succession of operations that represents the solving algorithm of the problem. However, there are practical problems with a high level of complexity, and for this kind of problems it is very hard or even impossible to establish a deterministic algorithm.

In the connectionist models like neural networks, we are not forced to give a solving algorithm dedicated to a certain problem; we have to offer to the ANN only a multitude of consistent examples in order to learn and generalize them. The network extracts the information from the training samples. In this way, it is able to synthesize implicitly a certain model of the problem. In other words, the neural network builds up alone an algorithm to solve a problem. The capacity of the neural network to solve complex practical problems using a multitude of samples gives them a highly large potential of applicability.

LSTM is a Recurrent Neural Network (RNN). RNNs represent one of the most optimal choices when working with data organized in time series models. Their work principle is based on combining nonlinear activation functions in a recurrent structure, which makes prediction possible and provides improved prediction accuracy, as stated in [1]. In contrast to the standard neural networks, which are usually represented using feedforward architectures, RNNs allow the information to be transferred both forward and backward, with the help of their feedback connections. Therefore, these neural networks benefit from the ability to work with dynamic data. An analysis regarding the applicability of RNNs for prediction purposes is presented in [162].

4.2.1. Related Work

The authors in [50] proposed a short-term approach to forecast the total power consumption of buildings using ANNs. The specificity of their approach consists in the prediction mechanism and its target. The multilayer perceptron neural network uses in the training process the type of day (labor activity parameter). Thus, for the prediction of consumption in a certain day the ANN will be trained only using days of the same type and weather characteristics as well. The goal of prediction is to accurately determine the load curve of energy consumption by identifying the energy process of each individual consumer, that will be aggregated later. The advantage is that each end user is independently related to network input parameters (schedule, weather, etc.). The solution has been tested at the University of Valencia, a complex institution with more than 60 buildings and whose energy consumption is about 11.5 MW, similar to the one of a big commercial consumer.

Optimization of energy consumption can also be achieved by adapting production to consumer demand. An important drawback of microgrids is represented by the dynamicity and irregularity of load curves in contrast to larger environments like national or regional power grids, more stationary from this point of view. In this sense, for microgrids a smart prediction mechanism is required to model the relation between consumption and demand of energy, taking into account all the parameters that influence this process, and it must be able to quickly adapt to any changes. In [106], the authors describe a three-tier architecture for load prediction in microgrids. The first layer includes a self-organizing map (SOM) dedicated to classifying patterns of electricity consumptions based on historical data. On the second layer, the K-means clustering algorithm is applied to the SOM-generated partitions. The last layer consists in the multilayer perceptron which predicts the load curve for each cluster. Weather characteristics, day and month type are used as ANN inputs for predicting the electricity demand. The model was validated with data from microgrids situated in Castile and León, Spain, facilitated by the Iberdrola company. SOM-based prediction is applied also in [26]. There, a complex model is described that forecasts PV power using a radial basis neural network which is trained through a system composed of the K-means clustering, nearest-neighbor and least squares methods. The SOM is used to classify the inputs of the weather predictions. The average daily values of parameters like solar irradiance, wind speed, humidity, and temperature represent the neural network's inputs, while the predicted daily power of PV plant is the output.

In [112], the power values produced by a small-scale solar PV panel are used as inputs for a multilayer perceptron using the Levenberg-Marquardt learning algorithm, aiming to find the time horizon with the best prediction accuracy of the generated electricity. For April, the best time horizons were 5 minutes for short term prediction and 35 minutes for medium term prediction while the best time horizons relative to August were 3 and 40 minutes, respectively. Due to small variations of the climatic parameters in August, the prediction of electricity with ANNs is facilitated. In this case, the prediction could be determined by averaging the power of PV panels on wider time horizons.

In [4], the authors predict the power one hour ahead using ANNs. The inputs of the ANN consist in the solar irradiance and the temperature, each value being predicted by a distinct nonlinear autoregressive neural network. Such dynamic ANNs have the great advantage that they can correlate the output not only with the current input, but with previous inputs, too. The proposed method was validated on the data of a PV generator from the University of Jaen, Spain.

Two experiments regarding a grid-connected photovoltaic (GCPV) plant of 20 kWp are presented in [145] and [12]. In the first one, the power generated by the GCPV plant is predicted with the help of two complex neural predictors. The system was installed and tested on the roof top of the Italian city Trieste. Two ANNs are tested – a multivariate and a univariate one – aiming to determine the influence of climate conditions on the functional regime of the GCPV. The main difference between them is that the univariate model receives as an input parameter just the solar irradiance while the multivariate model also considers air temperature. In [12], two forecasting methods are combined: a time series method based on Seasonal Auto-Regressive Integrated Moving Average (SARIMA) and the SVM method. The hybrid model performs better than each method separately.

Another system which anticipates the energy provided by a GCPV plant is presented in [53]. The predictive mechanism consists in a chain of three modules. The first two make predictions of meteorological parameters starting from coarse-grained coverage of environmental data (global forecasting information provided by the Spanish national center) and continuing with more accurate information (fine-grained weather prediction for points situated in the neighborhood of the location). The final module of the system that performs energy prediction over a 39-hour interval (15 hours from the first day and the whole next day) consists in different types of predictors like k-nearest neighbors (k-NN), multilayer perceptron, or time series models.

In [150], Monteiro et al. evaluate short-term statistical prediction of photovoltaic electricity production. Two models are proposed: one of them is analytical and the other one is using an MLP. The prediction relies on weather forecasting tools focused on the location of the photovoltaic plant, as well as on hourly recorded photovoltaic electricity production. The analytical model computes the sky irradiation based on hourly radiation forecasts and adjusts it with irradiation attenuation index and photovoltaic production attenuation index. The neural network was selected and configured using genetic algorithms and is using weather forecasts as input information. The proposed models were evaluated and compared on the same data collected from a grid-connected photovoltaic plant. The authors concluded that the two models have similar results and both are usable in the sight of selling electricity to the markets.

In [51], Fan et al. evaluated the hybrid prediction through data mining techniques of the next-day energy consumption in buildings. The proposed method has three steps. In the first step, an outlier identification and removal is performed. In the second step, a recursive feature elimination is applied in order to use the optimal inputs for the eight different predictors. In the final step, an ensemble model is optimized for the predictors through a genetic algorithm. The authors concluded that the proposed ensemble model can be efficient in fault detection.

4.2.2. Electricity Prediction through Long Short-Term Memory

An RNN can have multiple layers, steps or stages. Their work principle is described in Figure 62. Each stage from the above scheme corresponds to a given time T. The RNN at the time T+1 will use the RNN from the time T as one of its inputs. Each stage will send its output to the next stage.

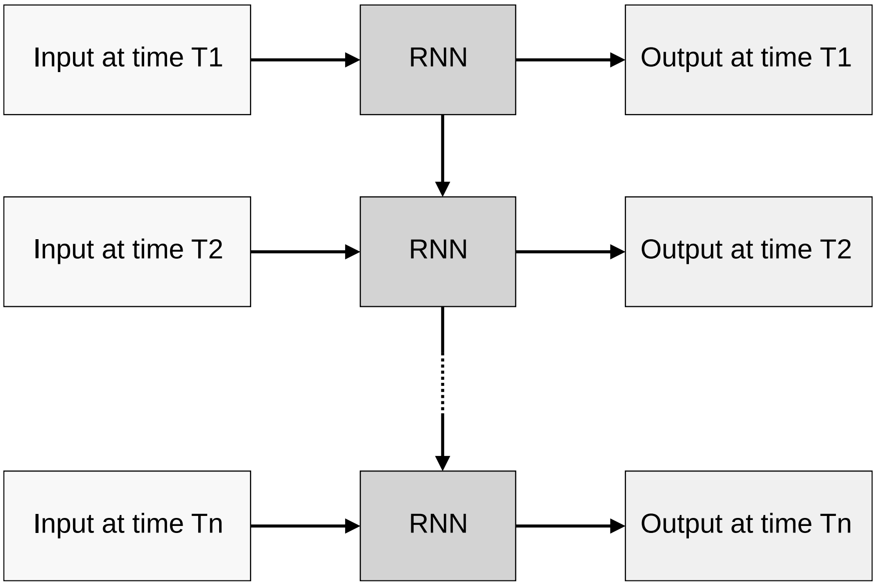


Figure 62. RNN

The key mechanism which makes the RNN to work well is represented by the hidden state information propagated from a certain stage to the next. The hidden state works as a memory capable of retaining information of the current stage. A layer from the RNN is processing the input data and is returning its internal state which is going to be used as an input in the next stage. More specifically, each stage is trained to transform the target sequence from a moment T into the input sequence but with a T+1 timestep offset. To achieve this, a backpropagation algorithm is used. The value of the loss obtained for each parameter is used to change the parameter values in the reverse direction with the purpose of minimizing the loss. As this movement is time based, each timestep contains its own loss value. In the process of modeling the dependencies between value sequences, the gradient of the timestep T depends on the gradient of the timestep T-1 and so on, and because of this, the further we progress with the timesteps, the gradient of the latter timestep matters less and less. This is known as the "vanishing gradient problem" whose effect is that the network cannot learn from long term dependencies, because the gradients of the early stages become smaller. LSTM networks are a solution to this problem.

LSTMs are RNNs that work with data that varies through time or sequentially, like language, stock market prices, weather recording sensors, etc. The way they work is similar to other RNNs, by using the outputs of a layer at a timestep T as inputs for the same layer at a timestep T+1. They have a component that acts as a memory which helps to transfer information learned at the timestep T to the next timesteps, and they can also forget irrelevant information from the preceding state and update the current state, allowing only important parts of the state to reach the output. The networks use activation functions to induce nonlinearity to the data. Among the most used activation functions are the sigmoid and the hyperbolic tangent. As in our datasets we had no negative values, we decided to use the sigmoid function in our LSTM network, as the interval of this function is [0,1]:

$$f(x) = \frac{1}{1+e^{-x}} \tag{28}$$

As mentioned before, LSTMs are capable of remembering and choosing which data is relevant as future inputs. They do this by using three gates that release data between hidden state and cell state. These gates are called "forget gate", "input gate" and "output gate". An LSTM neuron incorporates a cell, an input gate, an output gate, as well as a forget gate. The transformation process of information passing through a cell is described in [97], as follows. All the gates of the cell are collecting activations from the block and from the outside. A recurrent connection with the weight 1 keeps the current internal state of a cell. The input and output gates scale the input and output of the cell by using activation functions. The forget gate decides which information must be eliminated from the cell state. That is a sigmoid layer, which provides output values between 0 and 1, and scales the internal state, so as the values exiting the gate are ranged in the interval mentioned above.

For the purpose of our experiment [7], we implemented the LSTM network using Python and the TensorFlow framework with the Keras API. As input data, we used the datasets provided by the FENECON Energy Management System (FEMS) described in [52].

4.2.3. Experimental Results

In this section we focus on the results we obtained from our experiment presented in [7]. We tuned the LSTMs parameters in an effort to try and find the best configuration that would produce the smallest value for the MAE. We started with a standard configuration of 5 inputs, two hidden layers each containing 50 neurons, a learning rate of 0.01 and 30 epochs. The first parameter that we varied was the number of neurons in the first hidden layer, going from 5 values to 10, 25, 50 and 100, leaving the rest of the configuration unchanged. Due to the fact that the LSTM provides slightly different results in different runs because of its random initialization, we ran each dataset through the network 5 times for each changed parameter and calculated the average of the obtained MAE values. By increasing the input number, we noticed that the MAE value was increasing. After a series of experiments towards this direction, we concluded that 10 is the optimal value, and thus we obtained the MAE equal to 102.23 for this configuration. Figure 63 shows a graph with the values obtained following the tests.

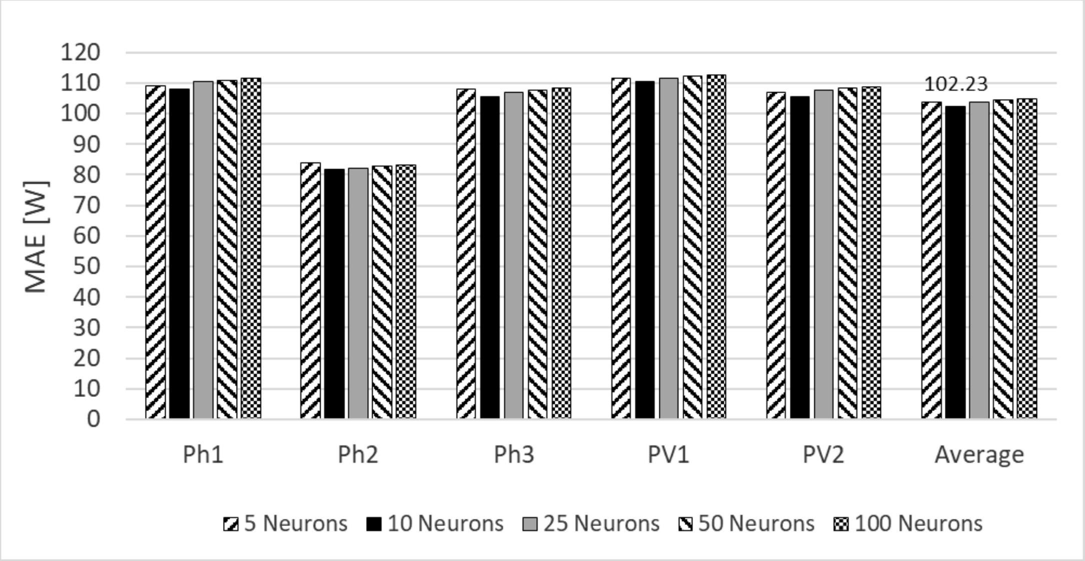


Figure 63. The influence of the number of neurons from the first hidden layer

Next, we varied the number of neurons from the second hidden layer, following the same pattern that we used for the previous varied parameter. Starting with the base configuration and adding the optimal value 10 for the first hidden layer, we experimented with the second layer starting with 5 neurons, then 10, 25, 50 and 100, and the smallest MAE value we obtained was 101.47, for 5 neurons on the second hidden layer. We established this value as being the optimal tune for this parameter. Figure 64 describes the results obtained by experimenting with the above-mentioned values through all the datasets.

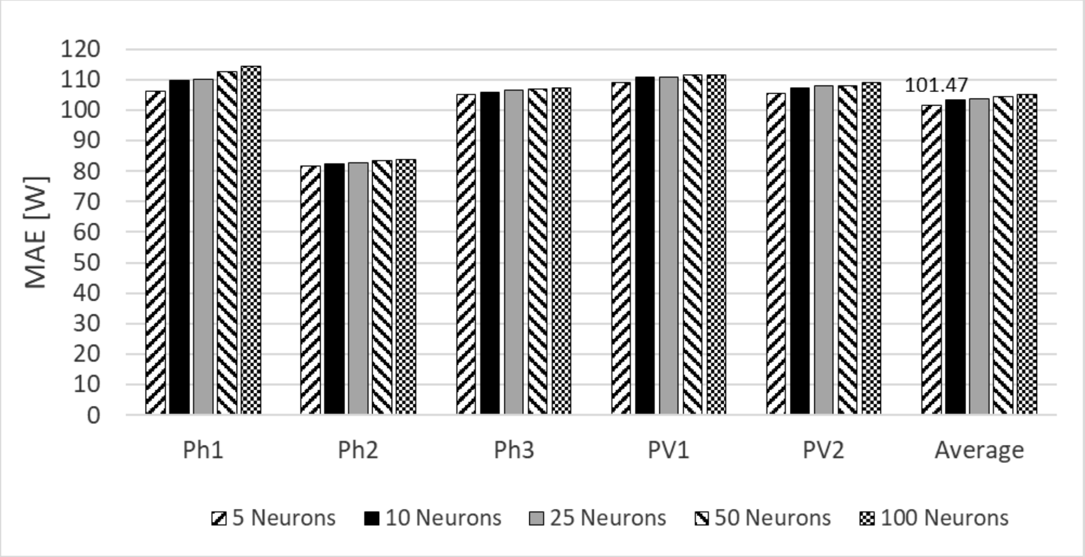


Figure 64. The influence of the number of neurons from the second hidden layer

The next parameter we tuned was the learning rate, starting from a value 0.01 and slightly increasing it to 0.02 and 0.03. We noticed that by increasing the learning rate, the MAE value also increased, to the point where we reached the value 107.1 with a 0.03 learning rate, so we decided to stop increasing it. The optimal configuration here is with a value of 0.01, having a MAE of 101.47, which means that this parameter already had an optimal value. The graph with the results can be seen in Figure 65.

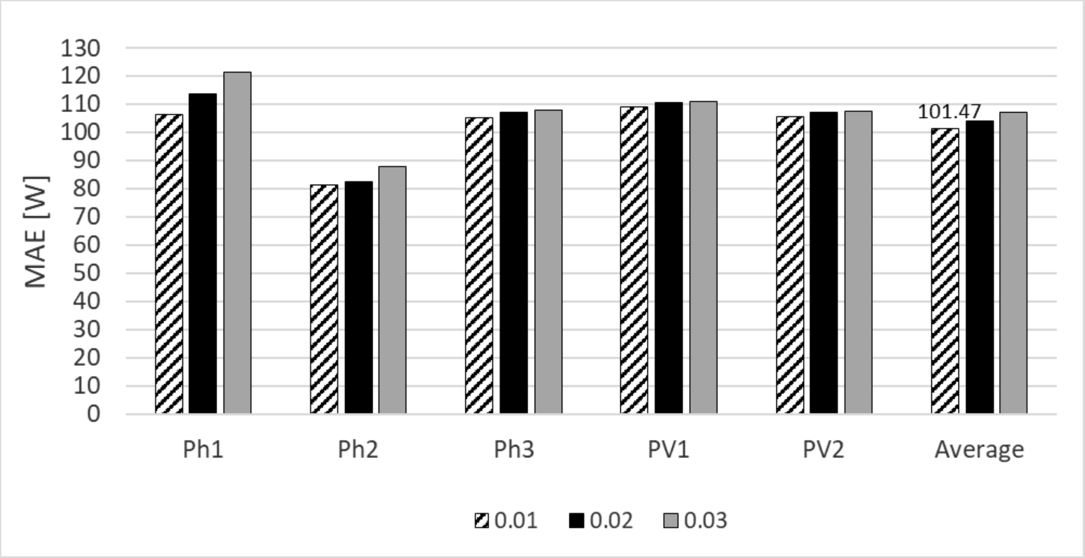


Figure 65. The influence of the learning rate

The next parameter we varied was the number of inputs. Having reached the MAE equal to 101.47 with our current configuration using 5 inputs, we increased the number to 10, 15, 20 and 25. We noticed that the higher the number of the inputs, the higher the value of the MAE became. So, we also decided to try a smaller number than the starting one and, thus, we went with 4 inputs. This proved to be the right decision, as we reached a MAE equal to 100.99. The results are visible in the graph from Figure 66.

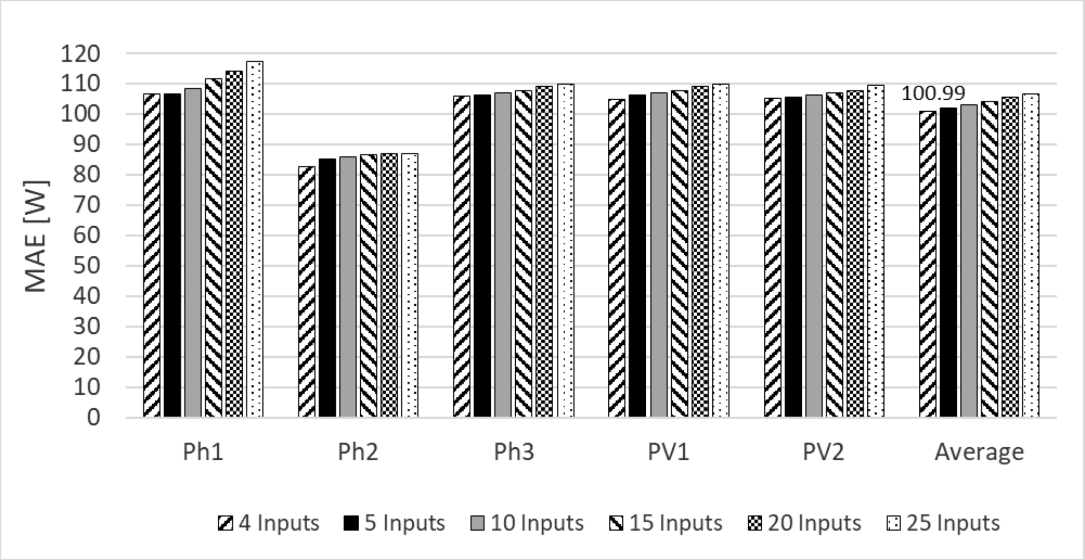


Figure 66. The influence of the input vector size

The last parameter that we decided to vary was the number of epochs. Our base configuration had 30 epochs which achieved the above-mentioned MAE, so we decided to increase this number. We varied through 50, 100 and 500 epochs. The results we obtained drove us to the conclusion that 50 epochs was the best configuration, having obtained a MAE equal to

100.77. We also tried to go below the starting value and we decided to run a series of tests with 25 epochs, but as we can see in the graph from Figure 67, the MAE was higher than the one we obtained with the optimal configuration.

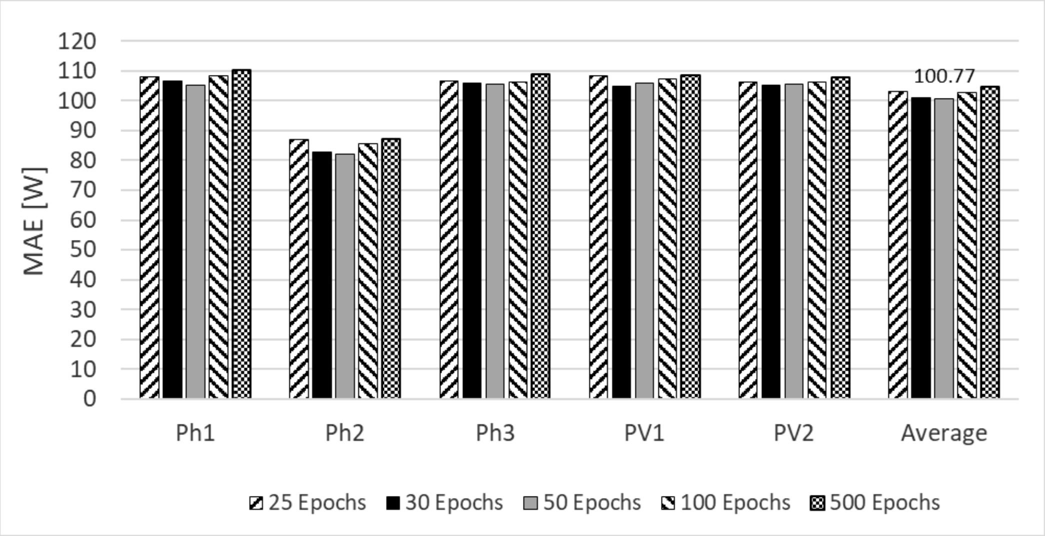


Figure 67. The influence of the number of epochs

After the experiment we concluded that the optimal LSTM configuration has 4 inputs, a first hidden layer with 10 neurons, a second hidden layer with 5 neurons, a learning rate of 0.01 and 50 epochs. Next, we made a comparison of our results with other methods used to calculate the MAE on the same datasets.

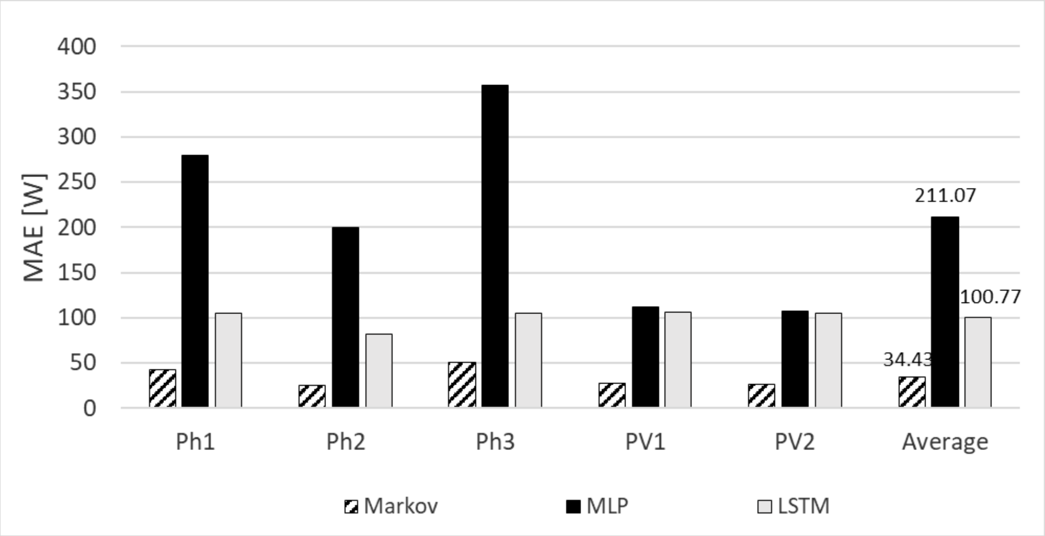


Figure 68. Comparison with other forecasting methods

As Figure 68 portraits, with a MAE of 100.77 Watts, our LSTM network outperformed the MLP, which had a MAE equal to 211.07, but had poorer performance than a Markov predictor with MAE 34.43.

4.2.4. Summary

In this section, we analyzed the LSTM used as a predictor of the electricity consumption and production in a smart house. The goal is to integrate such a predictor into a smart energy management system of a household, that might keep a balance between the electricity consumption and production avoiding demands from the grid. The evaluations performed on the datasets collected from a real household have shown that the LSTM’s mean average error is 100.77 Watts, which is half of the mean average error encountered by the MLP. The LSTM proved to be less accurate than the Markov predictor, but we can classify it among the best methods.

5. Image Restoration through Prediction Models

This chapter will present prediction-based image restoration methods proposed by us in [76], [77] and [79]. The well-known Markov chains were adapted to work with pixel intensities from 2D image areas. First, we will focus on predictive impulse noise filtering and we will evaluate different configurations of the proposed Markov model. Finally, we will adapt this Markov model to be applied for image inpainting.

5.1. Predictive Noise Filtering

As digital images are affected by noise during their acquisition or transfer, we are proposing a context-based method to eliminate salt-and-pepper noise from grayscale images, with an improved prediction scheme based on Markov chains. Salt-and-pepper is an impulse noise, consisting in white and black pixels altering the image. Our main goal is to restore the missing information and to preserve the unaffected pixels. Therefore, we are replacing the noisy pixel with the intensity having the highest number of occurrences in similar contexts within a limited surrounding area, like in a Markov chain. By using context information, our proposed filter can rebuild details in images altered by salt-and-pepper noise. In [77], we have applied a complete search in the limited surrounding area of the context, consisting in all the neighbor pixels. In [76], we have continued our research by studying different search methods and different context shapes.

For validation, we have compared our technique with several denoising methods from the current literature, by measuring the mean square error (MSE) on some well-known test images like “Cameraman”, “Boat” and “Airplane”. The experiments have shown that our Markov filter significantly outperforms many existing impulse noise filters.

5.1.1. Related Work

The median filter is one of the most employed methods to reduce impulse noise, with the drawback of being suitable only for low noise levels. Therefore, different improved median filter variants have been proposed over the years, which worked better on high noise densities. In [11], the authors proposed a method to overcome the shortcomings faced by the classical median filter at high noise densities, by considering only those pixels that are informative in the neighborhood. A filter employing two stages was proposed in [129], the noisy pixel being detected in the first stage, and replaced by the mean value of a 2×2 area noise-free pixels in the second stage. In [191], Srinivasan and Ebenezer proposed a decision-based method, which applies a 3×3 denoising window only on black and white pixels. A modified decision-based unsymmetrical median filter is proposed in [49], replacing the noisy pixel by the trimmed median value of the non-noisy pixels. When all the pixel values are 0 and 255, the noisy pixel is replaced by the mean value of the entire window. In [140], the authors recommend a modified directional-weighted-median filter to reconstruct images corrupted by salt-and-pepper noise. If the central pixel of a certain window is classified as noisy, it is replaced by a weighted median value on an optimum direction. In [102], the authors have introduced another median filter-based method, which relaxes the order statistic for intensity substitution. The authors of [213] have presented the progressive switching median filter, which applies through several iterations an impulse noise detection algorithm and filtering. Wang et al. presents in [214] a modified switching median filter, employing a two-phase denoising method. In the first phase, the adaptive vector median filter detection identifies pixels likely to have been corrupted by salt-and-pepper noise. In the second phase, the noisy candidates are evaluated by using four one-dimensional Laplacian operators, which allows edge preserving. The proposed approach can effectively preserve thin lines, fine details and edges. A soft-switching median filter for impulse noise removal was presented in [40], while Jassim [116] is proposing a Kriging interpolation filter to reduce salt and pepper noise from grayscale images. First, a sequential search is performed using k×k window size to determine non-noisy pixels. The non-noisy pixels are then passed to the Kriging interpolation method to predict their absent neighbor pixels detected in the first phase as being noisy. The experimental results are showing that the Kriging interpolation filter can achieve noise reduction without damaging edges and details.

In [200], the authors have introduced a two-level noise-adaptive fuzzy switching median filter. It identifies in the first stage the noisy pixels based on a histogram and replaces in the second stage the noisy pixels with the median of uncorrupted pixel values, applying also fuzzy reasoning. In [223], the authors introduced an adaptive progressive filtering technique, which detects corrupted pixels based on two-dimensional geometric and size features of the noise. Based on the result of the first stage, an adaptively sized and shaped filtering window (which in our work is fix sized and shaped) is employed in the second stage. Another two-stage scheme has been presented in [155] by Nasri et al., with noise detection in the first stage and Adaptive Gaussian Filtering in the second stage. The pixels detected as being noisy are stored in a binary noise matrix. The uncorrupted pixels from a fixed-size window are weighted by a Gaussian function. Then the denoised pixel intensity is computed as the normalized sum of these weighted values. In [133] Lin identifies impulse noise with Support Vector Machine and removes it with a fuzzy filter. Nair and Shankar [153] make use of a neural network to identify impulse noise in corrupted images and a modified median filter to remove the detected noise. The authors of [195] present another hybrid technique implying a neural network in the detection stage and a switching filter in the removal stage.

The main difference between the above-described methods and our denoising scheme is that we use context information and therefore we can reconstruct better the details in the corrupted images. Our Markov filter could be also applied for defect detection, as in [128]. Other context-based filters have been also proposed. In [120], the authors presented a probabilistic denoising technique consisting in Markov-Chain Monte Carlo sampling. A method employing a dissimilarity measure for the local neighborhood of the noisy pixel was presented by Berkovich et al. [8]. The content-based kernel uses a statistical model to exclude dissimilar intensities from the weighted average. The kernel was adjusted to the image content to preserve the edges or textures.

Universal filtering algorithms, which can be used on different types of noise, have been also proposed. Such a universal noise removal algorithm [27], working on both Gaussian and impulse noise, is introducing the spatial gradient into the Gaussian filtering framework for Gaussian noise removal and integrate their directional absolute relative differences statistic for impulse noise removal and combine them into a hybrid noise filter. Another two-stage filter which removes mixed impulse and Gaussian noise is proposed in [229]. Besides the very common impulse noise, a Poisson type noise distribution was analyzed by Mishra et al. [148]. This variety of noise is present especially in medical x-Ray imaging and affects low intensity regions. A modified version of the Bilateral Filter was introduced, followed by a performance comparison. In [189], Smolka and Kusnik presented a robust local similarity filter to reduce mixed Gaussian and impulse noise from the affected images. In order to determine the distortion level of a pixel, they compute the similarity of the pixels from the processing region and a small filtering window centered on the pixel being restored, as a sum of the smallest distances.

The denoising operation is generally affecting areas with discontinuities, producing an unwanted smoothing effect. In the paper of Rouf and Ward [182], the fact that chromatic discontinuities have lower gradients than luminance was used in order to restore image areas affected by noise. The method can be employed to recover deleted information and improve the denoising process. In the same direction, the Sorted Switching Median Filter presented in [109], is a three-stage filtering process that classifies the pixels and avoids the smoothing effect on uncorrupted areas. The multistage filtering process has been also employed by Liu et al. [135], with a new statistical process called ROD-ROAD and a fuzzy logic rule for the pixel classification at first, followed by a weighted mean filtering.

5.1.2. Filtering Impulse Noise Images with Markov Chains

Markov chains can be applied to compute the probability of a certain value in a sequence, as its number of occurrences in a considered context. Markov chains have been used in different computer science fields like bioinformatics [113], web access mining [82], pervasive computing [66], image retrieval [211], computational linguistics [151], etc. In an Rth order Markov model, the probability of the current state is computed based on R previous states, as follows:

$$P[q_t | q_{t-1},\ q_{t-2},\ ...] = P[q_t | q_{t-1},\ ...,\ q_{t-R}] \tag{29}$$

where q_t is the state at time t and R is the order of the Markov chain. A general prediction algorithm with Markov models, determining the next state of a 1D sequence based on the transition frequencies from the current state, was described in [92].

In [77], we reconstructed the grayscale images corrupted by impulse noise using Markov chains adapted for pixel intensities from 2D areas. The probability of pixel intensity in a certain context is computed as the number of its occurrences in similar contexts. The noisy pixel must be replaced with the predicted next state. The surrounding pixel values constitute the context and the search area is encoding the previous states. Thus, in grayscale images, the states are pixel intensities from the [0, 255] interval. The adjusted Rth order Markov model is given in (30), where CS is the context size (the width of the context square) and W and H specify the image width and height. Since the context is surrounding one pixel, its size can have only odd values. A certain pixel intensity $q_{x,y}$ depends on the neighbor context intensities. We have considered noisy the black and white pixels, as in [191].

$$\begin{aligned} &P[q_{x,y} | q_{i,j}, i = 0,...,W-1, j = 0,...,H-1, without\ (i = x\ and\ \ j = y)] = \\ &= P\left[q_{x,y} \middle| q_{x+i,y+j},\ i, j = -\frac{CS}{2},...,\frac{CS}{2},\ 0 \le x+i < W,\ 0 \le y+j < H, without\ i = j = 0 \right] \end{aligned} \tag{30}$$

Equation (30) implies searching the contexts in the entire image, which leads to a major disadvantage from the timing point of view. Therefore, we limit the search area, based on the search distance SD, as follows:

$$\begin{aligned} &P[q_{x,y} | q_{x+i,y+j}, i, j = -SD,...,SD,\ 0 \le x+i < W,\ 0 \le y+j < H, without\ i = j = 0] = \\ &= P\left[q_{x,y} \middle| q_{x+i,y+j},\ i, j = -\frac{CS}{2},...,\frac{CS}{2},\ 0 \le x+i < W,\ 0 \le y+j < H, without\ i = j = 0 \right] \end{aligned} \tag{31}$$

The adjusted Markov filter given in (31) is depicted in Figure 69, where the noisy pixel N is colored with black, the context pixels C are dark gray and the search area is light gray. A noisy pixel is replaced with the most frequent noise-free intensity occurred in similar contexts within a larger surrounding area limited by SD (without leaving the image boundaries). As it can be observed in Figure 69, in [77] we have applied a full search within the search area (limited by SD) of a full context consisting in all the neighboring pixels. Further, we denote that filter S0_C0.

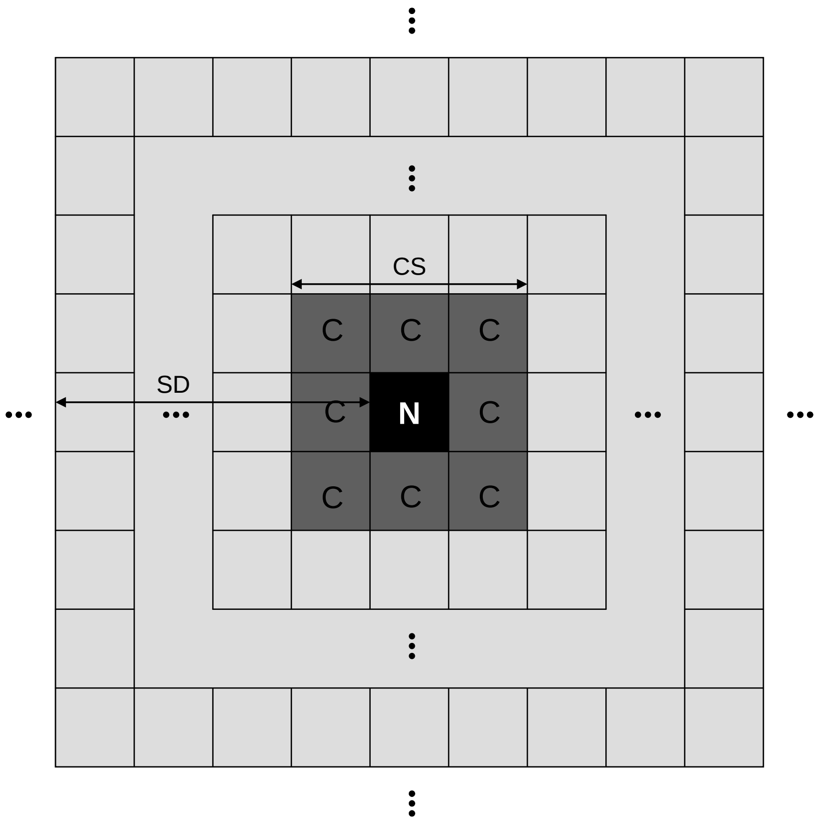


Figure 69. Image denoising with the S0_C0 *Markov filter*

In this section, we have investigated other simpler search rules and also different simpler context shapes. The goal is to improve the denoising performance and speed of the context-based filter. We tried to replace the full search (used in [77]) with a search in form of "+", "X" and also their combination in form of "*". We tried also to replace the full context (used in [77]) with different context shapes: the context in form of "+", "X" and their combination in form of "*".

The most efficient combination (determined on the test images), search in form of "*" for contexts in form of "+", is given in (32). This Markov filter is denoted S*_C+. Despite equation (32) seems more complicated than (31), in fact it is significantly simpler because it implies processing fewer pixels and, thus, we expect a faster filtering. We will also evaluate comparatively (31) and (32) and other variants in terms of denoising performance. Figure 70 presents the S*_C+ Markov filter. The considered context pixels are highlighted with dark gray (forming a "+") and the search rule with light gray (forming an "*").

$$
\begin{aligned}
&P[q_{x,y}\big|q_{x,y+j},\, j=-SD,...,SD,\ 0\le y+j<H, without\ \ j=0;\\
&\quad q_{x+i,y}, i=-SD,...,SD,\ 0\le x+i<W, without\ i=0;\\
&\quad q_{x+k,y+k}, k=-SD,...,SD,\ 0\le x+k<W, 0\le y+k<H, without\ k=0;\\
&\quad q_{x-l,y+l}, l=-SD,...,SD,\ 0\le x-l<W, 0\le y+l<H, without\ l=0]=\\
&=P\left[\, q_{x,y}\middle|q_{x,y+j},\ j=-\frac{CS}{2},...,\frac{CS}{2},\ 0\le y+j<H, without\ \ j=0;\right.\\
&\quad\left. q_{x+i,y},\ i=-\frac{CS}{2},...,\frac{CS}{2},\ 0\le x+i<W,\ without\ i=0\right]
\end{aligned}
\tag{32}
$$

Obviously, we have implemented and tested all the following combinations in which S denotes the search type and C the context type: S0_C0 [77], S0_C+, S0_CX, S+_C0, S+_C+, S+_CX, SX_C0, SX_C+, SX_CX and S*_C+.

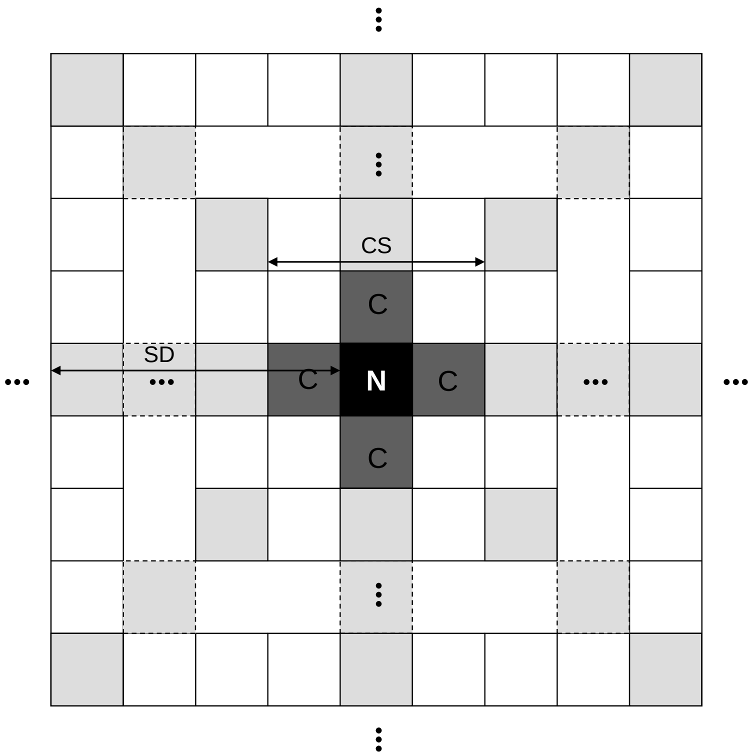


Figure 70. Image denoising with the S*_C+ *Markov filter*

The algorithm which replaces a noisy pixel through the S*_C+ Markov model is described in the following pseudocode:

```
Markov(x, y, CS, SD, T)
  For j:=y-SD to y+SD, 0≤j<H
    If j=y then Continue
    If SAD(x, y, x, j, CS)<T
      AND NOT Salt_Pepper(x, j) then
        Q[Color(x, j)]:=Q[Color(x, j)]+1
  For i:=x-SD to x+SD, 0≤i<W
    If i=x then Continue
    If SAD(x, y, i, y, CS)<T
      AND NOT Salt_Pepper(i, y) then
        Q[Color(i, y)]:=Q[Color(i, y)]+1
  For k:=-SD to SD, 0≤x+k<W, 0≤y+k<H
    If k=0 then Continue
    i:=x+k
    j:=y+k
    If SAD(x, y, i, j, CS)<T
      AND NOT Salt_Pepper(i, j) then
        Q[Color(i, j)]:=Q[Color(i, j)]+1
  For k:=-SD to SD, 0≤x-k<W, 0≤y+k<H
    If k=0 then Continue
    i:=x-k
    j:=y+k
    If SAD(x, y, i, j, CS)<T
      AND NOT Salt_Pepper(i, j) then
        Q[Color(i, j)]:=Q[Color(i, j)]+1
  If Q[Max(Q)]=0 then Return Color(x, y)
  Return Max(Q)
```

The parameters of the Markov function are: the line and the column of the current pixel, the context size CS, the search distance SD and the similarity threshold value T. The first two for instructions are performing the “+” search and the last two the “X” search for similar contexts. These two search rules used together constitutes the “*” search. We considered two image areas similar if the sum of absolute differences is less than T.

The following pseudocode presents how we compute the similarity degree as a Sum of Absolute Differences (SAD) in the S*_C+ Markov filter, the context having a “+” shape:

```
SAD(x1, y1, x2, y2, CS)
  S:=0
  For j:= -CS/2 to CS/2, 0≤j+y1<H, 0≤j+y2<H do
    If j=0 then Continue
    S:=S + |Color(x1, j+y1)-Color(x2, j+y2)|
  For i:= -CS/2 to CS/2, 0≤i+x1<W, 0≤i+x2<W, do
    If i=0 then Continue
    S:=S + |Color(i+x1, y1)-Color(i+x2, y2)|
  Return S
```

The first for instruction is processing the pixels from the vertical line and the second one from the horizontal line of the “+” context shape, both avoiding the middle pixel.

The frequencies of the noise-free pixel values occurring in similar contexts are kept in Q. The Max function returns the most frequent intensity which will replace the noisy pixel. The noisy pixel is not changed if the Markov function cannot find any similar context. The Salt_Pepper function checks if a pixel is noisy, by returning TRUE for black and white pixels. The Markov_Filter function, which calls the previously presented Markov function, is the same as in [77] and it is presented in the following pseudocode:

```
Markov_Filter(CS, SD, T)
  For i:=0 to W-1 do
    For j:=0 to H-1 do
      If Salt_Pepper(i, j) then
        Set_Color(i, j, Markov(i, j, CS, SD, T))
```

where the Set_Color function changes the intensity of the noisy pixel (i, j) with the value returned by the Markov function.

5.1.3. Evaluation Results

The proposed Markov filter was implemented in C# and we used for comparisons the available Matlab source codes of several filters. We performed the evaluations on the Cameraman, Boat and Airplane 512×512 grayscale PNG images having salt-and-pepper noise levels between 10% and 90%. The denoising performance has been determined using the MSE metric whose computation is given in (33):

$$MSE = \frac{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1}\left(F(i,j)-O(i,j)\right)^2}{W \cdot H} \qquad (33)$$

where W and H are the image width and height. The goal is to obtain the MSE as low as possible.

First, we have evaluated the S0_C0 filter by varying CS on a fixed SD=5 and T=500. As we have explained in 5.1.2, the CS can have only odd values and it must be at least 3. The MSE values obtained on the test images are presented in Figures 71-73.

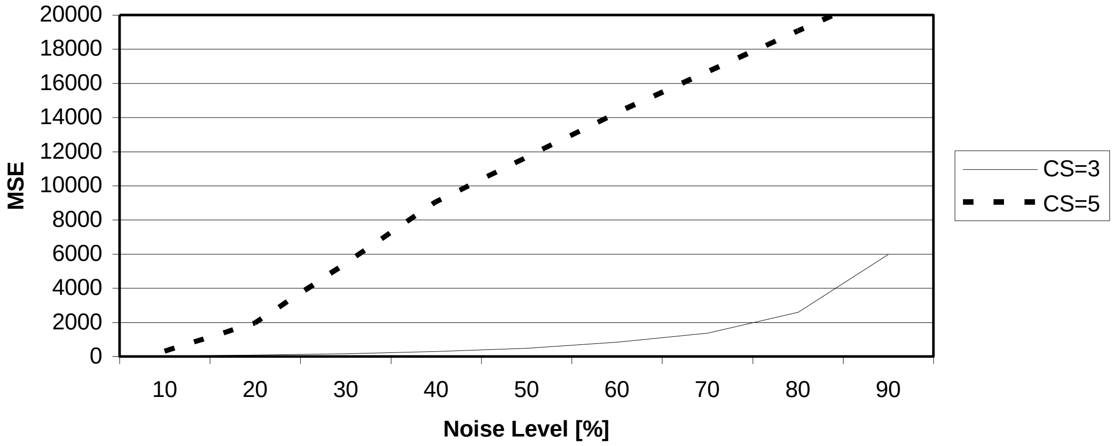


Figure 71. The MSE of the Cameraman image denoised using S0_C0 with different context sizes

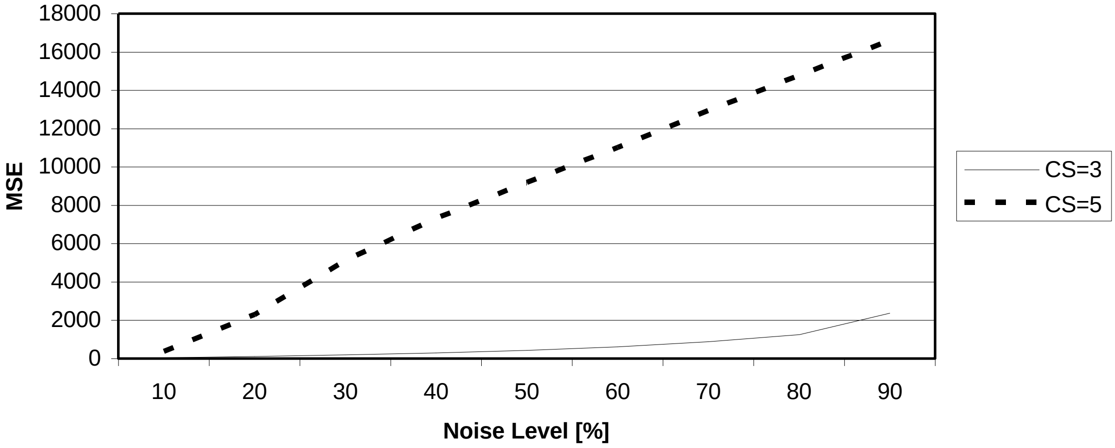


Figure 72. The MSE of the Boat image denoised using S0_C0 with different context sizes

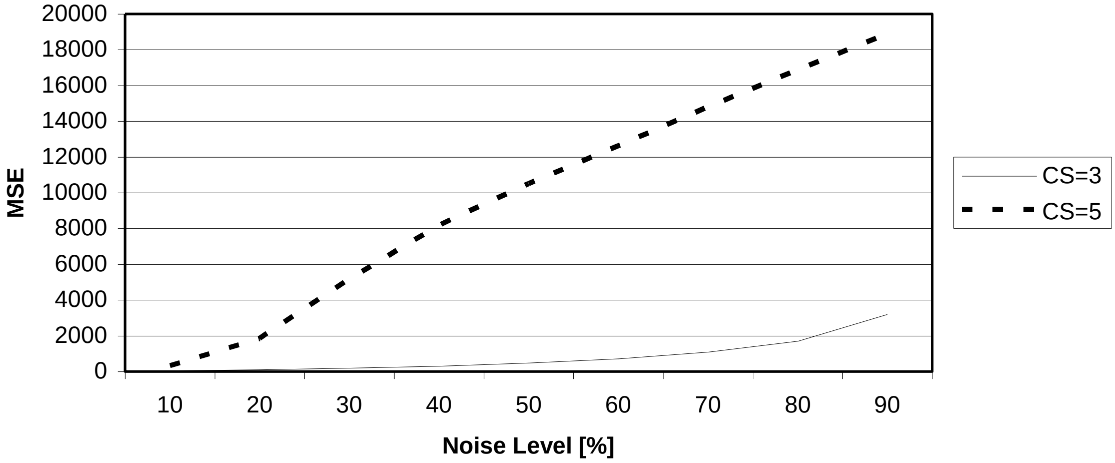


Figure 73. The MSE of the Airplane image denoised using S0_C0 with different context sizes

Figures 71-73 have shown that the best value for CS is 3, the S0_C0 filter being inefficient for higher contexts. A richer context leads to higher precision, but if it is too rich, the probability to find it is low. Therefore, usually the performance is increasing together with the context up to a certain size (which in our application is 3), after which it starts to decrease.

We have continued our evaluations by varying the search distance SD between 2 and 5, considering the best CS=3 and a fixed T=500. The MSE values obtained on the test images are presented in Figures 74-76.

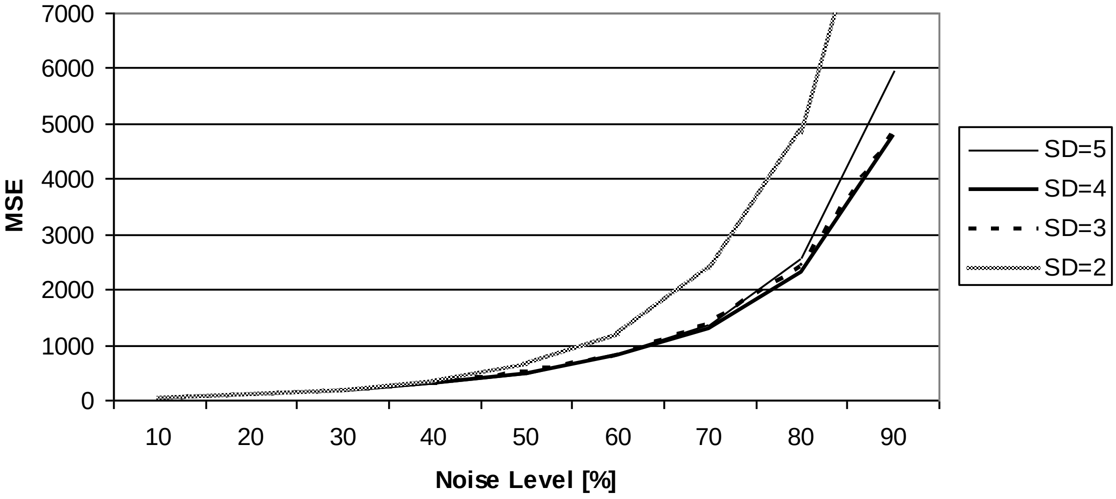


Figure 74. The MSE of the Cameraman image denoised using S0_C0 with different search radius values

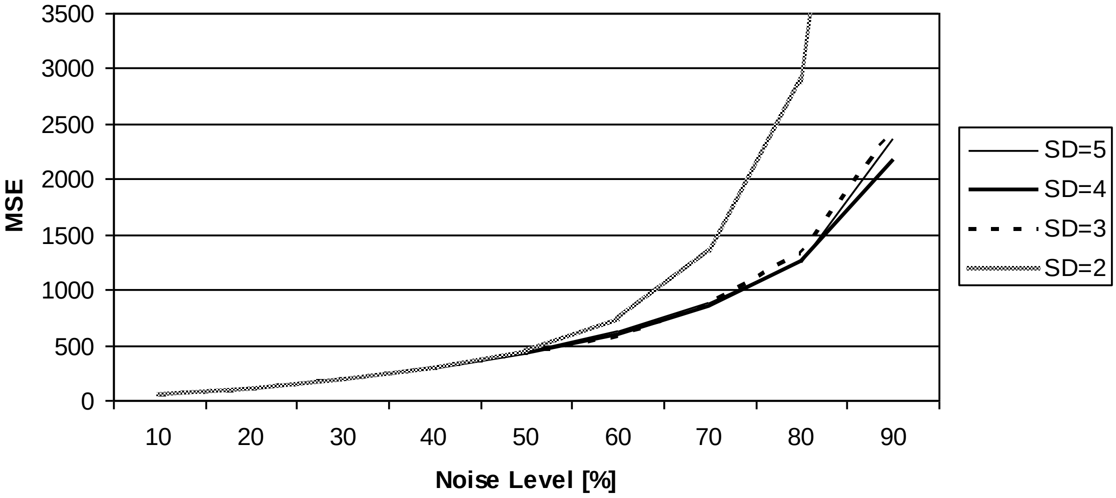


Figure 75. The MSE of the Boat image denoised using S0_C0 with different search radius values

One can observe that on the Boat image, an S0_C0 filter with SD value of 3 is better up to 60% noise level and for SD of 4 is better only starting with 70% noise density. On the Airplane image the SD of 2 is better up to 50%, while SD of 3 and 4 are very close and better starting with a noise of 60%. On the Cameraman image an SD of 4 performs best, it being just slightly outperformed by an SD of 2 on a noise up to 20%. Therefore, we consider that the optimal SD value will be 4. The conclusion after this evaluation step was that the search area might be sufficiently high to find the context, but if it is too high (SD≥5), the multiple pixel value choices can lead to uncertainty and thus to lower denoising ability.

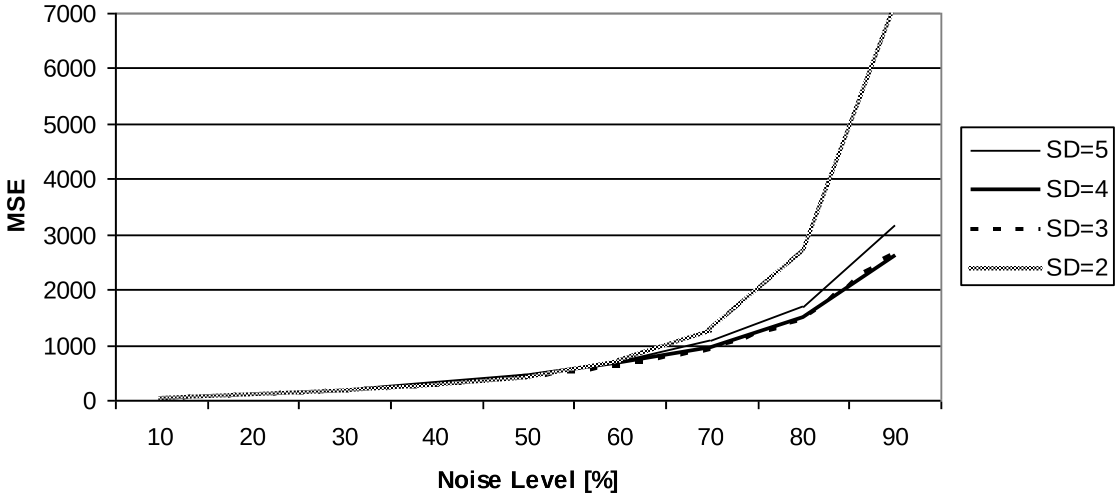


Figure 76. The MSE of the Airplane image denoised using S0_C0 with different search radius values

The next stage of our analysis consists in varying the similarity threshold T between 450 and 600, in steps of 50. As we have already explained, when we have searched for the context of the current noisy pixel, we have taken into account all the contexts whose similarity degree, computed as SAD, is less than T. Figures 77-79 present the MSE obtained for different similarity threshold values, considering the best CS=3 and the optimal SD=4.

Figures 77-79 showed that the best similarity threshold value is 500 up to 70% noise on the Boat image and even up to 80% noise on the Cameraman and Airplane images. Only on very high noise density, a threshold of 550 or 600 is slightly better. Therefore, we have considered that the optimal similarity threshold value will be T=500. A difference of 500 in the SAD between two compared image blocks, taking into account the best CS=3 (contexts of 8 pixels), results in a reasonable average per pixel difference of 62. Consequently, the optimal S0_C0 filter has SR=4, CS=3 and T=500.

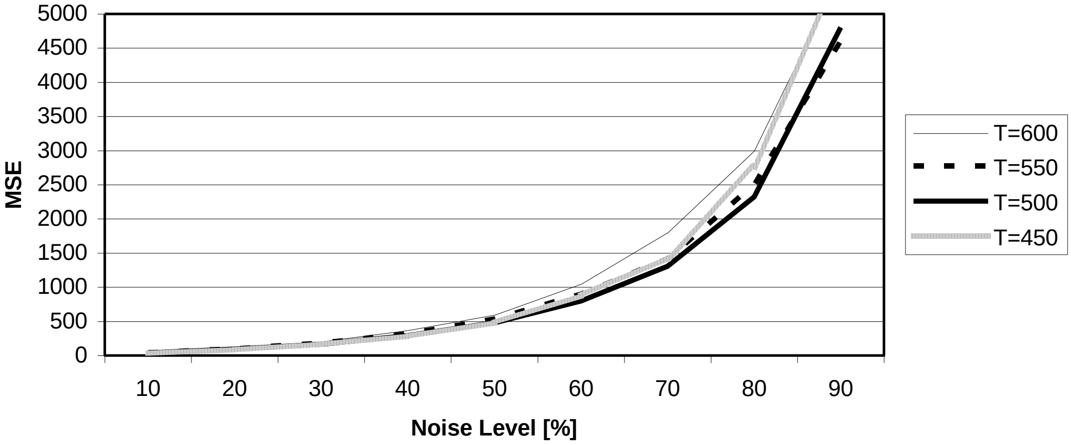


Figure 77. The MSE of the Cameraman image denoised using S0_C0 with different search similarity thresholds

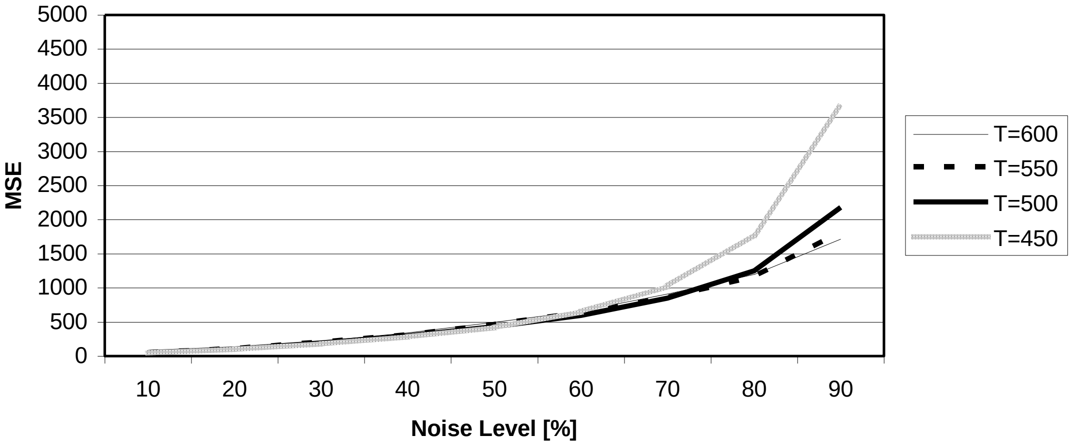


Figure 78. MSE of Boat denoised using S0_C0 with different search similarity thresholds

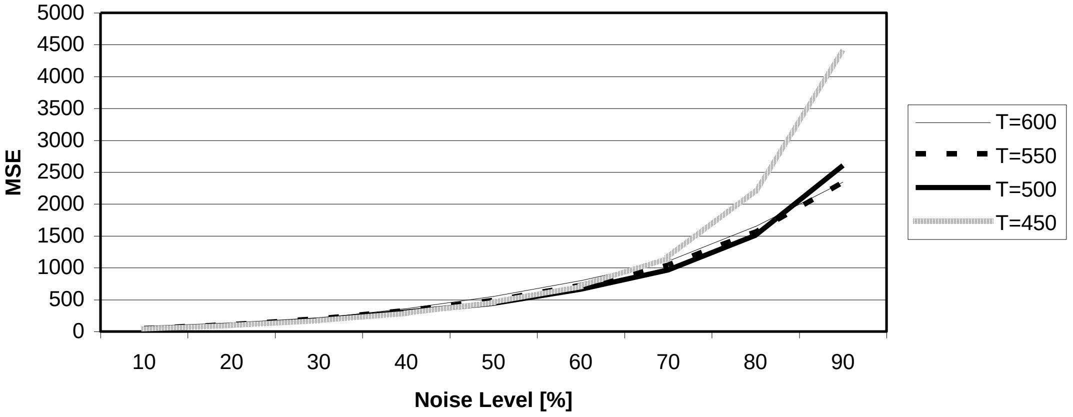


Figure 79. MSE of Airplane denoised using S0_C0 with different search similarity thresholds

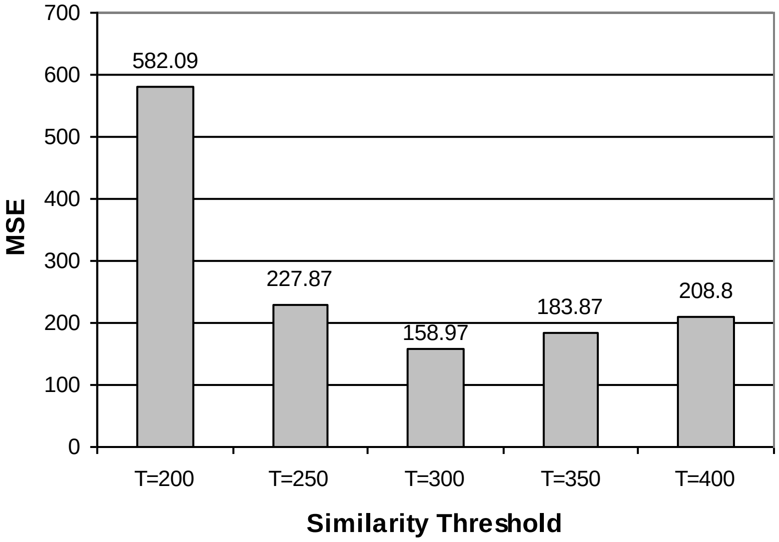


Figure 80. The MSE of the 30% noised Cameraman image filtered with S+_C+ *using different similarity thresholds*

Next, we analyze other simpler search rules and also different simpler context shapes. First, we have checked again the SD and CS parameters and the optimal values are the same as in [77]: CS=3 and SD=4. The optimal value of T for a full context was 500 in [77]. Since the number of pixels is reduced to the half in the "+" and "X" contexts, we expect a reduction of the optimal T to around 250 in the filters implying such contexts. In Figure 80 we have measured the MSE by varying the similarity threshold T around the expected optimal value. For this first parametrical setup we have chosen the S+_C+ Markov filter and the Cameraman test image with 30% noise. Figure 80 has shown that in the case of a "+" context the best value for T is 300. As we checked, for an "X" context the best T value is the same, which is obvious, since it implies the same number of pixels. Further we will use T=500 for a full context and T=300 for the "+" and "X" contexts.

Next, we have compared different search rule and context shape combinations. The MSE values obtained on the Cameraman, Boat and Airplane images are presented in Figures 81, 82 and 83, respectively. Since S0_CX was less performing than S0_C+ and also SX_C0 was less performing than S+_C0, we have checked but not included the other models that imply "X" search or "X" contexts (S+_CX, SX_CX and SX_C+) in these figures.

Noise density	Cameraman		Boat		Airplane	
	S0_C0	S*_C+	S0_C0	S*_C+	S0_C0	S*_C+
10%	9.38	2.29	10.63	2.63	10.51	2.41
20%	18.27	4.33	19.92	4.80	18.68	4.48
30%	26.56	6.27	27.88	6.74	26.64	6.39
40%	33.08	7.91	35.30	8.45	33.90	8.08
50%	40.02	9.49	41.87	9.96	41.00	9.68
60%	46.76	10.97	48.28	11.32	47.69	11.13
70%	53.21	12.45	54.43	12.59	53.81	12.50
80%	59.60	13.79	60.00	13.82	59.64	13.71
90%	65.95	14.88	66.51	14.84	66.38	14.82

Table 12. Comparison of the computation times in seconds

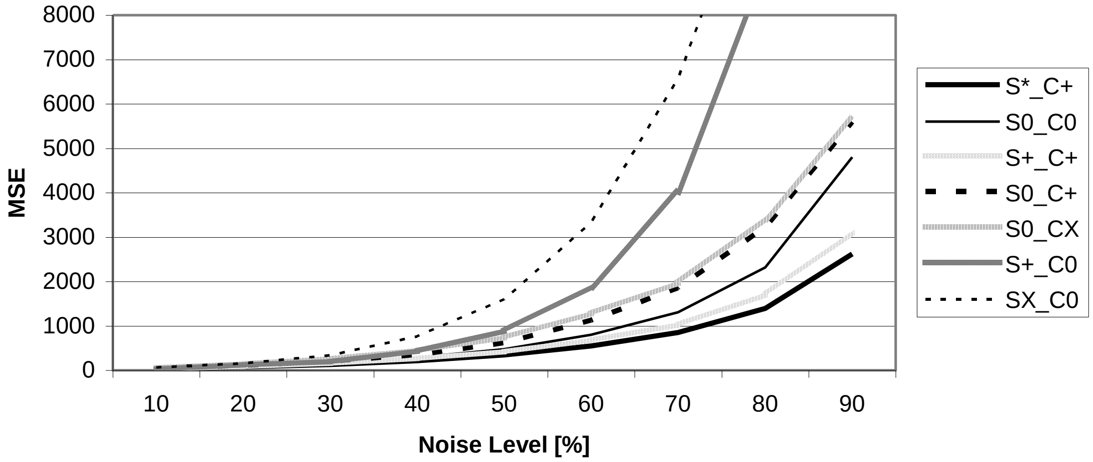


Figure 81. The MSE of the Cameraman test image denoised with different types of Markov models

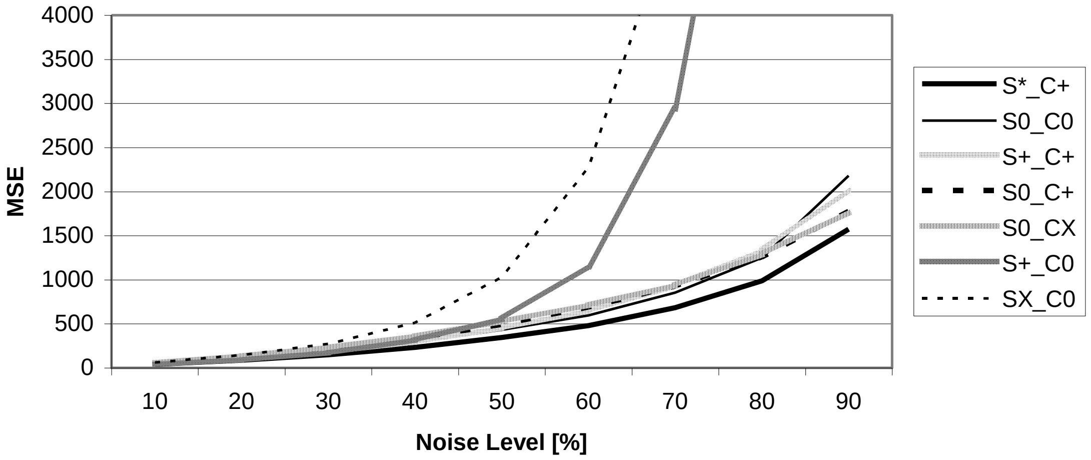


Figure 82. The MSE of the Boat test image denoised with different types of Markov models

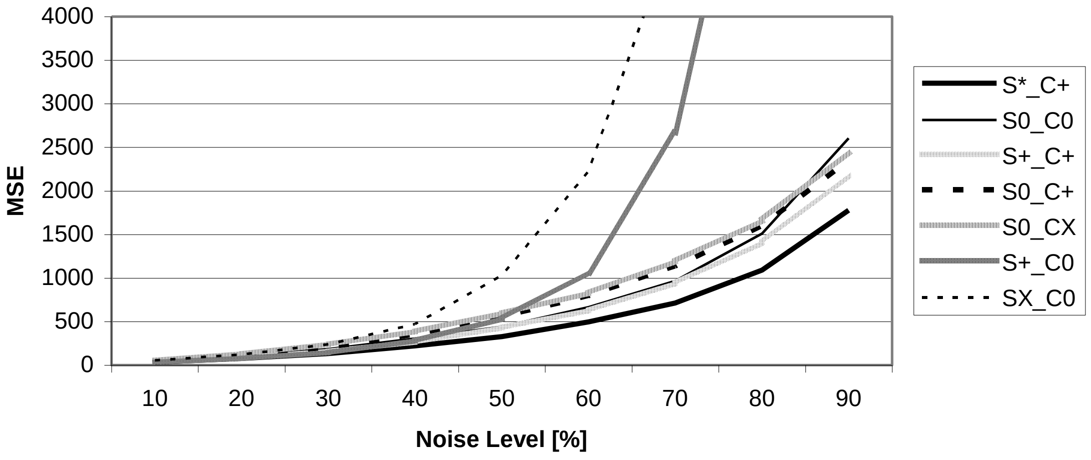


Figure 83. The MSE of the Airplane test image denoised with different types of Markov models

As Figures 81-83 show, the S*_C+ is the best Markov filter on all the three test images. Thus, even if an “X” search is less performing than a “+” search, their combination into “*” search provides the best results. The proposed S*_C+ model is significantly outperforming the initial S0_C0 filter from [77], on all the noise levels. The denoising speed also decreased on all noise levels. As Table 12 shows, the S*_C+ Markov filter is about four times faster than S0_C0.

The second best model is S+_C+, which is outperforming the initial S0_C0 on the Cameraman and the Airplane images but it is worse on Boat with noise between 40-80%.

Further, we have compared this best S*_C+ Markov filter with other existing filters: our S0_C0 filter [77], the Noise Adaptive Fuzzy Switching Median Filter (NAFSMF) [200], the Decision Based Algorithm (DBA) [191], the Median Filter (MF), the Progressive Switching Median Filter (PSMF) [213], the Relaxed Median Filter (RMF) [102] and the Analysis Prior Algorithm (APA) [142], whose source codes were available. Figures 84-86 are presenting comparatively the MSE for all the considered methods on the Cameraman, Boat and Airplane test images.

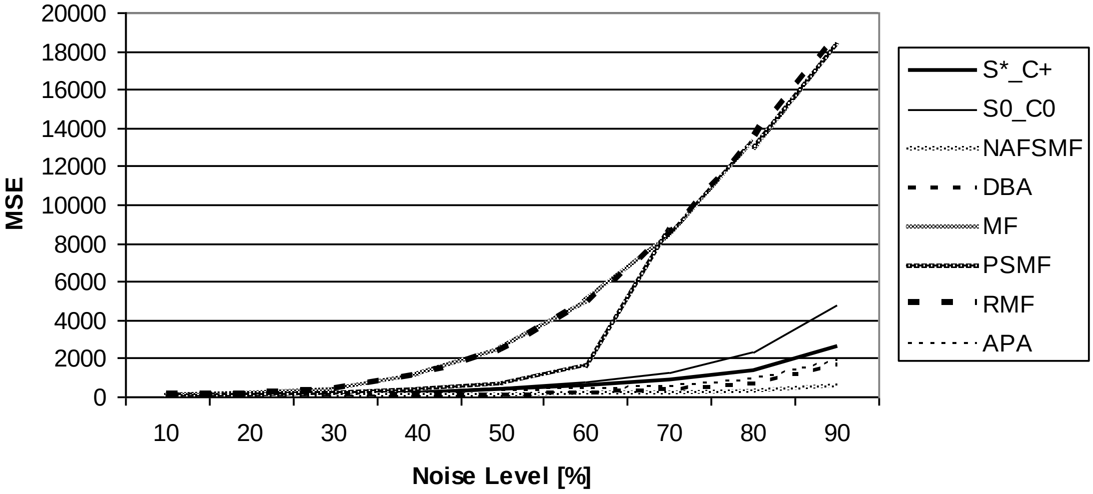


Figure 84. Comparing the MSE on the Cameraman image denoised with different existing methods

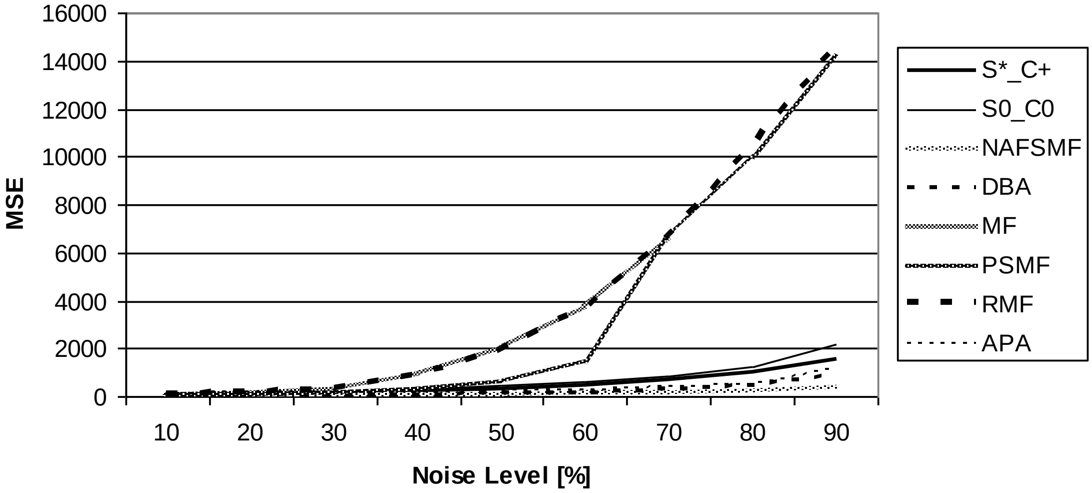


Figure 85. Comparing the MSE on the Boat image denoised with different existing methods

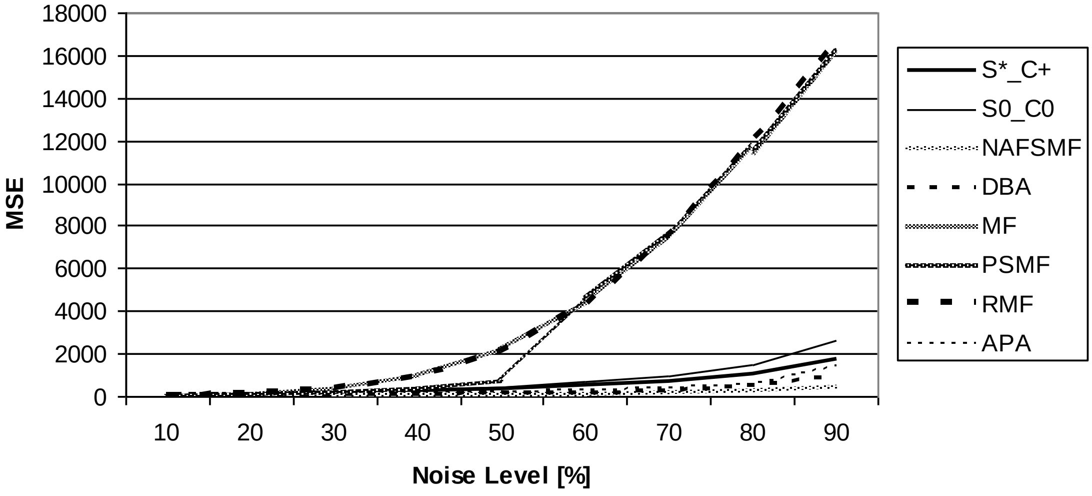


Figure 86. Comparing the MSE on the Airplane image denoised with different existing methods

Figures 84-86 show that the proposed S*_C+ Markov filter is better than the MF, PSMF, RMF and S0_C0 on all the noise levels. It is also better than APA on noise densities up to 30%. It is just slightly worse than the NAFSMF and DBA.

Figure 87 shows the Cameraman image having 60% salt-and-pepper noise (a) and the outputs obtained with our proposed S*_C+ Markov filter (b), as well as using S0_C0 (c), NAFSM (d), DBA (e), MF (f), PSMF (g), RMF (h), APA (i).

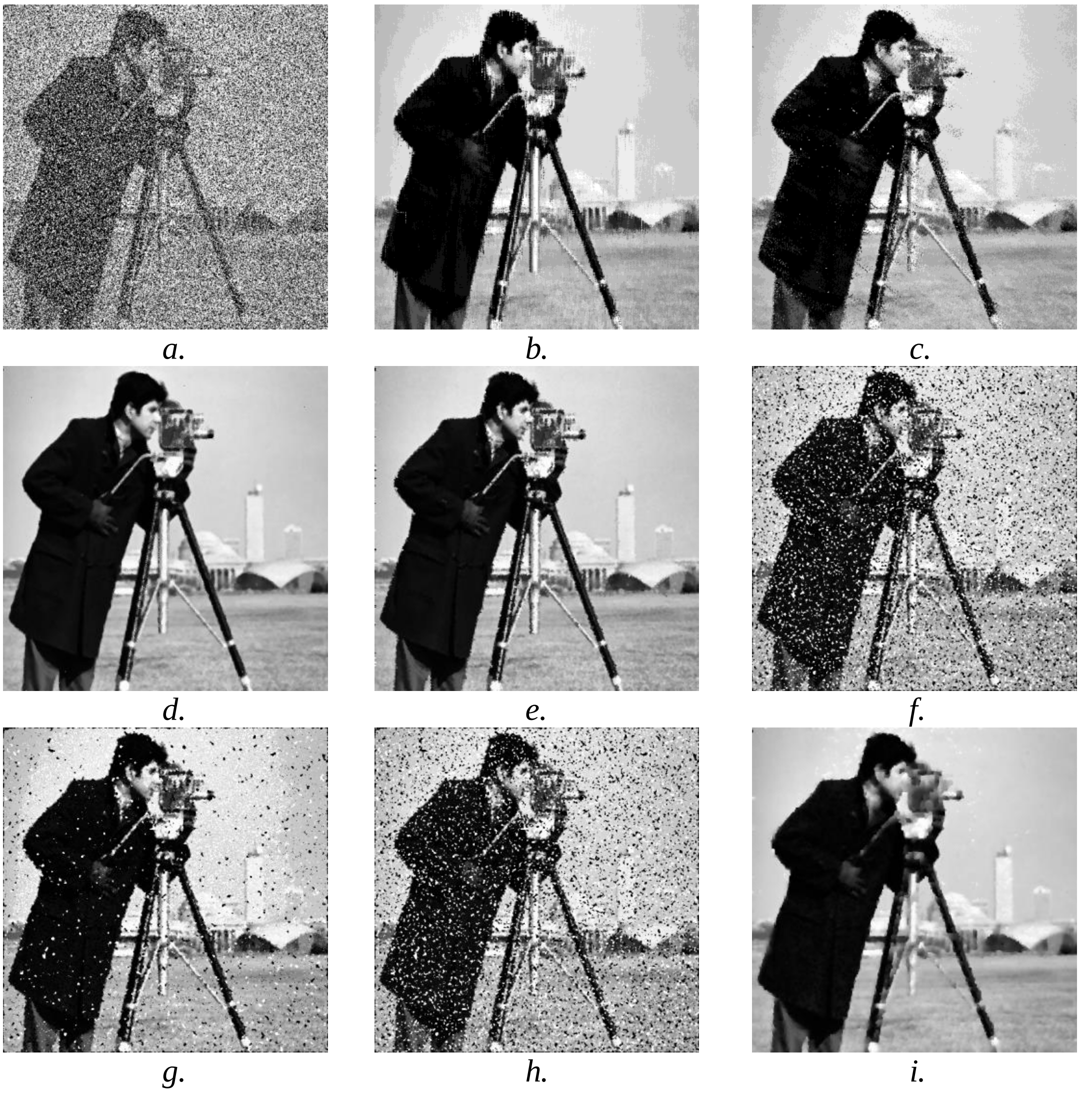

a. *b.* *c.*
d. *e.* *f.*
g. *h.* *i.*

Figure 87. Denoising the Cameraman image having 60% noise (a) using the S*_C+ *Markov filter (b), S0_C0 Markov filter (c), NAFSM (d), DBA (e), MF (f), PSMF (g), RMF (h), APA (i)*

As Figure 87 depicts, our proposed S*_C+ Markov filter is better than the S0_C0, MF, PSMF, RMF and APA techniques. We can observe again that the quality of the image denoised with the S*_C+ Markov filter is very close to the quality of the images filtered with NAFSM and DBA.

5.1.4. Summary

In this section, we have improved a context-based filter proposed in [77], to denoise grayscale images corrupted by impulse noise. Our filter is using Markov chains to replace the noisy pixel intensity with the pixel value having the highest number of occurrences in similar contexts. The context of a noisy pixel consists in the intensities of its neighbor pixels and is searched in a larger but limited surrounding area. The original contribution published in [76] consists in analyzing different search rules and different context shapes.

We have replaced the full search used in [77] with a search in form of "+", "X" and also their combination in form of a "*". We have also replaced the full context used in [77] with different context shapes: "+", "X" and "*". The MSE results obtained on the Cameraman, Boat and Airplane test images show that the most efficient model is the proposed S*_C+ Markov filter which applies the search in form of "*" of contexts in form of "+". This filter is better than our previous S0_C0 filter on all the noise levels, but also than the MF, PSMF, RMF and partially than APA and it is just slightly worse than the NAFSMF and DBA denoising methods. Beside the better denoising performance, the computational time has been also significantly improved with respect to the previous S0_C0 filter. The context information is a great advantage of our method, whereas the computational time, despite it was significantly improved, is still a slight disadvantage compared with some of the existing techniques.

A further work direction could try to adjust dynamically the search distance or the context size. Other research directions are the run-time computation of the similarity threshold proportionally with the context size and the utilization of the Markov filter together with fuzzy and neural techniques. Yet another further work direction is to make Markov filter usable on random-valued impulse noise, which implies an additional noise detection stage through machine learning techniques such as MLP, Long Short-Term Memory (LSTM) or fully convolutional networks.

5.2. Prediction-Based Image Inpainting

Inpainting is a technique which is replacing missing or affected parts from digital images or damaged films (regularly small areas). Three main categories of inpainting algorithms can be distinguished in the literature: structural, textural and combinations of both. All these methods are using the information from the unaffected areas in order to reconstruct the affected or missing areas. The structural inpainting is using geometrical operations to reestablish missing pixel colors. Textural inpainting algorithms are building up stochastic models based on the information from unaffected image areas and are using the obtained models to reconstruct the affected or missing areas, being thus able to restore textures, too. The combination of structural and textural inpainting in a hybrid approach, can exploit the advantage of both methods.

In this section, we are proposing a new context-based inpainting technique, which is relying on Markov chains to replace affected or missing image areas [79]. The area which must be reconstructed is defined by the user through points, which are connected afterwards by lines. The reconstruction is started from the exterior of the affected area, with pixels whose context (consisting in the surrounding pixels) is at least partially in an unaffected area. The unaffected context part is searched within a limited surrounding window. The affected or missing pixel is replaced with the color having the highest probability to occur in similar contexts or context parts. In the next iterations of the reconstruction process, the restored pixel colors can be used to restore other pixel colors. By using context information, the proposed method is appropriate to rebuild textures and details in images.

5.2.1. Related Work

Inpainting algorithms classified on categories, have been presented in [163], [99] and in [45].

A stochastic model based on Markov random fields has been used in [44], where the algorithm is reconstructing texture by starting from one pixel. Similar contexts are queried by using the Sum of Squared Differences (SSD) similarity metric. One of the intensities surrounded by contexts determined as being similar is randomly chosen to replace missing pixel. In contrast with this work, we use the Sum of Absolute Differences (SAD) as similarity metric and we build up a Markov chain model using the current context and similar contexts situated in a limited surrounding area. We replace the missing pixel color with the most probable color discovered in similar contexts. Thus, our method can better rebuild textures.

In [9], the authors proposed an inpainting method which restores the marked area by continuing in the same angle the isophote lines that arrive to the boundaries of the region. An important disadvantage of their technique is that it cannot restore textures [18]. Therefore, in [10] the authors are combining the structural inpainting algorithm presented in [9] with the texture synthesis algorithm proposed in [44], the results of both operations contributing to reconstruct the image.

Another technique which can replicate both structure and texture has been presented in [33]. In the proposed exemplar-based inpainting method, the reconstruction process is performed in the order given by the priority values computed for the patches along the fill front. High priority values are associated to the patches continuing strong edges and to the ones surrounded with high confidence pixels. The key difference between [33] and our work is that we apply pixel-level filling instead of patch-level filling. By focusing on a single pixel at each iteration of the algorithm, we expect to obtain better results. Exemplar-based inpainting using a locally linear neighbor embedding technique with low-dimensional neighborhood representation has been presented in [98]. In [6], the authors have analyzed the exemplar-based inpainting method from a variational viewpoint.

In [218], the authors presented a domain-based structural-aware image inpainting technique. They designed an iterative structure searching algorithm for structural restoration, by connecting the adjacent patches to form a repairing domain which assures coherency and accuracy. In [105], the authors proposed an inpainting method for images containing textures with gradually changed illumination. Based on an energy function model of the problematic area, gradually modified from the center to the boundary, a gradually changed directional priority function is used for the gradual propagation of texture synthesis. In [230], the authors combined a base reconstructor containing low-frequency information with a detail reconstructor containing high frequency information. The base layer can grasp the basic information, whereas the detail layer provides the local details. These two components are combined within a so-called base-detail generator. In [159], the authors proposed an inpainting technique which extends the isotropic diffusion model with diffusion barriers provided by the user. An improved model was introduced in [101].

Convolutional neural networks have been applied for image inpainting in [130] and [215]. In [136], the authors applied partial convolution by using only unaffected pixel intensities. Region-wise convolution have been combined in [141] with non-local correlation among regions. In [224], the authors proposed a hybrid inpainting approach which combines convolutional neural networks and multi-scale neural patch synthesis. In [222], the authors presented a foreground-aware image inpainting system which learns to predict the foreground contour and inpaints the missing region based on the predicted contour. The contour completion is performed by combining a generator and a discriminator used to encourage the generator to provide sharp contours. The generator is a cascade of a coarse network and a refinement network.

5.2.2. Inpainting with Markov Chains

The first step requests to the user to select the image area wanted to be reconstructed. The selection is performed by clicking on different points onto the boundary of the target area. These selected points are stored in a list and connected by lines. All the pixels falling inside and on this polygon defined by the user are considered as belonging to the target area which must be reconstructed. The selection is depicted in Figure 88 on the famous “cracked-plate” portrait of Abraham Lincoln, taken in 1865 by Alexander Gardner.

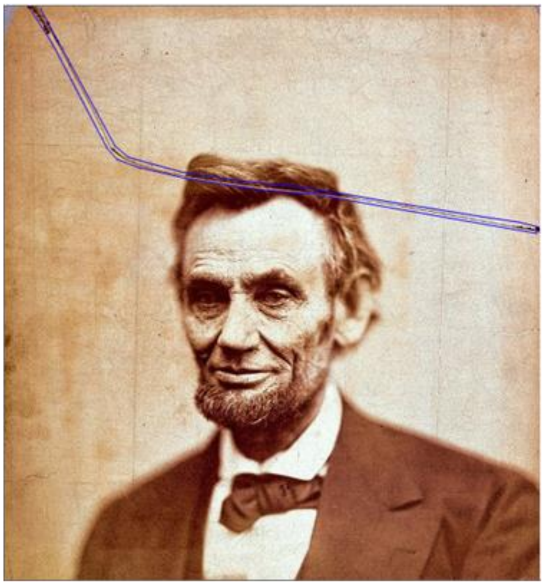


Figure 88. Image area selection for inpainting on the photo of Abraham Lincoln

The next step consists in using a stochastic method relying on Markov chains to reconstruct each pixel belonging to the selected target area. Markov chains can be used to determine the next probable value in a sequence, and have been successfully applied in bioinformatics [113], web mining [82], ubiquitous computing [75], speech recognition [151], image retrieval [211], image denoising [76], energy consumption modeling [73], etc. In a Markov chain of order R, the probability of the current state is depending on R previous states [77], as follows:

$$P[q_t | q_{t-1},\ q_{t-2},\ ...] = P[q_t | q_{t-1},\ ...,\ q_{t-R}] \tag{34}$$

where q_t is the state of the Markov chain at time t. A general prediction method relying on Markov chains, applied on 1D sequence, was described in [92]. In [76] and [77], we repaired grayscale images affected by impulse noise using Markov chains adapted for pixel intensities from 2D image areas. Similarly with the current work, the probability of a pixel color in a context was determined as the number of its occurrences in similar contexts.

For the inpainting, we have adapted the application to be able to work with colors instead of grayscale intensities. Thus, the states are pixel colors. Further, we adopted the notations already established in the inpainting literature, denoting with Ω the affected image area, with δΩ the boundary of the affected area, with Φ the search window and with Ψ_p the context of the replaceable pixel $p_{x,y}$.

In our proposed context-based method, the affected pixel color $p_{x,y}$ is replaced with the predicted color (next state). The unaffected surrounding pixel colors compose the context Ψ_p and the search window Φ is encoding the previous states. The adjusted Markov model of order R is presented in (35), where CS is the size of the context Ψ_p (more exactly its width, see Figure 89), SD is the search distance (used to define the search window Φ, as it is depicted in Figure 89), W is the width and H is the height of the image. As it can be observed, the color of a certain pixel $p_{x,y}$ depends on the neighbor colors (the context).

$$\begin{aligned} &P[p_{x,y} | p_{x+i,y+j} \notin \Omega, i, j = -SD, ..., SD,\ 0 \le x+i < W,\ 0 \le y+j < H, without\ i = j = 0] = \\ &= P\left[p_{x,y} \middle| p_{x+i,y+j} \notin \Omega,\ i, j = -\frac{CS}{2}, ..., \frac{CS}{2},\ 0 \le x+i < W,\ 0 \le y+j < H, without\ i = j = 0 \right] \end{aligned} \tag{35}$$

The adjusted Markov model from (35) is illustrated in Figure 89, where the replaceable pixel from the center of the window is marked with red and the unaffected context pixels from its vicinity are marked with green. That context is searched within the surrounding window (limited by SD without leaving the image). All the pixel colors (marked with red in the top-left corner of Figure 89) which are situated in similar contexts (marked with green in the top-left corner of Figure 89) are considered as candidates to replace the color of the affected pixel. The similarity is determined based on the SAD metric. After the search process, the color of the affected pixel is replaced with the most frequent unaffected color found in similar image contexts (without leaving the image boundaries).

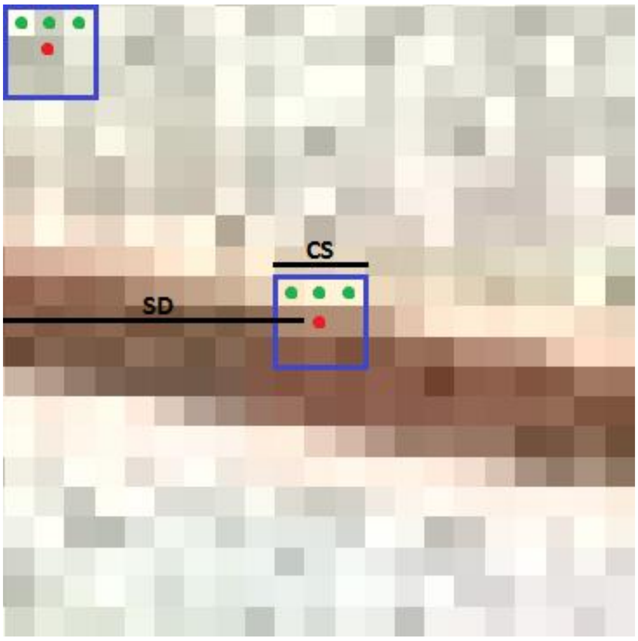


Figure 89. Image inpainting with the Markov chain (on an area extracted from the photo of Abraham Lincoln)

The next pseudocode presents the Markov function which replaces a certain pixel $p_{x,y}$ belonging to the affected area Ω:

```
Markov(x, y, CS, SD, T)
  For i:=x-SD to x+SD, 0≤i<W
    For j:=y-SD to y+SD, 0≤j<H
      If (i=x AND j=y) OR Pixel(i, j)∈Ω then
        Continue
      If SAD(x, y, i, j, CS)<T then
        F[Color(i, j)]:=F[Color(i, j)]+1
  Return Max(F)
```

The input parameters of the above presented Markov function are: the row and the column of the current pixel, the search distance SD, the context size CS and the similarity threshold T. Inside the for instructions, we are searching for similar contexts, avoiding obviously the current one centered in x, y. The similarity degree measurement applied between the context of the replaceable pixel p and the contexts of the candidate pixels q is given in (36):

$$SAD = \sum_{j=0}^{CS-1} \sum_{i=0}^{CS-1} \left| \Psi_p(i,j) - \Psi_q(i,j) \right|, \text{ without } i=j=\frac{CS}{2} \quad (36)$$

As lower the SAD value, as more similar the two compared contexts are. We have considered two contexts as being similar, if their SAD is less than T. The pseudocode of the SAD function, computing the similarity degree between two contexts, is defined as follows:

```
SAD (x1, y1, x2, y2, CS)
  S:=0
  For i:= -CS/2 to CS/2, 0≤i+x1<W, 0≤i+x2<W, do
    For j:= -CS/2 to CS/2, 0≤j+y1<H, 0≤j+y2<H do
      If i=0 AND j=0 then
        Continue
      S:=S + |I(i+x1, j+y1)-I(i+x2, j+y2)|
  Return S
```

The *for* instructions are summing the absolute differences between the pixels from the same position of the two contexts, avoiding the middle (which is not part of the context). The function can work with grayscale intensity (I) differences or with cumulative color component differences.

The frequencies of the unaffected pixel colors found in similar contexts are kept in F. The Max function returns the most frequent color which will be used to replace the color of the affected pixel $p_{x,y}$. The replacement is not performed if the Markov function fails finding similar contexts. Finally, the Inpainting function, which calls the Markov function, is given in the following pseudocode:

```
Inpainting(CS, SD, T)
  For each $p_{x,y}$ ∊ Ω do
    Set(x, y, Markov(x, y, CS, SD, T))
```

where the Set function is changing the color of $p_{x,y}$ with the color returned by the Markov function.

5.2.3. Evaluation

For a wider applicability and a higher processing speed, we have implemented our proposed inpainting method in a Windows Forms application, in C#.

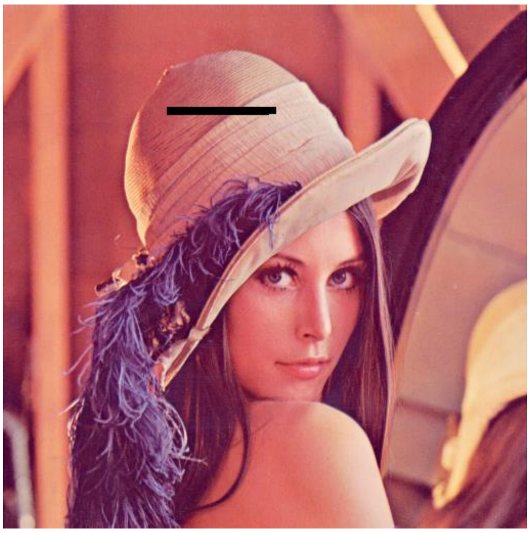


Figure 90. Artificial defect of 849 pixels on the Lena image

We performed the evaluations on the Lena, Peppers and Baboon color images, having artificial defects of different sizes. The defects were manually applied in arbitrary shape, size and position. Figure 90 presents an example of artificial defect (849 affected pixels) on the Lena image. We have also used in our evaluations the color photography of Abraham Lincoln, mentioned earlier in this section.

The inpainting performance has been determined using the mean square error (MSE) and the peak signal-to-noise ratio (PSNR) metrics. The MSE is given in (37):

$$MSE = \frac{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1}\left(R(i,j)-O(i,j)\right)^2}{W \cdot H} \tag{37}$$

where W and H are the image width and height, R is the repaired image and O is the original (obviously unaffected) image. The goal is to obtain low MSE values. The PSNR computation is presented in (38):

$$PSNR = 10 \cdot \log_{10} \frac{255^2}{MSE} \tag{38}$$

Our goal is to obtain high PSNR values. The MSE can be computed based on the PSNR values as follows:

$$MSE = 255^2 \cdot 10^{-\frac{PSNR}{10}} \tag{39}$$

The proposed Markov inpainting model has been configured step by step on the Lena, Peppers and Baboon test images. First, we have varied the SD parameter (and thus implicitly the size of the search window Φ), by maintaining CS on 3, and T on 300. The MSE and PSNR measurements are presented in Figures 91 and 92, respectively. It can be observed that the best SD value is 5. Lower values than 5 are not feasible since the search window Φ would be too small, with fewer chances to find the searched context. On the other hand, with high SD values, the search window Φ would be too large and not specific to the target area Ω which must be repaired.

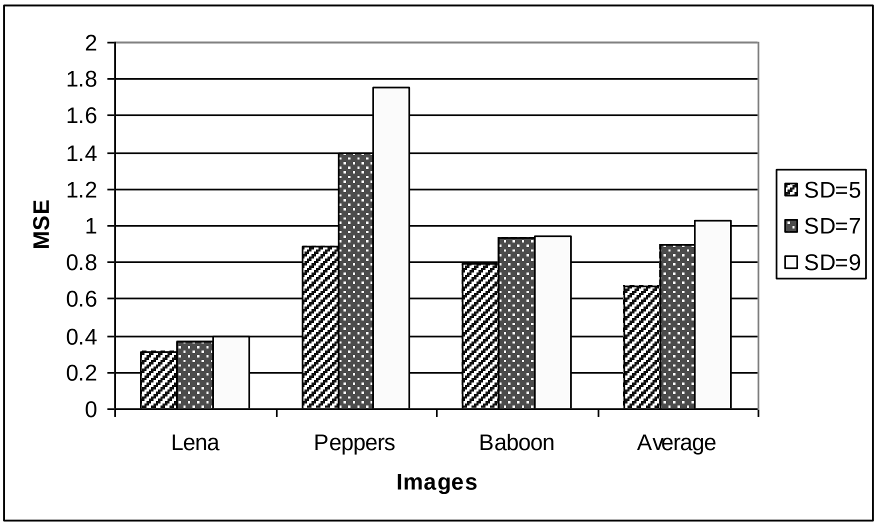


Figure 91. MSE by varying SD with fixed CS=3 and T=300

Next, we have varied the value of the CS parameter, by fixing the SD on 5 and still maintaining T on the initial value 300. The obtained MSE and PSNR values are depicted in Figures 93 and 94, respectively. As we can see, the best evaluated CS value is 7. Higher CS values would increase the processing time and would also reduce the chances to find a large context in a small search window Φ, already fixed in the previous step by SD=5.

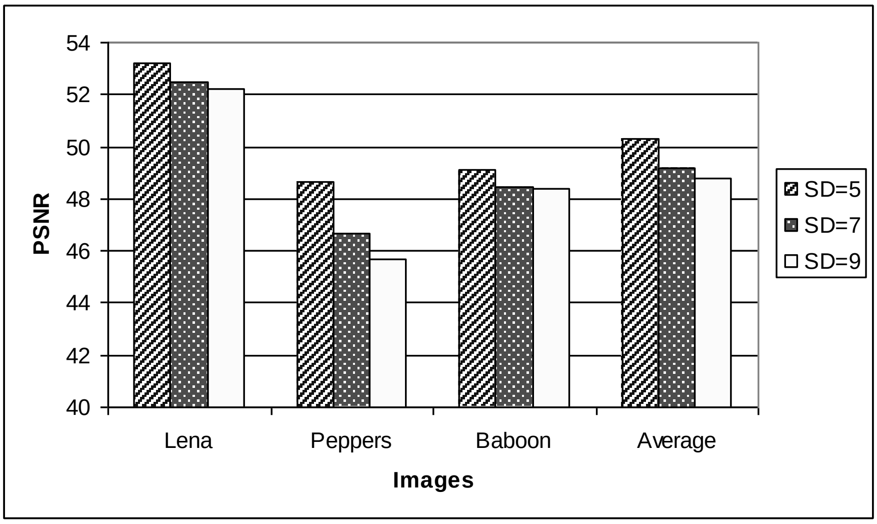


Figure 92. PSNR by varying SD with fixed CS=3 and T=300

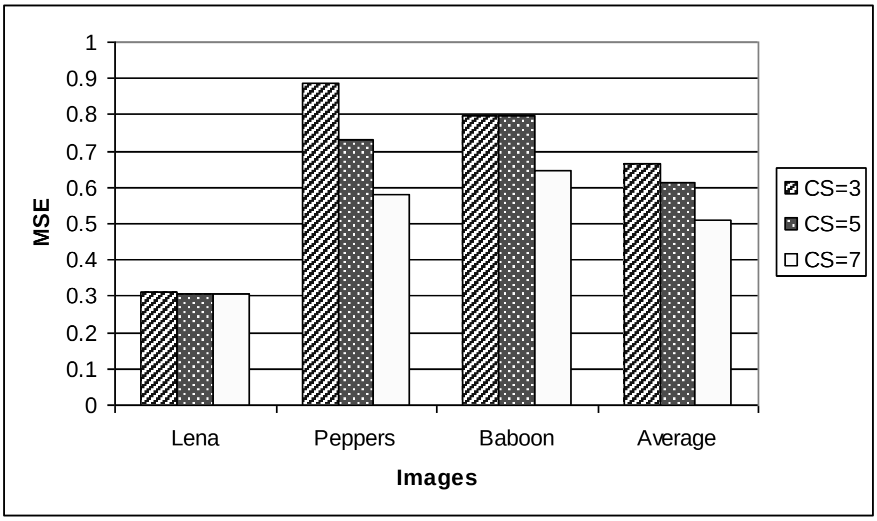


Figure 93. MSE by varying CS with fixed SD=5 and T=300

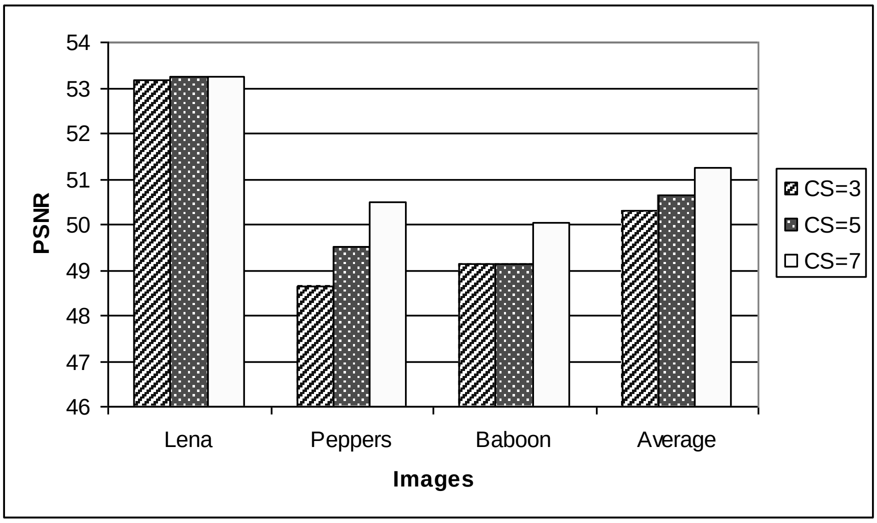


Figure 94. PSNR by varying CS with fixed SD=5 and T=300

The next varied parameter is the similarity threshold T. The MSE and PSNR measurements presented in Figures 95 and 96, respectively, were obtained by fixing the SD on 5 and the CS on 7. We can observe that the initial threshold value (T=300) was the optimal. Higher values of T are increasing the MSE and are decreasing the PSNR. In comparison with the value obtained in [76] and [77] for the same parameter T in image denoising, the optimal value 300 obtained here is reasonable, taking into account that the context is quite large. We have tried also lower values for T, but with poor inpainting results, since the lower similarity threshold implies the necessity of finding contexts with low SAD values, which is often impossible.

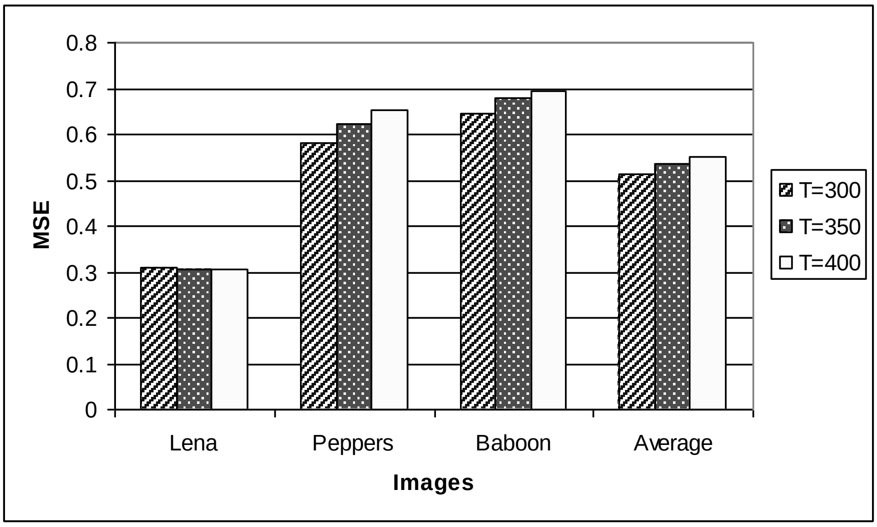


Figure 95. MSE by varying T with fixed SD=5 and CS=7

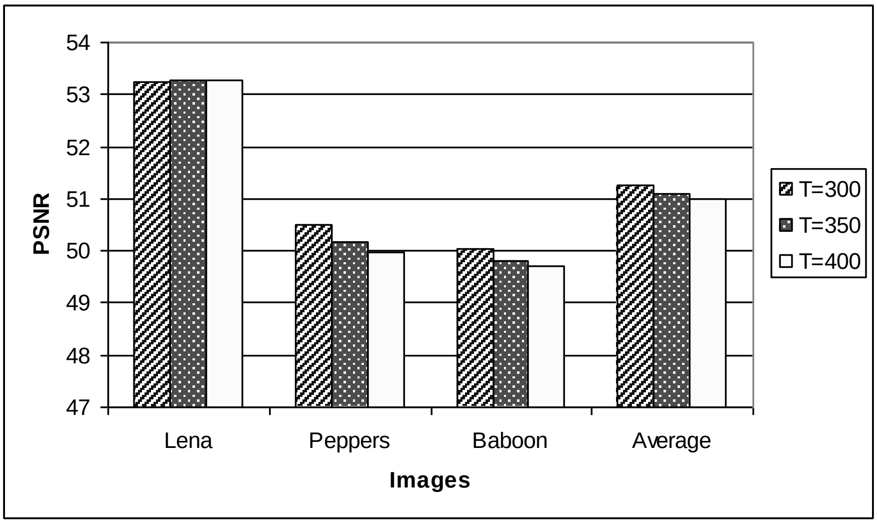


Figure 96. PSNR by varying T with fixed SD=5 and CS=7

Figure 97 presents the Lena image after the Markovian inpainting of the artificial defect presented in Figure 90. One can observe that the artificial defect was very well replaced.

Next, we will use this optimal configuration of our inpainting algorithm for comparisons with other existing techniques, in terms of MSE and PSNR. Figures 98 to 103 are presenting comparatively our Markov inpainting method (SD=5, CS=7, T=300) and the techniques developed by Bertalmio et al. [9], Oliveira et al. [159], Hadhoud et al. [101], Efros et al. [44] and Criminisi et al. [33], described in Section 2. As the comparative evaluations show, our proposed Markov inpainting method clearly outperforms all the other techniques on the Lena test image on all mask sizes (see Figures 98 and 99). On the Peppers test image, it is better than all the other techniques on high mask size and provides similar results on low and medium mask sizes (see

Figures 100 and 101). On the Baboon image, it is better than the methods developed by Oliveira et al., Hadhoud et al. and Criminisi et al. on high mask size and is similar with the method of Hadhoud et al. on low and medium mask sizes (see Figures 102 and 103).

Figure 97. The Lena image with artificial defect of 849 pixels after Markovian inpainting

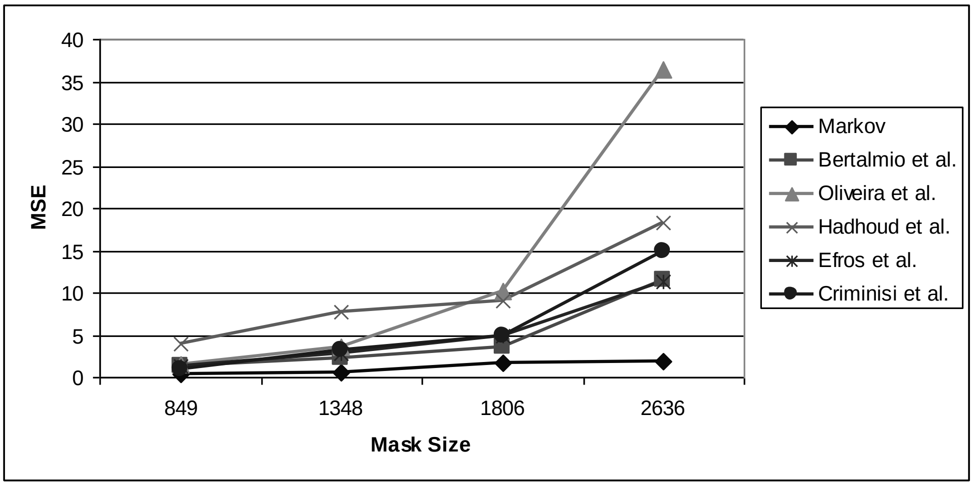


Figure 98. Comparing the MSE of our optimal Markov inpainting method with other existing techniques on the Lena image

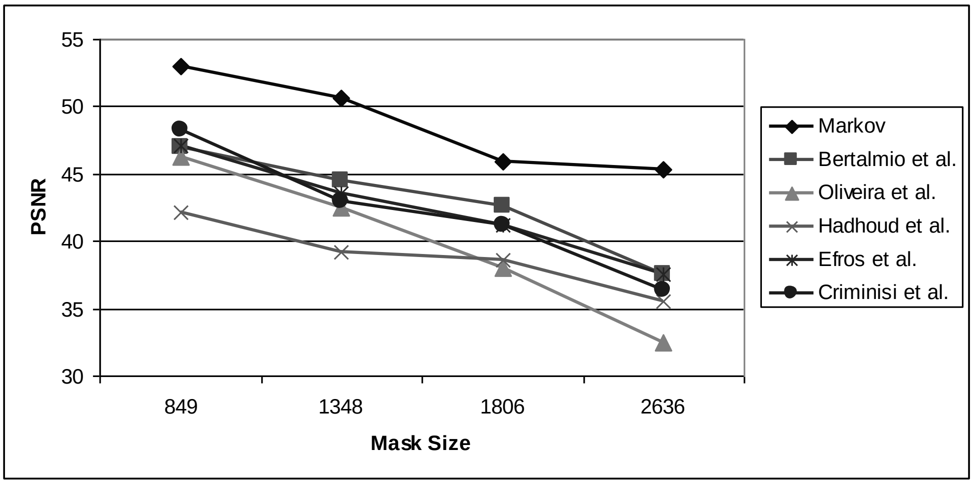


Figure 99. Comparing the PSNR of our optimal Markov inpainting method with other existing techniques on the Lena image

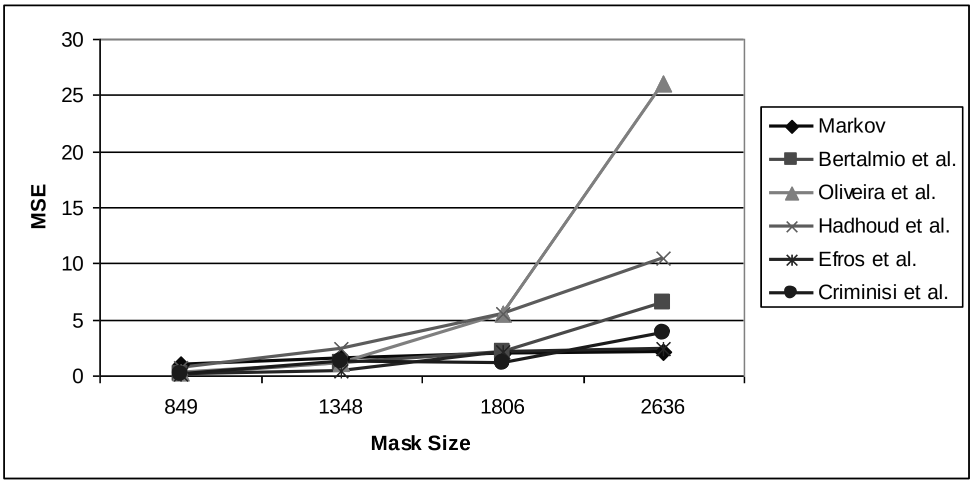


Figure 100. Comparing the MSE of our optimal Markov inpainting method with other techniques on the Peppers image

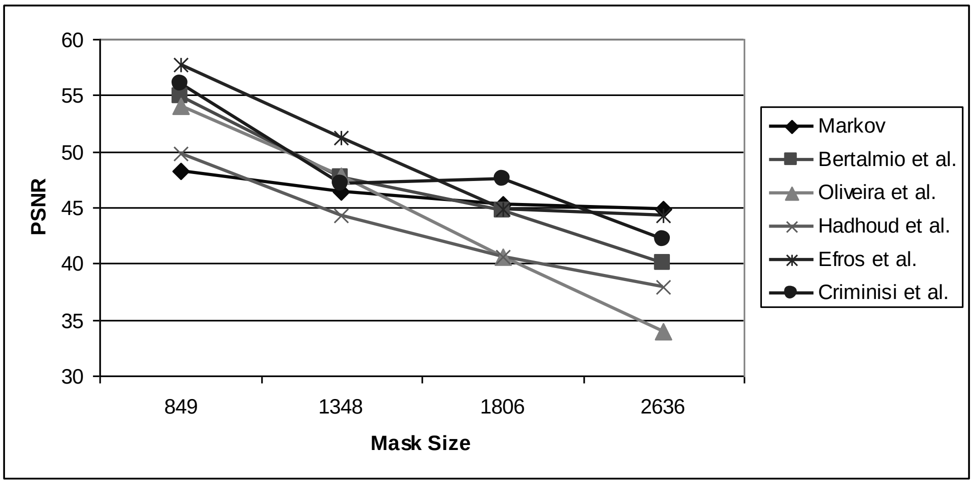


Figure 101. Comparing the PSNR of our optimal Markov inpainting method with other techniques on the Peppers image

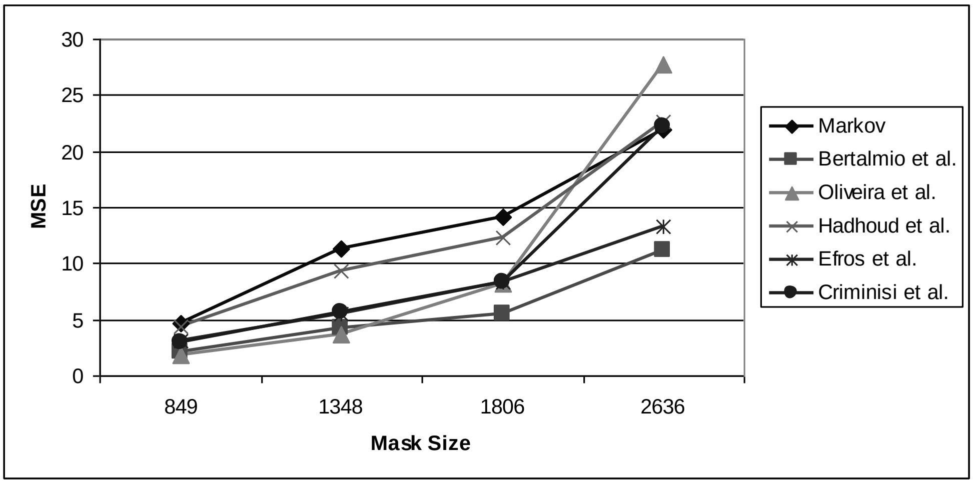


Figure 102. Comparing the MSE of our optimal Markov inpainting method with other techniques on the Baboon image

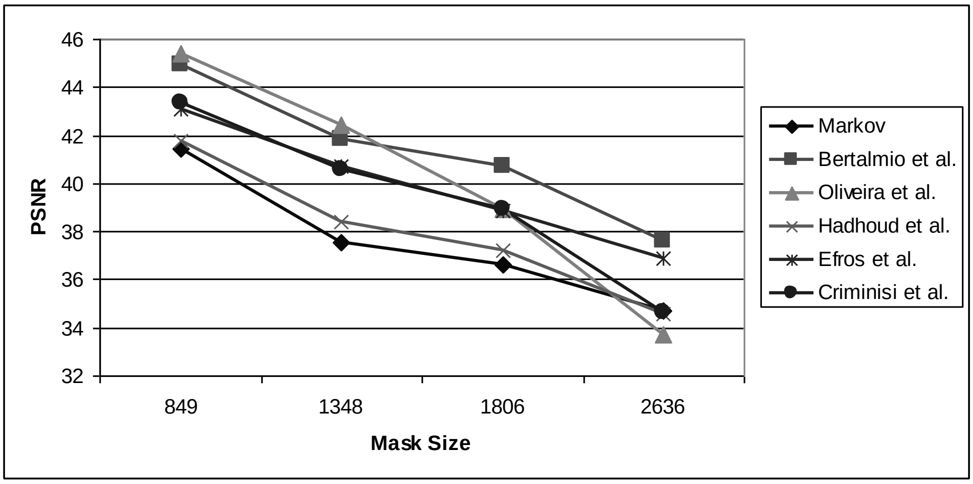


Figure 103. Comparing the PSNR of our optimal Markov inpainting method with other techniques on the Baboon image

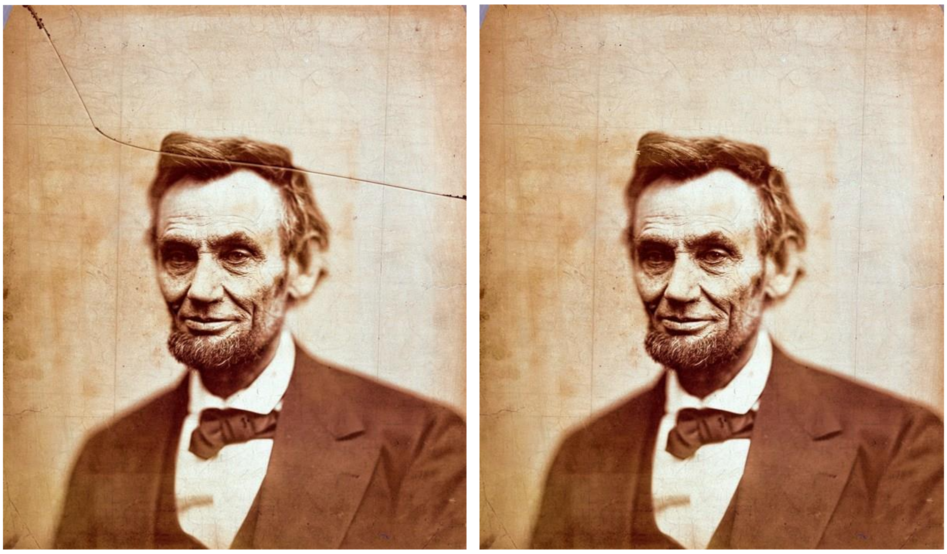


Figure 104. Markovian inpainting on the photo of Abraham Lincoln (the original on left, the repaired one on right)

Finally, we present the result of our inpainting method on the photograph of Abraham Lincoln. As Figure 104 shows, the crack could be successfully removed, being replaced with highly reliable colors. Some details in the background, but also in Lincoln's hair, were very well reconstructed due to the context information used by our method.

5.2.4. Summary

In this section, we introduced a new contextual inpainting method which is using Markov chains in order to repair pixel colors from images affected by external factors or to replace pixel colors belonging to image areas covered by objects or texts. The user must select the target (replaceable) area. Our inpainting algorithm is replacing each pixel from the target area based on the surrounding unaffected context information. Therefore, the restoration process is applied from the exterior to the interior within the selected target area. For the replacement of a certain pixel, we explored a limited surrounding image area to identify the color occurring with the highest probability in similar contexts. Since we use context information, the proposed inpainting technique can very well rebuild the image details. We have determined in our experiments the best parametrical configuration of the proposed Markov inpainting technique consisting in a context size of 7, a search distance of 5 and a similarity threshold of 300. We have compared our optimally configured method with other existing inpainting techniques and the results were better on some test images and comparable on others. The results obtained on the famous “cracked-plate” portrait of Abraham Lincoln, were remarkable.

In our opinion, this new inpainting method still has a good future development potential. As a further work direction, we intend to integrate our context-based technique, together with a structural one, into a hybrid inpainting method.

6. Predictive Web Prefetching

This chapter will present the prediction-based models proposed by us in [80] and [82] to prefetch web objects based on the users' web browsing history with the goal of decreasing the access latencies. We analyzed the PPM algorithm (a hybrid model with static prioritization), as well as Dynamic Decision Trees (hybrids with dynamic prioritization of their components).

6.1. Web Usage Mining through Prediction by Partial Matching

Nowadays the access to the Internet becomes more prevalent worldwide and often there are requests for higher bandwidths. Clients are frequently confronted with delays in accessing web pages, especially those ones that are limited by low-bandwidth modems or wireless routers. Many latency-tolerant techniques have been developed during the last years, the most important being caching and prefetching. Caching exploits the temporal locality principle by keeping the accessed pages and files in a cache structure, whereas prefetching anticipates the next accesses and loads the corresponding web pages or files into the cache. If the user accesses a web page or a file which is available in the cache, the browser can load it without any delays. Thus, when the users have long browsing sessions, prediction-based prefetching can be very effective by minimizing the access latencies.

In this section we analyze web page prefetching through prediction by partial matching (PPM) enhanced with a dynamic confidence mechanism [82]. The number of states used in such models tends to rise exponentially as the order of the model increases [36]. Therefore, we implemented the PPM algorithm as simple searches in the observation sequence instead of using high complexity tree-, graph- or table-based modelling. Our simple representation allows superior order Markov chains with high number of states and long histories, at low complexity. Thus, the proposed PPM implementation is significantly more efficient than previous implementations. We have also enhanced the predictor with a confidence mechanism, which classifies dynamically web pages as predictable or unpredictable. Predictions are generated selectively only from web pages classified as predictable, improving thus the accuracy. The confidence mechanism consists in dynamically adapted saturating counters attached to each web page. We evaluate the proposed predictor in terms of prediction accuracy on the BU benchmark set, generated by the "Ocean Group Research" from Boston University [34]. The goal is to find the most appropriate prediction technique for anticipating and prefetching web pages and to integrate it as an extension into browsers.

6.1.1. Related Work

In our previous work [83], we compared Markov chains, Hidden Markov Models and graph algorithms as web page prediction methods. We applied multi-page prediction by prefetching all the pages that appeared in the history after the considered context. The best prediction accuracy has been obtained with a hybrid predictor consisting in a Hidden Markov Model (HMM) and a graph-based predictor. The proposed hybrid predictor prefetched the web pages anticipated by both component predictors. In contrast, in this section we use the prediction by partial matching algorithm which combines different order Markov chains exploiting thus the advantage of each one. In order to reduce the additional network traffic, in this work we prefetch only one predicted web page from each confident state.

Link prediction based on Markov chains was presented in [188]. In [122] the author applied the Markov model together with the k-nearest neighbor classifier algorithm to enhance the performance of traditional Markov chains in predicting web pages. He obtained lower consumed memory, quite similar build time and evaluation time and higher accuracy. In [123] and [124] clustering is used to group homogeneous user sessions. Low order Markov chains are built on these clustered sessions and when Markov models cannot make clear predictions the association rules are used. In [125], the authors presented a survey of web page ranking for web personalization. They concluded that low order Markov models have higher accuracy and lower coverage, whereas the higher order models have a number of limitations associated with higher state complexity, reduced coverage and sometimes even worse prediction accuracy.

The PPM was first introduced by Cleary and Witten in [32] for data compression and represents an important context-based prediction method. It contains a set of simple Markov predictors. It uses the highest order Markov predictor which can provide prediction. The predicted value is the value that followed the context with the highest frequency. In [36], the authors used different PPM techniques for web page prefetching. They observed that as the order of the model increases, the number of states used for the model also increases dramatically. They reduced the complexity of the model by eliminating many of its states that are expected to have low prediction accuracy. But the behavior of a certain user can change in time and a state with low prediction accuracy can become a state with high accuracy and vice versa. In contrast, we select dynamically the confident states through the saturating counters attached to each web page. The great advantage of the saturating counters is that they can adapt fast to any changes in the user’s behavior. We reduced the complexity of PPM modelling by keeping all the states.

In [31], the authors proposed a page rank algorithm to predict the next page that will be visited by a web surfer. For the first set of sessions, they applied the page rank algorithm which provides the ranks for web pages. For each web page their method determines to which pages the user can navigate and, using the page ranks, it computes the probability of visiting them by dividing each rank to the sum of ranks, and the number of links to the total number of links, respectively.

In [21], Canali et al. proposed adaptive algorithms that combine predictive and social information and dynamically adjust their parameters according to the continuously changing workload characteristics. Their experimental results showed that such adaptive algorithms can achieve performance close to theoretical ideal algorithms. In [212], Wan et al. proposed an approach based on a vector space model, called random indexing, for the discovery of the intrinsic characteristics of web user activities. The underlying factors are then used for clustering individual user navigational patterns. The clustering results are used to predict and prefetch web requests for grouped users.

In [197], Temgire et al. presented a review on web prefetching techniques. User behavior prediction has been applied also in online advertisement industry [131] and financial service industry [232].

6.1.2. Web Page Prediction

Our goal is to integrate the most efficient PPM configuration as an extension into browsers, as in [83]. The browser extension, presented in Figure 105, collects and pre-processes the links accessed by the user: each link is codified with a unique number, ports and relative parts are eliminated from links, if there are two consecutive accesses of the same link only one is considered, links having the extension .gif and .xbm, which are usually small images, are also eliminated. The browser extension keeps a certain history of the accessed links. When the current link is accessed, the next link is anticipated using the PPM algorithm based on the history of the previously visited links. The predicted web page or file is prefetched in the cache in order to be available if the user accesses it.

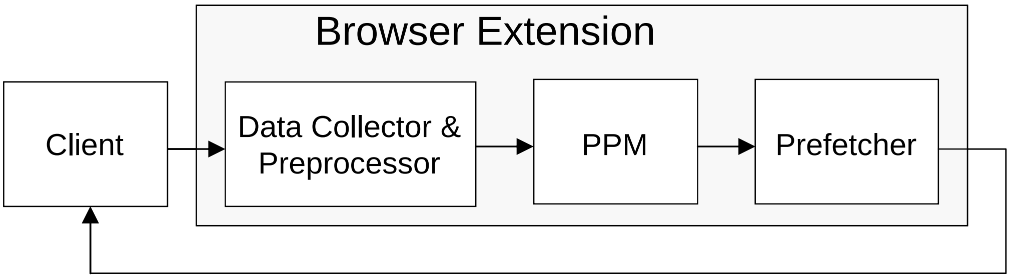


Figure 105. The structure of the application

In order to improve the prediction accuracy, the predictor is enhanced with a confidence mechanism as in [83], which consists in saturating counters, attached to all the links kept in the history sequence. The associated saturating counters are incremented on correct prediction and decremented on misprediction. A prediction is generated only if the confidence counter of the current link is in a predictable state. By using such a confidence mechanism, the number of predictions is lower, but the prediction accuracy is significantly higher. For a low traffic level, the proposed predictors are providing only one predicted web page, instead of a multi-page prediction presented in [83].

6.1.2.1 Markov Predictors

In a first order Markov chain with N states, the current state depends only on the previous state:

$$P[q_t \mid q_{t-1},\ ...,\ q_1] = P[q_t \mid q_{t-1}] \qquad (40)$$

where q_t is the state at time t. In a Markov chain of order R, the current state depends on R previous states [83]:

$$P[q_t \mid q_{t-1},\ ...,\ q_1] = P[q_t \mid q_{t-1},\ ...,\ q_{t-R}] \qquad (41)$$

In our application the states are represented by the web pages. The Markov chains can be used to predict the next web page by searching for the current context within the history of visited web pages. The web page that followed the context with the highest frequency is the predicted one. Figure 106 presents an example of prediction with a first order Markov chain: the context is the current link 1 and the prediction is 5 since it occurred the most frequently after 1.

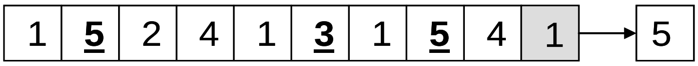


Figure 106. Prediction with a first order Markov chain

In a Markov chain of order R, the context consists in the last R web pages. The Markov predictor used for web page prefetching is presented in the following pseudocode:

```
MARKOV (SEQ, R)
    for k := 0 to R-1 do
      C[k] := SEQ[H-R+k]
    for i := R to H-1 do
      IS_CONTEXT := TRUE
      for k := 0 to R-1 do
        if SEQ[i-R+k] != C[k] then
          IS_CONTEXT := false
          break
```

```
  if(IS_CONTEXT)
    P[SEQ[i]] := P[SEQ[i]] + 1
PRED := 0
MAX := P[0]
for k := 1 to N-1 do
  if P[k] > MAX then
    MAX := P[k]
    PRED := k
if MAX > 0 then return PRED
return -1
```

where R is the order of the Markov chain, SEQ is the observation sequence, C is the context (the last R web pages from SEQ), H is the length of SEQ, P keeps the probability distribution for N distinct web pages, PRED is the predicted web page, and MAX is the number of occurrences of the predicted web page after the context C. If the context C is not found in SEQ, expressed by returning -1, we do not predict any web page. We have also presented the Java implementation of such a Markov predictor in [92].

Tree-, graph-, or transition table-based implementations are also possible, but such methods are inefficient for a high number of web pages and superior order Markov models. The temporal complexity of our Markov model implementation is Θ(H×R). The memory request is even lower: it needs to keep the web page sequence of size T in SEQ, the context of size R in C, the probability distribution of size N in P and other few variables (IS_CONTEXT, PRED, MAX), which is remarkable especially for systems with memory constraints.

6.1.2.2. Prediction by Partial Matching

The PPM of order R first tries to predict with the Markov model of order R based on the web page sequence SEQ. If the Markov model of order R cannot predict, then the Markov model of order R-1 is used, and so on, the last trial being the Markov model of order 1. In other words, if the Markov model of a certain order cannot provide prediction, it triggers the next lower order Markov chain. Figure 107 presents an example of prediction with a PPM of order 3: the Markov chain of order 3 cannot predict because it could not find the context of 3 links, thus the prediction 3 is generated by the Markov chain of order 2 because it followed once the context within the link sequence.

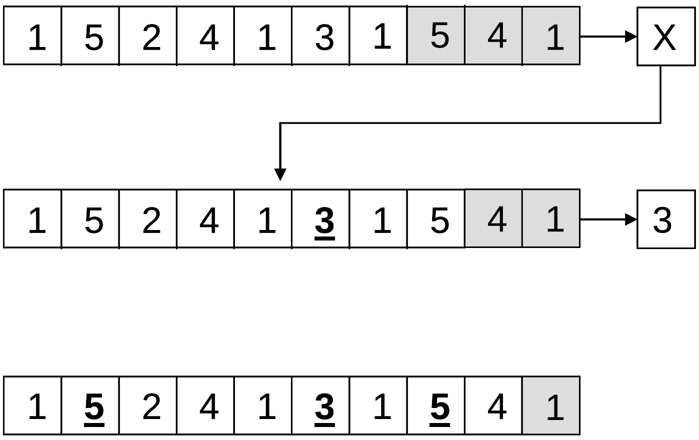


Figure 107. Prediction with PPM of order 3

We have not included the Markov model of order 0 because it is not using any contextual information. The PPM algorithm is given in the following pseudocode:

```
PPM (SEQ, R)
    for r := R downto 1 do
         PRED := MARKOV(SEQ, r)
       if PRED != -1 then return PRED
    return -1
```

The prediction mechanism is presented in Figure 108.

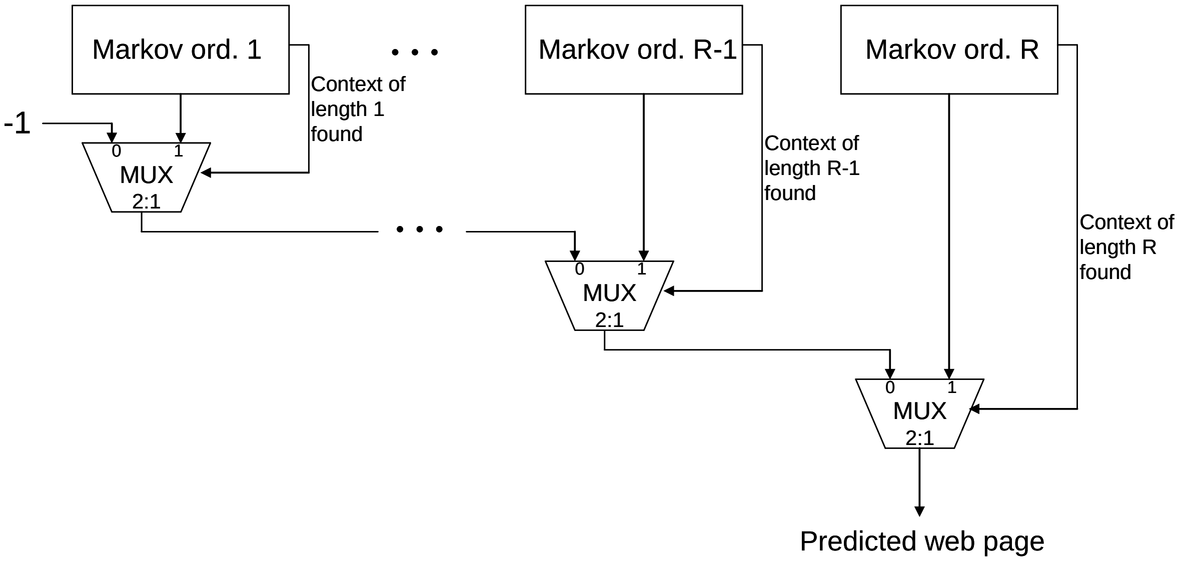


Figure 108. Prediction by partial matching

Since, in the most unfavourable case, the MARKOV function is called R times, the complexity of the PPM algorithm is $\Theta(H\times R^2)$. The memory request of the proposed PPM is similar with that of the above presented Markov model, which is remarkable, and significantly better than that of the other existing PPM implementations.

6.1.2.3. Graph-Based Prediction

The graphs are data structures used in many types of applications to model different processes. A graph consists in vertices which can be connected by edges. Figure 109 depicts a directed weighted graph whose vertices represent the accessed links. An edge between two vertices means at least one transition from a link to another in the direction shown by the arrow, whereas the weights are keeping the transition numbers.

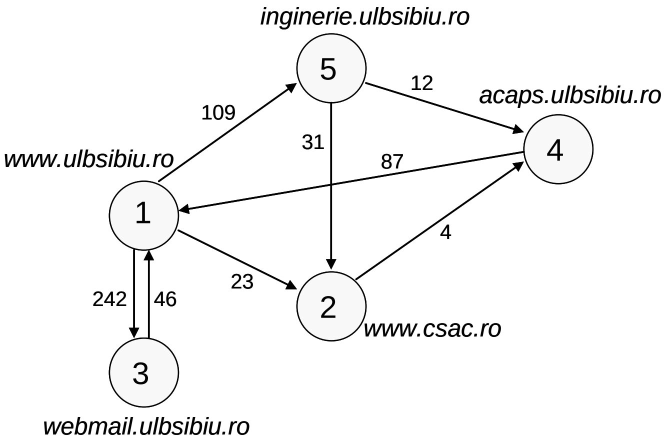


Figure 109. Modeling web page accesses using graphs

The graph-based predictor anticipates the next link as being the vertex whose edge with the current vertex has the highest weight. In fact, it is a first order Markov predictor implemented using a directed graph. The algorithm is presented in the following pseudocode:

```
DGRAPH()
    A := C.GET_PAIRS()
    MAX := A[0]
    foreach Edge E from A
        if E > MAX then MAX := E
    if MAX > 0 then return E.GET_PAIR(C)
    return -1
```

where C is the vertex corresponding to the current link, GET_PAIRS returns a list with all the adjacent vertices and GET_PAIR gives the pair vertex of a certain vertex from a given edge E. We assume that when the function is called, the directed graph is already constructed based on the sequence of visited web pages.

6.1.3. Experimental Results

The first step of our research is to analyze the proposed algorithms from the prediction accuracy point of view, by varying their input parameters. Such a study can be better highlighted on a set of log files. Therefore, the above presented algorithms have been implemented in Java and evaluated on the BU benchmarks.

The BU dataset was generated by the "Ocean Group Research" from Boston University [34] and consists in log files collected during 7 months on 37 workstations, spanning the timeframe from 21 November 1994 to 8 May 1995. Each log file name contains a user ID number, the machine on which the session took place and the Unix timestamp when the session started. Each line in a log corresponds to a single URL requested by the user; it contains the machine name, the timestamp when the request was made, the URL, the size of the document (including the overhead of the protocol) and the object retrieval time in seconds. The average number of links in the BU benchmarks is 1905. After we have pre-processed the original log files, as we described in Section 3, we named them conX where X is the user ID.

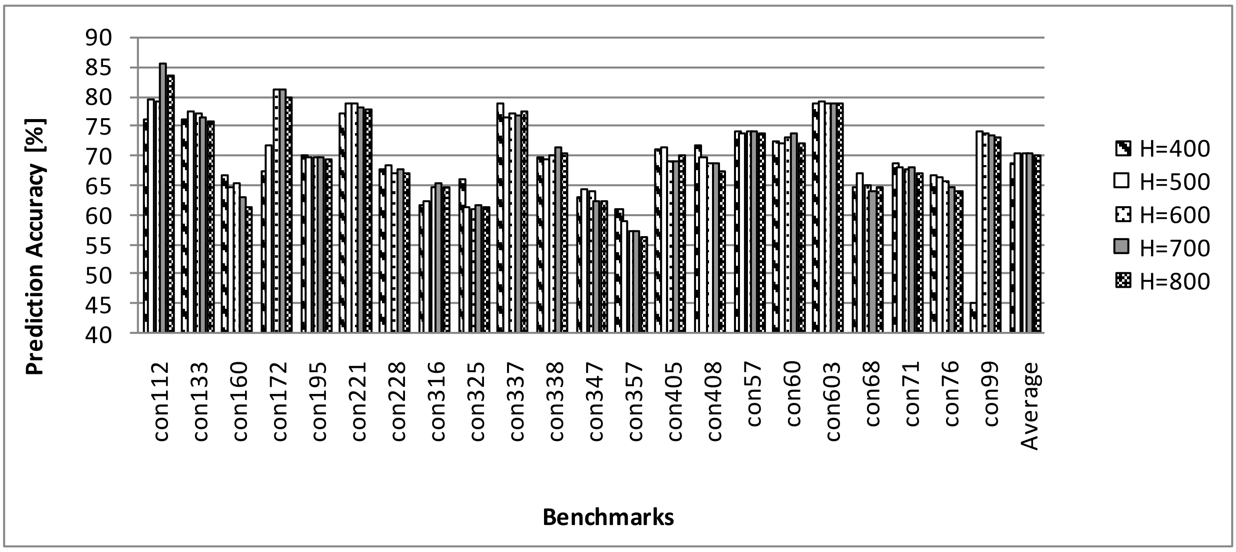


Figure 110. Prediction accuracy of 3^{rd} order PPM (R=3) for different histories

First, we have evaluated a 3rd order PPM by varying the size of the web page sequence, also called history (H). We used 4-state confidence counters, identified in [83] as being optimal for web page prediction. The results are presented in Figure 110. As Figure 110 presents, the highest average prediction accuracy has been obtained with a history length of 600. We can observe an increase in accuracy until H=600 and a fall after that, meaning that a rich history can lead to higher accuracy, but starting with a certain length it can behave as noise and the accuracy decreases. In fact, the accuracies obtained with history lengths of 500, 600 and 700 are very close – 70.14%, 70.42% and 70.40%, respectively –, therefore, we consider that the optimal history length is 500.

We have continued our study by varying the order of the confidence-based PPM, considering the optimal history length of 500 web pages. The results are presented in Figure 111:

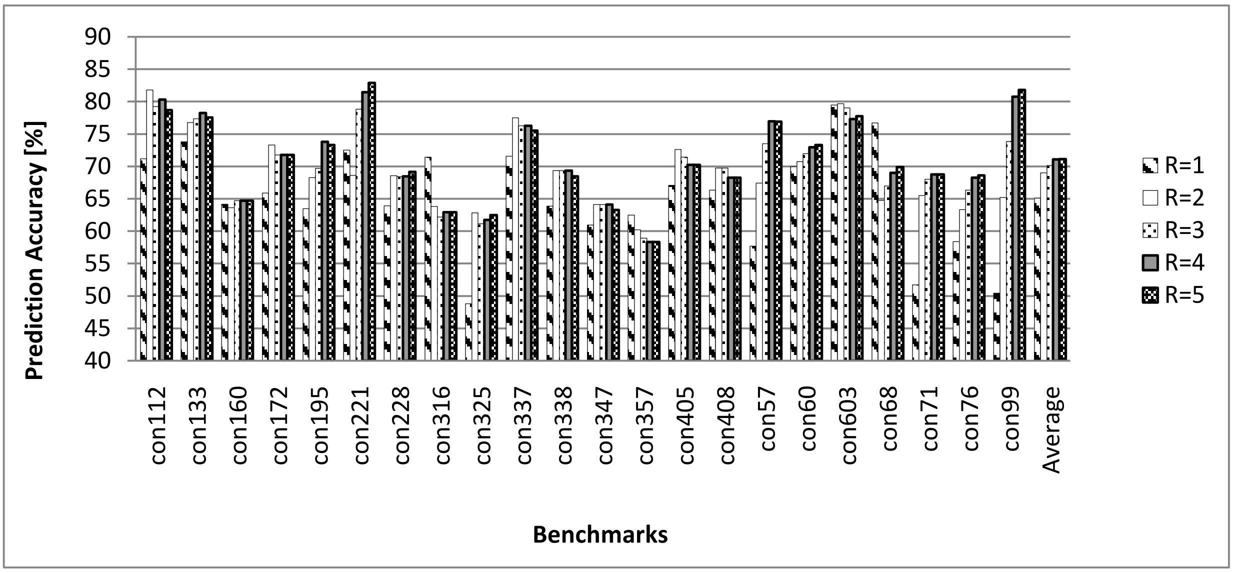


Figure 111. Prediction accuracy obtained with PPM of different orders and history of 500 (H=500)

Figure 111 shows that the prediction accuracy increases with the order of the PPM, but there is marginal benefit of increasing the order beyond 4. On some benchmarks we can observe inflexion points followed by accuracy decrease. For those benchmarks, the Markov chains of higher orders than the inflexion point have more mispredictions at higher complexity. The average prediction accuracies show that the PPM of order 4 is the optimal.

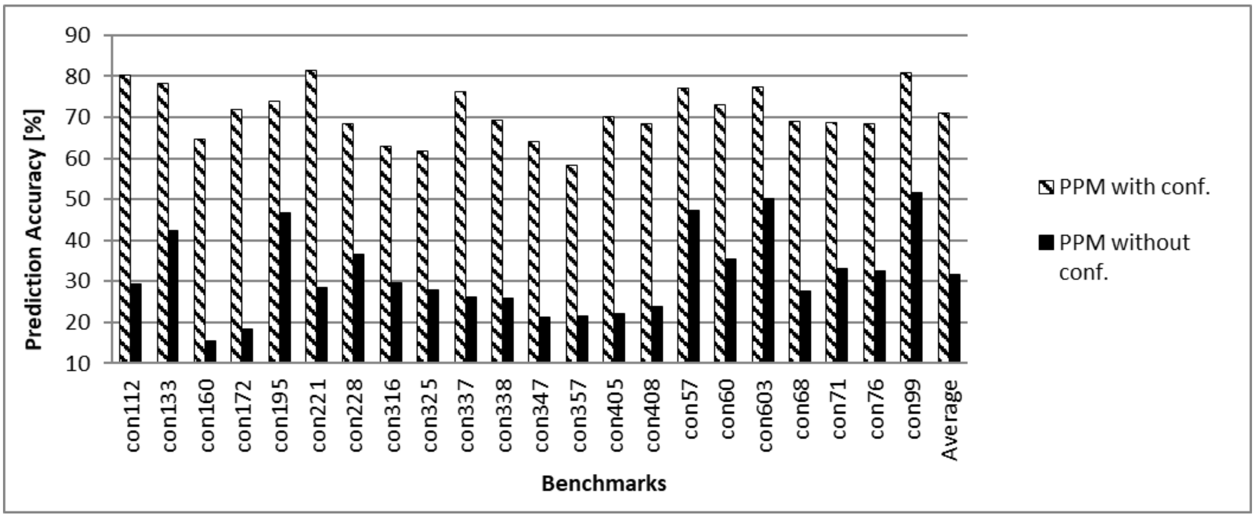


Figure 112. Comparing the optimal PPM (H=500, R=4) with and without confidence

We have compared the optimal PPM (H=500, R=4) with a PPM having the same configuration but without the 4-state confidence mechanism. As Figure 112 depicts, the 4-state confidence counters have a very high benefit. With this confidence mechanism we are able to selectively predict only from confident web pages, increasing thus the average prediction accuracy from 31.54% to 71.11%. Figure 113 shows how the prediction rate (the number of predicted web pages divided to the total number of web pages) is influenced by the selectivity of the 4-state confidence mechanisms in the case of the optimal PPM (H=500, R=4).

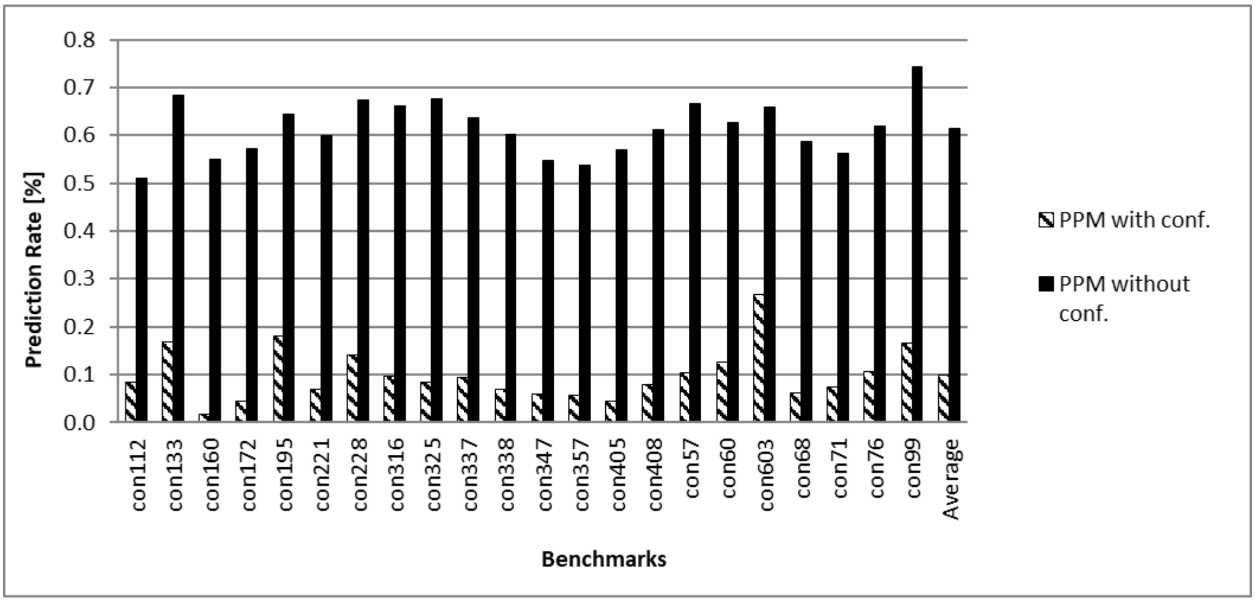


Figure 113. Comparing the prediction rates of the optimal PPM (H=500, R=4) with and without confidence

It can be observed that the attached 4-state confidence counters reduce the number of predictions from 62% to 10% but improve the accuracy from 31.54% to 71.11%. To not increase too much the network traffic, we prefer to predict fewer times but accurately.

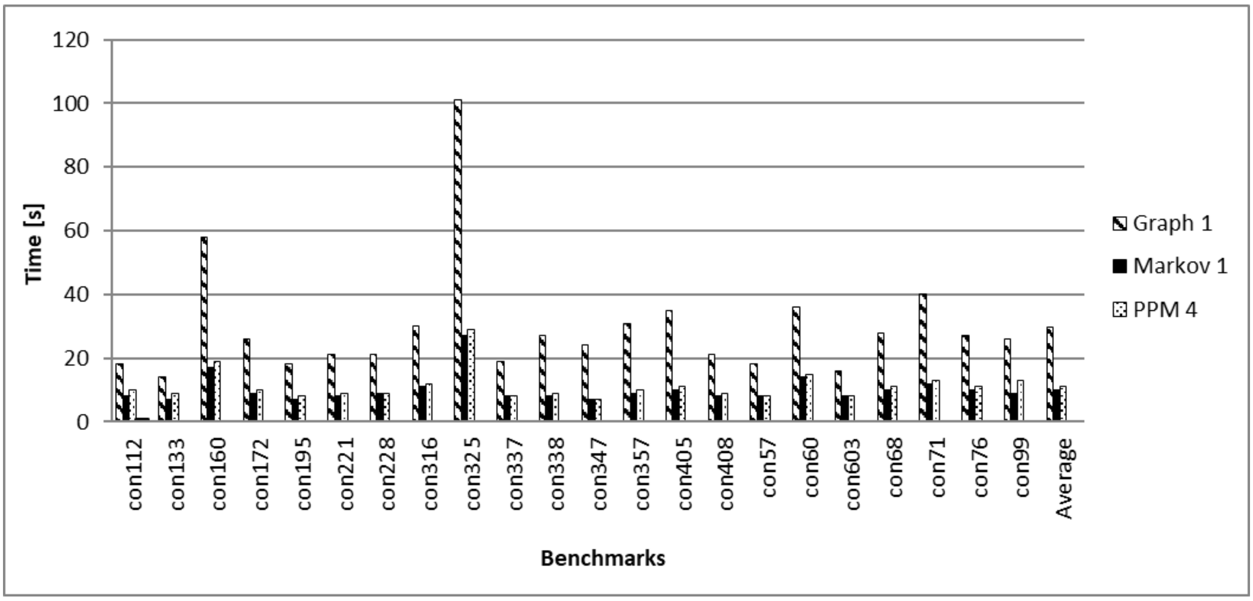


Figure 114. Simulation time

Finally, Figure 114 depicts the efficiency of our PPM implementation opposite to a graph-based implementation, as simulation time. The implementations which are implying transition tables, trees or graphs, as in [36] and [188], become very inefficient or even intractable for high number of distinct web pages and superior order models, because of the high state complexity. In Figure 114, Graph 1 is a first order Markov predictor implemented using a directed weighted graph, whereas Markov 1 and PPM 4 are our first order Markov and 4th order PPM implementations. We reported the time necessary to process and eventually predict all the links from each benchmark. As the evaluations show, our proposed Markov model implementation is more efficient than the graph-based one, and more important, our proposed PPM implementation has almost the same time-efficiency even for higher orders. The average execution time with the first order graph-based Markov predictor is 29.77 s, while with our first order Markov predictor and our 4th order PPM it was only 10.18 s and 11.27 s, respectively.

6.1.4. Summary

In this study we proposed prediction-based web prefetching methods in order to reduce the delays necessary to load the web pages and files visited by the users. We have presented an efficient way to implement the prediction by partial matching as simple searches in the observation sequence. By avoiding tree-, table- and graph-based implementations, the memory necessity of the proposed PPM is similar with that of the Markov model. This low complexity is remarkable, especially for the superior order models with high number of states, and significantly better than in the other existing PPM implementations. Our time-evaluations show that the prediction latency of the proposed model is just slightly affected by the order of the model, opposite to the implementations using the above-mentioned data structures where the order of the model affects exponentially the complexity.

The confidence mechanism has a high benefit because it allows to predict selectively from high-confidence contexts, decreasing thus the prediction rate from 62% to 10%, but increasing in the same time the accuracy from 31.54% to 71.11%. The optimal prediction accuracy of 71.11% was obtained with the PPM of order 4 using a history of 500 web pages and 4-state confidence counters.

A further work direction is the analysis of using neural networks for web page prediction as a first prediction level in a two-stage predictor. Another further work direction consists in developing and evaluating different hybrid predictors. The PPM is a hybrid predictor with prioritization-based static component selection and we intend to evaluate also dynamic selection in order to use the best predictor at a certain moment, for possible prediction accuracy improvements. Finding and exploiting similarity among users is another research challenge.

6.2. Hybrid Web Access Prediction through Dynamic Decision Trees

Several low-latency web access techniques have been proposed during the last years. Caching exploits the temporal locality principle by keeping the accessed pages and files in a cache structure. Prefetching anticipates the next accesses and loads the corresponding web pages or files into the cache. If the user accesses a web page or a file which is already in the cache, the browser can load it without any delays. Thus, when the users have long browsing sessions, prediction-based prefetching can be very effective by minimizing the access latencies. In this section, we investigate web prefetching through dynamic decision tree (DDT) which acts like a hybrid web predictor having different order Markov chains as components [80]. The idea of using such a hybrid predictor is based on our observation that in different log files, due to different user behaviors, Markov chains of different orders can predict better. Thus, we expect that a hybrid predictor, which can dynamically select the most appropriate component, could provide better prediction accuracy than any of the components considered alone. This dynamic component selection is performed, in our case, by the DDT. The novelty of the proposed method is that we use the predictions generated by the component Markov predictors as DDT features to anticipate the next accessed link.

The decision tree is a commonly used inductive inference machine learning method and due to its simplicity and accuracy it is best suited for our needs. Decision trees can classify items by keeping them sorted from the root to the leaf nodes [149]. Each node in the tree denotes a test of some feature and each descending branch corresponds to one possible value of that certain feature. An item is classified by testing the feature specified by the root and moving down on the branch corresponding to the value of that feature. The process is repeated in a recursive manner on the current subtree. The value of the leaf node situated at the end of the selected path represents the output of the decision tree.

The Markov chains are using as input the web access history of a certain size consisting in a codified web sequence. We consider as context the most recently accessed web files. Thus, a Markov chain of order R is using as context the R previous web files accessed by the user and predicts the web file that followed that context with the highest frequency as being the next access. For efficiency, the Markov chain is implemented as simple searches in the observation sequence, as in [82], instead of using high complexity tree-based modelling.

We use the decision tree to dynamically select the most predictive features from a considered feature set and we consider only the selected features to generate predictions. In our application the feature set includes the current link, the type of the current link (HTML, non-HTML) as well as the predictions of different order Markov chains. The first step of the algorithm is to choose the root of the tree from the existing attributes. That chosen attribute has the highest information gain on the entire set of examples and it does best classify this set. After that, for each possible value of the selected attribute a new attribute will be recursively chosen from the remaining attributes as the root of the subtree that will be created as a branch. After the tree is learned we will use it to predict and prefetch the next web page or file based on the current context. For a higher prediction accuracy, we have also attached a dynamic confidence mechanism to the predictor. We have evaluated the prediction accuracy of the proposed hybrid predictor on the Boston University benchmark set (BU), generated by the "Ocean Group Research" [34].

6.2.1. Related Work

In our previous work [83] we compared Markov chains, Hidden Markov Models and graph algorithms as web page prediction methods from their accuracy point of view. We applied multi-page prediction by prefetching all the web files occured in the user access history after the considered context. The best prediction accuracy has been obtained with a hybrid predictor consisting in a HMM and a graph-based predictor. The proposed hybrid predictor prefetched the web pages anticipated by both component predictors. In contrast, in this section we use DDT-based prediction which combines different order Markov chains, exploiting thus the advantage of each one. To keep low network traffic, in this work we predict and eventually prefetch only one web file at a time.

In [82], we have analyzed the prediction by partial matching (PPM) algorithm for web prefetching. We have implemented it in an efficient way, allowing thus long web access histories and higher order component Markov chains at low complexity. The evaluations performed on the BU dataset have shown that the optimal PPM configuration was the 4th order one, with a prediction accuracy of 71.11%. The PPM is a hybrid predictor with prioritization-based static component selection. In contrast, for higher prediction accuracy, in this work we use a DDT-based dynamic selection in order to use the best component predictor at a certain moment.

In [161], the authors combined All-kth order Markov models with fuzzy Adaptive Resonance Theory for web page request prediction and obtained a prediction accuracy less than 40%. In contrast, we dynamically select the order of the Markov chain using a DDT and we expect a better adaptability to different users and also to changes in the behavior of the same user.

In [41] the authors proposed a hybrid web page prediction method which combines support vector machines (SVM), association rule mining and Markov chains in order to enhance the efficiency. Their experimental results showed that the proposed hybrid method provided a prediction accuracy of about 63%, outperforming the individual predictors. Their method needs a pretraining stage, while our method is based only on run-time training and we expect faster adaptation to user behavior changes. In [14], the authors proposed a three-level technique in which they combined Markov models, association rule mining and clustering. The prefetching and prediction is done by preprocessing the user accesses and integrating the three mentioned techniques. They used only a first order Markov model and, therefore, we expect higher prediction accuracy with our superior order models. In [154], the authors proposed another hybrid predictor which combines Markov model and HMM. In our opinion, the complexity of a HMM-based prediction algorithm increases dramatically in the case of higher order model and high number of different accessed links.

In [139], Lowd and Davis improve Markov network structure learning with decision trees. In their approach, a tree is learned to predict the value of each variable and then the trees are converted into sets of conjunctive features. All the learned features are merged into a single global model. The weights of these features are then learned globally by a Markov network. In contrast, in this work we include the predictions of different order Markov chains as features within a dynamically adapted decision tree which provides the final prediction. A temporal modelling of web user behavior has been proposed by Radinsky et al. in [177] for future behavior prediction. They presented several modelling techniques based on time-series to capture trends, periodicities and surprises in web usage behavior. A decision tree is learned during a training stage to choose the best performing temporal model. In contrast, in our work we use different order Markov predictors in a decision tree which dynamically learns which one is better in each context.

6.2.2. Web Prediction using Markovian DDT

The browser extension, presented in Figure 115, collects and preprocesses the links accessed by the user: each link is codified with a unique number, ports and relative parts are eliminated from the links, if there are two consecutive accesses of the same link only one is considered, and the links having .gif and .xbm extension, which are usually small images, are eliminated. The browser extension keeps a link history of size H. When the current link is accessed, it is introduced into the link history which can be truncated to the last H unique web accesses.

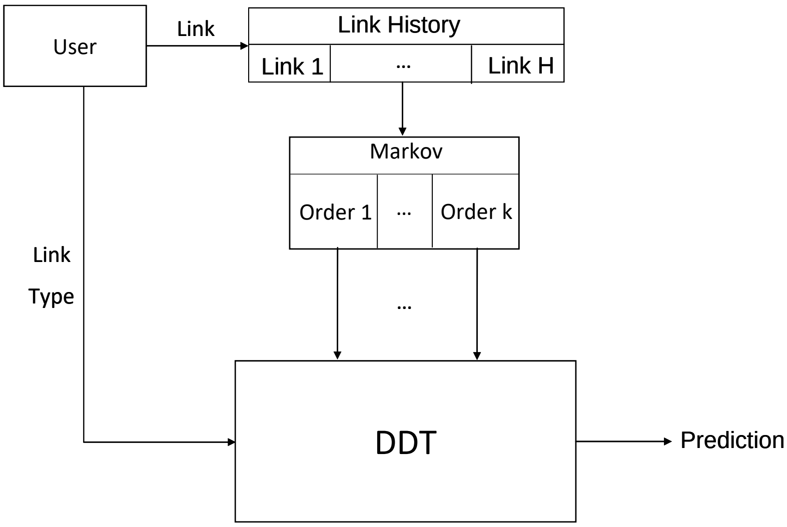


Figure 115. The structure of the application

The goal is to anticipate the next link using the DDT algorithm based on the history of the previously visited links as well as the type of the current link. For this, the proposed hybrid technique is composed of two prediction levels. The link history is translated into Markovian features consisting in the predictions using Markov chains of different orders. These Markovian features are used together with the link type feature (HTML and non-HTML) and the current link feature as input for the DDT algorithm to generate the final prediction. The predicted web page or file is prefetched into the cache in order to be available if the user will access it in the future.

For a higher prediction accuracy, the predictor is enhanced with a confidence mechanism as in [83], consisting in 4-state saturating counters attached to all the links kept in the history. The saturating counter associated to the current link is incremented on correct prediction and decremented on misprediction. A prediction is generated only if the confidence counter of the current link is in a predictable state. For a low traffic level, we predict and eventually prefetch only one web file at a time, instead of a multi-page prediction presented in [83]. In our application the states within the Markov chain are represented by the links (web pages and files). In a Markov chain of order R, the current link depends on R previous links [83]:

$$P_i[q_t = L_i \mid q_{t-1},\ ...,\ q_{t-H}] = P_i[q_t = L_i \mid q_{t-1},\ ...,\ q_{t-R}],\ i \in [1, N] \tag{42}$$

where q_t is the link at time t and P_i is the probability of the link (web page) L_i at time t, N is the number of distinct web pages and H is the size of the used web access history.

The Markov chain is implemented as in [82] as simple searches for the current context within the observation sequence, represented by the history of visited web pages. The web page that followed the context with the highest frequency is the predicted one. In a Markov chain of order R, the context consists in the last R web pages. The prediction of the Rth order Markov chain is computed as follows:

$$M_R = L_i,\ P_i = \max_{1 \le k \le N} P_k[q_t = L_k \mid q_{t-1},\ ...,\ q_{t-R}] \tag{43}$$

The Java implementation of this Markov predictor is given in [92]. Now we will describe the DDT method for predicting the next web page. We have used the ID3 learning algorithm [176] to create the decision tree from a training set of examples. Then we have used this tree to predict the next web page to be prefetched.

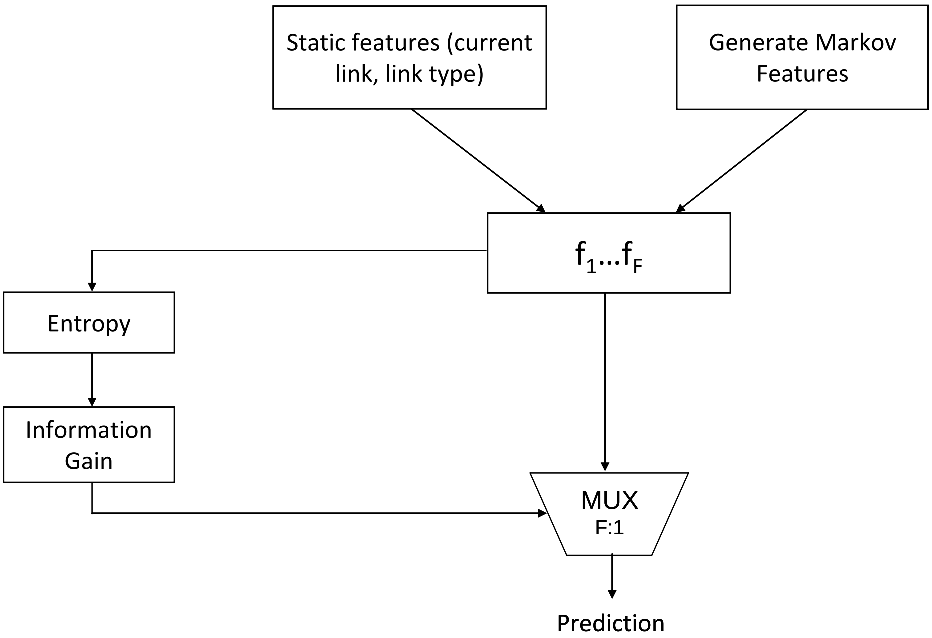


Figure 116. The structure of the DDT

Figure 116 presents the structure of our DDT. As it can be seen, we use two types of features, the static ones, which are the current link and the link type taken from the server logs, and the dynamic ones generated at runtime which are the predictions provided by the Markov models. We considered in our evaluations Markov chains with the order ranging between 0 and 7 and we observed that the best results were obtained with four Markov features: M_1, M_2, M_3 and M_4 (with M_R defined in equation (43)). As we detail below, the entropy and the information gain are used for feature selection.

The ID3 algorithm receives as parameters the set of examples S and the features F. It is a greedy algorithm that builds the decision tree T from top to down, by selecting at each node the feature that best classifies the training examples. The ID3 algorithm is given in the following pseudocode, where the union operation denotes node insertion:

```
ID3(S, F)
  T := Ø
  If F= Ø
    T := T ∪{the most common prediction in S}
  Endif
  Else
    Foreach $F_k$ ϵ F
      $G_{max}$ := 0
      $E(S) := \sum_{i=1}^{c}(-p_i \log_2 p_i)$
      $G(S,F_k) := E(S) - \sum_{v \in Values(F_k)} \frac{|S_v|}{|S|} E(S_v)$
      If $G(S, F_k) > G_{max}$
          $N := F_k$
          $G_{max}$ := G(S, $F_k$)
      Endif
    Endforeach
    T := T ∪{N}
    Foreach possible v of N
      If $S_v$ = Ø
        T := T ∪{the most common prediction in S}
      Endif
      Else
        T := T ∪ID3($S_v$, F-{N})
      Endelse
    Endforeach
  Endelse
  Return T
End
```

where c is the number of possible values of the target feature, N is the selected node, F denotes the features, S_v is the subset of the example set S for which feature F has the value v, E is the entropy and G is the information gain. The entropy represents the impurity of an example set. At the computation of the entropy, p_i is the proportion of S belonging to class i. The information gain is computed for each feature based on the entropy and it shows how well an attribute separates the training examples. The first feature that will be chosen does best split the set of examples S and will be the root of the tree. For each possible value of the selected feature a branch will be created whose descendent node will be recursively chosen from the remaining eligible features. The feature that has the highest information gain (meaning the lowest entropy) will be chosen as a node for the current branch. The tree is reconstructed after 100 links and the tree root can change, so it can provide good knowledge on the current situation. After this learning step, we use the DDT to predict the next link.

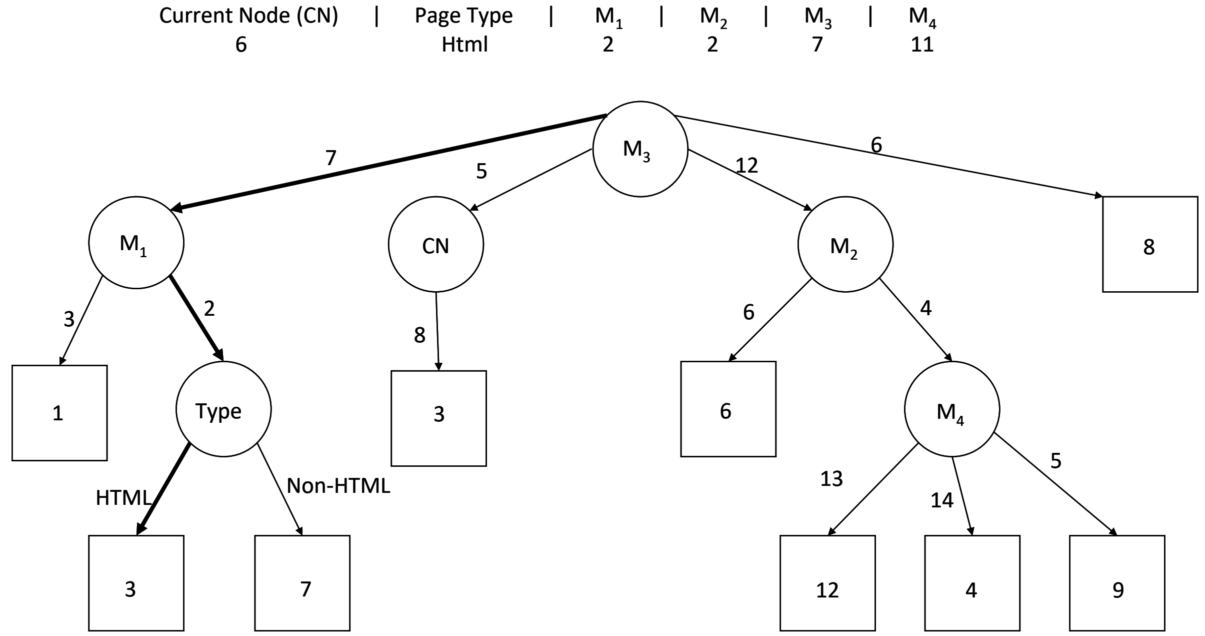


Figure 117. DDT Example

Figure 117 depicts a small example showing how a DDT-based prediction mechanism works by emphasizing the path from the root to the predicted leaf. The internal nodes (represented by circles) denote tests of the input features, the branches (represented by arrows) are the outcomes of the tests, whereas the leaf nodes (represented by squares) are the output classes. The feature with the highest information gain is M3 and, when the input data is available, that feature will be checked first choosing thus the branch that corresponds to that specific value. In this case, we have for the feature M_3 the value 7, which corresponds to the left sub-tree in which the root is M_1. The value for the M_1 feature in this example is 2 and we go through that branch reaching the Type node. The value of this feature is HTML, meaning that we go through the left branch and reach the leaf node 3 which will be the predicted value for this case. Basically, this step is a tree traversal having as input the static data provided by the link and the dynamic data provided by the Markov chains.

6.2.3. Experimental Results

We used the BU dataset to evaluate the proposed method which was implemented in Java. The BU benchmark set was generated by the “Ocean Group Research” from Boston University [34]. Each log file name contains a user ID number, the machine on which the session took place and the Unix timestamp when the session started. Each line in a log corresponds to a single URL requested by the user; it contains the machine name, the timestamp when the request was made, the URL, the size of the document in bytes (including the overhead of the protocol) and the object retrieval time in seconds.

We used as measures the prediction accuracy, the coverage and the prediction rate. The prediction accuracy is computed by dividing the number of correctly predicted links to the number of predicted links. The coverage is the division between the number of correctly predicted links and the total number of links. The prediction rate is the number of predicted links divided to the total number of links. The performance of prefetching techniques can be measured based on the hit rate and the additional bandwidth. In this work, the hit rate is expressed as prediction accuracy (PA). Since the bandwidth increase is directly proportional with the prediction rate (PR), we can approximate the relative prefetching benefit of a certain method 1 compared to another method 2, as being:

$$B = \frac{\frac{PA_1}{PR_1}}{\frac{PA_2}{PR_2}} \tag{44}$$

Obviously, there is a benefit if B is greater than 1. For all the evaluated methods we used a 4-state confidence mechanism as in [83]. First, we analyzed the proposed Markovian DDT algorithm from the prediction accuracy point of view, by varying its input parameters.

Figure 118 compares DDTs composed of different order Markov predictors using a history of 1000 links and 4-state confidence counters. DDT-M1 contains only a Markov predictor of order 1, DDT-M1-2 contains Markov predictors of orders 1 and 2, and so on. We denote DDT-Mx-y a decision tree using as features the predictions generated by Markov chains having the order between x and y. As Figure 118 shows, the inclusion of Markov chains having the order higher than 4 into the hybrid predictor is not improving the prediction accuracy, meaning that usually the web access patterns are not longer than 4. The DDT-M1-4 which uses the Markov predictors having the orders between 1 and 4 is the optimal since the inclusion of higher order Markov chains is not significantly improving the accuracy.

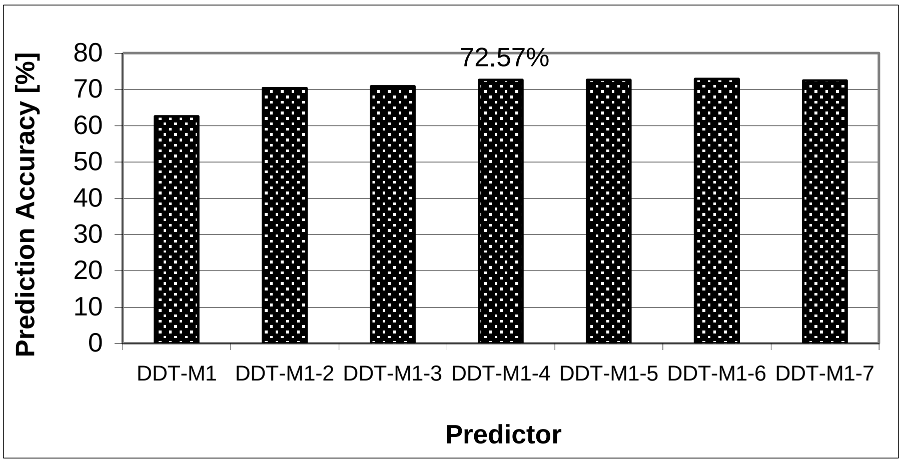


Figure 118. Average prediction accuracies obtained with different DDT predictors

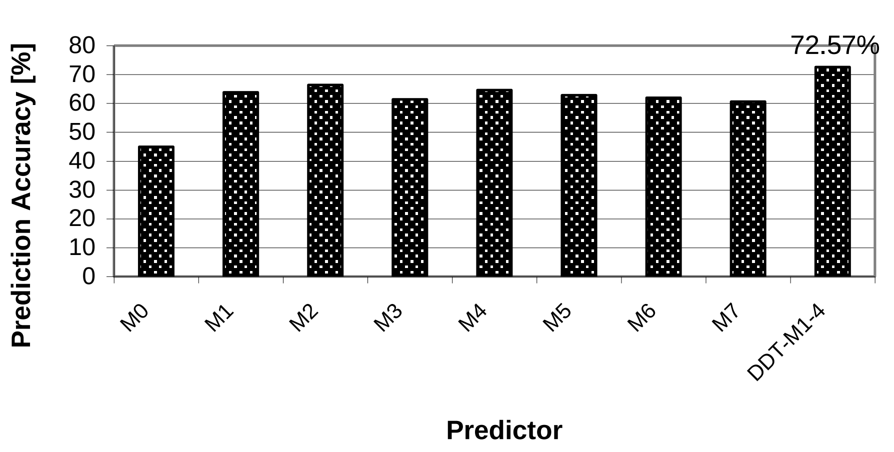


Figure 119. Average prediction accuracies obtained with Markov predictors of different orders and the optimal DDT-M1-4 predictor

Figure 119 compares the optimal DDT-M1-4 predictor with the Markov predictors having the orders from 0 to 7. All these compared predictors are using a history of 1000 links and 4-state confidence counters. As Figure 119 shows, the prediction accuracy is increasing together with the order of the Markov chain up to order 2 after which we can observe a decrease tendency, especially beyond order 4, which explains again the superiority in accuracy of DDT-M1-4 in Figure 118. In fact, we observed that starting with the Markov chain of order 6, the number of predictions was 0 on multiple benchmarks because the too rich contexts were not found within the web access history and/or the confidence counters were in unpredictable states.

The optimal DDT-M1-4 hybrid method outperforms all the evaluated Markov predictors, being thus an efficient two-level predictor, which provides better accuracy than all its components. The high average prediction accuracy of 72.57% is possible due to the 4-state confidence counters which dynamically classify the web pages as predictable or unpredictable and allow to predict and prefetch selectively only from high confidence contexts. Figure 120 shows comparatively the prediction rates and coverages of the same methods.

As Figure 120 depicts, due to the selective approach, the prediction rates are low. Only 6.7% of the web pages are predicted and, as the coverage shows, 5.18% of the total number of web accesses are correctly anticipated with the DDT-M1-4 predictor. By focusing on these 6.7% web page contexts, which were dynamically classified as predictable, the proposed hybrid technique can provide the prediction accuracy of 72.57%, outperforming all its components (see Figure 119). It is also outperforming the PPM algorithm presented in [82], whose average prediction accuracy is 71.11% (with a prediction rate of 10%) on the same dataset. By making fewer predictions but with good accuracy, we can avoid a high number of additional useless prefetches which helps to keep the traffic at reasonable level. The relative prefetching benefit of the optimal DDT-based method compared to the previous PPM-based technique, computed using equation (44), is 1.52.

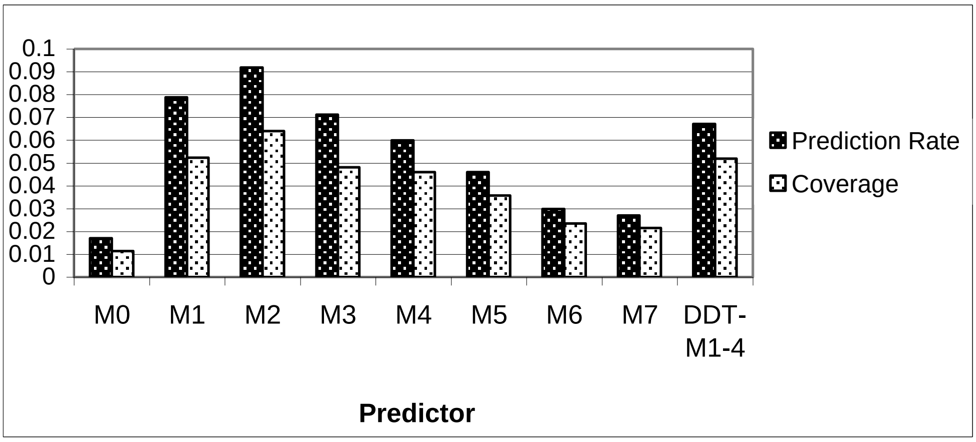


Figure 120. Comparing the average prediction rates and coverages on the BU dataset

Next, we varied the history size. Figure 121 presents the prediction accuracies obtained on the BU benchmarks using the optimal DDT-M1-4 predictor with 4-state confidence mechanism. We can observe that the optimal history size is 1000. For a history of 1500 the prediction accuracy is just slightly better.

Finally, we have extended the optimal DDT-M1-4 hybrid predictor with a Markov predictor component of order 0. A Markov chain of order 0 is not using any context information, it predicts the most frequent link from the considered history. Figure 122 presents comparatively the prediction accuracies obtained using DDT-M1-4 and DDT-M0-4, respectively. We have used the optimal history length of 1000 links.

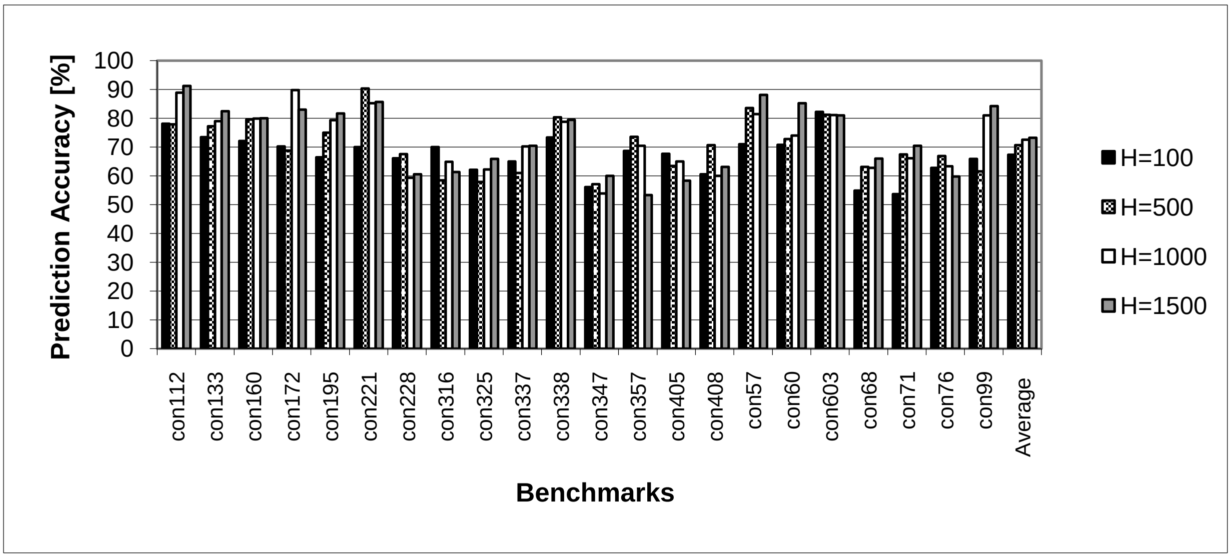


Figure 121. Prediction accuracies obtained with the DDT-M1-4 predictor using different link history lengths

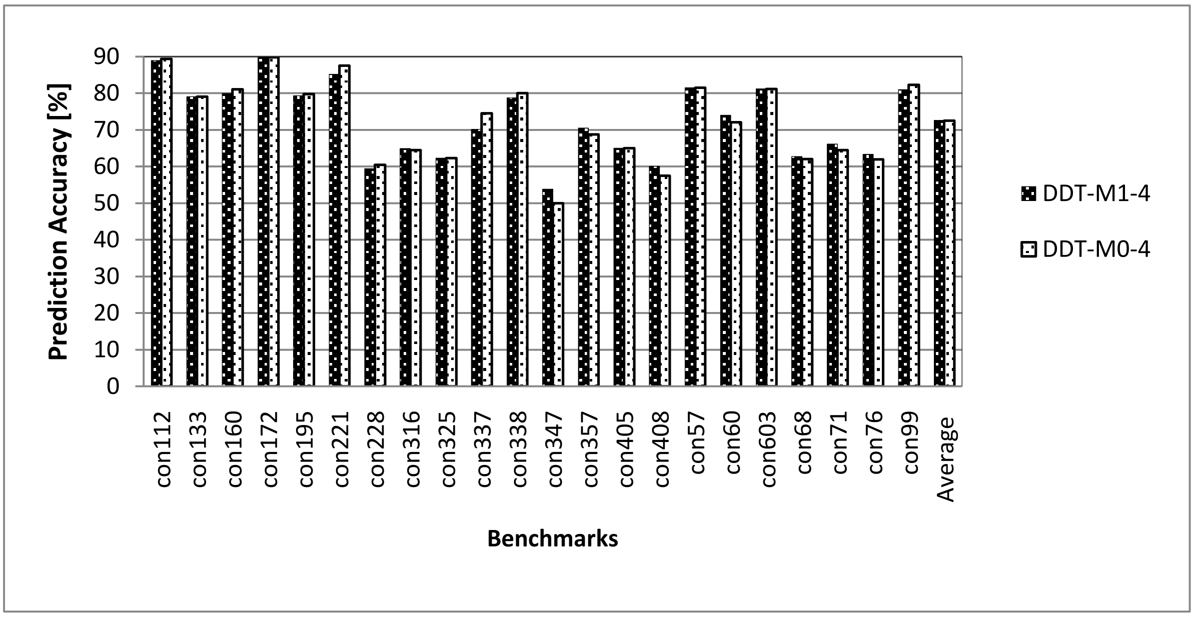


Figure 122. Extending the DDT-M1-4 hybrid predictor with a Markov predictor of order 0

As Figure 122 depicts, on some benchmarks is better to include the Markov predictor of order 0 and on others it is better without, such that the average prediction accuracies are almost the same: 72.57% without and 72.48% with 0 order Markov predictor, respectively. The Markov chain of order 0 cannot improve the average prediction accuracy due to its lack of context information. Thus, we prefer the DDT-M1-4 due to its simplicity with respect to the more complex hybridisation which additionally includes the 0 order Markov chain without any gain in accuracy. Thus, the optimal method remains DDT-M1-4 with an average prediction accuracy of 72.57%, but reaching almost 90% on some benchmarks (con112, con172).

6.2.4. Summary

In this section we presented a two-level hybrid web page prediction method consisting in a DDT with Markovian features. In the first level, the link history is translated into Markovian features represented by the predictions using Markov chains of different orders. These Markovian features are then used together with the static attributes, consisting in the current link and the link type, as input for the DDT algorithm to generate the final second level prediction. We have also used a confidence mechanism consisting in 4-state saturating counters, attached to all the links kept in the history, which allows selective prediction from high confidence contexts, improving thus the accuracy. The predicted web page or file is prefetched into the cache in order to be available for possible next accesses. The experiments performed on the BU dataset show that the optimal method is the DDT which uses as features the current link, the link type and the predictions provided by the Markov chains of orders 1-4. This optimal predictor uses a history of 1000 web accesses and achieves an average accuracy of 72.57%, outperforming all the Markov chains applied separately. It is also outperforming the PPM algorithm presented in [82] whose average prediction accuracy is 71.11%, measured on the same dataset. The DDT-based method has a relative prefetching benefit over the PPM-based algorithm of 1.52, which is remarkable. Furthermore, it is significantly outperforming the methods presented in [41], [123], [124] and [161], whose prediction accuracies have been mentioned in Section 2.

As further work, we intend to investigate other DDT features which can be relevant for even better prediction accuracy. Another further work direction consists in developing and evaluating SVM-based web page predictors.

II. DEVELOPMENT PLANS

7. Further Developments

Next, I will present my academic development plans in terms of teaching and research. I plan to always keep my teaching activities correlated with the research outcomes. Therefore, I will extend my lectures by covering the methods developed and applied during my research activities and the corresponding results and by exchanging information at national and international level. I will try to continuously improve my pedagogical skills and I intend to involve more the students in learning and research, especially at my courses from the master programs. I will support, attract and encourage students to work in research projects, to write scientific papers and to participate at conferences, scientific sessions and competitions. I will continue guiding students in elaborating bachelor and master theses and the Ph.D. students during their studies. I also intend to strengthen my collaborations with both academia (through projects, mobilities and joint research papers) and industry (through partnerships with companies). Some local and international collaborations were initiated within my recent research projects and I am going to start new such collaborations through other such projects. The research will be realized through projects and dissemination of the solutions and results in international journals and conferences and will focus on the topics and ideas presented in the next sections.

7.1. Anticipative Techniques in Multicore Architectures

In the previous works we focused on selective load value prediction as a hardware scheme which can improve superscalar, SMT and multicore microarchitectures in terms of performance and energy consumption. We would like to continue the investigations by developing such a selective predictor for other relatively high latency instructions too, like the multiply and the division. This time the selection can be performed earlier in the pipeline, more exactly in the dispatch stage, after the instruction is decoded and the opcode is available. However, the prediction table can be checked for hit or miss in the fetch stage. Taking into account the latency of such instructions, the gain will not probably be as high as in the case of load value prediction. The predicted results are forwarded to the dependent instructions which can thus be earlier executed. We are interested in accurate prediction, since any misprediction must be followed by a time-consuming recovery in order to reestablish the correct processor state.

Dynamic instruction reuse is another anticipative technique which can be applied selectively on the multiply and division instructions. We have previously applied this technique on superscalar and SMT architecture and we would like to evaluate its benefits in terms of performance and energy reduction in multicore architectures, too. As we concluded in [90] and [91], the reuse table can be accessed during the issue stage, because most of the multiply and division instructions do not have their operands ready in the dispatch stage. In contrast with the above-mentioned prediction method, dynamic instruction reuse is a non-speculative technique. If the multiply or division operation is found in the reuse buffer together with the actual operand values, the result can be forwarded to the dependent instructions whose earlier executions contribute to a better performance.

To evaluate the benefits of value prediction and reuse focused on the high-latency arithmetical operations in a multicore architecture, we need a simulator which provides access to the input and output operand values. Unfortunately, the necessary data is not available in the Sniper simulator [22] which was used in our recent research and, thus, we started to extend it with such capabilities. Sniper is a state-of-the-art multi-core simulator which uses the interval simulation method instead of a detailed cycle-by-cycle simulation of most of the existing simulators. The main idea is to approximate the performance of the cores using time intervals obtained using the encountered miss events (wrong branch prediction, cache misses, etc.) and to apply an analytical model to approximate the timing inside the intervals.

Sniper’s internal architecture contains four main clusters: frontend, Sniper Internal File Format (SIFT) pipes, scheduler and the backend. As a functional overview, the frontend is responsible to feed the scheduler via SIFT pipes with dynamic information about the executed instructions. The scheduler will associate for each application thread a simulated core performance model. The simulation flow starts with the binary file of a benchmark, which is executed on the host computer in real-time. Before execution, the binary file is instrumented in the simulator’s frontend. It uses the Intel Pin Tool to instrument, execute and analyze the binary file. During the execution of the program, the analyzer collects data about the execution flow and instructions. The data is feed in real-time towards the backend of the simulator via SIFT pipes. Sniper can also work with offline traces saved in previous runs, those can be loaded and sent directly to SIFT pipes. The frontend also adapts the instrumentation mode according to the selected simulation mode: fast-forward, warm-up or detailed. The Sniper Internal File Format trace contains an instruction stream which was generated by the execution of a program. Besides the execution order of instructions, it also contains additional information for each instruction.

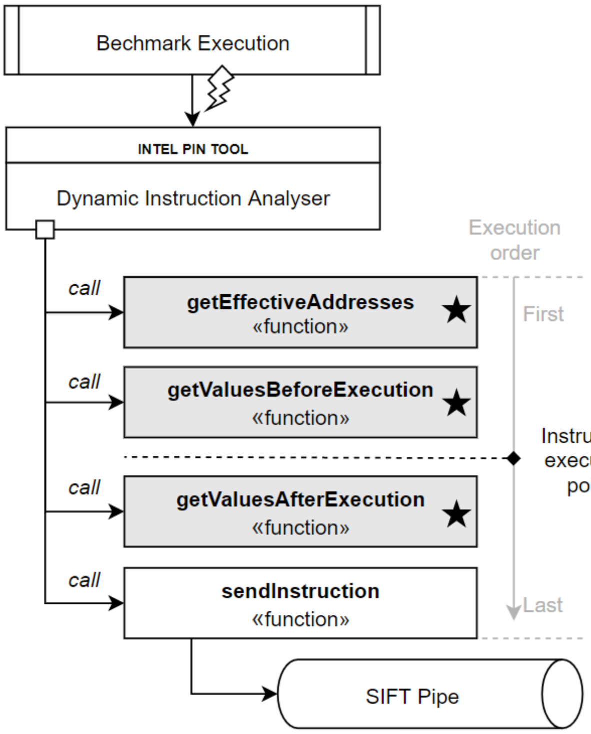

Bechmark Execution
INTEL PIN TOOL
Dynamic Instruction Analyser
Execution order
First
call
getEffectiveAddresses «function»
call
getValuesBeforeExecution «function»
Instruction execution point
call
getValuesAfterExecution «function»
call
sendInstruction «function»
Last
SIFT Pipe

Figure 123. Instrumentation flow after adaptations

Next, we present the modifications which were necessary in Sniper to access the operand values. The current version of Sniper 7.3 has limited access to the values of operands in the backend. The information is available only for branch instructions. We have adapted the simulator to provide details about the operands for multiple categories of instructions. In the frontend we take a snapshot of the values for all operands, before and after the execution of each instruction, and send it towards the backend using the SIFT pipes.

The frontend is the place where one can insert all the necessary callbacks, related to the instrumentation and analysis of a binary file. This process is done using the Intel Pin Tool. Initially it passes statically through basic blocks of instructions and inserts user defined calls for each instruction (code injection). This part is called binary instrumentation. It is performed at runtime on the already compiled binary files and there is no need to compile them again. The inserted code is executed during the program execution, this part being named the analysis process. For example, the injected code can receive access to specific context information (register contents, memory) snapshot, which gives possibilities to further analyze the behavior of the program.

The instrumentation process is done only once for all the static instructions, and the analysis is done for each dynamic instance of an instruction. Because of this, a connection between static information (program counter, opcode, number of operands, type of operands, etc.) and dynamic instruction data (e.g., effective addresses, value of operands, etc.) must be established. To store and later access the static information, we considered a thread-safe hash map using the program counter as the key.

Currently only one callback function is configured to be called for each instruction within a basic block trace. It starts with the execution of the benchmark via the Intel Pin Tool. The *sendInstruction* function is called for each dynamic instruction. Inside this function, all the dynamic information about the current instruction is gathered and sent via the corresponding SIFT Pipe. The information is propagated until the backend.

We adapted the frontend and added more callbacks (they are pointed with a star in Figure 123). Some of them are called before and others after the execution of each instruction. Depending on the simulation mode, those callbacks can be called or not. They are called in the detailed simulation mode. Those calls are registered in the instrumentation phase, for each static instruction. Also, at this point we store the static information about each instruction. Such information is: the instruction address (program counter), the number of operands and details about each operand (direction: read/write, type: register/memory/immediate, etc.). This static information is needed later in the analysis (dynamic) phase.

Those function calls will occur during the execution of each dynamic instruction. The execution order is as specified in the picture, first the *getEffectiveAddresses* function is called. Here the effective address of the memory operands is calculated and stored for later use (to read the memory location before/after the execution of the instruction). Next, the *getValuesBeforeExecution* and *getValuesAfterExecution* functions are called. In these functions the value for each operand is read either from a register or a memory location, the function name suggesting the reading point. The last called function from this chain is *sendInstruction*, which propagates the operand values, along with other dynamic information, to the backend of the simulator.

Further modifications were made also to the SIFT pipe infrastructure and the core performance model. The changes are related to extensions of data types to support additional information about the operands (value, type, and direction). The information is now available in the backend, more concrete, inside the core performance model.

The instruction reusability degree was measured in [17] using the following concept. A unique list was defined for each dynamic instruction. An element of the list contains the snapshot values of all the input (read) operands for this instruction at one execution point. The size of the list was varied. After the execution of each instruction, the list is updated in the following manner. If the list is empty, the elements will be added in order. The one at the end of the list will always be the newest. Before adding an element, all the existing elements are compared with the new operand values and if an existing element has the same value, then this element will be promoted as the newest. No new element will be added to the list in that case. This means that the instruction is executed again with the same input values, therefore it will be marked as reusable. In case the list is full and the current operand values are not present in the list, then the oldest element is removed and a new one is added with the actual operand values. Based on the presented results, the measured reusability looks promising and paves the way to implement a Dynamic Instruction Reuse (DIR) scheme in Sniper. We plan to enhance the Reuse Buffer (RB) from the DIR technique, to use a multi-value set-associative RB. It will be interesting to study the impact of the reuse technique, in a multi/many-core environment, considering the following microarchitectural metrics: computing performance, power consumption, total energy consumption and die temperature. It is also a point of interest for us to measure the performance gain by unlocking instruction chains due to data dependencies and compare them with our initial estimates.

A next step will be to also implement a selective value prediction technique in a multi-core system and to compare it with the DIR technique in fair conditions (to exploit the same value locality degree). The implementation should be done on the presented instruction types. The selectiveness is needed to consume less energy, because nowadays performance/Watt is more important than performance itself. A big challenge will be to correctly implement these techniques, taking care to not corrupt the coherency of the shared variables which are predicted by different execution threads.

Further, the proposed multi-core microprocessor architectures will be optimized using existing state-of-the-art multi-objective optimization techniques, integrated in an automatic design space exploration process. To achieve this, we plan to use FADSE, since it was specially developed to accelerate the design space exploration process using different methods (multi-objective heuristic algorithms, integrate human expert knowledge expressed through fuzzy rules, etc.). The optimization will be enhanced with novel multi-objective genetic algorithms (located in the Pareto-Fuzzy paradigm), using response surface modeling techniques and expert domain knowledge representation (e.g., fuzzy rules expressed in fuzzy logic).

7.2. Adaptive Assembly Assistance Systems

We intend to analyze the usefulness of a LSTM in modeling assembly processes, the idea being described in [173]. Thus, we will evaluate the ability of a LSTM to provide a possible next state, considering as input data the current state. In [86], we observed correlations of the assembly score and duration with the following characteristics of the workers: gender, eyeglasses wearer or not, height, and sleep quality in the preceding night. Therefore, besides the current assembly state we used all this additional input information in the prediction process. The LSTM is a recurrent neural network and therefore it could be appropriate in recognizing patterns in the input data.

LSTM has been specifically designed to deal with the long-term dependency of the information. In its current form, a LSTM unit is composed of a cell, tasked with memorising information over arbitrary periods of time, and three gates (the forget gate, the input gate and the output gate) tasked with regulating the flow of information in and out of the cell. The forget gate drops information that the model deems unnecessary in the decision-making process. The previous hidden state and the current input values are passed through a sigmoid function that dictates how much of the previous cell state should be dropped, with 0 representing drop all and 1 keep all. The input gate decides what new information should be learned. Like in the forget gate, the hidden state together with the input passes through the sigmoid in order to determine how much of the new information should be learned (ignore factor). Then, the hyperbolic tangent function squishes those values between -1 and 1. The cell state, the long-term memory, persists information from previous timesteps. The new cell state is obtained by multiplying the old state with the forget factor to which the new information from the input gate is added. The output gate decides what the next hidden state should be. First, we pass the current hidden state and the input through a sigmoid function. Then the newly modified cell state is passed through the tanh function. The information provided by the sigmoid and hyperbolic tangent functions is then multiplied in order to decide what information should the next hidden state carry. The newly computed hidden state is also the one used for predictions.

Due to the vast assembly possibilities of the tablet used as target product in the previous researches and lack of sufficient data, an LSTM network will be used to model the assembly process. The network is able to adapt to new assembly scenarios. For our implementation we will use the tensorflow library together with Keras. We will start the evaluations with a pre-configured LSTM and we will perform a design space exploration by systematically varying the main LSTM parameters in order to determine the optimal configuration for our application.

HMM is another candidate model. HMMs are doubly embedded stochastic processes composed of a hidden stochastic process and a set of observable stochastic processes, each state from the hidden stochastic process having associated an observable stochastic process (see Figure 124).

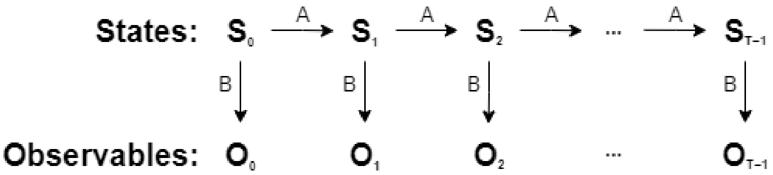


Figure 124. A generic HMM

There are three fundamental problems that an HMM should solve. These problems were introduced by Rabiner in his work [175]. We denote the observation sequence with O and the model with $\lambda = (\pi, A, B)$ where π is the set of initial hidden state probabilities, A is the set of transition probabilities between hidden states and B is the set of observable state probabilities. The first problem that an HMM should solve is called the likelihood problem, also known as the evaluation problem. It states that, given the HMM model $\lambda = (\pi, A, B)$ and the observation sequence O, the model should be able to determine the likelihood of that observation sequence, $P(O \mid \lambda)$. The second problem, called the decoding problem, states that given the model $\lambda = (\pi, A, B)$ and observation sequence O, the model should be able to tell what was the hidden state sequence that produced the observations. In the third problem, the learning problem, given the model λ and the observation sequence O, the model should adjust its parameters $\{A, B, \pi\}$ such that $P(O \mid \lambda)$ is maximized. According to Stamp [193], the third problem is used to train the model and the first problem is used to score the likelihood for a given observation sequence.

A possible HMM implementation is from the Pomegranate library. Pomegranate is a Python library developed by Jacob Schreiber that, among others, offers a good HMM implementation. The library offers us the possibility to train the model based on assembly sequences. The assembly steps are wrapped in an ObservableState, a wrapper class that contains information about the current assembly state and the worker’s characteristics, such as the gender (0 for female, 1 for male), height (0 for short and 1 for tall), eyeglass wearer or not (1 and 0, respectively), as well as sleep quality in the preceding night (0 for bad and 1 for good). After the model is trained on the assembly sequences, it can then be used to start predicting the next piece that the worker should mount.

To predict the next piece assembled by the worker, we need the history of assemblies done on the product so far. If no piece is assembled, then the history will contain an empty assembly (since the product has just begun being assembled) with the worker’s characteristics. The algorithm then proceeds to identify all the steps left to be done in the assembly sequence. After all the pieces left to be assembled have been identified, for each piece, a shadow assembly sequence containing the current assembly sequence and the piece that needs to be mounted is generated. Each shadow assembly sequence is then tested against the model to find the maximum likelihood of a shadow assembly to occur. The piece in the shadow assembly with the maximum likelihood is the one considered to be the next piece that should be assembled. Using this approach, of determining the maximum likelihood of an assembly sequence, enables the use of this algorithm for modeling unknown (earlier unseen) assemblies. A possible HMM-based implementation is proposed in [89]. Further in-depth comparisons are necessary.

We will also take into consideration Dynamic Bayesian Networks (DBN) as possible predictors in our assembly assistance system, some preliminary results being already presented in [172]. A Bayesian network, known also as a belief or causal network, is a probabilistic graphical method that models variables together with their conditional dependencies using a directed acyclic graph. The Bayesian networks that can model sequences considering the time factor and the evolution of variables through it, are known as DBN and are also called temporal Bayesian networks. A DBN connects variables to each other over adjacent time steps. The DBN can also be considered a two-time-slice Bayesian network (2TBN) since, at any given time, a variable's value can be computed using the internal regressors and the actual variable's value at time $t-1$. In the modelling of time series, the values of variables are observed at different time steps. Since time can move in only one direction, forward, the design of these networks is simplified and the directed edges should follow the direction of time.

Another possibility to build up assembly models is through decision trees (some preliminary results being presented in [190]) and through the A* algorithm (see the idea described in [88]).

7.3. Smart Energy Management Systems

Since Markov predictors are context-based, their main drawback is that predictions are solely based on the history accessed during the last time period. If a pattern appears for the first time, the context-based predictor cannot generate a prediction. Therefore, the next goal is to analyze some statistical modelling techniques such as the ARIMA and TBATS models.

The AutoRegressive Integrated Moving Average (ARIMA) model was introduced in 1970 by Box and Jenkins [13], and it is still widely used. The forecast R package provides a convenient interface to fitting an ARIMA model to data. These fitting and forecasting steps are somewhat equivalent to the "training" and "testing" terminology from the domain of neural networks for example. The ARIMA and SARIMA models only include information about the time-series itself in the model. The (S)ARIMAX algorithm incorporates external regressors.

The ARIMA model needs to be fed only with stationary data to yield good results. Data are said to be stationary when the statistical properties do not change regardless of the time window. This also means that statistical parameters of a time series, such as mean and variance, will be stable over time [111]. A statistical test that can determine whether a data set is stationary or not is the augmented Dickey-Fuller test [38]. The result of the test is a negative number. The lower this number, the higher the chances that the null hypothesis should be rejected (i.e., the tested time-series is non-stationary with a given percentage of confidence). Alternative stationarity tests also exist [47].

Seasonal time-series are not stationary, their properties changing depending on which time they are observed at. Transformations can be applied to them to make them stationary. One such transformation is differencing. Higher-order differences can be computed if needed. In a similar way, seasonal differencing is applied to seasonal data with a seasonal period of m. As noted in the description of SARIMA, the order of seasonal differences will be denoted by the D parameter for the model. Other examples of data transformations which make the series stationary include calendar transformations, logarithm transformations, power transformations, and the Box-Cox transformations that encompasses the preceding two [111].

The autocorrelation will drop to zero quite quickly for a stationary time series, while on the other hand the autocorrelation function of a non-stationary time series will decrease very slowly [111]. The autocorrelation and the partial autocorrelation will guide to choose the order of the autoregressive part and the order of the moving average part and also their seasonal counterparts

[111]. Once the parameters for the model are chosen and the model is fitted on the data, the residuals might be examined. Residuals should be zero-meaned, homoscedastic, not autocorrelated and ideally normally distributed [111]. The Ljung-Box test has been specifically developed for use with residuals generated by ARIMA models and this test statistic becomes helpful when comparing two ARIMA models [137].

SARIMA works best when individual observations represent aggregated data, because aggregation reduces the effect of noise and cyclic behavior is more evident. With data of finer granularity, dynamic harmonic regression could be better suited [111]. The seasonal pattern is modeled using Fourier series and the residuals are handled using ARIMA. The dynamic harmonic regression is included in the ARIMAX model, ARIMA with exogenous regressors. To determine how many Fourier terms to include, we will vary the number of Fourier terms and select the best performing model.

One-time events, for example public holidays, concerts or sport events that may last several days and influence electrical energy consumption can be modeled with indicator variables conventionally denoted as dummies.

Another statistical model is TBATS (Trigonometric Seasonal, Box-Cox Transformation, ARMA residuals, Trend and Seasonality) which can decompose seasonal time series into trend, seasonal and irregular components. The trigonometric terms from TBATS do not need normalization and the overall seasonal component can be decomposed to multiple seasonal components having different frequencies. TBATS allows automated model selection, which makes it easier to apply in comparison with ARIMA. Such statistical methods have been applied in [74] but further analysis and development is necessary.

Electricity consumption and production modeling is also possible through fuzzy logic, some preliminary results being presented in [158]. According to Figure 125, the fuzzy controller consists of four different modules or blocks: the fuzzification block, the inference engine, the rule base and the defuzzification block. Each module has its own set of tasks and they are all linked together in order to offer a result at the exit of the controller.

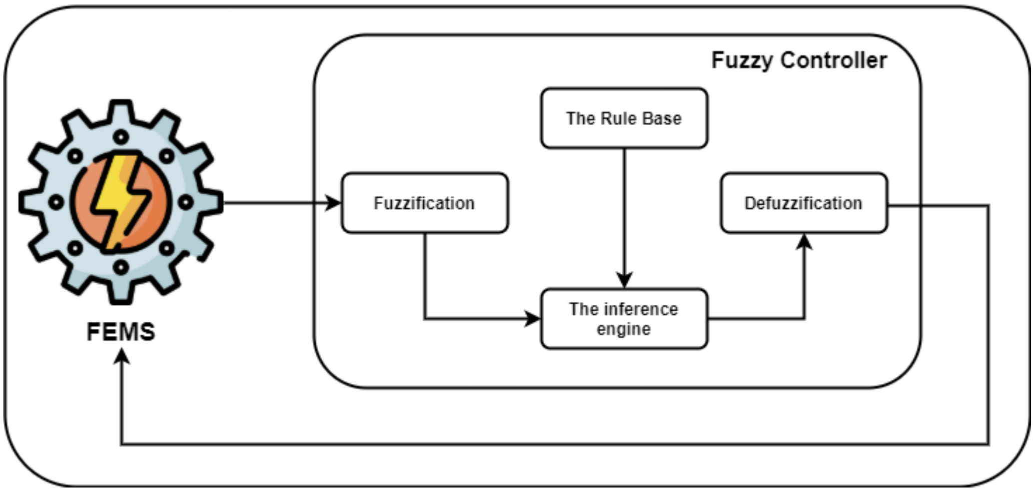


Figure 125. The structure of the fuzzy controller

We will follow the steps presented in [216] where a generic method of generating fuzzy rules from numerical data is shown. The first step consists of dividing the input and output data from the numerical data into fuzzy regions. This step is done in the fuzzification block and it is part of the preprocessing of the data. The fuzzification block will generate fuzzy rules from the data obtained in the previous step. Once the fuzzy regions are determined and the fuzzy rules are generated, the fuzzy rule base is created.

The next step would be finding the rules that would generate the output of the system. The defuzzification strategy for the pair of input values consists of combining the last fuzzy rules using the multiplication operation to determine the output corresponding to the given input. The rules are chosen by applying the input data onto every rule in the Fuzzy Rule Base. Thereby, if we have N rules in the rule base, it will generate N mappings, each one representing the membership degree of the input data to the rule. Fuzzy operators are used to determine the result of each mapping. The most common ones for "OR" and "AND" rules are the min and max operators. Finally, all the results are combined through the inference engine. It uses an inference method to compute all the mapping results and to determine the defuzzification area.

There are several defuzzification methods that can be used to process the output of the inference block. There is no method that would work on all the systems so it can be one of the configurable parameters of the system to find the best match. The most popular ones are the mean of maxima method, the weighted fuzzy mean method and the centroid method.

Using this five-step system, we end up choosing the fuzzification method, the membership function, the type of inference and the defuzzification method. To find the best configuration of the system, we need to analyze each output and change the parameters with the most suitable ones. Also, each block is linked to one another so a change in one block may affect the results of another one.

7.4. Context-Based Image Restoration

We have presented different context-based denoising and inpainting techniques implying Markov models. We intend as further work directions to extend the Markov filter to work not only on salt-and pepper noise, but also on random-valued impulse noise and, respectively, to apply Markov models to identify defects in images.

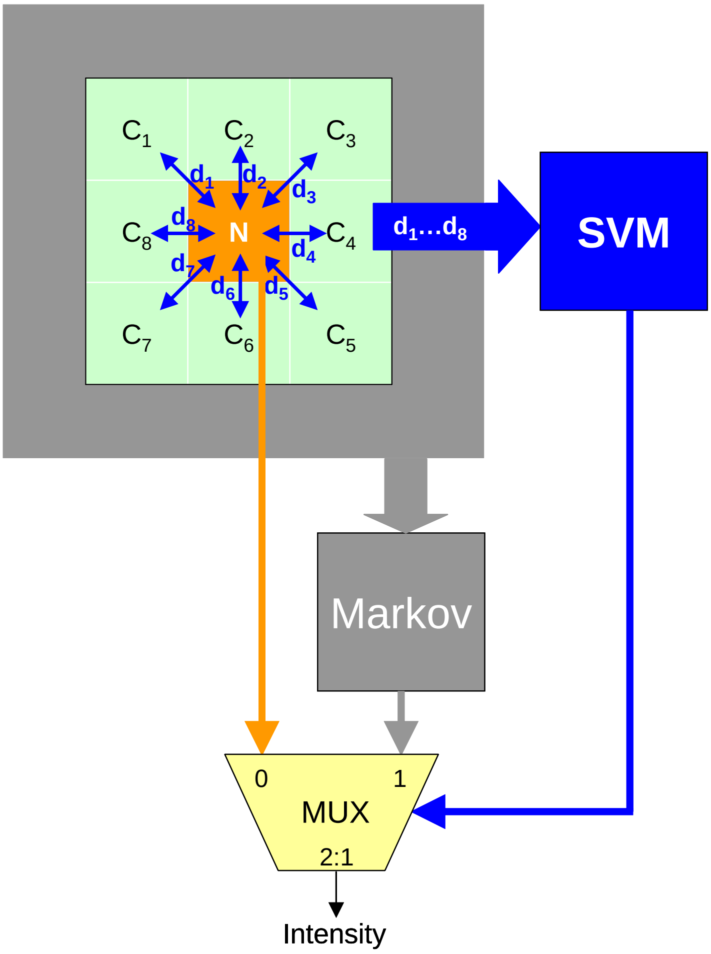


Figure 126. The scheme of the SVM-based Markov filter

Random-valued impulse noise is a pulse-type noise that alters pixels with random values over the entire range of [0, 255]. Therefore, to denoise images affected by such type of noise, first the noisy pixels might be identified and then the noisy pixels should be repaired. The pixels can be classified into noisy or unnoisy through different machine learning methods. We have already obtained some preliminary results in [78] by applying SVM, but further investigations are necessary. The method has two components: the SVM-based noise detection and the context-based filter applied on the detected noisy pixels. The scheme of the method is presented in Figure 126.

As Figure 126 illustrates, the differences between the intensity of the current pixel and the intensities of the surrounding 8 pixels are used by the SVM classifier to decide if the current pixel is noisy or not. If the current pixel is classified as unnoisy, then its intensity is not changed. On the other hand, if it is classified as noisy, then its intensity is replaced with the intensity provided by the Markov filter. We intend to apply also other potential neural classifiers such as MLP, LSTM or fully convolutional networks.

On the other hand, for defect detection in textured images we propose a context-based model to compute the difference between a defective and non-defective pixel in terms of intensity using values from their contexts. The given method can be used for textured images, where there is a greater diversity of pixel intensities, therefore we will use Markov chains to calculate the probability of occurrence for every pixel in the image. If the difference between the contexts is significant and the probability of the pixel occurring is small, we will consider the pixel as defective and we will highlight it. Compared to most defect detection methods for grayscale images, we propose a method that is able to identify defects in textured color images, to highlight the defect and to retain its color. Basically, for a certain pixel, the SADs between its context and the contexts of the pixels in the whole image are calculated. The contexts with SAD smaller than a threshold value were considered similar. We counted these similarities to obtain a frequency and based on that a probability of the pixel in the given context. In the end, if the probability is smaller than a probability threshold, we considered the initial pixel characterized by its context to be defective. A drawback of the method is the significantly increased execution time produced by the number of computations and another drawback is the difficulty of finding a threshold for different types of images that show specific differences in pixel intensity values. For a test image that has a larger difference between the two regions (texture and defect), the threshold value is different and considerably higher than a threshold value for an image in which the colors of the two regions (texture and defect) are closer.

To simplify the logic of the algorithm and at the same time to reduce the execution time, we intend to have a single data structure to store the information about the pixels. While before we used the SAD that went through two contexts pixel-by-pixel summing a maximum of eight differences for each RGB component every time we iterated through the image, now we intend to calculate only three differences, one for each RGB component, namely the differences between the arithmetic mean of each RGB component of the pixels from the two compared contexts. Thus, we approximate the SAD computed at the pixel level with the sum of average RGB distances, which can be obtained significantly faster.

7.5. Web Prefetching by Neural Hybrid Prediction

After we have evaluated the PPM algorithm which is a hybrid predictor with prioritization-based static component selection and the DDT which is a hybrid predictor with dynamic component selection, we would like to analyze also a neural hybrid web prefetching method based on the idea presented in [81]. Artificial neural networks are connectionist models composed of neurons organized on layers and can learn from examples to associate outputs to the given inputs. Each connection between two neurons has a weight, which indicates how strong a connection is. The inputs applied to the neural network are traversing the layers, being transformed according to the weights. If the output is wrong, the error can be backpropagated through the layers in order to correspondingly adjust the weights. The computation of the output for the given inputs does not require a specific algorithm, neural networks being thus widely used to solve high-complexity practical problems. Since we observed that the correct prediction is not provided always by the same predictor (it can differ among users, or it can change in time even for a certain user), a hybrid predictor could exploit the advantage of all its components. The structure of the browser extension is presented in Figure 127.

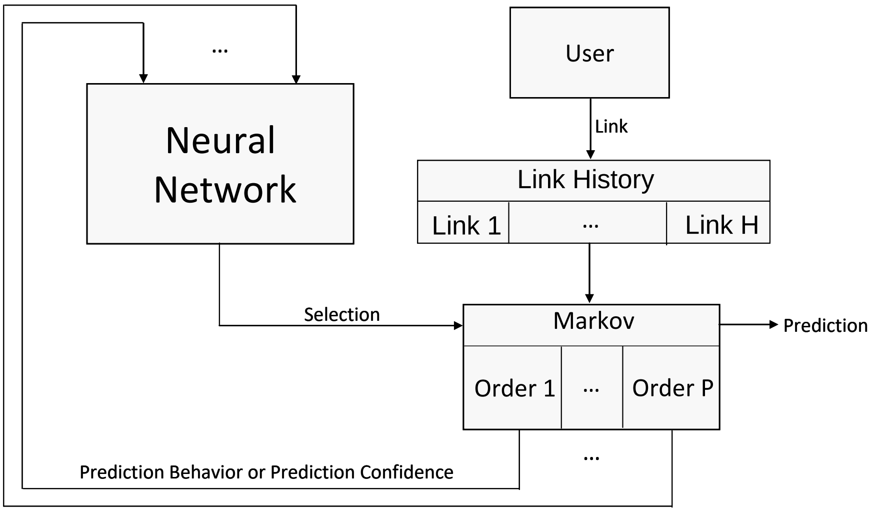


Figure 127. The structure of the browser extension

The proposed two-level predictor is using a neural network in its first level and contextual predictors in its second level. Our contextual predictors are Markov chains of different orders. The neural network from the first level selects based on the last behaviors of the component Markov chains of different orders from the second level, the one to be used to generate the prediction. The behaviors of a certain Markov model can be provided as codified confidence counters or as a limited history of correct / wrong predictions. The final prediction is then provided by the selected Markov model. The predicted web resources are prefetched into the cache of the browser in order to be available if the user will access them again in the future. The neural network has a good adaptability to dynamically select the best predictor. We expect that this novel neural hybrid web prefetcher will outperform the previously developed techniques in terms of prediction accuracy and rendering speedup, but further analysis and development is necessary.

III. REFERENCES

[1] Abdulkarim S., *Time series prediction with simple recurrent neural networks*, Bayero Journal of Pure and Applied Sciences, Vol. 9, No. 1, 2016.

[2] Aehnelt M., Bader S., *Information Assistance for Smart Assembly Stations. In: International Conference on Agents and Artificial Intelligence*, Vol. 2, pp. 143-150, 2015.

[3] Ahmed E., Yaqoob I., Hashem I.A.T., Khan I., Ahmed A.I.A., Imran M., Vasilakos A.V, *The role of big data analytics in Internet of Things.* Computer Networks, 129, pp. 459-471, 2017.

[4] Almonacid F., Pérez-Higueras P.J., Fernández E.F., Hontoria L., *A methodology based on dynamic artificial neural network for short-term forecasting of the power output of a PV generator*, Energy Conversion and Management, 85, pp. 389-398, 2014.

[5] Anderson B., Lin S., Newing A., Bahaj A., James P., *Electricity consumption and household characteristics: Implications for census-taking in a smart metered future*, Computers, Environment and Urban Systems, 63, pp. 58-67, 2017.

[6] Aujol J., Ladjal S., Masnou S., *Exemplar-based inpainting from a variational point of view*, SIAM Journal on Mathematical Analysis, 42, (3), pp. 1246–1285, 2010.

[7] Bachici M.-A., **Gellert A.**, *Modeling Electricity Consumption and Production in Smart Homes using LSTM Networks*, International Journal of Advanced Statistics and IT&C for Economics and Life Sciences, Vol. X, No. 1, pp. 80-90, December 2020.

[8] Berkovich H., Malah D., Barzohar M., *Non-local means denoising using a content-based search region and dissimilarity kernel*, 8th International Symposium on Image and Signal Processing and Analysis (ISPA), Trieste, pp. 10-15, 2013.

[9] Bertalmio M., Sapiro G., Caselles V., Ballester C., *Image inpainting*, 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 417–424, New Orleans, LA, USA, July 2000.

[10] Bertalmio M., Vese L., Sapiro G., Osher S., *Simultaneous structure and texture image inpainting*, IEEE Transactions on Image Processing, 12, (8), pp. 882–889, 2003.

[11] Bhatia A., Kulkarni R.K., *Removal of High Density Salt-And-Pepper Noise Through Improved Adaptive Median Filter*, International Conference on Computer Science and Information Technology, pp. 197-200, Bangalore, May 2012.

[12] Bouzerdoum M., Mellit A., Pavan A.M., *A hybrid model (SARIMA–SVM) for short-term power forecasting of a small-scale grid-connected photovoltaic plant*, Solar Energy, 98, pp. 226-235, 2013.

[13] Box G.E.P., Jenkins G.M., Reinsel G.C., Ljung G.M., *Time Series Analysis: Forecasting and Control*, 5th ed., Wiley, 2015.

[14] Brala M., Dhanda M., *An Improved Markov Model Approach to Predict Web Page Caching*, International Journal of Computer Science and Communication Networks, Vol. 2, pp. 393-399, 2012.

[15] Brooks D., Tiwari V., Martonosi M., *Wattch: A Framework for Architectural-Level Power Analysis and Optimizations*, Proc. of the 27th Int. Symposium on Computer Architecture, pp. 83-94, Vancouver, Canada, June 2000.

[16] Buduleci C., **Gellert A.**, Florea A., Chis R., Brad R., *Multi-Objective Optimization of Speculative and Anticipative Multi-Core Architectures*, Advanced Computer Architecture

and Compilation for High-performance Embedded Systems, pp. 11-14, Fiuggi, Italy, July 2020.

[17] Buduleci C., **Gellert A.**, Florea A., Matei A., *Extending Sniper with Support to Access Operand Values: A Case Study on Reusability Measurement*, 23rd International Carpathian Control Conference, Sinaia, May 2022.

[18] Bugeau A., Bertalmio M., Caselles V., Sapiro G., *A Comprehensive Framework for Image Inpainting*, IEEE Transactions on Image Processing, 19, (10), pp. 2634–2645, 2010.

[19] Burger D., Austin T., *The SimpleScalar Tool Set*, Version 2.0, Technical Report, University of Wisconsin, Madison, USA, June 1997.

[20] Calborean H., Jahr R., Ungerer T., Vintan L., *Optimizing a Superscalar System using Multi-objective Design Space Exploration*, Proc. of the 18th Int. Conf. on Control Systems and Computer Science (CSCS), pp. 339-346, Bucharest, Romania, May 2011.

[21] Canali C., Colajanni M., Lancellotti R., *Adaptive algorithms for efficient content management in social network services*, 10th International Conference on Computer and Information Technology, pp. 68-75, 2010.

[22] Carlson T.E., Heirman W., Eeckhout L., *Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation*, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Seatle, WA, USA, December 2011.

[23] Chang S.C., Li W.Y.H., Kuo Y.J., Chung C.P., *Early Load: Hiding Load Latency in Deep Pipeline Processor*, Proc. of the Asia-Pacific Computer Systems Architecture Conference (ACSAC), pp. 1-8, Taiwan, August 2008.

[24] Chen J., Dubois M., Stenström P., *Integrating Complete-System and User-Level Performance/Power Simulators: the SimWattch Approach*, Proc. of the IEEE Int. Symposium on Performance Analysis of Systems and Software, pp. 1-10, Austin, Texas, USA, March 2003.

[25] Chen I.-C.K., Coffey J.T., Mudge T.N., *Analysis of Branch Prediction via Data Compression*, ACM SIGPLAN Notices, 31(9), pp. 128-137, 1996.

[26] Chen C., Duan S., Cai T., Liu B., *Online 24-h solar power forecasting based on weather type classification using artificial neural network*, Solar Energy, 85(11), pp. 2856-2870, 2011.

[27] Chen S., Shi W., Zhang W., *An Efficient Universal Noise Removal Algorithm Combining Spatial Gradient and Impulse Statistic*, Mathematical Problems in Engineering, p. 480274, 2013.

[28] Chiacchio F., Petropoulos G., Pichler D., *The impact of industrial robots on EU employment and wages: A local labour market approach*, Working paper, Issue 2, Bruegel, 2018.

[29] Chis R., Vintan L., *Multi-Objective Hardware-Software Co-Optimization for the SNIPER Multi-Core Simulator*, 10th International Conference on Intelligent Computer Communication and Processing, pp. 3-9, Cluj-Napoca, September 2014.

[30] Cilardo A., Gallo L., Mazzocca N., *Design space exploration for high-level synthesis of multi-threaded applications*, *Journal of Systems Architecture*, Vol. 59, pp. 1171-1183, 2013.

[31] Ciobanu D., Dinuca C.E. *Predicting the next page that will be visited by a web surfer using Page Rank algorithm*, International Journal of Computers and Communications. Issue 1, Vol. 6, pp. 60-67, 2012.

[32] Cleary J., Witten I., *Data Compression Using Adaptive Coding and Partial String Matching*, IEEE Transactions on Communications. Vol. 32, No. 4, pp. 396-402, 1984.

[33] Criminisi A., Pérez P., Toyama K., *Region Filling and Object Removal by Exemplar-Based Image Inpainting*, IEEE Transactions on Image Processing, 13, (9), pp. 1200–1212, 2004.

[34] Cunha C.A., Bestavros A., Crovella M.E., *Characteristics of WWW Client Traces*, Technical Report TR-95-010, Boston University, Department of Computer Science, 1995.

[35] Deb K., Pratap A., Agarwal S., Meyarivan T., *A Fast and Elitist Multiobjective Genetic algorithm: NSGA-II*, IEEE Transactions On Evolutionary Computation, Vol. 6, No. 2, pp. 182-197, 2002.

[36] Deshpande M., Karypis G., *Selective Markov Models for Predicting Web-Page Accesses*, ACM Transactions on Internet Technology. Vol. 4, Issue 2, pp. 163-184, 2004.

[37] Desmet V., Girbal S., Temam O., France B. F., *Archexplorer. org: Joint compiler/hardware exploration for fair comparison of architectures,* Proc. of the 6th HiPEAC Industrial Workshop, Paris, France, November 2008.

[38] Dickey D.A., Fuller W.A., *Distribution of the estimators for autoregressive time series with a unit root*, J Am Stat Assoc, 74(366a), pp. 427–431, June 1979.

[39] Dijkman R. M., Sprenkels B., Peeters T., Janssen A., *Business models for the Internet of Things*, International Journal of Information Management, 35(6), pp. 672-678, 2015.

[40] Duan D., Mo Q., Wan Y., Han Z., *A Detail Preserving Filter for Impulse Noise Removal*, International Conference on Computer Application and System Modeling, pp. 265-268, Taiyuan, China, October 2010.

[41] Dubey S., Mishra N., *Web Page Prediction using Hybrid Model*, International Journal on Computer Science & Engineering, Vol. 3 Issue 5, pp. 2170-2176, 2011.

[42] Duranton M., Yehia S., De Sutter B., De Bosschere K., Cohen A., Falsafi B., Gaydadjiev G., Katevenis M., Maebe J., Munk H., Navarro N., Ramirez A., Temam O., Valero M., *The HiPEAC Vision*, HiPEAC Roadmap, 2010.

[43] Durillo J.J., Nebro A.J., Luna F., Dorronsoro B., Alba E., *jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics*, E.T.S.I. Informatica, Campus de Teatinos: Departamento de Lenguajes y Ciencias de la Computacion, University of Malaga, 2006.

[44] Efros A., Leung T., *Texture Synthesis by Non-parametric sampling*, 7th IEEE International Conference on Computer Vision, pp. 1033-1038, Corfu, Greece, September 1999.

[45] Elharrouss O., Almaadeed N., Al-Maadeed S., Akbari Y., *Image Inpainting: A Review*, Neural Processing Letters, 2019.

[46] Elkomy M., Abdelrahman Y., Funk M., Dingler T., Schmidt A., Abdennadher S., *ABBAS: An Adaptive Bio-sensors Based Assistive System*, Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 2543- 2550, 2017.

[47] Elliott G., Stock J., Rothenberg T.J., *Efficient tests for an autoregressive unit root*, Econometrica, 64, pp. 813–836, July 1996.

[48] Endo F.A., Perais A., Seznec A., *On the Interactions Between Value Prediction and Compiler Optimizations in the Context of EOLE*, ACM Transactions on Architecture and Code Optimization, Vol. 14, Issue 2, July 2017.

[49] Esakkirajan S., Veerakumar T., Subramanyam A.N., PremChand C.H., *Removal of High Density Saltand Pepper Noise Through Modified Decision Based Unsymmetric Trimmed Median Filter*, IEEE Signal Processing Letters, 18, (5), pp. 287-290, May 2011.

[50] Escrivá-Escrivá G., Álvarez-Bel C., Roldán-Blay C., Alcázar-Ortega M., *New artificial neural network prediction method for electrical consumption forecasting based on building end-uses*, Energy and Buildings, 43(11), pp. 3112-3119, 2011.

[51] Fan C., Xiao F., Wang S., *Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques*, Applied Energy, Vol. 127, 2014.

[52] Feilmeier S., *Loads management based on Photovoltaic and Energy Storage System*, M.Sc. Thesis, “Lucian Blaga” University of Sibiu, Supervisor: A. Gellert, 2015.

[53] Fernandez-Jimenez L.A., Muñoz-Jimenez A., Falces A., Mendoza-Villena M., Garcia-Garrido E., Lara-Santillan P.M., Zorzano-Santamaria P.J., *Short-term power forecasting system for photovoltaic plants*, Renewable Energy, 44, pp. 311-317, 2012.

[54] Florea A., Buduleci C., Chis R., **Gellert A.**, Vintan L., *Enhancing the Sniper Simulator with Thermal Measurement*, The Eighteenth International Conference on System Theory, Control and Computing, Sinaia, October 2014.

[55] Florea A., **Gellert A.**, Vintan L., Veltan M., *The Impact of Java Applications at Microarchitectural Level from Branch Prediction Perspective*, International Journal of Computers, Communications & Control, 4(1), pp 27-40, 2009.

[56] Florea A., Radu C., Calborean H., Crapciu A., **Gellert A.**, Vintan L., *Designing an Advanced Simulator for Unbiased Branches Prediction*, Proceedings of 9th International Symposium on Automatic Control and Computer Science, Iasi, 2007.

[57] Florea A., Radu C., Calborean H., Crapciu A., **Gellert A.**, Vintan L., *Understanding and Predicting Unbiased Branches in General-Purpose Applications*, Bulletin of the Polytechnic Institute of Iasi, Tom LIII (LVII), Fasc. 1-4, Section IV, Iasi, 2007.

[58] Florea A., **Gellert A.**, *Memory Wall – A Critical Factor in Current High-Performance Microprocessors*, Science and Supercomputing in Europe, pp. 257-264, Barcelona, Spain, 2006.

[59] Florea A., Ratiu A., **Gellert A.**, Vintan L., *A Visual Simulation Framework for Simultaneous Multithreading Architectures*, The 25th European Conference on Modelling and Simulation (ECMS 2011), Krakow, Poland, June 2011.

[60] Florea A., Klein A., Badea V., Stefanescu M., **Gellert A.**, *Using FOCAP Tool for Teaching Microarchitecture Simulation and Optimization*, The 17th International Conference on System Theory, Control and Computing (ICSTCC 2013), Sinaia, October 2013.

[61] Funk M., Dingler T., Cooper J., Schmidt A., *Stop helping me – I'm bored! Why assembly assistance needs to be adaptive*, 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and 2015 ACM International Symposium on Wearable Computers, pp. 1269-1273, Osaka, Japan, 2015.

[62] Funk M., Bächler A., Bächler L., Kosch T., Heidenreich T., Schmidt A., *Working with Augmented Reality? A Long-Term Analysis of In-Situ Instructions at the Assembly Workplace*, 10th ACM International Conference on Pervasive Technologies Related to Assistive Environments, pp. 222-229. Island of Rhodes, Greece, June 2017.

[63] Gabbay F., Mendelson A., *Using Value Prediction to Increase the Power of Speculative Execution Hardware*, ACM Transactions on Computer Systems, Vol. 16, Issue 3, August 1998.

[64] Gajowniczek K., Ząbkowski T., *Short term electricity forecasting using individual smart meter data*, Procedia Computer Science, 35, pp. 589-597, 2014.

[65] Gallant S.I., *Neural Networks and Expert Systems*, MIT Press., 1993.

[66] Gambs S., Killijian M.O., del Prado Cortez M.N., *Next Place Prediction using Mobility Markov Chains*, Proceedings of the First Workshop on Measurement, Privacy, and Mobility, p. 3, New York, USA, April 2012.

[67] **Gellert A.**, Florea A., Vintan L., *Exploiting Selective Instruction Reuse and Value Prediction in a Superscalar Architecture*, Journal of Systems Architecture, Elsevier, Volume 55, Issue 3, pp. 188-195, March 2009.

[68] **Gellert A.**, Palermo G., Zaccaria V., Florea A., Vintan L., Silvano C., *Energy-Performance Design Space Exploration in SMT Architectures Exploiting Selective Load Value Predictions*, Int. Conf. on Design, Automation and Test in Europe (DATE 2010), Dresden, Germany, pp. 271-274, March 2010.

[69] **Gellert A.**, Calborean H., Vintan L., Florea A., *Multi-Objective Optimizations for a Superscalar Architecture with Selective Value Prediction*, IET Computers & Digital Techniques, Vol. 6, Issue 4, pp. 205-213, Stevenage, United Kingdom, July 2012.

[70] **Gellert A.**, Florea A., Fiore U., Zanetti P., Vintan L., *Performance and Energy Optimisation in CPUs through Fuzzy Knowledge Representation*, Information Sciences, Elsevier, Vol. 476, pp. 375-391, February 2019.

[71] **Gellert A.**, Vintan L., *A Multicore Architecture with Selective Load Value Prediction*, Proceedings of the Romanian Academy, Series A, Vol. 19, No. 4, pp. 597-604, November 2018.

[72] **Gellert A.**, Vintan M., Vintan L., *Perceptron-Based Selective Load Value Prediction in a Multicore Architecture*, Romanian Journal of Information Science and Technology, Vol. 22, No. 3-4, pp. 215-227, November 2019.

[73] **Gellert A.**, Florea A., Fiore U., Palmieri F., Zanetti P., *A study on forecasting electricity production and consumption in smart cities and factories*, International Journal of Information Management, Vol. 49, pp. 546-556, December 2019.

[74] **Gellert A.**, Fiore U., Florea A., Chis R., Palmieri F., *Forecasting Electricity Consumption and Production in Smart Homes through Statistical Methods*, Sustainable Cities and Society, Vol. 76, January 2022.

[75] **Gellert A.**, Vintan L., *Person Movement Prediction Using Hidden Markov Models*, Studies in Informatics and Control, 15(1), pp. 17-30, March 2006.

[76] **Gellert A.**, Brad R., *Studying the influence of search rule and context shape in filtering impulse noise images with Markov chains*, Signal, Image and Video Processing, 12(2), pp. 315-322, February 2018.

[77] **Gellert A.**, Brad R., *Context-Based Prediction Filtering of Impulse Noise Images*', IET Image Processing, 10(6), pp. 429-437, June 2016.

[78] **Gellert A.**, Brad R., Morariu D., Neghina M., *Filtering Random Valued Impulse Noise from Grayscale Images through Support Vector Machine and Markov Chain*, International Journal of Advanced Statistics and IT&C for Economics and Life Sciences, Vol. 11, Issue 1, pp. 70-84, December 2021.

[79] **Gellert A.**, Brad R., *Image Inpainting with Markov Chains*, Signal, Image and Video Processing, Vol. 14, Issue 7, pp. 1335-1343, October 2020.

[80] **Gellert A.**, *Web Access Mining through Dynamic Decision Trees with Markovian Features*, Journal of Web Engineering, Vol. 16, Issue 5-6, pp. 524-536, 2017.

[81] **Gellert A.**, *Web Usage Mining by Neural Hybrid Prediction with Markov Chain Components*, Journal of Web Engineering, River Publishers, Vol. 20, Issue 5, pp. 1279-1296, Denmark, July 2021.

[82] **Gellert A.**, Florea A., *Web Prefetching through Efficient Prediction by Partial Matching*, World Wide Web: Internet and Web Information Systems, Vol. 19, Issue 5, pp. 921-932, September 2016.

[83] **Gellert A.**, Florea A., *Web Page Prediction Enhanced with Confidence Mechanism*, Journal of Web Engineering, Vol. 13, Issue 5-6, pp. 507-524, USA, November 2014.

[84] **Gellert A.**, Zamfirescu C.-B., *Using Two-Level Context-Based Predictors for Assembly Assistance in Smart Factories*, 8th International Conference on Computers Communications and Control, Oradea, May 2020.

[85] **Gellert A.**, Zamfirescu C.-B., *Assembly support systems with Markov predictors*, 20th Open Conference of the IFIP WG 8.3 on Decision Support, Wrocław, Poland, June 2020.

[86] **Gellert A.**, Precup S.A., Pirvu B.C., Zamfirescu C.B., *Prediction-Based Assembly Assistance System*, 25th International Conference on Emerging Technologies and Factory Automation, Vienna, Austria, September 2020.

[87] **Gellert A.**, Precup S.-A., Pirvu B.-C., Fiore U., Zamfirescu C.-B., Palmieri F., *An Empirical Evaluation of Prediction by Partial Matching in Assembly Assistance Systems*, Applied Sciences, Vol. 11, Issue 7, ISSN 2076-3417 (ISI Thomson Journals IF=2.474, Scopus), DOI 10.3390/app11073278, p. 3278, Switzerland, April 2021.

[88] **Gellert A.**, Sorostinean R., Pirvu B.-C., *Robust Assembly Assistance Using Informed Tree Search with Markov Chains*, Sensors, Vol. 22, Issue 2, p. 495, January 2022.

[89] **Gellert A.**, Precup S.-A., Matei A., Pirvu, B.-C., Zamfirescu C.-B., *Real-Time Assembly Support System with Hidden Markov Model and Hybrid Extensions*, Mathematics, Vol. 10, Issue 15, p. 2725, August 2022.

[90] **Gellert A.**, *Advanced prediction methods integrated into speculative computer architectures*, PhD Thesis, Computer Science Department, "Lucian Blaga" University of Sibiu, November 2008.

[91] **Gellert A.**, *Beyond the Limits of Modern Processors*, Matrix Rom Publishing House, Bucharest, 2008.

[92] **Gellert A.**, Florea A., *Investigating a new design pattern for efficient implementation of prediction algorithms*, Journal of Digital Information Management, Vol. 11, Issue 5, pp. 366-377, 2013.

[93] **Gellert A.**, Florea A., Vintan M., Egan C., Vintan L., *Unbiased Branches: An Open Problem*, Twelfth Asia-Pacific Computer Systems Architecture Conference, pp. 16-27, Seoul, Korea, August 2007.

[94] **Gellert A.**, Florea A., *Finding and Solving Difficult Predictable Branches*, Science and Supercomputing in Europe, pp. 265-271, Barcelona, Spain, 2006.

[95] **Gellert A.**, Vintan L., Florea A., *A Systematic Approach to Predict Unbiased Branches*, Lucian Blaga University Press, 111 pp., 2007.

[96] Gorecky D., Khamis M., Mura K., *Introduction and establishment of virtual training in the factory of the future*, International Journal of Computer Integrated Manufacturing, Vol. 30, Issue 1, pp. 182-190, 2017.

[97] Graves A., *Supervised Sequence Labelling with Recurrent Neural Networks*, Studies in Computational Intelligence, Springer, ISBN: 978-3-642-24796-5, January 2012.

[98] Guillemot C., Turkan M., Le Meur O., Ebdelli M., *Image Inpainting using LLE-LDNR and Linear Subspace Mappings*, 38th IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 1558–1562, Vancouver, Canada, May 2013.

[99] Guillemot C., Le Meur O., *Image inpainting: overview and recent advances*, IEEE Signal Processing Magazine, 31, (1), pp. 127–144, 2014.

[100] Guo Z., Zhou K., Zhang C., Lu X., Chen W., Yang S., *Residential electricity consumption behavior: Influencing factors, related theories and intervention strategies*, Renewable and Sustainable Energy Reviews, 81, pp. 399-412, 2018.

[101] Hadhoud M.M., Moustafa K.A., Shenoda S.Z., *Digital Images Inpainting using Modified Convolution Based Method*, Optical Pattern Recognition XX, 7340, 2009.

[102] Hamza A.B., Luque-Escamilla P., Martínez-Aroza J., Román-Roldán R., *Removing Noise and Preserving Details with Relaxed Median Filters*, Journal of Mathematical Imaging and Vision, 11(2), pp. 161-177, October 1999.

[103] Hancock P.A., Jagacinski R.J., Parasuraman R., Wickens C.D., Wilson G.F., Kaber D.B., *Human-Automation Interaction Research: Past, Present, and Future*, Ergonomics in Design: The Quarterly of Human Factors Applications, Vol. 21, Issue 2, pp. 9-14, April 2013.

[104] Hashem I.A.T., Chang V., Anuar N.B., Adewole K., Yaqoob I., Gani A., Ahmed E., Chiroma H., *The role of big data in smart city*, International Journal of Information Management, 36(5), pp. 748-758, 2016.

[105] He K., Shen C.-N, Niu J.-H., Huang W.-R., *Effective inpainting method for natural textures with gradually changed illumination*, Signal, Image and Video Processing, 13, pp. 69-77, 2019.

[106] Hernández L., Baladrón C., Aguiar J.M., Carro B., Sánchez-Esguevillas A., Lloret J., *Artificial neural networks for short-term load forecasting in microgrids environment*, Energy, 75, pp. 252-264, 2014.

[107] Hilton A.D., *Energy Efficient Load Latency Tolerance: Single-Thread Performance For The Multi-Core Era*', Publicly accessible Penn Dissertations. Paper 188, 2010.

[108] Hochreiter S., Schmidhuber J., *Long Short-Term Memory*, Neural Computation, Vol. 9, No. 8, pp. 1735-1780, 1997.

[109] Horng S.J., Hsu L.Y., Li T., Qiao S., Gong X., Chou H.H., Khan M.K., *Using sorted switching median filter to remove high-density impulse noises*, Journal of Visual Communication and Image Representation, 24(7), pp. 956-967 2013.

[110] Howell S., Rezgui Y., Hippolyte J.-L., Jayan B., Li H., *Towards the next generation of smart grids: Semantic and holonic multi-agent management of distributed energy resources*, Renewable and Sustainable Energy Reviews, 77, pp. 193-214, 2017.

[111] Hyndman R.J., Athanasopoulos G., *Forecasting: Principles and Practice,* 2nd ed., OTexts, Melbourne, Australia, 2018.

[112] Izgi E., Öztopal A., Yerli B., Kaymak M.K., Şahin A.D., *Short–mid-term solar power prediction by using artificial neural networks*, Solar Energy, 86(2), pp. 725-733, 2012.

[113] Jääskinen V., Parkkinen V., Cheng L., Corander J., *Bayesian clustering of DNA sequences using Markov chains and a stochastic partition model*, Statistical Applications in Genetics and Molecular Biology, 13(1), pp. 105-121, February 2014.

[114] Jahr R., Ungerer T., Calborean H., Vintan L., *Automatic Multi-Objective Optimization of Parameters for Hardware and Code Optimizations*, Proc. of the Int. Conf. on High Performance Computing & Simulation, pp. 308-316, Istanbul, Turkey, July 2011.

[115] Jahr R., Calborean H., Ungerer T., Vintan L., *Boosting Design Space Explorations with Existing or Automatically Learned Knowledge*, 16th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems" and "Dependability and Fault-Tolerance, Kaiserslautern, Germany, March 2012.

[116] Jassim F.A., *Kriging Interpolation Filter to Reduce High Density Salt and Pepper Noise*, World of Computer Science and Information Technology Journal, 3, (1), pp. 8-14, 2013.

[117] Jia Z.J., Pimentel A.D., Thompson M., Bautista T., Núnez A., *NASA: A generic infrastructure for system-level MP-SoC design space exploration*, 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia), pp. 41–50, 2010.

[118] Jiménez D., Lin C., *Dynamic Branch Prediction with Perceptrons*, Proceedings of the Seventh International Symposium on High Performance Computer Architecture (HPCA-7), Monterrey, Nuevo Leon, Mexico, January 2001.

[119] Jiménez D., Lin C., *Neural Methods for Dynamic Branch Prediction*, ACM Transactions on Computer Systems, Vol. 20, No. 4, November 2002.

[120] Jiménez D., *Fast Path-Based Neural Branch Prediction*, 36th International Symposium on Microarchitecture, San Diego, CA, USA, December 2003.

[121] Kang S., Kumar R., *Magellan: a search and machine learning-based framework for fast multi-core design space exploration and optimization*, Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1432-1437, Munich, Germany, 2008.

[122] Kaushal P., *Hybrid Markov model for better prediction of web page*, International Journal of Scientific and Research Publications. Vol. 2, Issue 8, 2012.

[123] Khalil F., Li J., Wang H., *Integrating Recommendation Models for Improved Web Page Prediction Accuracy*, Proceedings of the 31st Australasian Conference on Computer Science, Vol. 74, pp. 91-100, Wollongong, Australia, 2008.

[124] Khalil F., Li J., Wang H., *An Integrated Model for Next Page Access Prediction*, International Journal of Knowledge and Web Intelligence, Vol. 1, No. 1/2, pp. 48-80, 2009.

[125] Khanchana R., Punithavalli M., *Web Page Prediction for Web Personalization: A Review*, Global Journal of Computer Science and Technology, Vol. 11, Issue 7, pp. 39-44, 2011.

[126] Korn O., Schmidt A., Hörz T., *Assistive Systems in Production Environments: Exploring Motion Recognition and Gamification*, 5th International Conference on Pervasive Technologies Related to Assistive Environments, pp. 1-5. Heraklion, Crete, Greece, June 2012.

[127] Korn O., Schmidt A., Hörz T., *Augmented manufacturing: A Study with Impaired Persons on Assistive Systems Using In-Situ Projection,* 6th International Conference on Pervasive Technologies Related to Assistive Environments, Rhodes, Greece, May 2013.

[128] Krejcar O., Frischer R., *Non Destructive Defect Detection by Spectral Density Analysis*, Sensors, 11(3), pp. 2334-2346, 2011.

[129] Lal S., Kumar S., Chandra M., *Removal of High Density Salt & Pepper Noise Through Super Mean Filter for Natural Images*, International Journal of Computer Science Issues, 9, (3), pp. 303-309, May 2012.

[130] Laube P., Grunwald M., Franz M.O., Umlauf G., *Image Inpainting for High-Resolution Textures using CNN Texture Synthesis*, Computer Graphivs & Visual Computing, Swansea, UK, 2018.

[131] Lee J., Shi Y., Wang F., Lee H., Kim H.K., *Advertisement Clicking Prediction by using Multiple Criteria Mathematical Programming*, World Wide Web Journal, 19(4), July 2016.

[132] Lim C., Kim K.H., Kim M.J., Heo J.Y., Kim K.J., Maglio P.P., *From data to value: A nine-factor framework for data-based value creation in information-intensive services*, International Journal of Information Management, 39, pp. 121-135, 2018.

[133] Lin T.C., *SVM-based Filter Using Evidence Theory and Neural Network for Image Denoising*, Journal of Software Engineering and Applications, 6, (3B), pp. 106-110, 2013.

[134] Lipasti M.H., Wilkerson C.B., Shen J.P., *Value Locality and Load Value Prediction*, Proc. of the 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, pp. 138-147, Cambridge, Massachusetts, USA, October 1996.

[135] Liu L., Chen C.P., Zhou Y., You X., *A new weighted mean filter with a two-phase detector for removing impulse noise*, Information Sciences, 315, pp. 1-16, 2015.

[136] Liu G., Reda F., Shih K., Wang T.-C., Tao A., Catanzaro B., *Image Inpainting for Irregular Holes Using Partial Convolutions*, European Conference on Computer Vision, pp. 85-100, Munich, Germany, 2018.

[137] Ljung G.M., Box G.E.P., *On a measure of lack of fit in time series models*, Biometrika, 65(2), pp. 297–303, March 1978.

[138] Loskyll M., Heck I., Schlick J., Schwarz M., *Context-Based Orchestration for Control of Resource-Efficient manufacturing Processes*, Future Internet, 4(3), pp. 737-761, 2012.

[139] Lowd D., Davis J., *Improving Markov Network Structure Learning Using Decision Trees*, Journal of Machine Learning Research, Vol. 15, pp. 501-532, 2014.

[140] Lu C.T., Chou T.C., *Denoising of Salt-and-Pepper Noise Corrupted Image using Modified Directional-Weighted-Median Filter*, Pattern Recognition Letters, 33, (10), pp. 1287-1295, July 2012.

[141] Ma Y., Liu X., Bai S., Wang L., He D., Liu A., *Coarse-to-Fine Image Inpainting via Region-wise Convolutions and Non-Local Correlation*, 28th International Joint Conference on Artificial Intelligence, pp. 3123-3129, Macao, China, 2019.

[142] Majumdar A., Ward R.K., *Synthesis and Analysis Prior Algorithms for Joint-Sparse Recovery*, IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3421-3424, March 2012.

[143] Mamdani E.H., Assilian S., *An experiment in linguistic synthesis with a fuzzy logic controller*, International Journal of Man-Machine Studies, 7(1), pp.1–15, 1975.

[144] Martin M.M.K., Sorin D.J., Cain H.W., Hill M.D., Lipasti M.H., *Correctly Implementing Value Prediction in Microprocessors that Support Multithreading or Multiprocessing*, 34th Annual ACM/IEEE International Symposium on Microarchitecture, Austin, Texas, December 2001.

[145] Mellit A., Pavan A.M., *Performance prediction of 20 kWp grid-connected photovoltaic plant at Trieste (Italy) using artificial neural network.*, Energy Conversion and Management, 51(12), pp. 2431-2441, 2010.

[146] Miguel J.S., Badr M., Jerger N.E., *Load Value Approximation*, Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 127-139, Cambridge, UK, December 2014.

[147] Miguel J.S., Albericio J., Jerger N.E., Jaleel A., *The Bunker Cache for Spatio-Value Approximation*, 49th Annual IEEE/ACM International Symposium on Microarchitecture, Taipei, Taiwan, October 2016.

[148] Mishra B.K., Bharadi V.A., Nemade B., Thakur K.V., Damodare O.H., Sapkal A.M., *Poisson Noise Reducing Bilateral Filter*, Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016, Procedia Computer Science, Vol. 79, pp. 861-865, 2016.

[149] Mitchell T., *Machine Learning*, McGraw-Hill, 1997.

[150] Monteiro C., Fernandez-Jimenez L.A., Ramirez-Rosado I.J., Muñoz-Jimenez A., P.M. Lara-Santillan, *Short-Term Forecasting Models for Photovoltaic Plants: Analytical versus Soft-Computing Techniques*, Mathematical Problems in Engineering, Vol. 2013, 2013.

[151] Mushtaq A., Lee C.-H., *An Integrated Approach to Feature Compensation Combining Particle Filters and Hidden Markov Model for Robust Speech Recognition*, IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 4757-4760, Kyoto, Japan, March 2012.

[152] Mutlu O., Kim H., Patt Y.N., *Address-Value Delta (AVD) Prediction: A Hardware Technique for Efficiently Parallelizing Dependent Cache Misses*, IEEE Transactions on Computers, Vol. 55, No. 12, pp. 1491-1508, 2006.

[153] Nair M.S., Shankar V., *Predictive-Based Adaptive Switching Median Filter for Impulse Noise Removal using Neural Network-Based Noise Detector*, Signal and Video Processing, 7, (6), pp. 1041-1070, November 2013.

[154] Narvekar M., Banu S.S., *Predicting User's Web Navigation Behavior Using Hybrid Approach*, International Conference on Advanced Computing Technologies and Applications, Vol. 45, 3-12, Mumbai, India, March 2015.

[155] Nasri M., Saryazdi S., Nezamabadi-pour H., *A fast Adaptive salt and pepper noise reduction method in images*, Circuits, Systems, and Signal Processing, 32(4), pp. 1839-1857, 2013.

[156] Nyberg R.A., *Using 'smartness' to reorganise sectors: Energy infrastructure and information engagement*, International Journal of Information Management, 39, pp. 60-68, 2018.

[157] Oancea M., **Gellert A.**, Florea A., Vinţan L., *Analyzing Branch Prediction Contexts Influence*, Advanced Computer Architecture and Compilation for Embedded Systems (ACACES 2006), pp. 5-8, L'Aquila, Italy, July 2006.

[158] Olaru L.M., **Gellert A.**, Fiore U., Palmieri F., *Electricity Production and Consumption Modeling through Fuzzy Logic*, International Journal of Intelligent Systems, Vol. 37, Issue 11, pp. 8348-8364, November 2022.

[159] Oliveira M.M., Bowen B., McKenna R, Chang Y.-S., *Fast digital image inpainting*, Proceedings of the International Conference on Visualization, Imaging and Image Processing, pp. 261–266, Marbella, Spain, 2001.

[160] Orosa L., Azevedo R., Mutlu O., *AVPP: Address-First Value-Next Predictor with Value Prefetching for Improving the Efficiency of Load Value Prediction*, ACM Transactions on Architecture and Code Optimization, Vol. 15, Issue 4, January 2019.

[161] Pamutha T., Chimphlee S., Kimpan C., Sanguansat P., *Web Page Access Prediction on Server Side. Journal of Convergence Information Technology*, Vol. 9, No. 5, September 2014.

[162] Park J., Yi D., Ji S., *Analysis of Recurrent Neural Network and Predictions*, Symmetry, Vol. 12, No. 4, 2020.

[163] Patel P., Prajapati A., Mishra S., *Review of Different Inpainting Algorithms', International Journal of Computer Applications*, 59, (18), pp. 30–34, 2012.

[164] Perais A., Seznec A., *Practical Data Value Speculation for Future High-End Processors*, 20th International Symposium on High Performance Computer Architecture, pp. 428-439, Orlando, FL, USA, February 2014.

[165] Perais A., Seznec A., *EOLE: Paving the Way for an Effective Implementation of Value Prediction*, 41st Annual International Symposium on Computer Architecture, pp. 481-492, Minneapolis, MN, USA, June 2014.

[166] Perais A., Seznec A., *BeBoP: A Cost Effective Predictor Infrastructure for Superscalar Value Prediction*, 21st International Symposium on High Performance Computer Architecture, pp. 13-25, San Francisco, CA, USA, February 2015.

[167] Perais A., Seznec A., *EOLE: Combining Static and Dynamic Scheduling through Value Prediction to Reduce Complexity and Increase Performance*, ACM Transactions on Computer Systems, Vol. 34, Issue 2, May 2016.

[168] Peruzzini M., Pellicciari M., *A framework to design a human-centred adaptive manufacturing system for aging workers*, Advanced Engineering Informatics, Vol. 33, pp. 330-349, 2017.

[169] Petzold J., Bagci F., Trumler W., Ungerer T., Vintan L., *Global State Context Prediction Techniques Applied to a Smart Office Building*, Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, California, January 2004.

[170] Pirvu B.-C., Zamfirescu C.-B., Gorecky D., *Engineering insights from an anthropocentric cyber-physical system: A case study for an assembly station*, Mechatronics, Vol. 34, pp. 147-159, 2016.

[171] Powell A., Savvas-Bouganis C., Cheung P.Y.K., *High-level power and performance estimation of FPGA-based soft processors and its application to design space exploration*, Journal of Systems Architecture, Vol. 59, pp. 1144-1156, 2013.

[172] Precup S.-A., **Gellert A.**, Matei A., Gita M., Zamfirescu C.-B., *Towards an Assembly Support System with Dynamic Bayesian Network*, Applied Sciences, Vol. 12, Issue 3, p. 985, Switzerland, February 2022.

[173] Precup S.-A., **Gellert A.**, Dorobantiu A., Zamfirescu C.-B., *Assembly Process Modeling Through Long Short-Term Memory*, 13th Asian Conference on Intelligent Information and Database Systems, pp. 28-39, Phuket, Thailand, April 2021.

[174] Prost-Boucle A., Muller O., Rousseau F., *Fast and Standalone Design Space Exploration for High-Level Synthesis under Resource Constraints*, Journal of Systems Architecture, 60(1), pp. 79-93, January 2014.

[175] Rabiner L.R., *A tutorial on hidden Markov models and selected applications in speech recognition*, Proc. IEEE, Vol. 77, Issue 2, pp. 257–286, 1989.

[176] Quinlan J.R., *Induction of Decision Trees*, Machine Learning, Vol. 1, Issue 1, pp. 81-106, 1986.

[177] Radinsky K., Svore, K.M., Dumais, S.T., Shokouhi, M., Teevan, J., Bocharov, A., Horvitz, E., *Behavioral Dynamics on the Web: Learning, Modeling, and Prediction*, ACM Transactions on Information Systems, Vol. 31, Issue 3, July 2013.

[178] Radu C., Calborean H., Crapciu A., **Gellert A.**, Florea A., *An Interactive Graphical Trace-Driven Simulator for Teaching Branch Prediction in Computer Architecture*, The 6th EUROSIM Congress on Modelling and Simulation, (EUROSIM 2007), Ljubljana, Slovenia, September 2007.

[179] Radu C., Calborean H., Florea A., **Gellert A.**, Vintan L., *Exploring Some Multicore Research Opportunities. A First Attempt*, Advanced Computer Architecture and Compilation for Embedded Systems (ACACES 2009), Terrassa, Spain, 2009.

[180] Rojas R., Rauch E., Dallasega P., Matt D.T., *Safe Human-Machine Centered Design of an Assembly Station in a Learning Factory Environment*, International Conference on Industrial Engineering and Operations Management, pp. 403-411, Bandung, Indonesia, March 2018.

[181] Romero D., Noran O., Stahre J., Bernus P., Fast-Berglund Å., *Towards a Human-Centred Reference Architecture for Next Generation Balanced Automation Systems: Human-Automation Symbiosis*, IFIP International Conference on Advances in Production Management Systems, pp. 556-566, Tokyo, Japan, 2015.

[182] Rouf M., Ward R.K., *Retrieving information lost by image denoising*, 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 1066-1070, Orlando, FL, 2015.

[183] Sembroiz D., Careglio D., Ricciardi S., Fiore U., *Planning and Operational energy optimization solutions for Smart Buildings*, Information Sciences, 476, pp. 439-452, February 2019.

[184] Shafiq M., Hussain G.A., Elkalashy, N.I., Hyvonen P., Lehtonen M., *Integration of online proactive diagnostic scheme for partial discharge in distribution networks.*, IEEE Transactions on Dielectrics and Electrical Insulation, 22(1), pp. 436-447, 2015.

[185] Sharkey J., Ponomarev D., Ghose K., *M-SIM: A Flexible, Multithreaded Architectural Simulation Environment*, Technical Report CS-TR-05-DP01, Department of Computer Science, State University of New York at Binghamton, October 2005.

[186] Shyamkumar T., Naveen M., Ho A.J., Jouppi N., *CACTI 5.1*, Technical Report HPL-2008-20, 2008.

[187] Silvano C., Fornaciari W., Palermo G., Zaccaria V., Castro F., Martinez M., Bocchio S., Zafalon R., Avasare P., Vanmeerbeeck G., Ykman-Couvreur C., Wouters M., Kavka C., Onesti L., Turco A., Bondi U., Mariani G., Posadas H., Villar E., Wu C., Dongrui F., Hao Z., Shibin T., *MULTICUBE: Multi-Objective Design Space Exploration of Multi-Core Architectures*, Proc. of the IEEE Int. Annual Symposium on VLSI, pp. 488–493, Lixouri Kefalonia, Greece, 2010.

[188] Singhai N., Nigam R.K., *A Novel Technique to Predict Oftenly Used Web Pages from Usage Patterns*, International Journal of Emerging Trends & Technology in Computer Science. Vol. 1, Issue 4, 49-55, 2012.

[189] Smolka B., Kusnik D., *Robust local similarity filter for the reduction of mixed Gaussian and impulsive noise in color digital images*, Signal, Image and Video Processing, 9(1), pp. 49-56, 2015.

[190] Sorostinean R., **Gellert A.**, Pirvu B.-C., *Assembly Assistance System with Decision Trees and Ensemble Learning*, Sensors, Vol. 21, Issue 11, p. 3580, May 2021.

[191] Srinivasan K.S., Ebenezer D., *A New Fast and Efficient Decision Based Algorithm for Removal of High Density Impulse Noise*, IEEE Signal Processing Letters, 14(3), pp. 189-192, 2007.

[192] Srivastava S., Baptista M.S., *Markovian language model of the DNA and its information content*, Royal Society Open Science, January 2016.

[193] Stamp M., *A Revealing Introduction to Hidden Markov Models*, Introduction to Machine Learning with Applications in Information Security, 1st ed.; Chapman and Hall/CRC: New York, NY, USA, 2017.

[194] Stork S., Schubö A., *Human cognition in manual assembly: Theories and applications*, Advanced Engineering Informatics, 24(3), pp. 320-328, 2010.

[195] Surrah H.A., *Impulse Noise Removal from Highly Corrupted Images using new Hybrid Technique based on Neural Networks and Switching Filters*, Journal of Global Research in Computer Science, 5, (3), pp. 1-7, 2014.

[196] Tan Q., Tong Y., Wu S., Li D. *Anthropocentric Approach for Smart Assembly: Integration and Collaboration*, Journal of Robotics, February 2019.

[197] Temgire S., Gupta P., *Review on Web Prefetching Techniques*, International Journal of Technology Enhancements and Emerging Engineering Research. Vol. 1, Issue 4, pp. 100-105, 2013.

[198] Thompson M., Pimentel A.D., *Exploiting domain knowledge in system-level MPSoC design space exploration*, J. Syst. Architect., 59(7), pp. 351-360, 2013.

[199] Tocu N.A., **Gellert A.**, Stefan I.R., Nitescu T.M., Luca G.A., *The Impact of Virtual reality Simulators in Manufacturing Industry*, 12th Annual International Conference on Education and New Learning Technologies, Palma de Mallorca, Spain, July 2020.

[200] Toh K.K.V., Isa N.A.M., *Noise Adaptive Fuzzy Switching Median Filter for Salt-and-Pepper Noise Reduction*, IEEE Signal Processing Letters, 17(3), pp. 281-284, March 2010.

[201] Vatjus-Anttila J., Kreku J., Korpi J., Khan S., Saastamoinen J., Tiensyrjä, K., *Early-phase performance exploration of embedded systems with ABSOLUT framework.*, Journal of Systems Architecture, Vol. 59, pp. 1128-1143, 2013.

[202] Vintan L., **Gellert A.**, Florea A., Oancea M., Egan C., *Understanding Prediction Limits Through Unbiased Branches*, Eleventh Asia-Pacific Computer Systems Architecture Conference (ACSAC 2006), published in Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, Vol. 4186/2006, pp. 480-487, Shanghai, China, September 2006.

[203] Vintan L., Florea A., **Gellert A.**, *Forcing Some Architectural Ceilings of the Actual Processor Paradigm*, Invited Paper, The 3rd Conference of The Academy of Technical Sciences from Romania (ASTR), Cluj-Napoca, November 2008.

[204] Vintan L., Florea A., **Gellert A.**, *Random Degrees of Unbiased Branches*, Proceedings of the Romanian Academy, Series A, Vol. 9, No. 3, 2008.

[205] Vintan L., Florea A., **Gellert A.**, *Focalising dynamic value prediction to CPU's context*, IEE Proceedings – Computers and Digital Techniques, Stevenage, United Kingdom, Volume 152, Issue 4, pp. 473-481, July 2005.

[206] Vintan L., **Gellert A.**, Florea A., *Register Value Prediction Using Metapredictors*, Proceedings of the Eighth International Symposium on Automatic Control and Computer Science, Iasi, October 2004.

[207] Vintan L., **Gellert A.**, Florea A., *Value Prediction Focalized on CPU Registers*, Advanced Computer Architecture and Compilation for Embedded Systems (ACACES 2005), Academia Press, pp. 181-184, Ghent, Belgium, July 2005.

[208] Vintan L., Chis R., Ali Ismail Md., Cotofana C., *Improving Computing Systems Automatic Multi-Objective Optimization through Meta-Optimization*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 35, Issue 7, pp. 1125-1129, July 2016.

[209] Vintan L., **Gellert A.**, Petzold J., Ungerer T., *Person Movement Prediction Using Neural Networks*, Technical Report 2004-10, Institute of Computer Science, University of Augsburg, Germany, 2004.

[210] Vintan L., **Gellert A.**, Petzold J., Ungerer T., *Person Movement Prediction Using Neural Networks*, Proceedings of the KI2004 International Workshop on Modeling and Retrieval of Context, Vol-114, Ulm, Germany, September 2004.

[211] Waghchawre A.J., Shinde J.V., *Query Mining for Image Retrieval System Using Markov Chain Model*', International Conference on Emerging Trends in Computer Engineering, Science and Information Technology, pp. 109-112, India, 2015.

[212] Wan M., Jönsson A., Wang C., Li L., Yang Y., *Web user clustering and Web prefetching using Random Indexing with weight functions*, Knowledge and Information Systems. Vol. 33, Issue 1, pp. 89-115 2012.

[213] Wang Z., Zhang D., *Progressive Switching Median Filter for the Removal of Impulse Noise from Highly Corrupted Images*, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 46(1), pp. 78-80, 1999.

[214] Wang G., Li D., Pan W., Zang Z., *Modified Switching Median Filter for Impulse Noise Removal*, Signal Processing, 90, (12), pp. 3213-3218, December 2010.

[215] Wang Y., Tao X., Qi X., Shen X., Jia J., *Image Inpainting via Generative Multi-Column Convolutional Neural Networks*, 32nd Conference on Neural Information Processing Systems, Montréal, Canada, 2018.

[216] Wang L.-X., Mendel J.M., *Generating fuzzy rules by learning from examples*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 12, Issue 6, 1992.

[217] Watson J. Pek A., *Welcome to the sunny-side of energy*, The United Nations Climate Change Conference (COP23), Bonn, Germany, pp. 42-45, 2017.

[218] Wei Y., Liu S., *Domain-based structure-aware image inpainting*, Signal, Image and Video Processing, 10, (5), pp. 911-919, 2016.

[219] Weron R., *Electricity price forecasting: A review of the state-of-the-art with a look into the future*, International Journal of Forecasting, 30(4), pp. 1030-1081, 2014.

[220] Wong A., Mishra A., Zhang W., Fieguth P., Clausi D.A., *Stochastic Image Denoising Based on Markov-Chain Monte Carlo Sampling*, Signal Processing, 91(8), pp. 2112-2120, 2011.

[221] Woo S.C., Ohara M., Torrie E., Singh J.P., Gupta A., *The SPLASH-2 Programs: Characterization and Methodological Considerations*, 22nd International Symposium on Computer Architecture, pp. 24-36, Italy, June 1995.

[222] Xiong W., Yu J., Lin Z., Yang J., Lu X., Barnes C., Luo J., *Foreground-Aware Image Inpainting*, 2019 IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 2019.

[223] Xu Z., Wu H.R., Yu X., Qiu B., *Adaptive progressive filter to remove impulse noise in highly corrupted color images*, Signal, Image and Video Processing, 7(5), pp. 817-831, 2013.

[224] Yang C., Lu X., Lin Z., Shechtman E., Wang O., Li H., *High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis*, 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 2017.

[225] Ye Z., Kim M.K., *Predicting Electricity Consumption in a Building Using an Optimized Back-propagation and Levenberg–Marquardt Back-propagation Neural Network: Case Study of a Shopping Mall in China*, Sustainable Cities and Society, 2018.

[226] Yin J., Sharma P., Gorton I., Akyoli B., *Large-scale data challenges in future power grids*, 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 324-328, March 2013.

[227] Yoon B.-J., *Hidden Markov Models and their Applications in Biological Sequence Analysis*, Current Genomics, 10(6), pp. 402-415, September 2009.

[228] Zadeh L.A., *Fuzzy Sets*, Journal of Information and Control, Vol. 8, No. 3, pp. 338-353, 1965.

[229] Zeng X., Yang L., *Mixed Impulse and Gaussian Noise Removal using Detail-Preserving Regularization*, Optical Egineering, 49(9), p. 097002, September 2010.

[230] Zhang R., Ren Y., Qiu J., Li G., *Base-Detail Image Inpainting*, 30th British Machine Vision Conference, Cardiff, UK, 2019.

[231] Zhao H.X. Magoulès F., *A review on the prediction of building energy consumption*, Renewable and Sustainable Energy Reviews, 16(6), pp. 3586-3592, 2012.

[232] Zheng Z., Wei W., Liu C., Cao W., Cao L., Bhatia M., *An Effective Contrast Sequential Pattern Mining Approach to Taxpayer Behavior Analysis*, World Wide Web Journal, Vol. 19, pp. 633-651, 2016.