ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **20**, Number 3, 2017, 241–255

Towards Synergic Meta–Algorithmic Approaches in Complex Computing Systems

Lucian N. VINŢAN

Computer Science and Electrical Engineering Department,
Lucian Blaga University of Sibiu,
E. Cioran Str. No. 4, 550025, Sibiu, Romania
Email: lucian.vintan@ulbsibiu.ro

Abstract. Today for almost any complex problem there are a set of state of the art refined powerful basic algorithms, quite effective in solving it. Frequently, there is not an ideal algorithm for a given complex research problem. Thus, for such complex applications it would be recommendable to use state of the art algorithms that will be concurrently orchestrated through a meta-algorithmic abstraction layer. In this paper we review this fertile idea. So, we analysed a meta-optimization method proposed in order to find the best (Pareto) individuals in a bi-objective space. Some qualitative and quantitative results were presented and explained. Also it was shown that meta-algorithms were efficient in documents' classification domain. An effective adaptive meta-classification scheme for text documents was presented. Finally, it was shown that meta-algorithmic approaches are very effective in developing meta-predictors, too. A generic adaptive meta-prediction scheme was developed. In all these validation cases it was shown that the meta-algorithmic approaches involve useful synergism.

Key-words: Complex Computing Systems, Meta–Algorithm, Meta–Optimization, Multi-Objective (Pareto) Optimization, Meta–Classification, Meta–Prediction.

1. Introduction

In our days for almost any computing systems' problem there are a set of state of the art refined powerful algorithms effective in solving it. Frequently there is not an ideal algorithm for a given complex research problem. Almost all of these effective algorithms are already implemented in some (often free access) libraries; therefore we can use them without knowing all their intrinsic implementation details. For example, the well-known jMetal library provides many state-of-the-art multi-objective evolutionary and bio-inspired algorithms, including some corresponding quality indicators (ex. hyper-volume), too. It is difficult, but obviously not impossible,

to improve these state of the art basic algorithms. However, instead of focusing on such a difficult task, sometimes it would be more recommendable to use such state of the art complementary algorithms that will be concurrently orchestrated through a meta-algorithmic abstraction layer, in order to solve a complex difficult problem. In this work we review this fertile idea, mainly based on some of our concrete research experiences, but also based on some other fertile published researches. Thus, in the further paragraphs we are trying to present and to qualitatively analyse, some useful applications of the meta-algorithmic approach in meta-optimization, meta-classification and meta-prediction.

A meta-algorithm mainly means an algorithm that was developed to manipulate some other component (base) algorithms. Meta-algorithms are powerful methods in order to design intelligent high effective robust (especially learning) systems. Meta-algorithms differ from algorithms at the level of abstraction they are implemented and at the level of adaptation, too. They are exploiting the accumulated experiences and performances of the component base algorithms, which are working together on a given task. Adaptive meta-algorithms are trying to understand the run-time contexts influence on the component algorithms, and, based on this understanding, they would decide about the most appropriate decision, at a given time. If in a certain meta-algorithmic approach the interaction of the base algorithms generates greater effects than the sum of their individual effects, we say that the meta-algorithm involves synergy that clearly shows their added performance value. It might be useful to observe here that meta-algorithms and synergy are natural in parallel systems. Taking into account the present-day extremely complex applications, often involving hierarchical control systems, the importance of migrating algorithms at a superior hierarchy abstraction level – called meta-algorithmic level - becomes more and more important. In fact, the idea of the paper was quite intuitive: I realized that during my research work I and my colleagues implicitly developed meta-algorithmic approaches in order to break the algorithmic limits and to manage the projects' complexity. Therefore, I considered useful to write a paper that explicitly points out this fact.

This work mainly represents a personal critical review of meta-algorithms development, implementation, utilization and evaluation in optimization, classification, and prediction research fields. The work is based on some fertile valuable references, but also on some of the author's original published research in this field. We want to point out that, due to impressive complexity growth of the present-day computing systems, meta-algorithmic approaches are a must in designing such hardware-software systems, in order to exploit in a synergic manner the complementary qualities of different state of the art effective component algorithms, and, therefore, to break the algorithms' intrinsic limitations.

This article is structured as follows: Section 2 provides an overview of the meta-optimization concept, including authors personal research experience. Meta-Classification and Run-time Meta-Prediction concepts are presented in Sections 3 and 4. Section 5 concludes the paper.

2. Review of Meta-Optimization

Meta-optimization has been used in Compiler Heuristics [1], Portfolio Selection [2], Direct Search Optimization Methods [3], Large Scale Parameter Optimization [4], etc. For example, meta-optimization has been used as a method to automatically fine-tune compiler heuristics [1]. The developed meta-optimization algorithms used machine-learning methods to automatically search the compiler heuristics' space. It is shown that the proposed meta-optimization method significantly reduces the compilers design complexity. Their developed software system uses

a genetic algorithm to automatically find several effective compiler heuristics. In [2] the authors analysed meta-algorithms which work by adaptively combining some iterative optimization techniques from a library of base optimization algorithms. It was rigorously shown that the performance of such meta-algorithms was competitive with the best convex combination of the iterative optimization methods that is remarkable. In other words, the meta-optimization results were comparable with the superposition of the optimization methods' results. The authors have adequately proven that solutions from multiple iterative algorithms can be combined by meta-algorithms to outperform the single best base algorithm. In [3] the so-called Complex-RF method is analysed and optimized, by processing a range of representative problems. In order to evaluate the developed optimization method it is proposed a new performance index derived from information theory. It was shown that this performance index can be used to optimize parameters within the optimization algorithm itself, and to compare different optimization algorithms, too. Additionally, the authors are showing that it is possible to concretely state how difficult an objective function is to optimize, compared to a reference objective function. In [4] the authors present the meta-optimization implementation for their heuristic optimization developed environment. It is proven that the search for optimal parameters' values can be seen as an optimization problem itself which can be solved by a meta-heuristic optimization algorithm, called meta-optimization algorithm. The approach of using a meta-level algorithm to find the optimal parameters' values has proven to work very well on some certain representative validation cases.

Recently, as far as we know, we were the first that introduced the idea of multi-objective meta-optimization in Computer Architecture domain [5]. The main aim of this research was to automatically find the Pareto individuals, in a huge superscalar processor design space, by minimizing two contradictory objectives: the clocks per cycle number (performance metric) and the hardware complexity. The meta-optimization rationale consists in the fact that state of the art multi-objective optimization algorithms might be very effective from complementary points of view. For example, some of them perform better from the solutions performance point of view while other algorithms are converging faster [6]. Thus, based on our experience in this research field, there is not a "best optimization algorithm". In other words, according to the "no free lunch theorem", there is no optimization algorithm that can perform always better than all the other algorithms. In our meta-optimization approach, we selected two complementary well-known genetic multi-objective algorithms: NSGA-II and SPEA2 [6], that work together in order to improve both the solutions' quality and the convergence speed. Thus, the main aim was to obtain Pareto quasi-optimal solutions, with better convergence speed and diversity, in the same amount of time as a single optimization algorithm.

In our view meta-optimization represents a scientific method where an optimization algorithm is applied – as a meta-level optimizer – to other optimization (usually heuristic) algorithms. According to Figure 1, the algorithms which are optimized are called base-level algorithms (meta-heuristics in the presented case), while the one that solves the global parameterized optimization problem, is called meta-level algorithm or hyper-heuristic level. The main difference between hyper-heuristics and meta-optimization is that the former uses heuristics to drive other low level meta-heuristics, while the latter uses various forms of control at any layer of the problem.

Therefore, instead of focusing on improving some state of the art multi-objective meta-heuristics (genetic algorithms, bio-inspired, swarm optimization, etc.) we tried to develop an upper abstraction level containing an adaptive hyper-heuristic algorithm integrated into a meta-optimization layer. This layer controls the two multi-objective genetic algorithms working on

the same size populations. This hyper-heuristic algorithm – implementing the meta-optimization level – dynamically adjusts the proportion of individuals submitted by each meta-heuristic to the master population, according to its global performance.

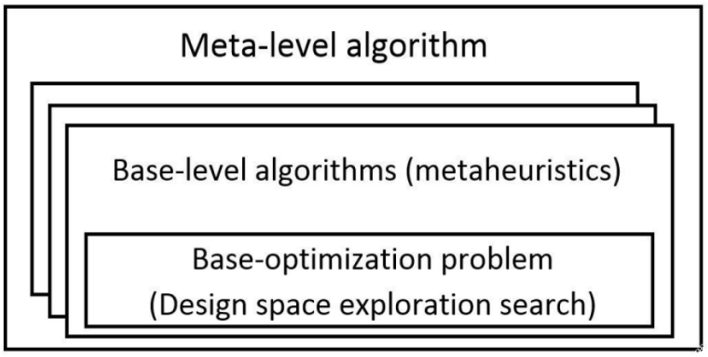

Fig. 1. The Meta-Optimization Metaphor.

The multi-objective evolutionary algorithms have some associated quality metrics like Hyper-volume, Two Set Difference Hyper-volume, Coverage of Two Sets and others [5, 6]. For a maximization problem, Hyper-volume (HV) computes the volume enclosed by the current Pareto front and the orthogonal axes. Its evolution from a generation to another shows if the algorithm makes progress or not, implementing thus a useful convergence indicator. It represents a solution quality indicator, too. For a minimization problem, a hyper-volume reference point must be established. This point will replace the origin in the hyper-volume computation. The hyper-volume reference point is set to the coordinates of the maximum values of the objectives. The values returned by the hyper-volume computation are normalized to the interval [0, 1].

Two Set Difference Hyper-volume (TSDH) is defined as:

TSDH (X′, X″) = HV (X′∨X″) – HV (X″)

where X′, X″ ⊆ X are two sets of decision vectors, HV (X) represents the hyper-volume of the space covered by the decision vector X, and X′∨ X″ represents the non-dominated decision vector obtained after the union (combination) of sets X′ and X″. TSDH computes the normalised hyper-volume of the space that is dominated by X′ but not by X″.

Coverage of Two Sets is used to compare different algorithms or different runs of the same algorithm. It computes the percentage of individuals from a population dominated by individuals from another population. Let X′, X″ ⊆ X be two sets of decision vectors. The function C maps the pair (X′, X″) to the interval [0, 1]:

$$C(X', X'') = \frac{|\{a'' \in X''; \exists a' \in X' : a' \succcurlyeq a''\}|}{|X''|}$$

If all the points from X″ are dominated (or even equal) by the points belonging to X′ then C(X′, X″) = 1. If C(X′, X″) = 0 then none of the points belonging to X″ are dominated by the points from X′. Usually C(X′, X″) ≠ C(X″, X′).

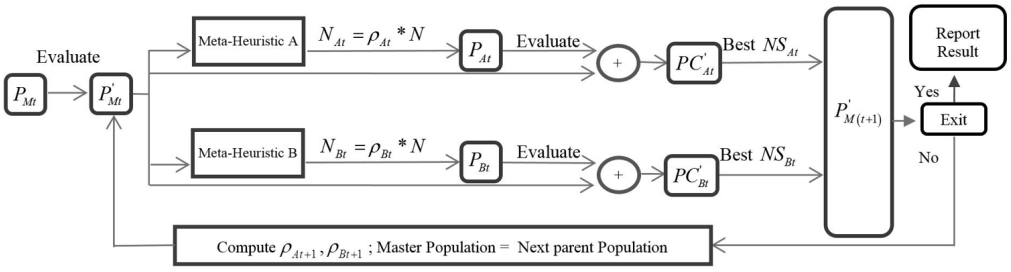

Fig. 2. Meta-Optimization System.

Figure 2 intuitively presents the adaptive meta-optimization approach developed by us in [5]. The current master population noted $P_{M(t)}$ is automatically evaluated, usually through a dedicated simulator. After this (usually time consuming) process, each individual will have associated a certain fitness rate for each of its objectives. This evaluated population called $P'_{M(t)}$ is concurrently processed by each of the two meta-heuristics A and B, acting as optimization algorithms, using their specific genetic operators (mutation and cross-over).

Meta-heuristic A will select from its off-springs, according to its intrinsic selection algorithm, $\rho_{A(t)}N$ individuals. Analogously, meta-heuristic B will generate other $\rho_{B(t)}N$=(1-$\rho_{A(t)}$) N individuals. Both these two new populations of individuals will be automatically evaluated through the dedicated simulator, in order to estimate their objectives' values (fitness rates). After this evaluation process each of these two sets will be combined with the current master population named $P'_{M(t)}$. Through the specific selection mechanisms implemented in NSGA-II (A) and SPEA2 (B) meta-heuristics, the best individuals at a given generation (t) named NS$_{At}$ and NS$_{Bt}$, are generated and combined to form the new master population $P'_{M(t+1)}$. It was considering that each generation contains N=NS$_{At}$+NS$_{Bt}$ distinct individuals (chromosomes). This new master population $P'_{M(t+1)}$ is then evaluated through two already presented quality metrics called Two Set Difference Hyper-volume and Coverage. Considering these two optimization algorithms (A and B), the multi-objective meta-heuristic's global performance (ρ) represents an aggregation between the two normalized quality metrics, called Two Set Difference Hyper-volume ($\widehat{TSDH}$) and Coverage ($\widehat{C}$), like in the following formulas:

$$\rho_{A(t+1)} = \alpha * \widehat{TSDH}_{At} + (1-\alpha) * \widehat{C}_{At},$$
$$\rho_{B(t+1)} = \alpha * \widehat{TSDH}_{Bt} + (1-\alpha) * \widehat{C}_{Bt}, \alpha \in [0,1]$$
$$\widehat{TSDH}_{At} = 1 - \widehat{TSDH}_{Bt} \; and \; \widehat{C}_{At} = 1 - \widehat{C}_{Bt}$$
$$\rightarrow \rho_{A(t+1)} + \rho_{B(t+1)} = 1,$$ where "t" represents the index of the t[th] generation.

Unfortunately, the process of the individuals' selection from the combined population may lead to duplicate individuals. These individuals belong to the previous master population $P_{M(t)}$, and have been selected by both meta-heuristics A and B. An effective iterative solution for this problem was presented in [5]. According to this solution, the best *N* individuals selected can be grouped according to each meta-heuristic's selection algorithm: $NSel_A$ individuals selected only by meta-heuristic A, $NSel_B$ individuals selected only by meta-heuristic B and $NSel_{AB}$ individuals selected by both of them. If we consider NS_A the number of all the individuals

selected by A meta-heuristic at a given generation and NS_B the number of all the individuals selected by B meta-heuristic, then we have the following identities:

$NS_A = NSel_A + NSel_{AB}$
$NS_B = NSel_B + NSel_{AB}$

Totally there were obtained $NSel_A + NSel_B + 2NSel_{AB} = N$ individuals, out of which only $NSel_A + NSel_B + NSel_{AB}$ are unique, while the other $NSel_{AB}$ are duplicates. Thus, it is necessary to further select $NSel_{AB}$ more individuals in order to obtain finally *N* unique solutions, as it is requested. It is necessary to determine meta-heuristic A to select $nNSel_A$ more new individuals and, analogously, meta-heuristic B to select $nNSel_B$ additional individuals. At the end, summing all the individuals selected during the generations, each algorithm will have selected nNS_A, respectively nNS_B, individuals, with:

$nNS_A = NS_A + nNSel_A = NS_A + \rho_A NSel_{AB}$
$nNS_B = NS_B + nNSel_B = NS_B + \rho_B NSel_{AB}$
$nNS_A + nNS_B = N$

Duplicates can once again appear, thus the solution is to apply these selections iteratively, until finally will be selected exactly *N* unique individuals. The final stop condition will be met because the worst case is that $P_{M(t+1)} = P_{M(t)}$.

The obtained quantitative results were encouraging. It has been shown that the proposed meta-optimization method produces better results than the combination of two runs with different meta-heuristics (A and B), in half of the time and with half of the number of total simulations. The "combination" means the superposition of the two obtained Pareto fronts, generated by the two meta-heuristics, involving thus a better Pareto front. More precisely, taking into account that our optimization problem was a minimization one in a bi-objective space (average clocks per cycle and hardware complexity) we obtained that: HV(M-O)>HV(A∨B), where HV means the hyper-volume, A (NSGA-II) and B (SPEA2) represent the two used meta-heuristics, A∨B mean the combination (union, superposition) of populations generated by A and B genetic algorithms, and M-O represent the meta-optimization algorithm. This fulfilled inequality shows that the meta-optimization approach involves a useful efficient synergism. It was also shown that the meta-optimization has 3 times more non-dominated space than the combination of NSGA-II (A) and SPEA2 (B) meta-heuristics. Generalising this meta-optimization method by using M meta-heuristics, the number of individuals will decrease from M*N (the maximum number of individuals necessary to run M meta-heuristics) to just N, and the simulation time from M*T to just T (the time necessary to run a single meta-heuristic). Having more algorithms incorporated in the Meta-Optimization layer could involve better results, too, because the method is easily scalable.

As we already pointed out, Coverage can be used to compare two different algorithms by the fraction of individuals from one algorithm that are non-dominated by individuals from the other algorithm. Figures 3 and 4 are presenting for the first time the Coverage over the 100 generations in a special superscalar microprocessor's optimization problem that was described in [5]. It is clear that the meta-optimization approach is significantly better than each of the NSGA-II (A) and SPEA2 (B) meta-heuristics separately, because more individuals from the meta-optimization are non-dominated by individuals from the other two ones. However it can be observed that there are still more than 20% of the individuals generated by the two base algorithms (A and B) that

are not dominated by individuals from the meta-optimization algorithm. Therefore a base-level meta-heuristic could offer valuable individuals – still not detected by the meta-optimization – belonging to the huge design space.

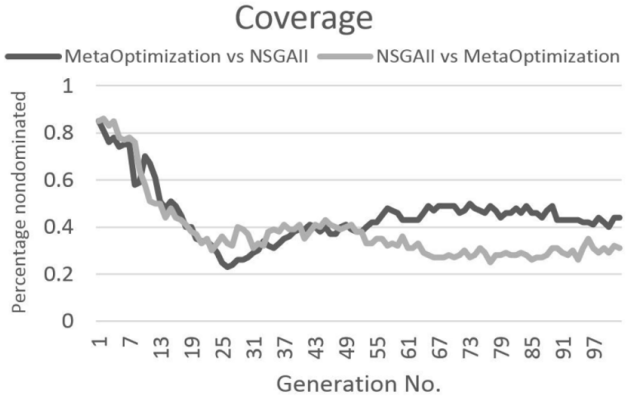

Fig. 3. Coverage (Meta-optimization, NSGA-II).

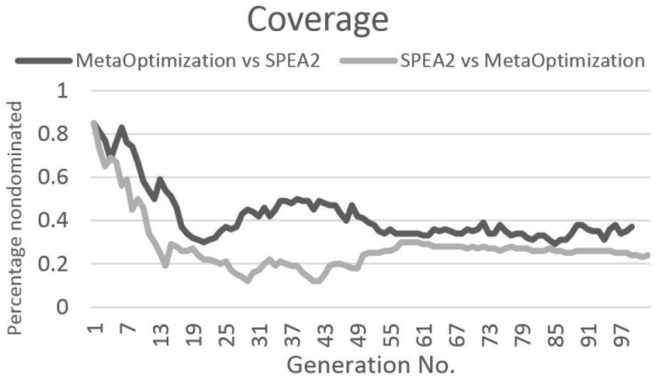

Fig. 4. Coverage (Meta-optimization, SPEA2).

Another meta-optimization approach might be the following. The hyper-heuristic level would implement a certain (mono-objective) genetic algorithm by itself. An individual (chromosome) at this hyper-heuristic level would represent a so called hyper-individual, in fact a certain instance of the genetic algorithm(s) used for master population's optimization. (The master population represents the target individuals that we want to optimise.) This chromosome might contain the following genes (parameters): Selection Operator Type (SOP), Probability of SOP, Mutation Operator Type (MOT), Probability of MOT, Crossover Operator Type (COT), Probability of COT, etc. A hyper-generation would be a generation of hyper-individuals. Each hyper-individual would work during a certain number of generations on a global master population (P_{Mt}) that represents the target individuals of the whole optimization process. Of course, this master population contains chromosomes, too. These chromosomes are codifying the target individuals to be optimised (processor's parameters in our case). Based on the quality indicator value of its

current developed target generation, each hyper-individual would proportionally contribute to the new global master population of target individuals (P_{Mt+1} generation). This global quality indicator - attached to a certain hyper-individual – would represent its fitness rate. Like in all (mono-objective) genetic algorithms, this fitness rate will be essential in developing the new hyper-generation. Unfortunately, due to the many hyper-individuals to be evaluated, such an approach is more complex than the previously described one. Multithreading programming techniques used on multi-core systems would facilitate the implementation of such meta-optimization ideas. At this moment we are trying to implement and evaluate this meta-optimization approach, in the research group led by the author.

3. Meta-Classification for Text Documents

It is very difficult for a certain classifier to accurately classify a large set of text documents. Each particular classifier has its intrinsic advantages and disadvantages. Thus, it would be better to combine into a single meta-classifier several specific classifiers. Adequately selecting these base classifiers, in order to cover the entire problem's space, is a challenging research task. The role of the meta-classifier is to combine the classifiers' results and to predict the correct documents' class. In contrast with the hardware run-time meta-predictors (see paper's Paragraph 4) these software approaches have no hard real time constraints. In [7] we developed a meta-classifier having a multi-layer perceptron structure and based on the back-propagation learning algorithm. This adaptive neural meta-classifier used 8 Support Vector Machine (SVM) classifiers and one Naïve Bayes (NB) classifier in order to achieve the transposition of the input data f rom a large-scale representation space, given by the Vector Space Model documents' representation, into a much smaller size space (with only 16 attributes, corresponding to the 16 distinct classes). Each particular classifier returns a vector, where each scalar value represents the confidence given to the current document to be classified in that certain class. The authors tested and presented 5 distinct configurations for the feed-forward neural network architectures that mainly differ by the number of neurons from the hidden layer. The best obtained results in terms of classification accuracy (about 99%) were obtained using a neural network with 192 neurons in the hidden layer. It was clearly shown that the neural meta-classifier significantly outperforms some simple, non-adaptive meta-classifiers (like voters for example).

Figure 5 presents a possible generic scheme, as it may be in our view, for a (document) meta-classification system. It contains *m* distinct (hybrid) classifiers (SVM, NB, neural, decision tree, etc.). Each classifier will classify the input document in one of the *n* possible output classes. For example, a classifier's output might be a vector containing *n* scalars; each scalar represents the value of the classification confidence corresponding to one of the *n* classes. As an extreme case, a scalar can be a single bit. In this case, if a certain classifier (*i*) classifies the input document as belonging to the class number *k*, the bit $Vi_k = 1$ and all the other bits $Vi_j = 0$, for all j≠k, where $i, j, k \in \{1, 2, \ldots, n\}$. All the *m* classifiers' output vectors are entries in the meta-classification sub-system. The meta-classifier sub-system cascades a non-adaptive meta-classifier (for example a simple voter or a fixed priority selection circuit) and an adaptive one (for example a supervised neural network, as it was already presented.) The non-adaptive meta-classifier aggregates, using a certain algorithm, the *n* vectors (*V1* to *Vn*) into a single one, called *Average Vector* (*AV*). This *AV* vector enters in the adaptive meta-classifier that classifies the document in one of the *n* possible classes (*C1* to *Cn*). Therefore, the final meta-classifier will try to "predict" the right class for the current document. For example, a response of this meta-classifier might contain the

value "1" for the correct class and the value "0" otherwise. One advantage of a supervised meta-classifier consists in the pre-training possibility. Obviously, in this case corresponding updates are necessary. The performance is given in this case not only by the classification accuracy but also by the response time.

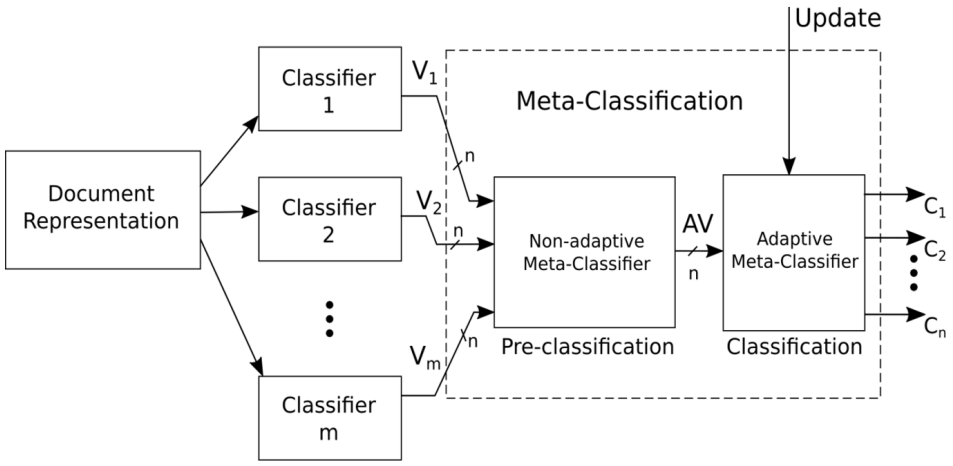

Fig. 5. Document Meta-Classification System.

Using this proposed generic approach we can better understand almost all of the particular implementations. For example in [14] the authors developed classifiers that automatically assign documents to categories, by exploiting features from the corresponding inputs. They used hybrid-classifier that combines state of the art text and image based classifiers (SVM and Adaboost) in order to make optimal joint decisions. Laborious experiments on some real world databases have proven the important benefits of the joint exploitation of these modalities. In [8] it is presented an original efficient method that adaptively combins multiple heterogeneous approaches for document classification, using some identified meta-algorithmic design patterns. These design patterns are exploring the error space in multiple base engines, and provide improved results in comparison to the simple voting schemes and also to the output of any individual base classifiers. Significant quantitative results were reported (for example the developed method reduces document classification error rates by up to 13% comparing with to the single best classifier engine.) In [9] the author developed a two phase meta-classifier. During the first phase there are used four different classifiers to classify the current document. An important difference comparing with the usual meta-classification methods is that reliability indicators allow the meta-classifier to weigh each classifier according to its local reliability in the neighborhood of the current classification process. Thus, based on the reliability indicators functions - that use the words from the document and the outputs of the classifiers from the first processing phase - are generated a set of so-called reliability indicators for the current document. These indicators, together with the document's representation, are then used as the input for the meta-classifier. During this second phase it is done a final document classification using a Decision Tree respectively a SVM (with linear kernel) algorithm.

Meta-classifiers have been widely used in Natural Language Processing, especially for text classification [15]. According to the authors, meta-classification could improve the classification accuracy by combining different classification algorithms. They adequately reviewed some meta-

classification approaches named ensemble techniques, that try to combine different classifiers and aggregate their outputs through several static techniques like voting, weighted combinations, etc. However, as a personal criticism, according to the well-known Arrow's impossibility theorem (see for example https://en.wikipedia.org/wiki/Arrow%27s%_impossibility_theorem), an adequate aggregation is impossible while also meeting a specified set of common-sense criteria. In order to avoid such drawbacks we strongly recommend the design of some adaptive meta-classifiers inspired from machine learning field (voting is a non-adaptive method). According to our experience, it would be really useful for the meta-classifier to learn based on some training data in order to obtain better results during the evaluation phase. The authors are also focusing on some boosting and bagging meta-classification methods that mainly train the same classifier on different parts of the training data to create different classification models. As it is shown, such techniques are succesful in many validation cases. However, suplementary to the authors' criticism related to boosting and bagging meta-classification methods, we add an obvious one: they can't exploit potential useful performance compementarity between different classification algorithms (like Support Vector Machine and Naive Bayes, for example).

In [16] it is presented a quite interesting meta-classifier, called Meta-Consensus (Meta Net). It has two main structural (topological) characteristics. The first one consists in the fact that the weights connecting the basic classifiers' outputs with the meta-classifier outputs depends by the classifier's sensitivity and precision. The second one consists in the fact that "each cell of the confusion matrix of every basic classifier will generate a specific weight connecting all the outputs of the basic classifiers to all the outputs of the Meta Net". Therefore, the second new characteristic is that this meta-classifier additionally considers inhibitory credibility of each basic classifier, too. However, the used datasets are very particular and small, thus it is not clear if the proposed method classifies well other datasets types, like huge documents or images, for example.

In [17] the authors present an automated method for analyzing which meta-classication techniques are most effective for different types of brain-computer interfaces data. They are developing and evaluating five distinct well-known meta-classification algorithms. The authors have shown that their approach scales well with the number of basic classifiers, which is useful. However, it is not sufficiently clear from a qualitative point of view why the proposed meta-classification algorithms are performing better or worse, in the given context. Another open question is if the proposed meta-algorithms could accurately classify other types of objects (*e.g.* text documents), too.

4. Dynamic Meta-Prediction

Prediction methods are frequently used in Computer Science and Computer Engineering research, applications and commercial hardware/software products. For example, almost all present-day pipelined multiple instruction issue microprocessors (superscalar microprocessors) are using advanced run-time (dynamic) branch prediction techniques in order to increase Instruction Level Parallelism (ILP) degrees. Through dynamic branch prediction microprocessors are speculatively processing multiple basic-blocks in parallel and therefore their ability to increase ILP is significantly better. Actually, dynamic branch prediction could be viewed as a classification problem. Each dynamic branch instruction is predicted based on its binary context in two classes: Taken and Not Taken. Dynamic Instruction Value Prediction (VP) is a speculative processing technique that generalizes branch prediction concept. Its main aim is to early predict

the instructions' results, during their fetch or decode pipeline stages, and, therefore, to speculatively execute the instructions flow. The predicted value can then be used as an input to some subsequent dependent instructions, so that they can execute earlier in a speculative manner. If the prediction is incorrect, some recovery mechanisms must be employed to squash speculative results and re-execute all instructions that already used the incorrectly predicted values (selective re-issue). An important challenge for VP techniques is to compress the programs dynamic critical path and therefore to (partially) solve the so-called Issue Bottleneck. Therefore, the VP technique tries to avoid a fundamental limitation in the present-day computing paradigm, the intrinsic program's sequential execution.

A contextual value predictor predicts the next value based on a particular stored binary pattern (context) that is repetitively generated in the values' sequence, in a Markovian stochastic manner. Theoretically, these context predictors can predict any stochastically repetitive values' sequences. A context predictor is of order k if its context information includes the last k values produced by the current instruction. The search is done using this pattern of k values length. A contextual predictor of order k derives from the k-value locality statistical principle that represents an idealised k-context predictor [10]. This principle says that the statistical probability for a certain instruction to produce a value belonging to the N previous value instruction's instances is high and it increases if N is greater. There is not an ideal value predictor or branch predictor. For example, Two Level Adaptive Branch predictors and neural (perceptron) branch predictors are complementary. Digital neural branch predictors could be better than the classical Two Level Adaptive Branch Predictors from the prediction's accuracy point of view but their prediction and learning latencies are usually higher. Perceptron predictors are effective in predicting linear separable branches but they cannot successfully predict branches that are not linear separable, due to their intrinsic limitations (the perceptron's output represents a hyper-plane). In contrast, Two Level Adaptive Branch Predictors could be effective in predicting such difficult branches. In [11] is presented an optimized version of the so-called Scaled Neural Analog Predictor (SNAP) [18], hybridized with two Two-Level Adaptive Branch predictors. SNAP uses electrical current summation for branch prediction's computation instead of high latency digital dot-product computations. The dot product computation is done in this case by summing of electrical currents through Kirchhoff's law. Its main advantages are related to lower power consumption and prediction latency, comparing with a digital implementation. The developed meta-prediction algorithm decides based on the base predictors' confidences. It is clearly shown that the hardware complexity of this effective branch predictor requires less than 64 Kbits that is remarkable and make it feasible to commercial microprocessors' implementations.

In the dynamic value prediction domain, for example contextual (Markovian) value predictors and computational value predictors (for example incremental predictors) are complementary, too. More than this, the best predictor is workload-dependent. Therefore, the idea of hybrid branch/value prediction is natural, meaning that some different (heterogeneous) predictors are working together to exploit the potential synergism. For example, many superscalar microprocessors combined two or even three distinct branch predictors (usually global history / local history predictors and a loop branch predictor). In this case a new challenge is to design an effective meta-predictor that would dynamically select the best predictor at a given moment.

A hybrid instruction value predictor combines two or more component predictors, too. The hybrid predictor needs a selection mechanism, called meta-predictor, to predict which of the predictors is likely to be the most accurate at a certain moment. In [12] we simultaneously used the following component predictors: a last value predictor, a stride predictor and a context-based

predictor. Every of these predictors provide two values: the predicted value and the predictor's confidence. A confidence mechanism performs instructions' speculation control by limiting the predictions only to those that are likely to be correct. Thus, the meta-predictor selects dynamically, based on their last behaviours (confidences), one of the three base predictors (last value, stride or contextual) in order to predict the next value. We designed several different adaptive/non-adaptive meta-prediction structures, in order to properly select the current best predictor. The experimental results obtained using meta-prediction has shown an average prediction accuracy of over 91%, measured on SPEC 2000 benchmarks that were quite remarkable. In [13] we developed a superscalar/Simultaneous Multithreaded microarchitecture that selectively anticipates the values produced by some high-latency instructions like Multiply, Division and Loads with miss in data caches. For doing this, we designed a Dynamic Instruction Reuse scheme for the Multiply / Division instructions and respectively a Last Value Predictor for the long-latency Load instructions. Thus, again we used a hybrid anticipating scheme with an implicit simple meta-prediction algorithm that consists in pre-allocating the two anticipatory schemes to different long-latency instructions' types. In this case the meta-prediction method is not explicitly shown. It was pointed out that this improved microarchitecture architecture achieves significant speedups and reduced energy consumption comparing with a classical one. Other proposed hybrid value prediction schemes were carefully analysed in [10].

Figure 6 presents our proposal for a possible Dynamic Instruction Value Meta-Predictor generic scheme. It generalizes a concrete hybrid branch predictor scheme or a dynamic instruction value predictor scheme, too. During the current instruction's fetch/decode stage, based on the instruction's binary context (Program Counter and other information), each of the*m* branch predictors will generate a Taken/Not Taken prediction. In the case of a Dynamic Instruction Value Predictor each predictor will generate the instructions predicted value (PV). Each predictor has attached its own confidence. The predictor's confidence could be represented as a binary number on k bits showing the predictor's historical behaviour during its last *k* instances (1–correct prediction, 0–incorrect prediction). Based on these *m* confidences' degrees (*Confid. 1* to *Confid. m*) the meta-predictor (non-adaptive or adaptive) will select the most likely predicted value from all of the *m* possibilities (PV1 to PVm). If the meta-predictor is an adaptive supervised one, after the current instruction is finished (Commit phase), it will correspondingly update its state (Update_L2). Also the *m* instruction value predictors (*Pred. 1* to *Pred. m*) will update their states, at the same time. Actually, designing an efficient hardware adaptive meta-predictor for branch prediction or instruction value prediction is an extremely challenging task due to the limited complexity (transistor budget), power consumption restriction and timing restrictions, too. Recent progresses in artificial neural networks hardware implementations, in particular based on different memristor-like devices [19], might accelerate these progresses and would make the commercial implementations feasible.

Meta-prediction algorithms might be very fertile because prediction and pro-active methods are frequently implemented in many Information Technology applications, like: network adaptive routing, image reconstruction, speech recognition (especially Hidden Markov Prediction Models are used here), handwritten character recognition, strategic computer games, gesture understanding, pro-active web browsers, artificial intelligence applications, GPS & Digital Maps applications, next location prediction in ubiquitous systems, mobile phone communications, etc.

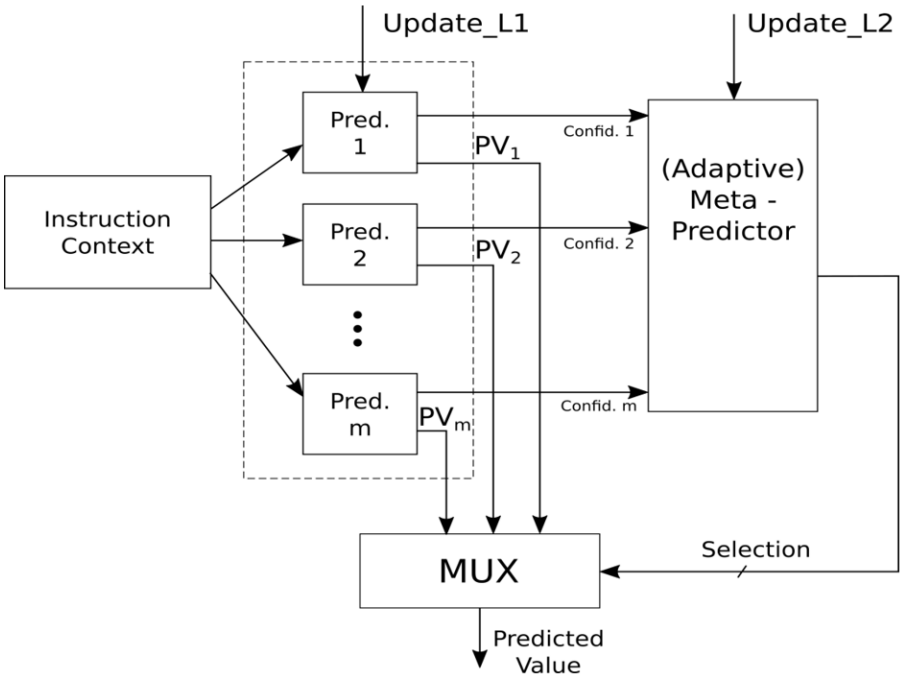

Fig. 6. Document Meta-Classification System.

5. Conclusion

Solving nowadays complex problems needs appropriate research methodologies. State of the art algorithms are very mature and refined for almost all the Computing Systems research challenges. Therefore a meta-algorithmic approach, implementing a new hierarchical abstraction level, might be very useful and effective. Based on some of our concrete scientific experience we presented some approaches that pointed out and confirm this general methodological view. First, we analysed an adaptive refined meta-optimization method proposed in order to find the best (Pareto) individuals in a bi-objective space. It was detailed the meta-algorithm that orchestrates two state of the art multi-objective optimization evolutionary algorithms (NSGA-II and SPEA2). Some qualitative and quantitative results were presented and explained. Also other interesting meta-optimization approaches were synthetically described and analyzed. An alternative possible meta-optimization algorithm, based on a hyper-heuristic algorithm that is managing the meta-heuristics optimization algorithms, was suggested, too.

After this we had shown that meta-algorithms were fertile and efficient in documents' and patterns' classification domain, too. Some hybrid classification approaches were analyzed. A generic adaptive meta-classification scheme was developed. Finally, it was shown that meta-algorithms are very effective in developing meta-predictors with applications in dynamic branch prediction and instruction value prediction fields. In all these validation cases it was clearly shown that the discussed meta-algorithmic approaches involve a very useful synergism that brings an important added performance value.

Meta-algorithmic approach might be an almost ubiquitous solution in order to improve present-

day computing systems multi-criterial performance, due to their exponential hardware-software complexity growth. Designing and implementing a new hierarchic meta-algorithmic layer would allow a synergic exploitation of the lowest level state of the art complementary algorithms, breaking their limitations. Today, it is difficult to create a new effective alternative basic algorithm related to an open research problem. But a meta-algorithmic approach could easily bring a performance gain.

Acknowledgements. I would like to thank the anonymous reviewers for their detailed, useful recommendations.

References

[1] M. STEPHENSON, S. AMARASINGHE, M. MARTIN, U.-M. O'REILLY, *Meta Optimization: Improving Compiler Heuristics with Machine Learning*, Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '03), San Diego, California, USA. ACM, 2003.

[2] P. DAS, A. BANERJEE, *Meta Optimization and its Application to Portfolio Selection*, Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining, ACM, NY USA, 2011.

[3] P. KRUS, J. ÖLVANDER, *Performance index and meta-optimization of a direct search optimization method*, Engineering Optimization, **5**(6), pp. 345–357, 2012.

[4] C. NEUMÜLLER, A. SCHEIBENPFLUG, S. WAGNER, A. BEHAM, M. AFFENZELLER, *Large Scale Parameter Meta-Optimization of Metaheuristic Optimization Algorithms with HeuristicLab Hive*, Proceedings of the 8-th Spanish Congress on Metaheuristics, Evolutionary and Bioinspired Algorithms (MAEB'2012), Albacete, Spain, 2012.

[5] L. VINTAN, R. CHIS, Md. A. ISMAIL, C. COTOFANA, *Improving Computing Systems Automatic Multi-Objective Optimization through Meta-Optimization*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **35**(7), pp. 1125–1129, 2016.

[6] R. CHIS, M. VINTAN, L. VINTAN, *Multi-objective DSE Algorithms' Evaluations on Processor Optimization*, Proceedings of 9th International Conference on Intelligent Computer Communication and Processing, pp. 2–34, IEEE Computer Society Press, Cluj-Napoca, 2013.

[7] R. CRETULESCU, D. MORARIU, L. VINTAN, I. D. COMAN, *An Adaptive Meta-classifier for Text Documents*, The 16th International Conference on Information Systems Analysis and Synthesis (ISAS 2010), **2**, pp. 372–377, Orlando Florida, USA, 2010.

[8] S. J. SIMSKE *et al.*, *Meta-Algorithmic Systems for Document Classification*, ACM Symposium on Document Engineering (DocEng'06), Amsterdam, The Netherlands, 2006.

[9] P. BENNETT, *Building Reliable Metaclassifiers for Text Learning*, PhD Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, available at http://reports-archive.adm.cs.cmu.edu/anon/2006/CMU-CS-06-121.pdf, 2006.

[10] L. VINŢAN, *Prediction Techniques in Advanced Computing Architectures*, Matrix Rom Publishing House, Bucharest, 2007.

[11] D. A. JIMNEZ, *OH-SNAP: Optimized Hybrid Scaled Neural Analog Predictor*, Proceedings of the 3-rd Championship on Branch Prediction, San Jose, CA, USA, 2011.

[12] L. VINTAN L., A. GELLERT, A. FLOREA, *Register Value Prediction using Metapredictors*, Proceedings of the 8-th International Symposium on Automation Control and Computer Science (SACCS 2004), Iasi, Romania, 2004.

[13] A. GELLERT, A. FLOREA, L. VINTAN, *Exploiting Selective Instruction Reuse and Value Prediction in a Superscalar Architecture*, Journal of Systems Architecture, **55**(3), pp. 188–195, Elsevier, 2009.

[14] S. D. CHEN, V. MONGA, P. MOULIN, *Meta-classifiers for Multimodal Document Classification*, 11-th IEEE International Workshop on Multimedia Signal Processing, MMSP'09, pp. 1–6, Rio de Janeiro, Brazil, 2009.

[15] C. C. AGGARWAL, C. X. ZHAI, *A survey of text classification algorithms*, Mining Text Data (Vol. 9781461432234, pp. 163–222), 2012.

[16] M. BUSCEMA, W. TASTLE, *A new meta-classifier*, Annual Meeting of the North American Fuzzy Information Processing Society, IEEE, Toronto, Canada, 2010.

[17] P. HAMMON, V. DE SA, *Preprocessing and Meta-Classication for Brain-Computer Interfaces*, IEEE Transactions on Bio-Medical Engineering, **54**(3), pp. 518–525, 2007.

[18] R. St. AMANT, D. A. JIMENEZ, D. BURGER, *Mixed-signal approximate computation: A neural predictor case study*, IEEE Micro – Top Picks from Computer Architecture Conferences, 29(1):104115, 2009.

[19] E. ROSENTHAL *et al.*, *A fully analog memristor-based neural network with online gradient training*, IEEE International Symposium on Circuits and Systems, Montreal, QC, Canada, 2016.