Procedia Computer Science

Volume 51, 2015, Pages 2563–2572

ICCS 2015 International Conference On Computational Science

# Multiobjective Design Optimization in the Lightweight Dataflow for DDDAS Environment (LiD4E) *

Kishan Sudusinghe[1], Yang Jiao[1], Haifa Ben Salem[1], Mihaela van der Schaar[2], and Shuvra S. Bhattacharyya[1]

[1] University of Maryland College Park, College Park, Maryland, U.S.A
`kishans@umd.edu, yjiao1@umd.edu, hbensale@umd.edu, ssb@umd.edu`
[2] University of California Los Angeles, Los Angeles, California, U.S.A.
`mihaela@ee.ucla.edu`

**Abstract**

In this paper, we introduce new methods for multiobjective, system-level optimization that have been incorporated into the Lightweight Dataflow for Dynamic Data Driven Application Systems (DDDAS) Environment (LiD4E). LiD4E is a design tool for optimized implementation of dynamic, data-driven stream mining systems using high-level dataflow models of computation. More specifically, we develop in this paper new methods for integrated modeling and optimization of real-time stream mining constraints, multidimensional stream mining performance (precision and recall), and energy efficiency. Using a design methodology centered on data-driven control of and coordination between alternative dataflow subsystems for stream mining (classification modes), we develop systematic methods for exploring complex, multidimensional design spaces associated with dynamic stream mining systems, and deriving sets of Pareto-optimal system configurations that can be switched among based on data characteristics and operating constraints.

*Keywords:* Dataflow, DDDAS, model-based design, stream mining, machine learning.

## 1 Introduction

The proliferation of sensing devices and cost- and energy-efficient embedded processors has contributed to the increasing interest in adaptive stream mining (ASM) systems. In this class of application systems, streams of data are analyzed in real-time from various data sources for diverse purposes, such as environmental monitoring, surveillance, structural health management, and cyber-security [8]. For cost- and energy-efficient operation, adaptive stream mining must be performed in a data-driven manner, so that the applied classifiers do not excessively over- or under-perform with respect to current data characteristics and operational constraints [15].

*This research is sponsored by the US Air Force Office of Scientific Research under the DDDAS Program.

doi:10.1016/j.procs.2015.05.364

A challenging aspect of ASM systems is the diverse sets of operational constraints and objectives under which they must be deployed. The specific constraints that take precedence may depend strongly on the operational scenario and associated data. For example, in the midst of a security breach or intrusion, conserving energy is less important while delay is key; on the other hand, if there is no detected threat, conserving energy may be critical. Furthermore, such multidimensional constraints and objectives are often in conflict with one another so that trade-offs must be carefully guided and rigorously optimized to achieve results that yield acceptable levels of reliability and quality of service. For example, classification accuracy (the rate of correct classifications), false positive rates in classification, processing latency, processing throughput, and energy consumption per classification operation are metrics that may all be relevant to some degree in a particular stream mining deployment. Conventional approaches to design and implementation of ASM systems often focus on small subsets of relevant metrics in isolation (e.g., the trade-off between accuracy and false positive rate) or orient the implementation process toward a particular subspace (e.g., throughput-constrained accuracy maximization).

Motivated by these complex, multidimensional, data-dependent design spaces in ASM systems, we develop in this paper methods for integrated modeling and multiobjective design optimization of real-time stream mining systems. Our proposed design framework is readily adaptable to different kinds of operational constraints and objectives. For concreteness, we develop our methods in the context of real-time performance, multidimensional stream mining performance (precision and recall), and energy efficiency. These metrics are discussed in detail in Section 4. Using a design methodology centered on data-driven control of and coordination between alternative dataflow subsystems for stream mining (classification modes), we develop systematic methods for exploring complex, multidimensional design spaces associated with dynamic stream mining systems, and deriving sets of Pareto-optimal system configurations that can be switched among based on data characteristics and operating constraints.

We demonstrate and experiment with our methods for data-driven, multiobjective optimization through their integration in the *Lightweight Dataflow for Dynamic Data-Driven Application Systems Environment* (LiD4E), which is a software tool for experimentation with and optimization of dataflow-based design methods for ASM systems [16]. Using LiD4E together with our new methods for multiobjective optimization, we experiment with a multiclass vehicle classification system that categorizes vehicles among three distinct classes — cars, buses and vans — from images. Through experiments on this vehicle classification application, we demonstrate the effectiveness of our methods in deriving Pareto-optimal design options and quantifying complex implementation trade-offs. These capabilities can provide significant insight to the system designer to identify the set of design configurations that best matches the targeted set of application scenarios and their associated system requirements.

The remainder of this paper is organized as follows. In Section 2, we discuss related work in DDDAS methods, real-time stream mining, and dataflow-based design methodologies for signal processing systems. Section 3 introduces our proposed new multiobjective design optimization framework, and Section 4 presents a vehicle classification case study to demonstrate the framework. In Section 5, we present experimental results from this case study. Finally, Section 6 provides conclusions and future research directions.

## 2 Related Work

The work presented in this paper is rooted in core concepts of the DDDAS paradigm [5]; real-time stream mining [8]; and dataflow-based design methodologies for signal processing systems (e.g., see [11, 3]). In this form of dataflow modeling, applications are represented in terms of *dataflow graphs*, where graph vertices (*actors*) represent signal processing tasks of arbitrary complexity, and edges represent logical FIFO communication channels between pairs of actors. In this paper, we apply dataflow as a programming model with semantics that are matched to the domain of adaptive stream mining systems [16, 15]. This modeling approach differs from uses of dataflow as a compiler intermediate representation (e.g., see [12]), and as a form of computer architecture [7].

The work presented in this paper builds upon our previous work on ASMs for multimedia applications [16], and extends the dynamic, multi-mode stream mining framework presented in [15] with powerful capabilities for multiobjective design space exploration and optimization. More specifically, contributions introduced in this paper include new techniques for modeling, control and optimization of multiobjective design spaces in ASM implementation; extension of the multi-mode design framework of [15] to multiclass recognition systems (i.e., to classifiers that map to two or more different output classes); and application to a multiclass vehicle detection problem that is relevant to surveillance and traffic monitoring.

Design and implementation techniques for stream mining systems have been studied before in a statically configured environment, and with relatively fine granularity (low level) optimizations on application performance (e.g., see [14, 13, 10]). Here, by "statically-configured," we mean that the processing methods are not adapted dynamically in response to data characteristics or operational context. Our work in this paper deviates from this body of prior work in that our focus is on a dynamic, data-driven implementation context, and also, we focus on coarser granularity optimizations — in particular, optimizations for configuring and coordinating across different stream mining classification subsystems and application modes.

Incorporation of data-driven operation in individual signal processing functional components has been studied in [4, 1]. This related work has been developed in the context of speech recognition. Although this work, relates to the dynamic, data-driven theme of our contribution in this paper, the approach that we develop in this paper is more flexible in terms of data-driven operation since we consider adaptation of application modes globally (at the dataflow graph and scheduling levels) as well as locally (at the level of individual actors or subsystems). In contrast, this body of related work on data-driven speech processing focuses on local optimizations. However, techniques derived from works such as [4, 1] can provide useful building blocks (parameterized actor and subsystem designs) for the DDDAS design framework developed in this paper. Integration of such building blocks into our proposed framework is a useful direction for future work.

We emphasize that the objective of this paper is not to introduce new types of classification techniques nor to endorse a particular type of classifier, but rather to provide a systematic framework for optimized configuration, control, and coordination across arbitrary sets of complementary classifiers (i.e., classifiers with complementary profiles of operational trade-offs). In our implementations and experiments, we utilize Support Vector Machine (SVM) classifiers, although our design framework is readily adaptable to the use of other types of classifiers. Use of SVM classifiers for low-sample data sets, and as efficient, robust components for general classification purposes has been motivated extensively in the literature (e.g., see [9, 17]).

## 3 Design Methodology

In this section, we introduce the system model that we employ in our new multiobjective design optimization framework, which we refer to as the *ASM multiobjective design optimization framework*, abbreviated as AMDO. AMDO is built upon the DDDAS-HCFDF-Multi-Mode (DHMM) scheduling framework introduced in [15]. Here, HCFDF stands for hierarchical core functional dataflow [16], which is the underlying model of computation for the DHMM framework.

## 3.1 Background on DHMM

We first review in this section key aspects of the DHMM system model that are inherited by AMDO. The developments in this paper build on the DHMM model, and incorporate flexible and powerful new capabilities for multiobjective optimization and design space exploration. In DHMM, an ASM system design is represented as a set of mutually exclusive *application modes* $S_M = \{\mu_1, \mu_2, \ldots, \mu_N\}$. Here, each $\mu_i$ represents a set of application subsystems that are active during the corresponding mode together with the configurations, such as actor-, application- and schedule-level parameters, that are to be applied to the subsystems whenever $\mu_i$ executes. Each design is also characterized by a set of measurements, corresponding to the associated DDDAS-based instrumentation subsystem, $M = m_1, m_2, \ldots, m_k$. These measurements can be made from arbitrary sources, including the system input, target platform, system output or operating environment. Each $m_i$ corresponds to a distinct metric, such as power consumption, remaining battery capacity, or selected frequency content profiles for some kind of sensor data.

A key aspect of the DHMM model is a state machine $S_{DHMM}$ in which states correspond to application modes, and transitions correspond to changes made by the executing system to the current mode in response to input data that is monitored by $S_{DHMM}$. This input data comes from the measurements $m_i$, which are performed iteratively according to periodic processes or other kinds of timing patterns (e.g., dependent on the current mode).

In DHMM, the functionality of specific application modes is represented using the hierarchical core functional dataflow (HCFDF) model of computation [16], while $S_{DHMM}$ is employed for dynamic and adaptive model-based coordination and parameter control across different modes. In HCFDF-based dataflow graph specifications, software components (actors) are specified in terms of sets of processing *modes*, where each mode has static *dataflow rates* — i.e., each mode produces and consumes a fixed number of data values (tokens) on each actor port. However, different modes of the same actor can have different dataflow rates, and the actor mode can change from one actor execution (firing) to the next, thereby allowing for dynamic dataflow behavior (dynamic rates). Additionally, HCFDF allows dataflow graphs to be hierarchically embedded within actors of higher level HCFDF graphs, thereby allowing complex systems to be constructed and analyzed in a scalable manner. For further details on the HCFDF model of computation, we refer the reader to [16].

## 3.2 AMDO Design Methodology

The AMDO design methodology incorporates the instrumentation subsystem $M$ and mode-transition state machine $S_{DHMM}$ of DHMM. The methodology additionally incorporates a parameterization $P$ of $S_{DHMM}$ for use in exploring the design space associated with implementations that are controlled by $S_{DHMM}$ in conjunction with the underlying application modes. More specifically, $P = (p_1, p_2, \ldots, p_K)$, called the *design space parameter set* (*DSPS*) of the AMDO model, is a sequence of parameters of $S_{DHMM}$, where each $p_i$ has an associated domain *domain*$(i)$, which gives the set of admissible parameter settings (*configurations*) for $p_i$ during execution of $S_{DHMM}$. For clarity and conciseness, we assume in the remainder of this paper that the *domain*$(i) \subset \mathbb{R}$ for all $i$, where $\mathbb{R}$ denotes the set of real numbers.

When applying the AMDO methodology, the parameterization $P$ of $S_{DHMM}$ is central to the processes of design space exploration and multiobjective optimization. Different parameter configurations of $S_{DHMM}$ in general lead to different ways in which data-driven adaptation is controlled, and in which the multidimensional design evaluation metrics, such as energy consumption, real-time performance, and stream mining accuracy, are traded-off throughout the execution process. Additionally, the high level dataflow model of the targeted ASM application together with the FSM-driven application governed by $S_{DHMM}$ provides a model-based representation that can be employed for efficient simulation so that a wide variety of alternative parameter configurations and associated design points can be evaluated.

Two other aspects in the operation of an AMDO-based stream mining implementation are periodic performance assessment (PPA), and performance assessment actors (PAAs). In each state of $S_{DHMM}$, the recent performance of the system is assessed in terms of the set of relevant metrics $M$. This PPA process helps to determine whether $S_{DHMM}$ should remain in its current state or whether a transition should be made to a different state. The operation of the PAAs may in general depend on the values of parameters in $P$. The determination of whether or not a transition is made and which new state should be the target of each PPA-related transition is made by $S_{DHMM}$ with input from the PAAs. Each PAA $A$ is a software component (dataflow actor) that takes as input a selected subset of data obtained from the DDDAS instrumentation subsystem during a window of recent operation (e.g., during the last 10ms or last 100 processed data packets).

On each execution of $A$, the output of $A$ is a member of the set $s_{PAA} = \{\gamma_o, \gamma_i, \gamma_u\}$, where $\gamma_o$ represents an indication by $A$ that the system is currently overperforming with respect to the form of performance assessment carried out by $A$. Similarly, $\gamma_u$ represents and indication by $A$ that the system is underperforming, and $\gamma_i$ indicates that the performance of the system is within an intermediate range — neither too high (at potential expense of other objectives) nor too low . Intuitively, a PAA can be viewed as a standard interface for capturing data-dependent characteristics of system operation, and relating them dynamically to a compact set of values ($\gamma_o$, $\gamma_i$, and $\gamma_u$). The values generated by the different PAAs can then be processed in an integrated way by $S_{DHMM}$ to control overall system operation.

For example, an AMDO system could be designed with three PAAs $A_1, A_2, A_3$ that correspond, respectively, to performance assessment for speech processing quality (accuracy), energy consumption, and processing speed. During each PPA, these PAAs would each provide an input to the controller for $S_{DHMM}$ indicating the "health" of the system's recent performance with respect to the corresponding assessment considerations. Logic within the controller would then process these inputs to determine whether or not to remain in the current state, and what state to transition to if a transition is to be made. For example, if the system is found to be underperforming in terms of energy consumption (i.e., consuming excessive amounts of energy), this may favor a transition to a more energy-efficient application mode. Similarly, overperforming with respect to speed may lead to transition to a processing mode that is slower and more favorable in terms of other objectives, such as energy consumption or quality.

As with the state machine parameterization $P$, the design of the PAAs, and the associated controller logic for processing the PAA outputs are design issues of the given AMDO. The objective of the AMDO design methodology is thus to raise the level of abstraction for stream mining system implementation in a structured manner so that the system designer can focus on a standard, well-defined set of DDDAS-based system components — $S_{DHMM}$, $P$, the PAA set — that interact in a systematic manner. Thus, we represent an AMDO system $\alpha$ by a tuple $\alpha = (S_{DHMM}, P, T)$, where the elements of this tuple respectively specify the state machine, parameterization, and PAA set associated with $\alpha$.

Using an AMDO system $\alpha = (S_{DHMM}, P, T)$, the designer can evaluate multidimensional system performance for a variety of parameter settings within $P$ to generate alternative design points, while each parameter setting influences system operation (through $S_{DHMM}$ and

$T$) to trade off different performance objectives in a specific way. In Section 4 and Section 5, we demonstrate the application of the AMDO design methodology on a practical surveillance application case study involving vehicle detection. This case study helps to make the developments in this section more concrete, and to demonstrate the utility of the AMDO methodology as a framework for multiobjective design space exploration and optimization of ASM systems.

## 4 Case Study: Vehicle Classification

To validate and demonstrate the AMDO framework, we have developed a multiobjective optimization case study of a data-driven ASM application that is relevant to surveillance systems. Specifically, our case study involves a vehicle classification system in which images of detected vehicles are analyzed to classify each vehicle as either a bus, car or van. The classification system is assumed to be a mobile system that is capable of being deployed with agility and low cost in operational environments. This mobile deployment feature makes energy efficiency an important metric to consider in the design evaluation space for the system.

We have performed extensive simulations to evaluate a complex, five-dimensional design evaluation space (i.e., a space of trade-offs involving selected implementation metrics) that is based on several relevant, and often competing deployment objectives. Specifically, the design evaluation space considered encompasses the metrics of throughput (data rate), deadline miss rate (real-time performance), energy efficiency, precision for detecting cars (one objective related to classification accuracy), and recall for detecting cars (another accuracy-related objective). Thus, a main goal of the case study is to expose Pareto points in a complex multidimensional space of designs for deploying the vehicle classification application on a targeted mobile device.

Here the throughput, in terms of images per second, gives the rate at which the system can process images. If $T$ denotes the throughput, then the reciprocal $(1/T)$ specifies the *deadline*, which is in units of seconds per image, and gives the maximum time allowed to process a single image. Whenever the AMDO system fails to process an image within its associated deadline period, a deadline miss occurs. For a given stream mining execution consisting of an input stream that contains $I$ images, the deadline miss rate $r$ is computed as $(N_{miss}/I)$, where $N_{miss}$ is the total number of deadline misses encountered throughout the execution.

Figure 1 provides an illustration of the state machine $S_{DHMM}$ for our AMDO-based vehicle classification system. The state machine includes three application modes, labeled $Z_{M,1}, Z_{M,2}, Z_{M,3}$, which represent one-against-one (1A1) support vector machine (SVM) classifier subsystems with varied parameter configurations. For background on this type of classifier, we refer the reader to [9]. The classifiers are configured with Gaussian radial basis function kernels that have different combinations of sigma and box constraint values. These three alternative application modes yield different operational trade-offs in terms of execution time, energy consumption, precision, and recall. The AMDO framework provides a systematic way to exploit such variety in application modes to derive diverse sets of alternative design points (Pareto designs) during multiobjective optimization. The states in Figure 1 with labels of the form $Z_{P,i}$ correspond to PPA points. Each of these states encapsulates a single PAA, and is entered periodically from its associated application mode. The transitions in the state machine are executed either from periodic interrupts that trigger PPAs or from decisions that are computed from the relevant PAAs. Further details on the state machine operation are omitted due to space limitations.

The state labeled $Z_{M,E}$ in Figure 1 is a special state that is dedicated to providing graceful shutdown of the system once the battery capacity $c$ has fallen to a value that is less than or equal to a pre-defined threshold $\epsilon$. In our experiments (see Section 5), we employed $\epsilon = 5\%$.

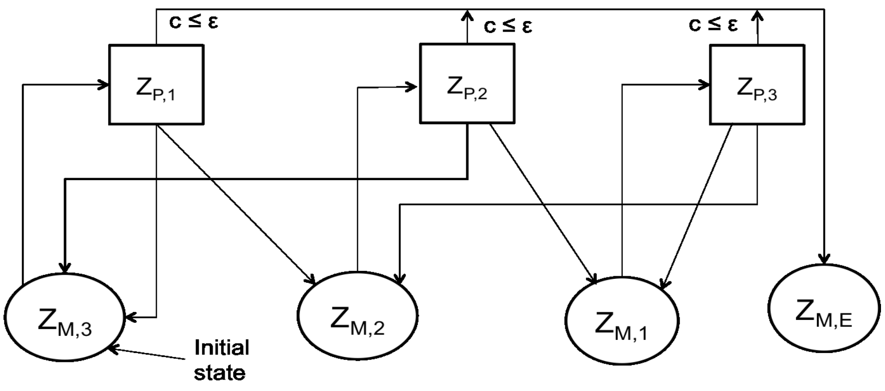

Figure 1: An illustration of $S_{DHMM}$ for the experimental vehicle classification system.

Since our targeted application is a multiclass classification application (a classification application that involves more than two classes), we employ *precision* and *recall* as metrics for assessing classification accuracy. These metrics are commonly used for multiclass classification systems. We arbitrarily choose cars as the relevant vehicle class for the precision and recall calculations. Thus, the precision is calculated as $TP/(TP + FP)$ and the recall is calculated as $TP/(TP + FN)$, where $TP$, $FP$, and $FN$ denote, respectively, the numbers of true positives, false positives, and false negatives as related to detection of cars (the selected relevant class).

## 5 Experiments

In this section, we present experimental results derived from applying the AMDO framework on the vehicle classification application introduced in Section 4.

### 5.1 FSM Parameterization

Recall from Section 3 that an AMDO system can be expressed by a tuple $(S_{DHMM}, P, T)$, where the elements of this tuple specify the state machine, parameterization, and PAA set for the system. In our experimental vehicle classification system, the employed $S_{DHMM}$ is illustrated in Figure 1. The PAA set consists of 3 actors, which provide performance assessment in terms of deadline miss rate, execution speed, and remaining battery capacity.

The FSM parameterization $P$ that we employed in our experiments can be expressed as $P = (p_1, p_2, p_3, p_4, p_5)$. Here, $p_1$ represents the deadline for processing each image (i.e., the reciprocal of the supported image processing throughput); $p_2$ represents the *deadline miss tolerance*, which specifies what percentage of deadlines can be tolerated before the system is considered to be underperforming in terms of real-time operation; $p_3$ represents an analogous tolerance on execution-time overperformance — the system is considered to be overperforming in terms of execution time if the average execution time of an application mode is less than the product $(p_1 \times p_3)$; $p_4$ specifies what percentage of system battery capacity must be exceeded for the system to be overperforming in terms of energy availability; and similarly, $p_5$ specifies a minimum threshold (percentage) on battery capacity below which the system is considered to be underperforming in terms of battery capacity. Collectively, the five parameters in the vector $P$ defined above control how the PAAs in $S_{DHMM}$ cooperate, in a deeply data-driven manner, to explore different regions of the overall design evaluation space — as these parameters are varied, different design trade-offs are concretely realized.

This particular parameterization $P$ is one specific parameterization that we experimented with to concretely demonstrate the AMDO framework; other parameterizations can be derived to drive data-driven, multiobjective optimization in different ways. A central contribution of the AMDO framework is to structure and raise the level of abstraction in data-driven multiobjective optimization by introducing this kind of parameterization as a first class citizen in the design process for ASM systems. This is an advance over conventional methods for ASM system implementation, which focus on ad-hoc fine-tuning of control code, on analysis of static (non-data-driven) design configurations, or on individual design metrics in isolation.

## 5.2 Experimental Setup

We have implemented a simulation model for the vehicle classification system on a desktop computer using the model-based design approach underlying AMDO. The developed simulation environment provides validation of the vehicle classification functionality, along with multidimensional performance assessment of system operation. Using the LiD4E environment described in Section 1, we have also implemented the classifier subsystems (application modes) employed for vehicle classification on a mobile platform (Android-based, Nexus 7, first-generation tablet). We performed extensive profiling of the performance of these mobile-device-targeted subsystem implementations. Data from this mobile-device-based profiling, including execution time and energy consumption data, was employed to provide characterizations of classifier operation that were applied in the simulation model.

We used 561 vehicle silhouettes from the Statlog dataset [2] for training and 281 images for experimentation. The image sets for training and testing were chosen randomly. We made a minor modification to the annotations in the Statlog dataset by combining the two distinct class labels for cars, "Saab" and "Opel", into a single class labeled "cars". Hence, as described in Section 1, our modified dataset consists of three class labels in total — buses, cars, and vans.

## 5.3 Experimental Results

Using the AMDO system design and experimental setup described in Section 5.1 and Section 5.2, we simulated 26 different design points corresponding to 26 different configurations of the FSM parameter SET $P$. The alternative combinations of parameter settings were selected manually with a view towards experimenting with diverse combinations of parameter settings. Alternatively, one could generate and simulate parameter settings using an automated approach, such as an approach that employs a multiobjective evolutionary algorithm (e.g., see [18]) to maintain populations of parameter settings, and employs our AMDO simulation framework for fitness evaluation. Such automated design space exploration using the AMDO framework is a useful direction for future work.

As discussed in Section 4, the design evaluation metrics considered in our experiments are throughput; deadline miss rate; energy efficiency; and both precision and recall for detecting cars. Here, energy efficiency is measured as the number of images that were processed (excluding deadline misses) for a given amount of initial battery capacity. The amount of initial battery capacity employed in the experiments was 432.5 milliampere-hours (mAh). The metric employed for energy efficiency thus gives an indication of the total volume of data that can be processed before the given amount of battery capacity expires.

Table 1 lists the set of Pareto-optimal designs from among the set $Y$ of 26 design points that we generated in our experiments. Here, we say that a point $y \in Y$ is Pareto-optimal if for any other point $y\prime \in Y$, $y\prime$ is inferior to $y$ in terms of at least one design evaluation metric. For general background on Pareto optimization in the context of electronic system design, we refer the reader to [6]. Intuitively, a Pareto point represents a useful design point to keep track of during design space exploration because such a design point cannot be improved upon in any dimension without sacrificing quality in at least one other dimension. Among the 26 design points explored in our experiments, 16 (61%) were found to be Pareto-optimal. These 16 points are the ones that are listed in Table 1 along with their simulated performance results in terms of the five targeted design evaluation metrics.

In summary, the experiments and results presented in this section demonstrate concretely how AMDO enables designers to rapidly investigate diverse sets of alternative design points for an ASM system (1) relative to a complex multidimensional design evaluation space, and (2) in a manner that systematically takes into account data-driven adaptation of application modes and system implementation parameters within a unified framework.

| Design ID | Energy Efficiency (images processed) | Deadline Miss Rate (%) | Throughput (images per second) | Average Precision for Cars (%) | Average Recall for Cars (%) |
|---|---|---|---|---|---|
| AMDO-1 | 861237 | 0.13 | 83.33 | 99.15 | 95.42 |
| AMDO-2 | 847853 | 0.36 | 90.91 | 99.15 | 95.49 |
| AMDO-3 | 679407 | 2.01 | 90.91 | 98.27 | 97.34 |
| AMDO-4 | 656775 | 0.36 | 66.67 | 97.90 | 97.90 |
| AMDO-5 | 861438 | 0.07 | 66.67 | 99.15 | 95.42 |
| AMDO-6 | 651649 | 4.20 | 100.00 | 98.09 | 97.61 |
| AMDO-7 | 656566 | 0.35 | 62.50 | 97.90 | 97.90 |
| AMDO-8 | 821935 | 0.06 | 62.50 | 99.02 | 96.76 |
| AMDO-9 | 861654 | 0.03 | 62.50 | 99.15 | 95.42 |
| AMDO-10 | 861312 | 0.10 | 76.92 | 99.15 | 95.42 |
| AMDO-11 | 861162 | 0.13 | 83.33 | 95.42 | 99.15 |
| AMDO-12 | 653875 | 1.06 | 83.33 | 97.90 | 97.90 |
| AMDO-13 | 849897 | 0.03 | 62.50 | 99.15 | 95.50 |
| AMDO-14 | 723423 | 18.13 | 135.14 | 99.27 | 95.17 |
| AMDO-15 | 18470 | 98.09 | 169.49 | 99.27 | 95.10 |
| AMDO-16 | 155060 | 83.75 | 166.67 | 99.27 | 95.10 |

Table 1: Pareto-optimal design points derived through design space exploration.

## 6 Conclusions

In this paper, we have introduced a new multiobjective design optimization framework for adaptive stream stream mining systems (ASMs). The framework, called the ASM multiobjective design optimization (AMDO) framework, employs a novel design methodology centered on data-driven control of and coordination between alternative dataflow subsystems for stream mining. AMDO allows system designers to efficiently explore complex, multidimensional design evaluation spaces in a data-driven manner, and is readily adaptable to different kinds of operational constraints and objectives. We have integrated AMDO into the Lightweight Dataflow for DDDAS Environment (LiD4E) tool for design and implementation ASM systems, and demonstrated the framework using a case study involving real-time and energy-constrained multiclass vehicle classification. Useful directions for future work include development of automated design space exploration methods using the AMDO framework, such as integration of AMDO methods with multiobjective evolutionary algorithms.

## References

[1] G. Aradilla, J. Vepa, and H. Bourlard. Improving speech recognition using a data-driven approach. Technical Report IDIAP-RR 05-66, IDIAP Research Institute, April 2005.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] S. S. Bhattacharyya, E. Deprettere, R. Leupers, and J. Takala, editors. *Handbook of Signal Processing Systems*. Springer, second edition, 2013. ISBN: 978-1-4614-6858-5 (Print); 978-1-4614-6859-2 (Online).

[4] G. Chollet, K. McTait, and D. Petrovska-Delacrétaz. Data driven approaches to speech and language processing. In *Nonlinear Speech Modeling and Applications*, pages 164–198. Springer, 2005.

[5] F. Darema. Grid computing and beyond: The context of dynamic data driven applications systems. *Proceedings of the IEEE*, 93(2):692–697, 2005.

[6] G. De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

[7] J. B. Dennis. Dataflow supercomputers. *Computer*, 13(11), November 1980.

[8] R. Ducasse and M. van der Schaar. Finding it now: Construction and configuration of networked classifiers in real-time stream mining systems. In S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, editors, *Handbook of Signal Processing Systems*, pages 97–134. Springer, second edition, 2013.

[9] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[10] F. König, D. Boers, F. Slomka, U. Margull, M. Niemetz, and G. Wirrer. Application specific performance indicators for quantitative evaluation of the timing behavior for embedded real-time systems. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pages 519–523, 2009.

[11] E. A. Lee and T. M. Parks. Dataflow process networks. *Proceedings of the IEEE*, pages 773–799, May 1995.

[12] W. Najjar, B. Draper, W. Bohm, and R. Beveridge. The cameron project: High-level programming of image processing applications on reconfigurable computing machines. In *Proceedings of the PACT Workshop on Reconfigurable Computing*, 1998.

[13] J. Pisharath, N. Jiang, and A. Choudhary. Evaluation of application-aware heterogeneous embedded systems for performance and energy consumption. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*, pages 124–132, 2003.

[14] U. Ramacher. Software-defined radio prospects for multistandard mobile phones. *Computer*, 40(10):62–69, 2007.

[15] K. Sudusinghe, I. Cho, M. van der Schaar, and S. S. Bhattacharyya. Model based design environment for data-driven embedded signal processing systems. In *Proceedings of the International Conference on Computational Science*, pages 1193–1202, Cairns, Australia, June 2014.

[16] K. Sudusinghe, S. Won, M. van der Schaar, and S. S. Bhattacharyya. A novel framework for design and implementation of adaptive stream mining systems. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1–6, San Jose, California, July 2013.

[17] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.

[18] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.