Pocket Data Mining: Towards Collaborative Data Mining in Mobile Computing Environments

Frederic Stahl, Mohamed Medhat Gaber and Max Bramer
School of Computing
University of Portsmouth
Portsmouth, Hampshire
PO1 3HE, UK
Email:{Frederic.Stahl, Mohamed.Gaber, Max.Bramer}@port.ac.uk

Philip S. Yu
Department of Computer Science
University of Illinois at Chicago
851 South Morgan Street
Chicago, IL 60607-7053
Email:psyu@cs.uic.edu

***Abstract*—Pocket Data Mining *PDM* is our new term describing collaborative mining of streaming data in mobile and distributed computing environments. With sheer amounts of data streams are now available for subscription on our smart mobile phones, the potential of using this data for decision making using data stream mining techniques has now been achievable owing to the increasing power of these handheld devices. Wireless communication among these devices using *Bluetooth* and *WiFi* technologies has opened the door wide for collaborative mining among the mobile devices within the same range that are running data mining techniques targeting the same application. This paper proposes a new architecture that we have prototyped for realizing the significant applications in this area. We have proposed using mobile software agents in this application for several reasons. Most importantly the autonomic intelligent behaviour of the agent technology has been the driving force for using it in this application. Other efficiency reasons are discussed in details in this paper. Experimental results showing the feasibility of the proposed architecture are presented and discussed.**

I. Introduction

With the continuous advances in handheld mobile devices including smart phones, PDAs (Personal Digital Assistants) and smart sensors, there is an unpreceded opportunity to perform significantly useful data analysis tasks in an ad hoc computing environment. This can be realized with the help of several established areas of study including: (a) data stream mining [3]; (b) mobile software agents [22], [18]; and (c) embedded programming.

A typical scenario for this ad hoc data analysis would include number computationally capable devices like smart phones and sensors, and number of applications that run onboard these devices. An agent platform like the Java Agent DEvelopment Framework *JADE* [2] would be running on all the devices. A computational task would be initiated by one of these devices firing a number of mobile software agents roaming an ad hoc formed network. The agents would discover the data sources, the computational capabilities of the devices that formed the network and the available applications onboard these devices. The agents in turn would take a collective decision on the distribution of the processing subtasks to perform the initiated task according to several criteria like proximity to the source of data, the available applications to perform the process, etc.

This generic scenario, when applied to collaborative data mining, would include mobile software agents of different types. These types could be identified as follows:

- *(Mobile) agent miners (AM)*: these agents are either distributed over the network when the mining task is initiated or are already located on the mobile device.
- *Mobile agent resource discoverers (MRD)*: these agents are used to explore the available computational resources, processing techniques, and data sources.
- *Mobile agent decision makers (MADM)*: these agents roam the network consulting the mobile agent miners to collaborate in reaching the final decision.

Our proposed *PDM* framework in this paper makes use of the above types of agents. Details of the roles of the different agents will be discussed in details when discussing the system architecture in Section III. The primary motive for developing this framework is to enable seamless collaboration among users of mobile data mining applications. Two constraints necessitate the distribution of the task resulting in a collaborative environment. First, the large amounts of data that challenge the state-of-the-art of our smart phones and embedded devices. Second, subscription fees that apply for this data to be streamed to the user's mobile device. Thus, collaborative mining addresses these constraints realising the potential of this important application.

Two stimulating factors have motivated us to use the mobile software technology in this application. The first is the autonomous behaviour that the agent framework supports. This is important to cope with the dynamic nature of the application of varying number of nodes and the data mining algorithms used. Communication efficiency as reported in [21], [16] of using mobile software agents in distributed data mining has been the second factor.

The paper is organized as follows. Section II discusses related work in the mobile data mining area. Our *PDM* architecture and its components are thoroughly discussed in Section III. Experimental results that prove the feasibility of the architecture are presented in Section IV. Finally, the paper is concluded in Section V.

II. Related Work

Related work in this area includes systems developed by Kargupta el al in [15], [20], [17], [13], [1] for mobile data mining for mobile brokering and road safety, and by Pirttikangas et al [19] for context-aware health club. Brief descriptions of these systems will follow.

Kargupta et al [15], [20], [17] have developed the first ubiquitous data stream mining system termed *MobiMine*. It is a client/server PDA-based distributed data mining application for financial data streams. The system prototype has been developed using a single data source and multiple mobile clients; however the system is designed to handle multiple data sources. The server functionalities in the proposed system are data collection from different financial web sites and storage, selection of active stocks using common statistics methods, and applying online data mining techniques to the stock data. The client functionalities are portfolio management using a mobile micro-database to store portfolio data and information about user's preferences, and construction of the WatchList and this is the first point of interaction between the client and the server. The server computes the most active stocks in the market, and the client in turn selects a subset of this list to construct the personalized WatchList according to an optimisation module. The second point of interaction between the client and the server is that the server performs online data mining and then transforms the results using Fourier transformation and finally sends this to the client. The client in turn visualises the results on the PDA screen. It is worth pointing out that the data mining process in MobiMine has been performed at the server side given the resource constraints of a mobile device.

With the increase need for onboard data mining in resource-constrained computing environments, Kargupta et al [13] have developed *Vehicle Data Stream Mining System (VEDAS)*. It is a ubiquitous data stream mining system that allows continuous monitoring and pattern extraction from data streams generated on-board a moving vehicle. The mining component is located on the *PDA*. *VEDAS* uses online incremental clustering for modeling of driving behaviour. A commercial version of *VEDAS* termed as *MineFleet* has been successfully deployed [14], [1].

Pirttikangas et al [19] have implemented a mobile agent-based ubiquitous data mining for a context-aware health club for cyclists. The system is called *Genie of the Net*. The process starts by collecting information from sensors and databases in order to recognize the needed information for the specific application. This information includes user's context and other needed information collected by mobile agents. The main scenario for the health club system is that the user has a plan for an exercise. All the needed information about the health such as heart rate is recorded during the exercise. This information is analysed using data mining techniques to advise the user after each exercise.

Other related work includes the large body of data stream mining algorithms. Key techniques and approaches in the area are discussed in [3] and more recently in the tutorial presented by Gama et al in [9].

Addressing the resource constraints of small computational devices like smart phones and Personal Digital Assistants *PDAs* has been reported in work conducted by Gaber el al in [4], [6], [5], [7]. The approach taken in this body of work has been termed as *Granularity-based* approach. It adapts the data mining algorithm to adjust the resource consumption pattern according to availability of resources. Notably, successful applications of the approach in road safety and healthcare have been reported in [10], [11], [12].

III. PDM Architecture

The architecture of our *PDM* framework is illustrated in Figure 1. From this point onwards in the paper, we shall use the terms *PDM architecture* and *PDM framework* interchangeably. As the figure shows, the data stream mining process runs onboard the users' smart mobile phones. As the data streams in, the model is continuously updated to cope with the possible concept drift of the streaming environments. The process of stream mining is carried out using an *Agent Miner*, denoted as *AM*. *AMs* are distributed at the beginning of initiating the mining task. Some of these miners could be stationary and some others could be mobile. Stationary agents are instructed by the task initiator to mine the streaming data to the mobile device without making any hops. However, the mobile agents could travel to one or more nodes in order to perform the mining task. The choice of using stationary or mobile agent relies on the nature of the task and the number of nodes involved in the processing. Typically, AMs are data stream classification techniques. But the use of other techniques is also possible according to the required task.

If at any point in time, a user decides to use the models built using the different *AMs* on all the mobile phones to collaborate in finding the class label of a set of unlabeled instances, a *Mobile Agent Decision Maker MADM* is fired to visit the nodes consulting the models about the local class label. While these agents are visiting the different nodes, it may decide to terminate its itinerary given that clearly there is a dominant class. This clearly makes the agent framework the suitable technology for this task.

A simple flow chart of the process of collaborative data mining using our *PDM* architecture is given in Figure 2.

Our *PDM* framework raises a number of research issues that are important to be addressed to optimise the task. These issues are currently being investigated by our research team. The following is a list of these issues.

- The number of *AMs* that are decided by the task initiator. In an ad hoc environment, the number of participants may vary. Thus, it is important to involve the number of *AMs* that cover the largest number of attributes and data instances. For example, if the number of mobile phones in a setting is 5, and 2 of these share the same instances and attributes and run the same data mining technique, only one of the two would be chosen for the task depending on

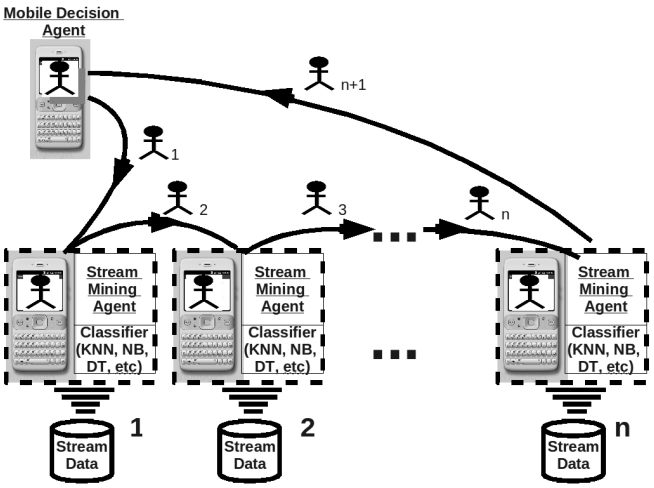

Fig. 1. PDM Architecture

other factors like computational resources and proximity to the data source.

- The number of *MADMs* that are decided by a participant. This is done when that participant is ready to use the output of the stream mining process for decision making. This number relies on the number of participants. Although it would be faster to fire a number of *MADMs* that is equivalent to the number of participants, this could be a burden on the communication network if the number of participants is large. Also, the decision relies on the different data mining techniques that run concurrently. If using a particular model is known to take longer time due to the structure the technique produces (rules, trees, etc.), another agent could visit two different nodes while one agent is consulting the loner running time node. Thus, an optimum number of *MADMs* has to be decided.
- The combination of data mining techniques to be used is essential to ensure that the built models are of optimum accuracy. Once the *mobile agent resource discoverers MRDs* have found out the different techniques and the data sources in the network, stream mining techniques that will be used are decided. This is done according to the request made by the task initiator. For example, if a classification task is initiated and some nodes host more than one classification technique, it is important to decide whether the node runs the two techniques on the same streaming data, or runs one of them according to availability of resources and importance of the attributes in the decision making. It is also essential to decide whether these techniques would be stationary or mobile. If the mobility of agents is decided, it is important to decide the timing of the migration from one node to the other.
- The combination of attributes and data instances that is used as input to the data mining algorithm. This decision relies on what has been discovered by the mobile agent

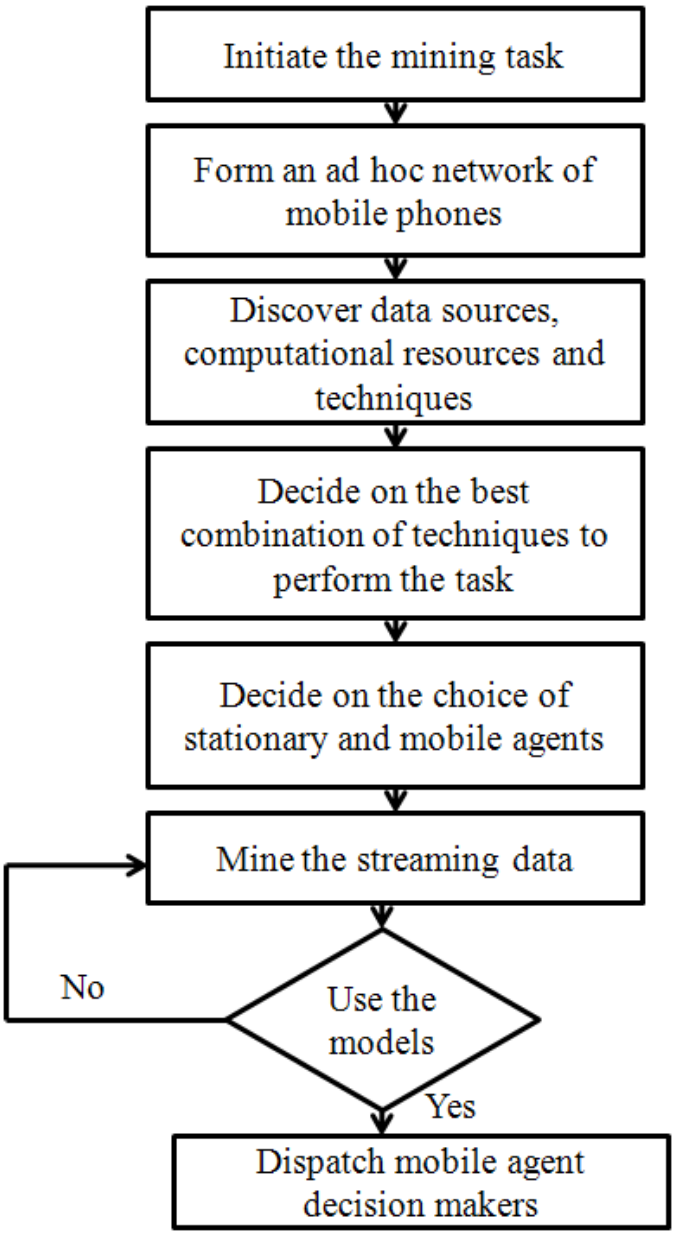

Fig. 2. A Flow Chart of the Collaborative Data Mining Process

resource discoverers. If some attributes and instances are shared among classifiers, it is important to decide whether to use the shared attributes and instances by two or more different classifiers, or use disjoint subsets of the data to accelerate to the process.

- The decision on whether each node is required to adapt to resource availability is important. Some nodes would be in a better position to cope with high speed streaming data due to its high performance computational power. However, some other would need to adapt. As previously mentioned in Section II, the *Granularity-based* approach developed by Gaber et al [4], [6], [5] is able to adjust the algorithm settings in real-time to change the consumption pattern of the algorithm to cope with low availability

of resources. The decision of using one or more of the algorithm granularity settings will be taken in the light of the running techniques and the configuration of the mobile phone.

This paper addresses the first couple of issues listed above which are discussed in more detail and in terms of computational efficiency in Section IV. The rest of the issues are the subject to our currently undertaken investigations and experimental study.

IV. Evaluation of the PDM Framework

A prototype of the PDM framework as described in Section III has been implemented and empirically evaluated in a LAN. For the implementation the well known JADE framework has been used [2], with the reasoning that there exist a version of JADE, *JADE-LEAP (Java Agent Development Environment-Lightweight Extensible Agent Platform)*, that is designed for the implementation of agents on mobile devices and can be retrieved from the JADE project website as an 'add on' [23]. As JADE works on standard PCs as well as on mobile devices it was possible to develop and test the first prototype of the PDM framework on the LAN described below. The LAN consisted of four computers interconnected with a network switch. Three of the computers (computers A, B and C) have a CUP with a 2.8 GHz clock-speed and 1 GB of memory, the fourth computer (computers D) has a CPU with 2.20 GHz clock-speed and 500 MB of memory. The switch used is a standard *CISCO* Systems switch of the catalyst 2950 series. Computer D was used as the base from which all MADMs were started from. Computers A, B and C were hosting AMs that actually implement the data mining algorithms. Computers A, B and C where hosting more than just one AM in order to simulate more nodes in the network than actual computers were available. In order to have a realistic scenario the MADMs were not permitted to visit two consecutively AMs located on the same machine. One constraint in this setup is that it may happen that two or more MADMs visit different AMs hosted on the same computer at the same time. That makes it more likely that there are collisions in the network, which would not be the case if all AMs were hosted on separate computers. So the performance of the PDM framework in the current setup will be worse compared with a setup with one physical computer per AM, like it would be in a real application of the PDM framework.

Figure 3 illustrates the GUI of an MADM which is displayed at the start of the MADM. The text field "Test Instance" takes a data instance to be test for a classification, the values are entered separated by a comma. Further information that the MADM needs in order to calculate the order in which AMs are visited, is the total number of AMs or "stream mining agents available" in the LAN, the number of AMs per machine, the number or ID if this particular MADM and the number of AMs this MADM shall visit. As mentioned before the order of the agents to visit is relevant for this evaluation as several AMs may be hosted on the same computer and an MADM should not visit consecutively AMs that are hosted on the same computer. In a real application only the test instance would be required as a input parameter. The information that the GUI requests would normally be collected from the MRD agent, which has not been implemented yet.

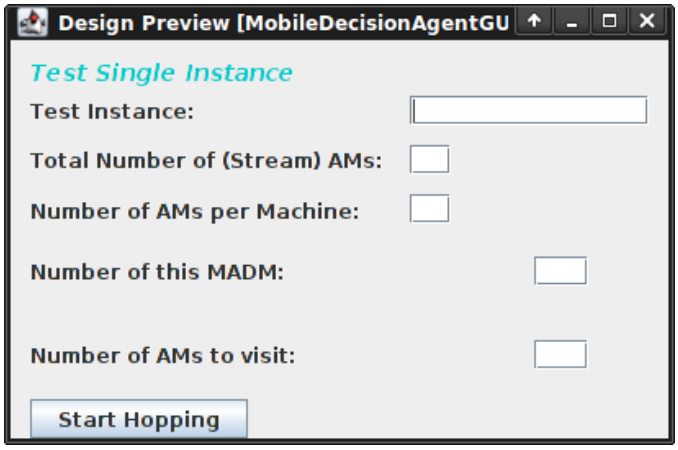

Fig. 3. GUI of an MADM.

Figures 4, 5, 6, and 7 show screenshots of the JADE agents running on computers A, B, and C. While computers A, B, and C are running the data stream classification process, another computer fires an MADM to consult the three aforementioned computers. This is shown in the screenshots depicted in figures 4, 5, and 6. Figure 7 shows that MADM *bob1* has been initiated by computer D and started its itinerary by visiting *Container-1*, which corresponds to computer A. It also shows that after *bob1* finished its itinerary by visiting all the three computers, it is now in a position to take a decision depending on the accumulated results. Note that symbols A, B, and C in Figure 7 refer to the inferred class labels and not the computers that were running the classification process. It is also worth noting that it is just a coincidence to have each computer produces a different label.

Fig. 4. A Screenshot of Computer A.

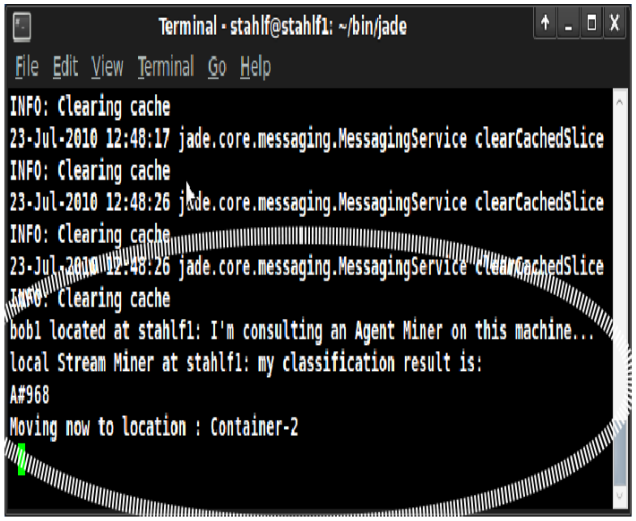

Fig. 5. A Screenshot of Computer B.

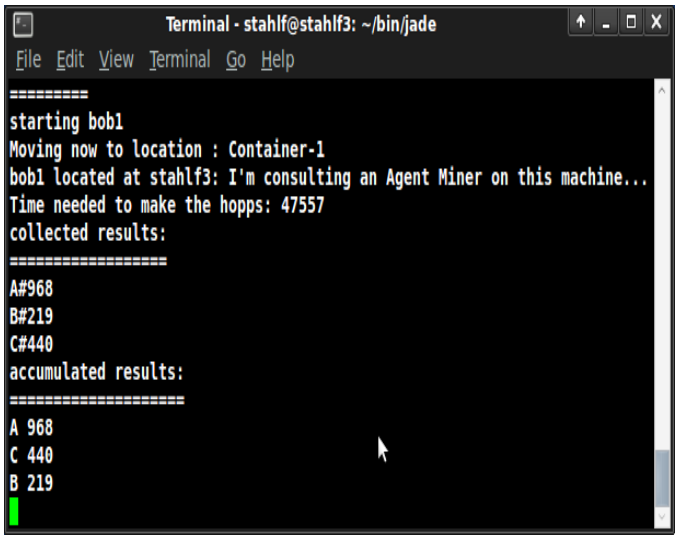

Fig. 7. A Screenshot of Computer D.

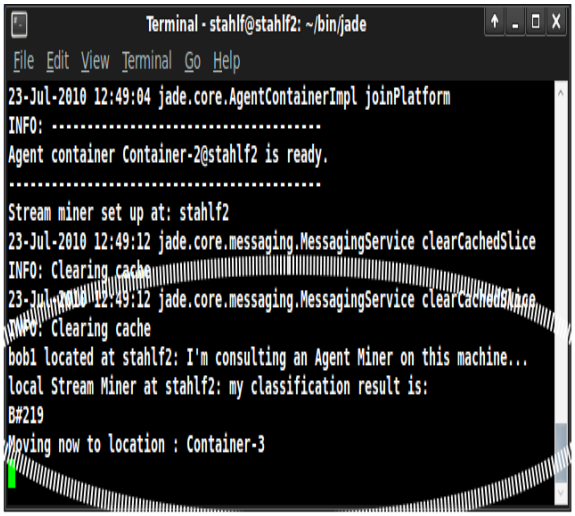

Fig. 6. A Screenshot of Computer C.

The purpose of this section is to evaluate the feasibility of the PDM framework in computational terms, in particular the communication performance and its parallel processing performance. With respect to the communication performance, the property of interest is the time needed for the MADMs to visit all the AMs and return to computer D. With respect to the parallel performance the property of interest is how much faster the PMD framework becomes the more MADMs are used.

A. Evaluation of the Communication Performance

In order to evaluate the communication performance the actual data mining algorithms of the AMs were replaced by a random result generator. Assuming a classification task, the result produced by each AM would consist of the class label and a weight to indicate how reliable or important the classification produced by this particular AM is. The random result generator simply generates a random class label and a random weight. The reason for doing this is that generating a random result consumes only a very little amount of CPU time compared with a actual classification algorithm. In order to measure how quickly an MADM visits all the nodes it is important to bring the execution time of the AM visited to a minimum.

Figure 8 depicts the time needed by one or more MADMs to visit all AMs in the network on several network configurations (different numbers of AMs). What can be seen is that the time needed for communication is decreasing at a high rate at the beginning for using more MADMs. The rate of decrease levels off very quickly and the time needed for communication seems to increase slightly for larger numbers of MADMs. The decrease of communication time with only a few MADMs can be explained by the fact that several MADMs perform their hops in parallel and the more MADMs are used the less ”hops” each MADM has to perform. However the benefit of using more MADMs is contradicted by the fact that the more MADMS are used the higher the traffic on the network and thus the risk of collisions. Regarding that the risk of collisions is increased by the experimental setup itself, as mentioned earlier in this Section, in particular by hosting several AMs on the same computers, it is likely that the communication time would still decrease for a larger number of MADMs if there would be more computers available for the experimental setup. Nevertheless it can be seen that the communication overhead using one ore more MADMs is very low in general.

B. Evaluation of the Parallel Performance using Speedup Factors

One may say that regarding low communication overhead it not necessary to use more than one or two MADMs. However taking into consideration that the actual data mining algorithms embedded in the AMs will consume CPU time draws a different picture. If an MADM visits a AM it waits

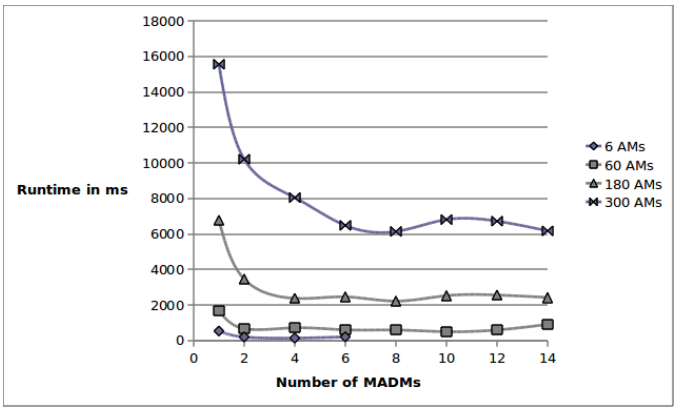

Fig. 8. Time consumed by the MADMs to visit all AMs

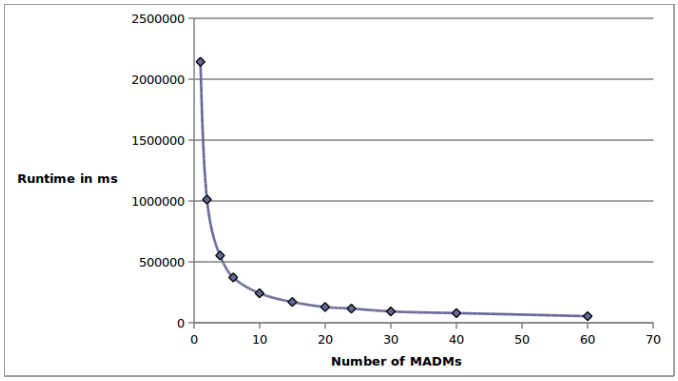

Fig. 9. Runtime of the PDM framework on a set up with 120 AMs and a variable number of MADMs

until the AM has derived a result and then the MADM will visit the next AM. So the more MADMs are used the more AMs can be visited by different MADMs and are executed in parallel (at the same time) and thus reduce the overall runtime.

In order to evaluate this parallel behaviour all AMs have been forced to wait a period amount of time in order to simulate the execution time of their local data mining algorithm. The simulated execution time of the AMs is a random number between 5 to 30 seconds. For the experiments in this Section 120 MAs have been set up evenly distributed between computers A, B and C so each computer hosting 40 AMs. A concrete implementation of the PDM framework using KNN and has been implemented, however the reason for using a random time generator rather than the actual algorithms lies if the limited amount computer hardware available for these experiments. If two AMs or more AMs are hosted on the same machine and two or more AMs are visited by different MADMs at the same time then they would technically be executed in a serial fashion rather than in parallel. However if the system time of the actual computer is used to make the AMs waiting, then the waiting literally happens in parallel if the concerning AMs are executed on the same computer at the same time. Hence using a random number to simulate the runtime of the AMs draws a more realistic picture for the experiments described in this Section.

The experiments conducted in this Section are speedup experiments. The number of work is kept constant at 120 AMs while the number of MADMs and thus the number of CPUs used to execute the AMs is increased step by step.

It is expected that the larger number of MADMs the shorter execution time of the PDM framework as more AMs are consulted in parallel (at the same time).

Figure 9 depicts the runtimes of the PDM framework using 120 AMs and from 1 up to 60 MADMs. What can be seen is that the more MADMs are used the shorter the execution time of the framework. However what can also be seen is that the larger the amount of MADMs the less the framework benefits from each individual MADM. This loss of performance can be explained by the fact that different algorithms consume different amount of time, which leads to that some MADMs are finished earlier in visiting their allocation of AM than others. Also the larger the number of MADMs the larger the communication overhead. Furthermore the more MADMs are used the more time is needed to combine the by the MADMs collected results. At the moment the combining of classification results is a simple weighted majority voting, however different data mining tasks may require different combining strategies which can be implemented in the MADMs. The fact that MADMs finish visiting their allocation of AMs at different times could be avoided by a more dynamic allocation of AMs to each MADM. For example once an MADM that finishes could take over some of the allocated AMs of a different MADM. This may well lower the communication overhead.

Two standard metrics to evaluate a parallel algorithms or architectures like the PDM framework are the speedup factors and the efficiency [24], [25]. They are a convenient way of looking at the overheads mentioned observed in figure 5. The speedup factor S_p is the runtime R_1 of the PDM framework using one MADM divided by the runtime R_p using p MADMs as shown in formula (1) .

$$S_p = \frac{R_1}{R_p} \tag{1}$$

With the speedup factor it is possible to compare how much a parallel version of a system is faster using p computational nodes with one. p in this case is the number of MADMs. In the ideal case the S_p is equal to the number of MADMs. For example if two instead of one MADMs are used then the ideal S_p would be 2, or loosely speaking, using twice the number of MADMs makes the PDM framework twice as fast. The actual speedup factors based on the results depicted in Figure 9 are shown in Figure 10.

The speedup factors in Figure 10 show that even 60 AMs still have a positive impact. However the ideal speedup for 60 MAs would be 60 and is in fact 38, which reflects the communication overhead observed earlier.

A different way of looking at the speedup factors is the efficiency. The efficiency in formula (2) shows how much speedup is achieved per computational node and in the case of the PMD framework this is how much speedup is achieved per

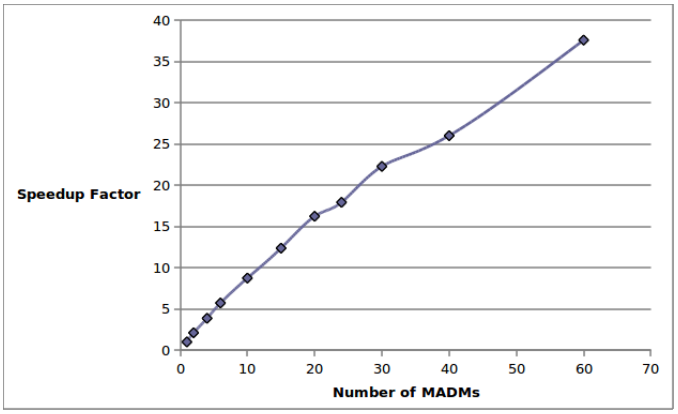

Fig. 10. Speeup factors obtained using the PDM framework with 120 AMs

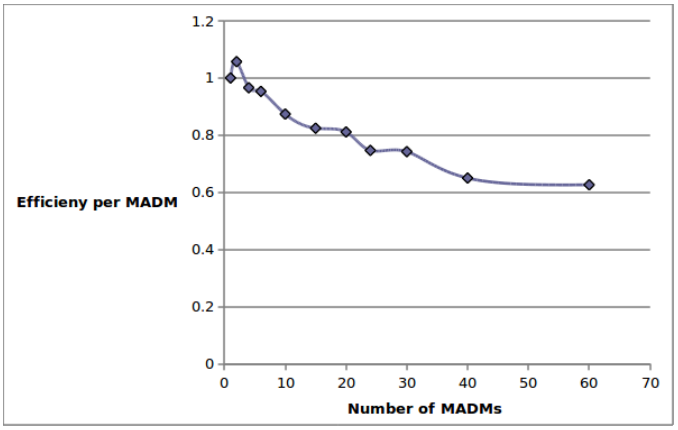

Fig. 11. The Efficiencies obtained using the PDM framework with 120 AMs

MADM. It is the percentage with which the PDM framework profits from each MADM and is caculated by dividing the speedup factors by the number of computational nodes or in our case all MADMs:

$$E_p = \frac{S_p}{p} \tag{2}$$

The efficiencies of the PDM framework are illustrated in Figure 11 and are decreasing the more MADMs are used, which is a normal and expected behaviour caused by the same overheads as stated for the speedup factors. However for using two MADMs the efficiency unexpectedly increases slightly, which is most likely a outlier as for using further MADMs the the efficiencies always decrease. In general what can be seen is that the PDM framework is scaling nicely with respect to the number of MADMs used.

V. Conclusion

The paper introduced our *Pocket Data Mining* framework to enable collaborative mining of streaming data in mobile environments. The framework uses the mobile software agents technology benefiting from its autonomous behaviour and computational efficiency. Experimental results using *JADE* toolkit have proved the applicability of the system.

Future directions in our research would explore the numerous alternatives of collaborative mining techniques and strategies. These include varying the mining techniques, the sharing of attributes and instances of the data among nodes, and the distribution of the roles among agents that would yield the highest accuracy.

References

[1] Agnik, MineFleet Description, http://www.agnik.com/minefleet.html

[2] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa, Developing multi-agent systems with JADE. In Cristiano Castelfranchi and Yves Lesperance, editors, Intelligent Agents VII. Agent Theories Architectures and Lan- guages, 7th International Workshop, ATAL 2000, Boston, MA, USA, July 7-9, 2000, Proceedings, volume 1986 of Lecture Notes in Computer Science, pages 89 - 103. Springer Verlag, 2000.

[3] Gaber, M, M., Zaslavsky, A., and Krishnaswamy, S., Mining Data Streams: A Review, ACM SIGMOD Record, Vol. 34, No. 1, pp. 18-26, June 2005, ISSN: 0163-5808.

[4] Gaber M. M., and Yu P. S., A Holistic Approach for Resource-aware Adaptive Data Stream Mining, Journal of New Generation Computing, ISSN 0288-3635 (Print) 1882-7055 (Online), Volume 25, Number 1, November, 2006, pp. 95-115, Ohmsha, Ltd., and Springer Verlag.

[5] Phung N. D., Gaber M. M., and Rhm U, Resource-aware Online Data Mining in Wireless Sensor Networks, Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2007, pp. 139-146, part of the IEEE Symposium Series on Computational Intelligence 2007, Honolulu, Hawaii, USA, 1-5 April 2007. IEEE 2007, ISBN: 1-4244-0705-2.

[6] Gaber M. M., Data Stream Mining Using Granularity-based Approach, a book chapter in Foundations of Computational Intelligence Volume 6, Abraham A., Hassanien A., Carvalho A., and Snase V. (Eds), Volume 206/2009, pp. 47-66, ISSN 1860-949X (Print) 1860-9503 (Online), ISBN 978-3-642-01090-3, Springer Berlin/Heidelberg, Germany, 2009.

[7] Gaber, M, M., Zaslavsky, A., and Krishnaswamy, S., A Cost-Efficient Model for Ubiquitous Data Stream Mining, Proceedings of the tenth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2004), pp. 747-754, Perugia Italy, July 4-9.

[8] Gama J., and Gaber M. M. (Eds), Learning from Data Streams: Processing Techniques in Sensor Networks, a book published by Springer Verlag, ISBN 3540736786, 9783540736783, 2007.

[9] Gama J., Gaber M. M., and Krishnaswamy S., Data Stream Mining: From Theory to Applications and From Stationary to Mobile, presented in the ACM 25^{th} Symposium On Applied Computing, available online at: http://www.csse.monash.edu.au/ shonali/ACM-SAC10-DS-Tutorial/Tutorial-SAC10-Final.pdf

[10] Haghighi P. D., Zaslavsky A., Krishnaswamy S., Gaber M. M., Mobile Data Mining for Intelligent Healthcare Support, Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS08), pp. 1-10, Hawaii, USA, January 5-8, 2009, IEEE 2009.

[11] Horovitz O., Gaber M. M., and Krishnaswamy S., Making Sense of Ubiquitous Data Streams: A Fuzzy Logic Approach, Rajiv Khosla, Robert J. Howlett, Lakhmi C. Jain (Eds.): Knowledge-Based Intelligent Information and Engineering Systems, 9th International Conference, KES 2005, pp. 922-928, Melbourne, Australia, September 14-16, 2005, Proceedings, Part II. Lecture Notes in Computer Science 3682 Springer 2005, ISBN 3-540-28895-3.

[12] Horovitz, O., Krishnaswamy, S., and Gaber, M, M., A Fuzzy Approach for Interpretation of Ubiquitous Data Stream Clustering and Its Application in Road Safety, Intelligent Data Analysis, Special Issue on Knowledge Discovery from Data Streams JooGama and Jesus Aguilar-Ruiz (Eds.), Vol. 25, No. 1, pp 89-108, 2007, 1088-467X (Print) 1571-4128 (Online), IOS Press.

[13] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, and D. Handy. (2004). VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. Proceedings of the SIAM International Data Mining Conference, Orlando.

[14] H. Kargupta, V. Puttagunta, M. Klein, K. Sarkar, On-board Vehicle Data Stream Monitoring using MineFleet and Fast Resource Constrained Monitoring of Correlation Matrices. Next Generation Computing. Invited

submission for special issue on learning from data streams, volume 25, no. 1, pp. 5–32, 2007.

[15] H. Kargupta, B. Park, S. Pittie, L. Liu, D. Kushraj, and K. Sarkar (2002). MobiMine: Monitoring the Stock Market from a PDA. ACM SIGKDD Explorations. January 2002. Volume 3, Issue 2. Pp. 37–46. ACM Press.

[16] H. Kargupta, I. Hamzaoglu and B. Stafford, Scalable, Distributed Data Mining Using an Agent-Based Architecture. Proceedings of Knowledge Discovery and Data Mining. Eds: D. Heckerman, H. Mannila, D. Pregibon and R. Uthurusamy, pp. 211214, 1997, AAAI Press.

[17] H. Kargupta, K. Sivakumar, and S. Ghosh. (2002). Dependency Detection in MobiMine and Random Matrices. Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pp. 250–262. Helsinki, Finland .

[18] Page J., Padovitz A., and Gaber M. M., Mobility in Agents, a Stumbling or a Building Block?, Proceedings of Second International Conference on Intelligent Computing and Information Systems, Cairo, Egypt, 5-7 March 2005.

[19] S. Pirttikangas, J. Riekki, J. Kaartinen, J. Miettinen, S. Nissila, and J. Roning. Genie Of The Net: A New Approach For A Context-Aware Health Club. In Proceedings of Joint 12th ECML'01 and 5th European Conference on PKDD'01. September 3-7, 2001, Freiburg, Germany.

[20] S. Pittie, H. Kargupta, and B. Park. (2003). Dependency Detection in MobiMine: A Systems Perspective. Information Sciences Journal. Volume 155, Issues 3-4, pp. 227-243, Elsevier.

[21] J. da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch, Distributed Data Mining and Agents, Engineering Applications of Artificial Intelligence Journal, 2005 volume 18, pp. 791–807.

[22] Zaslavsky A., Mobile Agents: Can They Assist with Context Awareness?, IEEE MDM, Jan. 2004 ,Berkeley, California.

[23] JADE-LEAP: http://jade.tilab.com/.

[24] Hennessy J. L., and Patterson D. A. (2003). Computer architecture a quantitative approach (Third ed.). USA: Morgan Kaufmann.

[25] Hwang K., and Briggs F. A. (1987). Computer architecture and parallel processing (International ed.). McGraw-Hill Book Co.