Applied Computing Review

2011, Vol. 11, No. 4

Frontmatter

Research Articles

Applied Computing Review

A Message from the Editor

Welcome to Applied Computing Review! I am happy to announce that we have begun publishing the ACR newsletter electronically on a quarterly basis. ACR contains invited papers from world-renowned researchers and selected papers presented by prominent scholars and professionals of the Symposium on Applied Computing (SAC). The selected papers from the 2011 SAC, held in Taichung, Taiwan, have been expanded, revised, and peer-reviewed again to be included in the current issue of ACR.

Our goal is to provide you with a platform for exchanging a wide variety of novel and promising ideas in numerous fields of computer science. We are proud that we have provided a springboard for further improvements and that we have given excellent service to quite a few technical communities. We believe that our effort has made it possible to serve the scientific computing society in a productive and efficient manner. We look forward to working with the ACM SIG Governing Board to further expand SIGAPP by increasing membership and developing a new journal on applied computing.

The papers included in ACR represent the modern applied computing research trends and I would like to thank the authors for contributing to the state-of-the-art methods and technology in applied computing. I am grateful to the highly qualified peer reviewers that coordinated an outstanding lineup of technical paper reviews. I also would like to show my appreciation to the track chairs who are serving on the editorial board of ACR and to the SIGAPP officers who are serving as the associate editors. I especially wish to thank Dr. John Kim, technical editor of ACR, and Ms. Irene Frawley for their dedicated work and support. This issue of ACR couldn't have been published without significant efforts made by everyone who I have mentioned above and I want to express my sincere gratitude to them.

Best Regards,

Sung Shin
Editor in Chief

Next Issue

The planned release for the next issue of ACR is March 2012.

SAC 2012 Preview

The 27th Annual edition of the ACM Symposium on Applied Computing (SAC) will be held in the Congress Center of Riva del Garda (Trento), Italy, March 26-30, 2012. Riva del Garda is a small picturesque town situated on the northern shore of Lake Garda, Italy's largest lake. Riva del Garda lies in Trentino, a beautiful region in Northern Italy where the Dolomites and the Southern Alps show themselves in all their beauty.

SAC 2012 is hosted by The Microsoft Research – University of Trento Center for Computational and Systems Biology. It is sponsored by the ACM Special Interest Group on Applied Computing, Provincia Autonoma di Trento, and Riva del Garda Congress Center. The local organizing committee consists of Dr. Paola Lecca, Conference Vice-Chair; Dr. Mirtis Conci, Local Arrangements Chair; and Dr. Elisabetta Nones and Ms. Susan Longhi, committee members.

The conference starts on Monday with the Tutorials Program, organized by Dr. Dan Tulpan. The planned program offers tutorials for all attendees free of charge. The local committee is planning a social luncheon for tutorial attendees who choose to participate, for a minimal fee. The planned tutorials include *Requirements Meet Interaction Design, Evaluation of Recommender Systems, Metaheuristic Algorithms: Theory and Applications,* and *Web Crawling.* Please visit the conference website for more details at http://www.acm.org/conferences/sac/sac2012/.

The technical program starts on Tuesday with keynote address session by Professor Letizia Tanca, from Dipartimento di Elettronica e Informazione Politecnico di Milano, Italy. On Thursday, the keynote speaker will be Professor Anthony Finkelsteinr, from University College London, UK. Details are posted on the conference website.

The four-day technical program, coordinated by Drs. Chih-Cheng Hung and Jiman Hong, offers a wide range of sessions from the 30+ tracks organized this year by international groups of experts lead by Track Chairs and Co-Chairs. The complete listing of sessions and accepted papers in this year's edition are listed on the conference website. The open Call for Track Proposals resulted in 34 tracks that were finally accepted after rigorous review by the organizing committee. The Call for Papers attracted over 1000 submissions from 64 countries. All submissions underwent a blind review process and resulted in the acceptance of 272 papers, making SAC 2012 acceptance rate around 26%. Papers are organized into five themes: *AI and Agents, Distributed Systems, Information Systems, Software Development, System Software and Security.*

The posters program, coordinated by Dr. Mathew Palakal, offers a selection of invited posters that represent high quality work from SAC tracks. The posters program is planned for Thursday, morning and afternoon sessions. Posters are invited papers that have received high review scores. A total of 77 posters will be presented in this year's posters program.

This year, the organizing committee is planning two receptions on Tuesday and Wednesday. Tuesday reception is sponsored by SIGAPP and will take place at the conference venue. The Wednesday reception is sponsored by *Municipality of Riva del Garda* and hosted by the city Mayor at the *La Rocca di Riva del Garda*, near the conference venue. In addition, free lunches are planned for Tuesday, Wednesday, and Thursday. Morning and afternoon coffee breaks are also planned for all four days of the conference. The conference Banquet event is scheduled for Thursday evening at the *CAFFÈ CASINÒ , Città di Arco - Salone delle Feste*, a historic palace near Riva del Garda. Transportation will be provided, departing from the conference venue.

A paper from each theme will be selected as a Best Paper. A Best Poster will also be selected from the Posters Program. The winners will be honored during the Banquet's awards program and will receive both certificates and prizes. Other awards will be presented during the program.

Riva del Garda is surrounded by the mountains, offering beautiful landscapes and healthy and wholesome agricultural products. The atmosphere is a typical Mediterranean with inviting lakeside beaches and villas reached on leisurely steamers. To enjoy the city and surrounding area, Rivatour (the official SAC 2012 travel agency) will manage transportation, hotel reservations, and excursions for SAC attendees. Rivatour is offering special rates for all these services. Please visit the conference website for more details. For those who plan to join us in Riva del Garda and on behalf of the local host and organizing committee, we hope you enjoy the conference program and have a pleasant stay at Riva del Garda.

On Behalf of SAC Steering Committee,

Hisham Haddad
Member of the Steering Committee
Member of SAC 2012 Organzing Commitee

Efficient Hole Bypass Routing Scheme Using Observer Packets for Geographic Routing in Wireless Sensor Networks

Hyunseung Choo
School of Information and Communication Engineering
Sungkyunkwan University
Korea
choo@ece.skku.ac.kr

Mooshik Choi
Smart Networking Division
LS Industrial Systems
Korea
cmszzang@naver.com

Minhan Shon
School of Information and Communication Engineering
Sungkyunkwan University
Korea
minari95@skku.edu

Doogsoo Stephen Kim
Indiana University
Purdue University Indianapolis
USA
dskim@iupui.edu

ABSTRACT
Greedy forwarding fails due to void area (called hole) where no nodes can be deployed in realistic wireless sensor networks. This is known as the local minimum problem. In this paper, we propose *BHOP-GR(Bypassing Hole Scheme Using Observer Packets for Geographic Routing)* that selects the optimum bypassing path while minimizing overhead to maintain hole boundary information. Several schemes have been recently proposed to solve this problem. However, they have overhead about nodes that are adjacent to the hole that have to maintain hole boundary information and increase hop count due to additional data packet transmission. *BHOP-GR* detects the hole boundary using an observer packet previously. When it transmits the data packet using a time delay, this can be arranged to decrease the average hop counts and routing path length. Also, using a virtual regular hexagon model, which can exactly cover a hole to be detoured, minimizes overhead to maintain hole boundary information. In the performance evaluation, we show that *BHOP-GR* has at least 34% and 25% of the average hop counts and routing path length compared to SLGF(Safety Information based Limited Geographic greedy Forwarding) and Virtual Circle, respectively.

Categories and Subject Descriptors
C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms
Protocol, Design, Performance.

Keywords
Wireless Sensor Network, hole detour, routing, greedy forwarding

A short version of this paper appeared in the proceedings of the International Conference on Information Networking (ICOIN), January 2011.

1. INTRODUCTION
Geographic routing [1-3] is a typical routing technique used in forwarding packets from a source node to a destination node based on geographic information in a wireless sensor network environment. The geographic routing only uses local information within a transmission range. Thus, it is efficient, simple, and scalable. The advantages are adequate for a wireless sensor network environment that has limited memory, energy and computation capacities.

Greedy forwarding [4, 5] is a typical geographic routing technique, in which a node selects one of the neighbor nodes closest to a given destination as the next forwarding node. Greedy forwarding fails in packet transmission when reaching a local minimum problem[6], that a node has no neighbors closer to the destination than itself. The occurrence of area where no nodes can be deployed in realistic wireless sensor networks can be caused by many factors, such as sparse deployment, physical obstacles, node failures, communication jamming, power exhaustion, and animus interference [6-11]. This area, termed a hole, is the cause of the local minimum problem.

Some noteworthy approaches have been proposed recently to solve the local minimum problem. Karp *et al*. proposed the Greedy Perimeter Stateless Routing (GPSR) scheme [6], in which the data packets tend to be routed along the boundaries of a hole. Jiang *et al*. proposed the Safety Information based Limited Geographic greedy Forwarding (SLGF) [12]. It uses a new information model, in which nodes are labeled as safe or unsafe nodes in four quadrants. Tian *et al*. proposed a virtual ellipse scheme[13]. It forms a virtual ellipse that covers the hole to be detoured. After detouring, the information of the ellipse is sent to nodes inside the ellipse because it uses a detour around the hole. Similarly Yu et al. proposed a virtual circle method. In this scheme, nodes around a hole are identified using a boundhole algorithm. A virtual circle covering the hole is formed [14]. However, SLGF and virtual ellipse schemes have overhead on nodes that are adjacent to the hole since they have to maintain the hole boundary information. In the virtual circle, to avoid the hole

problem, a source node forwards the data packet twice. A source node in the virtual circle cannot forward the data packet to a destination.

In this paper, we propose the Bypassing Hole Scheme Using Observer Packets for Geographic Routing (BHOP-GR). It selects the optimum bypassing path and also minimizes overhead to maintain hole boundary information. In BHOP-GR, a node can detect if it is located on the boundary of a hole by the boundhole algorithm [15] then one node calculates the center point of the hole. The source node forwards the Observer packet to the destination using greedy forwarding to obtain hole information. Previously, it obtained efficient hole boundary information through the Observer packet. The hole boundary node receiving the Observer packet forwards hole information to the source node. The intermediate node calculates a virtual regular hexagon based on hole information, and selects the detour points from points of the virtual regular hexagon. Using detour points, it forwards data packets to the destination node through the ideal routing path. BHOP-GR using the Observer packets, the model of virtual hexagon decreases the average hop counts and routing path length and minimizes overhead to maintain hole boundary information.

The remainder of the paper is organized as follows. In section 2, we briefly describe related work. Our proposed scheme BHOP-GR is presented in section 3. Section 4 uses simulation to evaluate performance. Finally, we conclude our work in the last section.

2. RELATED WORKS

GPSR [6] is a traditional geographic routing scheme used to bypass a hole. In GPSR, a source node knows the location of the destination and each node knows the location of its neighbor nodes. A source node forwards data packets including the location of the destination. A node receiving data packets selects the neighbor node closest to the destination as the next forwarding node. When the routing process reaches a hole at an intermediate node, it will start a perimeter routing phase. In this, the packet is routed by the "right-hand rule", until it reaches a node that is closer to the destination than the hole boundary node. Then, the routing returns to the greedy forwarding phase. However, without sufficient shape information about the hole, this routing scheme may use a long detour path in the perimeter routing, compared to the shortest path to the destination. Also, the excessive energy consumption may occur in the hole boundary nodes. The excessive energy consumption may lead to the hole diffusion problem.

Jiang et al. [12] recently proposed Safety Information based Limited Geographic greedy Forwarding (SLGF). SLGF uses a new information model, in which nodes are labeled as safe and unsafe nodes in four quadrants. A node is labeled as an unsafe node in a quadrant if there is no safe neighbor in that quadrant. The unsafe information is spread until no new node that is unsafe formed. SLGF tries to avoid forwarding packets to the unsafe nodes, since using the unsafe nodes will cause the local minimum. When a node has a packet, it tries to find a safe neighbor in the requesting zone, which is a rectangle formed by its coordinate, and the coordinate of the destination to which the packet will be forwarded. If it cannot find a safe node in that area, it finds the first safe node in a counter clockwise direction. In the case where the destination is an unsafe node and is in the request zone of the current node, unsafe nodes are used to forward packets to the destination. However, without sufficient shape information about the hole, such a routing may use a long detour path in the perimeter routing and it has overhead about nodes that are adjacent to the hole that have to maintain the hole boundary information.

Nodes around the hole are identified using the boundhole algorithm [9]. A virtual circle that covers the hole detour is formed. After detouring, the information of the virtual circle is transferred to nodes in the boundary of the hole. When a source node has data, it sends the data to the destination node using geographic routing. If this data arrives at a node in the boundary, this node informs the source node the information of the virtual circle. Next, the source node calculates the anchor point that is the intersection point of two tangent lines to the circle at the source and destination nodes. The source node sends the data to the node that is closest to the anchor point and this node forwards the data to the destination node. However in a virtual circle, to avoid the hole problem, a source node forwards the data packet twice. Moreover, a source node in the virtual circle cannot forward the data packet to its destination.

Instead of forming the virtual circle as the work in the previous session, the work in [13] forms a virtual ellipse that covers the hole. Next, the information of the ellipse is sent to nodes inside the ellipse. When the source node has data to send, it uses greedy forwarding to send to the destination node. If the data packet arrives at a node in the ellipse, this node finds an anchor point that lies on the tangent line to the ellipse at the node. A node is randomly selected from nodes whose distance to the anchor point is less than a predefined value. This node then does what the source node does. Finally, the data packet arrives at the destination node. However, in the virtual ellipse like SLGF, nodes that are adjacent to the hole have to maintain the hole information.

3. EFFICIENT HOLE BYPASS SCHEME

3.1 Motivation and Overview

In GPSR and SLGF schemes, without sufficient shape information about the hole, such a routing may use a long detour path in the perimeter routing. Also, the excessive energy consumption in the hole boundary nodes may lead to the hole diffusion problem. In the SLGF and the virtual ellipse, nodes that are adjacent to the hole have to maintain the hole information. Finally in the virtual circle, to avoid the hole problem, a source node forwards the data packet twice and a source node in the virtual circle cannot forward the data packet to a destination. Thus, excessive energy consumption and overhead to maintain hole boundary information must be solved only with the local information to bypass hole routing.

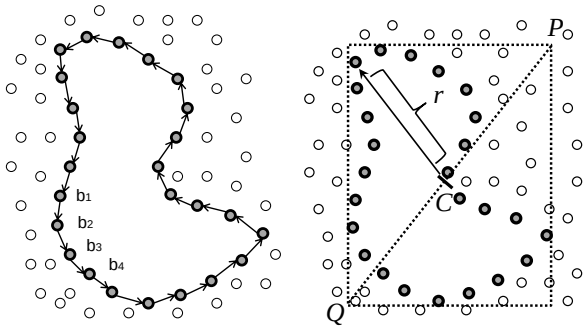
In this section, we propose the Bypassing Hole Scheme Using Observer Packets for Geographic Routing, BHOP-GR. The goal of the proposed scheme is to reduce the number of data packet transmissions and minimize the number of nodes that have hole information. In BHOP-GR, hole boundary is detected using an observer packet. When a node transmits a data packet using the time delay, it can arranged to decrease the average hop count and routing path length. BHOP-GR has three steps: (1) hole boundary creation phase, (2) packet forwarding phase, (3) hole detour phase. BHOP-GR decreases average hop counts and minimizes the overhead to maintain hole boundary information through these three phases.

In this work, we consider the following assumptions.

1. Each node has its own ID and same constant radio range.
2. Each node knows its location information via GPS[16] or some distributed localization methods[17-19] when GPS is not available.
3. The source node knows the position of the destination node.
4. Each node knows the location of its neighbor nodes by exchanging HELLO messages.
5. Our study does not consider other means of energy savings
   such as sleep/awake scheduling and transmission power control.
6. The network is static and has relatively low congestion.
7. Computation cost is very small compared to transmission cost.

3.2 Hole Boundary Creation Phase

Fig.1 shows the hole boundary creation phase. A node can detect if it is located on the boundary of a hole by the BOUNDHOLE algorithm. The node that first detects the hole sends out the hole boundary detection (HBD) message around the hole by the well-known right hand rule. The mission of the HBD message is to trace the location of all nodes on the boundary of the hole. As shown in Fig. 1(a), node b_1 first detects the hole and initiates a HBD packet marked with its own ID and forwards the HBD packet to the neighbor hole boundary node b_1 by the right hand rule. It inserts its own location coordinate into the received HBD packet and then forwards it to node b_2 by the right hand rule. This process continues until the HBD packet has traveled around the hole and is eventually received by the initiator node b_1. Node b_2 gets the location coordinate of all nodes on the boundary of the hole from the received HBD packet.



(a) (b)

Figure 1. Hole Boundary Creation Phase.

Then, node b_1 locates two points, P and Q, in the network with coordinates. As shown in Fig. 1(b) C is the center point of line $\overline{PQ}$ with the coordinate:

$$C \in \{(x_c, y_c) | x_c = (x_p + x_q)/2, y_c = (y_p + y_q)/2\} \quad (1)$$

Then, node b_1 calculates C as the center and r as radius as shown in Fig. 1(b), where r is the longest distance between C and all hole boundary nodes.

$$r = \max_{1 \le i \le n} \{\overline{b_i C}\} \quad (2)$$

Then, node b_1 sends the packet including the coordinate of the circle center C and circle radius r to all hole boundary nodes by the right hand rule again. Then all hole boundary nodes are aware of the center and radius of the virtual hexagon.

3.3 Packet Forwarding Phase

As shown in Fig. 2, the source node S forwards the Observer packets to the destination using greedy forwarding and waits for a time delay τ. Intermediate nodes store the previous node ID using the observer packets at each step. After detecting the hole information, the observer packet returns to the source node and intermediate nodes on the way will save the previous node ID to give hole information to a data packet on the same route. After time delay τ, the source node S forwards a data packet to the destination D using greedy forwarding. Each data packet header includes an anchor location field, and a flag field indicating if the packet is in straightforward mode or detour mode. The data packet is initially set to straightforward mode and its detour location field is set to void by the source node S. Source node S also includes the geographic location of the destination in each data packet. The destination location field is only set by source node S, and left unchanged as the packet is forwarded through the network.

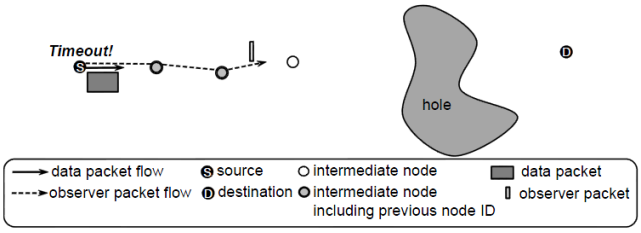


Figure 2. Forwarding Observer Packets.

When a hole boundary node b_1 receives the Observer packet, b_1 sends the hole information message to the source node S. The hole information message contains the information of the virtual hexagon (center point C, radius r). As shown in Fig. 3, when the data packet reaches the intermediate node which has the hole information, the intermediate node forms a virtual hexagon. In this scheme, we define the intermediate node as junction node J.

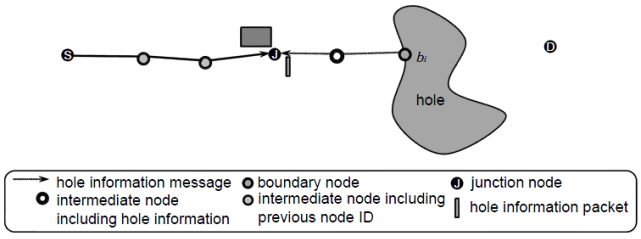


Figure 3. Selection of Junction Node.

3.4 Hole Detour Phase

Node J calculates the vertices of the virtual regular hexagon. Using a virtual regular hexagon, BHOP-GR has an advantage that the source node in a virtual regular hexagon easily calculates detour points. As shown in Fig. 4(a), we identify vertices of a virtual regular hexagon as V_1 , V_2 , V_3 , V_4 , V_5 and V_6 . When a node *J* creates virtual regular hexagon, it creates $\overline{V_2 V_3}$ and $\overline{J D}$. They are parallel as shown in Fig. 4(b). As shown in Fig. 4(b), node *J* generates a virtual regular hexagon based on the location of $\overline{J D}$. In this study, we define the dynamic formation of the virtual regular hexagon procedure. BHOP-GR prevents the hole diffusion problem through this procedure.

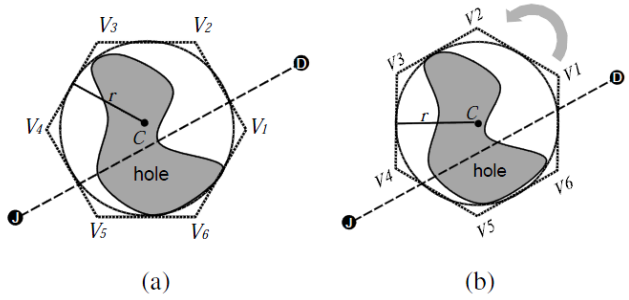


Figure 4. Formation of Virtual Regular Hexagon.

Table 1 shows the detour points selection algorithm. After the formation of a virtual regular hexagon (line 1), node *J* calculates detour points *K*. As shown in Fig. 5, a node J divides the hexagon into two areas by using $\overline{J D}$. Node *J* deletes V_4 closest to node *J* and V_1 closest to the destination *D* from candidates of detour points among vertices of virtual regular hexagon (lines 2-3). It selects vertices of the virtual regular hexagon without V_1 , V_4 as candidates of detour points. Node *J* selects a node V_3 closest to node *J* among candidates of detour points. Finally, it decides the 1st detour point V_3 which has the minimum distance between V_3 and $\overline{J D}$; then, selects the 2nd detour point V_2 which exist in the same area including V_3 (lines 4-9).

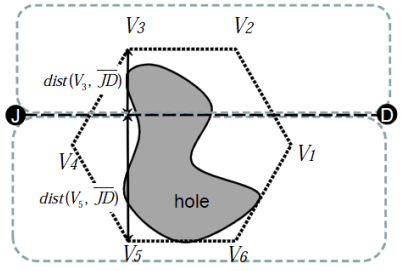


Figure 5. Detour Point Selection.

Table 1. Detour Points Selection Algorithm

ALGORITHM 1: Detour Points Selection Algorithm

Input: *junction node J, and destination D, Center point C, Radius R*
Output: *set of anchor points*
DPSA*(J, D, R,C)*
1: $Calc\_hexa\_point(C, R)$ *//calculation 6 points of virtual hexagon*
2: $x_{V_U} \leftarrow x_{V_3}, y_{V_U} \leftarrow y_{V_3}$ *// V_U: Upper hexagon point*
3: $x_{V_L} \leftarrow x_{V_3}, y_{V_L} \leftarrow y_{V_5}$ *// V_L: Lower hexagon point*
4: **if** $dist(V_U, \overline{JD}) < dist(V_L, \overline{JD})$ **then**
5: $x_{K_1} \leftarrow x_{V_U}, y_{K_1} \leftarrow y_{V_U}$ *// K_1: 1^{st} anchor point*
6: $x_{K_2} \leftarrow x_{V_2}, y_{K_2} \leftarrow y_{V_2}$ *// K_2: 2^{nd} anchor point*
7: **else**
8: $x_{K_1} \leftarrow x_{V_L}, y_{K_1} \leftarrow y_{V_L}$
9: $x_{K_2} \leftarrow x_{V_6}, y_{K_2} \leftarrow y_{V_6}$
10: **end if**
11: $GFRouting(J,D,K_1,K_2)$

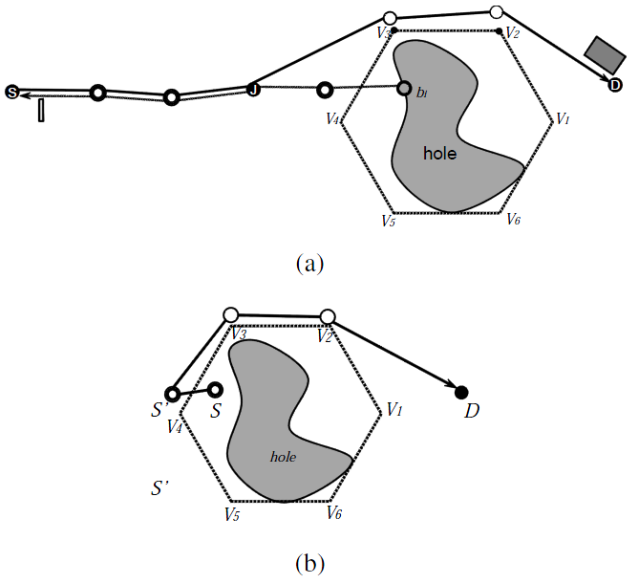


Figure 6. Hole Detour Phase.

Fig. 6 shows the hole detour phase in BHOP-GR. Node J sets its flag to detour mode, and then forwards the data packet to the node geographically closest to the detour points set by greedy forwarding. When the node closest to the 2nd detour point receives the data packet, it resets the packet to straightforward mode and resets the detour location field to void; then, forwards the data packets to the original destination by greedy forwarding. As shown Fig. 6, if the source node S is located in a virtual regular hexagon, the source node S selects temporary node S' which is the closest node to a vertex of the hexagon. The temporary node S' selects detour points set and bypass hole using the hole detour phase.

A routing scheme has two recovery methods for the local minimum problem. First, in the Link Recovey method, when the routing process becomes a hole at an intermediate node, it will bypass a hole. In section 2, GPSR is a Link Recovery method. The other method is Path Recovery, in which a source node detect

hole information in advance then, it will bypass a hole using a new routing path based on hole information. In section 2, the virtual circle is a Path Recovery method. In our scheme, a source node waits for time delay τ before transmitting data packets. In this scheme, if the value of τ increases, it is similar to Path Recovery method, if the value of τ decreases, it is similar to the Link Recovery method. We need to select optimal value of time delay τ in BHOP-GR, to benefit from the path recovery method. In this paper, the data and observer packet are 100bytes and 3bytes, respectively [21, 22]. Through simulation results, time delay τ sets the observer packet a round trip time of 1-hop. Our future study will focus on the ways to select an efficient value for the time delay τ through various schemes and data.

4. PERFORMANCE EVALUATION

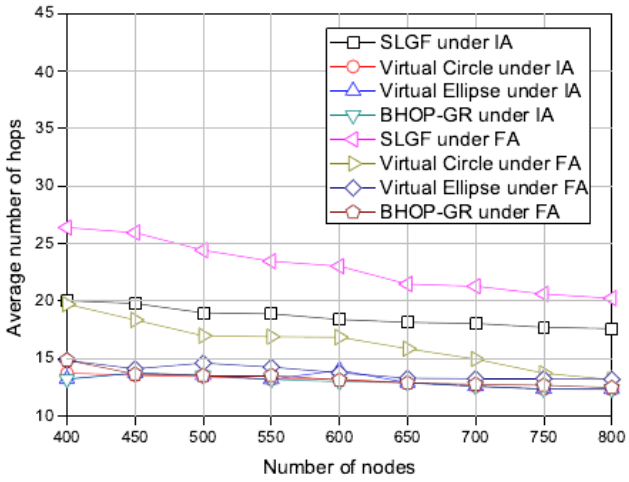
4.1 Simulation Environment

This section implements and compares the performance of BHOP-GR to SLGF, virtual ellipse, virtual circle schemes. The simulation focuses on evaluating the effects of physical-layer packet loss without concerning the packet loss from extraneous factors such as MAC collisions. We run the experiments using a simulator built in C++. Table 1 and summarizes the parameters were used in [20, 21]. This section simulates networks with different number of nodes ranging from 400 to 800 with 50 increments under the same propagation characteristics. Nodes are deployed randomly in the network and 100 packets are forwarded between randomly selected source node and destination node. We use the average of 100 runs to evaluate performance. The data and observer packets are 100 bytes and 3 bytes, respectively [21, 22].

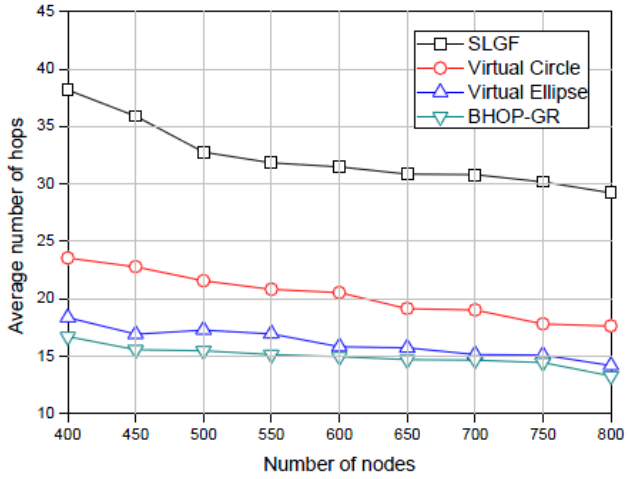
Table 1. Simulation parameters

Radio Parameters	
Modulation	16-ary orth.
Frequency band	2.4~2.4835*GHz*
Output power	0*dBm*
Bit rate	250*kbps*
Time delay configuration	
Observer payload size	3*bytes*
Observer packet delay	2.208*ms*
Data payload size	100*bytes*
Data packet delay	5.312*ms*
Deployment configuration	
Area height	200*m*
Area width	200*m*
Number of nodes	400~800
Range	20*m*

Nodes are deployed to cover an area of 200m x 200m under different deployment models in the simulation [12]. First, the nodes will be deployed uniformly. This is the ideal area model (denoted by IA), in which the hole is only caused by a sparse



(a)



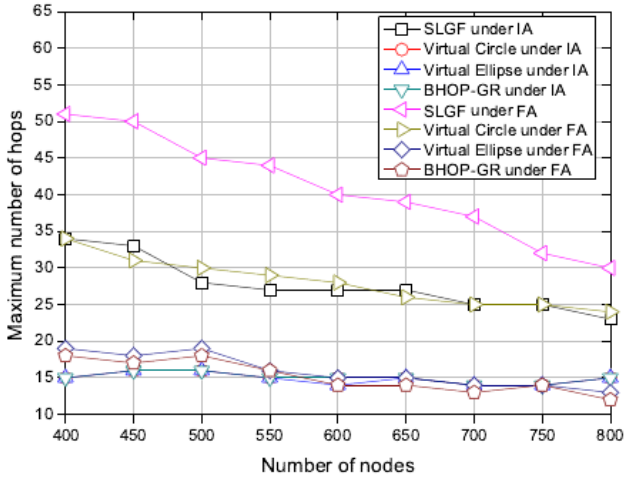
(b)

Figure 7. (a) The average number of hops under IA and FA (hole size 50m x 50m) model (b) The average number of hops under FA model with hole size 100m x 100m.

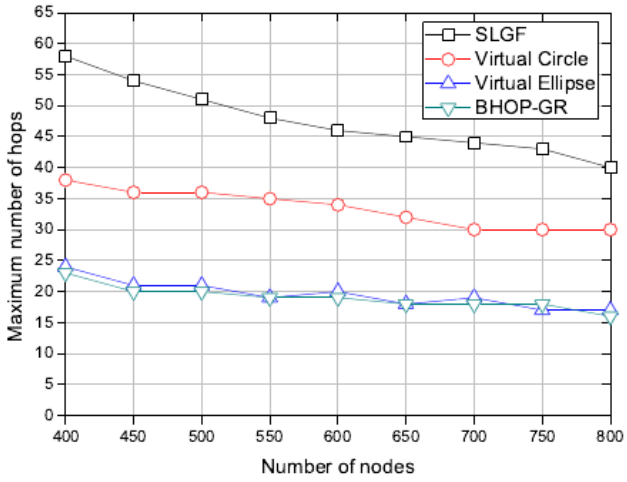
deployment. Usually, the hole size is very small. Second, we randomly set some forbidden areas inside the network area, in which no nodes can be deployed. The forbidden areas, which may be irregular, are constructed to study the impact of larger holes in related methods and in BHOP-GR. Such a model is denoted by FA.

4.2 Simulation Results & Analysis

In Fig. 7(a), we show the average number of hops at different numbers of nodes under both IA and FA models. Under the IA model, the average number of hops of SLGF is greater than the average number of hops of the virtual circle, virtual ellipse, and BHOP-GR. BHOP-GR is similar to the virtual circle, and virtual ellipse in the average number of hops, since the hole size is very

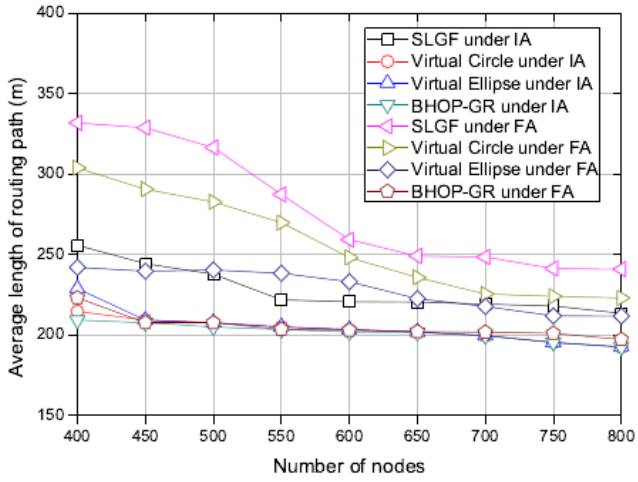


(a)

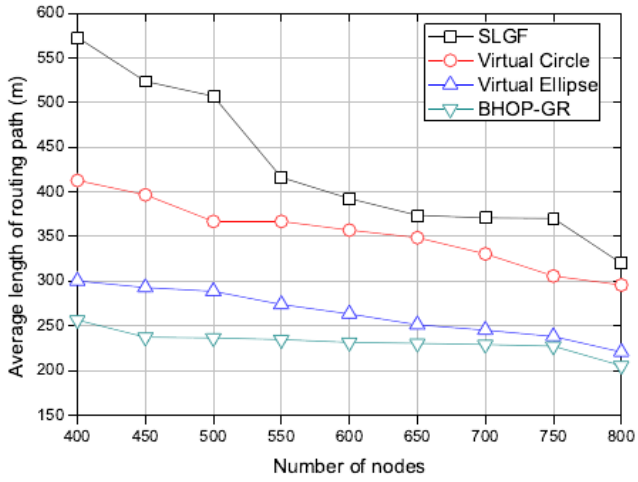


(b)

Figure 8. (a) The maximum number of hops under IA and FA (hole size 50m x 50m) model (b) The maximum number of hops under FA model with hole size 100m x 100m.



(a)



(b)

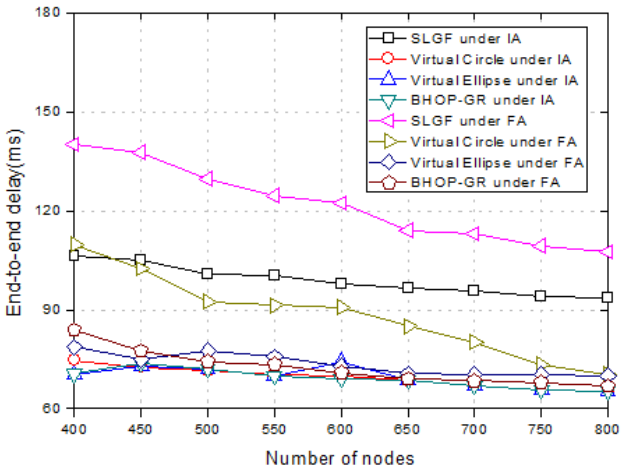
Figure 9. (a) The average Length of Routing Path under IA and FA (hole size 50m x 50m) model (b) The average Length of Routing Path under FA model with hole size 100m x 100m.

small or the hole does not exist under the IA model. However, the FA model is constructed to study the impact of larger holes on the proposed algorithms. In Fig. 7(b), previously the BHOP-GR detects the hole boundary using the observer packet and when it transmits the data packet using a time delay, it can arrange to decrease the average hop counts. Overall, the curve of the greedy forwarding protocols decrease when the number of nodes increase, because the hole size decreases.

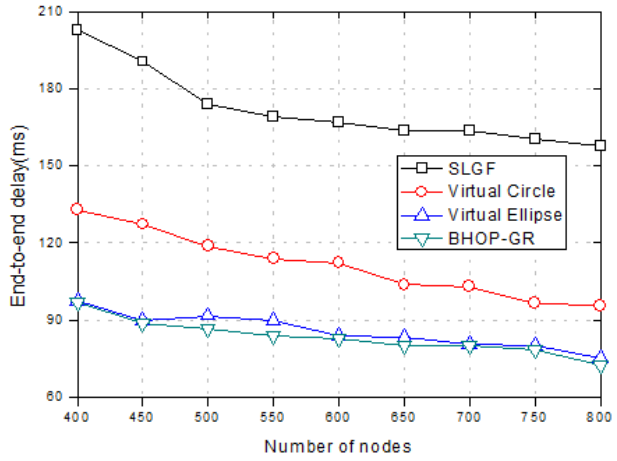
Fig. 8 shows the upper bound of the number of hops of the routing path for different numbers of nodes, under both IA and FA models. In Fig. 8(a), as we mentioned in the previous paragraph, BHOP-GR is similar to the virtual circle, and virtual ellipse in the maximum number of hops because the hole size is very small or the hole does not exist under the IA model.

However, when larger hole occur under the FA model, the routing protocols, including BHOP-GR, show different results compared to results under the IA model. The maximum number of hops of BHOP-GR is significantly lower than the maximum number of hops of SLGF and virtual circle and is slightly lower than that of the virtual ellipse. In WSNs, the packet is forwarded hop-by-hop along the path. Reducing the number of hops can reduce end-to-end delay and furthermore support quick responses to routing requests. The results show that the routing under our proposed protocol always requires the least number of hops in the detour compared to other protocols.

Fig. 9 shows the corresponding average length of the entire routing path. In Fig. 9(a), as we showed in the previous paragraph, BHOP-GR is similar to the virtual circle, and virtual ellipse in the

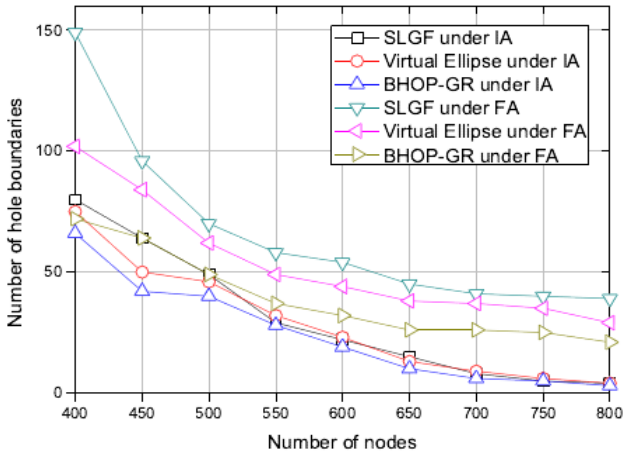


(a)

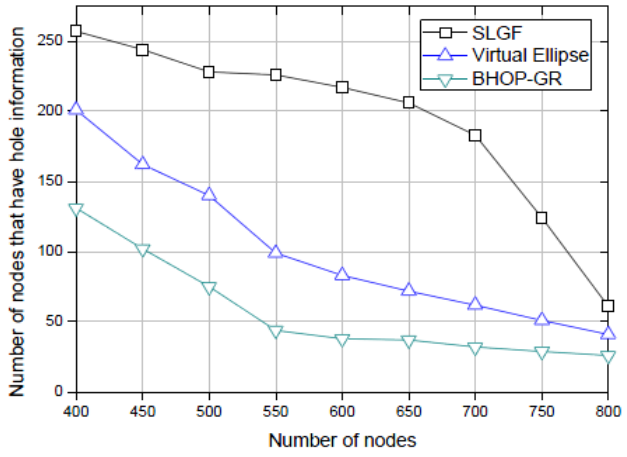


(b)

Figure 10. (a) The end-to-end delay under IA and FA (hole size 50m x 50m) model (b) The end-to-end delay under FA model with hole size 100m x 100m



(a)



(b)

Figure 11. (a) The number of node including hole information under IA and FA (hole size 50m x 50m) model (b) The number of node including hole information under FA model with hole size 100m x 100m

average length of the routing path, because the hole size is very small or the hole does not exist under the IA model. The average routing path is similar to the average number of hops, because the average routing path was affected by the average number of hops. Through Fig. 9, we show that the average length of the routing path increases when hole size increases. These results prove the routing under our proposed protocol can always achieve a shorter path and conserve more energy consumed in data transmission because BHOP-GR previously detects the hole boundary using an observer packet and transmits data packet using the virtual regular hexagon model which can be arranged to decrease routing path length.

Fig. 10 shows the end-to-end delay for BHOP-GR, virtual circle, virtual Ellipse, and SLGF, respectively, under both IA and FA models. The results of end-to-end delay are similar to Fig. 7, because the end-to-end delay is affected by the average hop counts. In Fig. 10(a), BHOP-GR dose not differ in performance compared with other routing protocols under the IA model. However, in Fig. 10(b), we found the performance of BHOP-GR under the FA model, whereby the end-to-end delay increases when hole size increases and the end-to-end delay decreases when the number of nodes increases because the hole size decreases. The end-to-end delay of BHOP-GR is significantly better than the end-to-end delay of the virtual circle and SLGF, and is slightly better than the virtual ellipse under the FA model, since BHOP-GR selects the optimum bypass path using the observer packet and time delay.

In this paragraph, we implement and compare the overhead to maintain hole boundary information of BHOP-GR to other routing protocols: SLGF and Virtual Ellipse. Fig. 11 shows the number of

nodes including hole information at a different number of nodes under both the IA and FA models. In Fig. 11(a), the results of BHOP-GR do not differ in the number of nodes including hole information compare to other routing protocols under the IA model. We show that the number of nodes that include hole information is increased, because hole size increases. We simulate implementation, with the exception of the virtual circle, because the hole detection procedure of BHOP-GR and the hole detection procedure of the virtual circle are the same. The number of nodes including hole information of BHOP-GR is significantly better than for SLGF and virtual ellipse under the FA model, since the virtual ellipse and SLGF have overhead about nodes that are adjacent to the hole that have to maintain the hole boundary information.

5. CONCLUSION

This paper propose a Bypassing Hole Scheme Using Observer Packets for Geographic Routing (BHOP-GR) to solve the local minimum problem. It selects the optimum bypass path and minimizes overhead to maintain hole boundary information. The BHOP-GR detects the hole boundary using an observer packet. When it transmits the data packet using a time delay, it can arrange to decrease the average hop count and routing path length. Also, using a virtual regular hexagon model which can exactly cover a hole to be detoured, it minimizes overhead to maintain hole boundary information. Simulations, with varying node densities, using two network deployment models demonstrated that BHOP-GR has reduced hop counts and routing path length and also minimizes overhead to maintain hole boundary information compared to the existing routing schemes. BHOP-GR is expected to be applied to the management of facilities and structures such as Bridge, intellectual farming, and environment monitoring. Each node in the WSN has limited battery resources. Therefore, our future study will focus on the ways of using the sensor node's energy more efficiently by considering the sensor duty cycle and power control.

6. ACKNOWLEDGMENTS

This research was supported in part by MKE and MEST, Korean government, under ITRC NIPA-2011-(C1090-1121-0008), NGICDP(2011-0020517) and PRCP(2011-0018397) through NRF, respectively.

7. REFERENCES

[1] M. Mauve, J. Widmer and H. Hartenstein, “A Survey on Position-based Routing in Mobile Ad Hoc Networks,” IEEE Network Magazine, vol. 15, No. 6, pp. 30–39, 2001.

[2] K. Seada and A. Helmy, “Geographic Protocols in Sensor Networks,” USC Technical Report, 2004.

[3] J. N. Al-Karaki and A. E. Kamal, “Routing Techniques in Wireless Sensor Networks: A Survey,” IEEE Wireless Communications, vol. 11, issue. 6, pp. 6-28, 2004.

[4] B. Karp and H. T. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks,” Proc. of the 6th annual international conference on Mobile computing and networking, pp. 243-254, 2000.

[5] Y. Ko and N. Vaidya, “Location-aided routing (LAR) in mobile ad hoc networks,” Wireless Networks, Vol. 6, No. 4, pp.307-321, 2000.

[6] N. Ahmed, S. Kanhere, and S. Jha, “The holes problem in wireless sensor networks: A survey,” ACM Sigmobile Mobile Computing and Communication Review, vol. 9, No. 2, pp. 4-18, 2005.

[7] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, “Geometric ad hoc routing: of theory and practice,” Proc. of ACM Symposium on the Principles of Distributed Computing (PODC), pp. 243–254, 2003.

[8] C. Chang, K. Shih, S. Lee, and S. Chang, “RGP: Active route guiding protocol for wireless sensor networks with obstacles,” Proc. of the 3rd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS’06), pp. 367-376, 2006.

[9] K. Liu, N. Abu-Ghazaleh, and K. Kang, “Location verification and trust management for resilient geographic routing,” Journal of Parallel and Distributed Computing, vol. 67, No. 2, pp. 215-228, 2007.

[10] S. Olariu and I. Stojmenovic, “Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting,” Proc. of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM’06), pp. 1-12, 2006.

[11] X.Wu, G. Chen, and S. Das, “On the energy hole problem of nonuniform node distribution in wireless sensor networks,” Proc. of the 3rd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS’06), pp. 180-187, 2006.

[12] Z. Jiang, J. Ma, W. Lou and J. Wu, “An information model for geographic greedy forwarding in wireless ad-hoc sensor networks,” Proc. of the 27th Annual Joint Conference of the IEEE Computer and Communication Societies (IEEE INFOCOM'08), pp. 825-833, 2008.

[13] Y. Tian, F. Yu, Y. Choi, S. Park, E. Lee, M. Jin, and S. Kim, "Energy-Efficient Data Dissemination Protocol for Detouring Routing Holes in Wireless Sensor Networks," Proc. of IEEE International Conference on Communications, pp. 2322-2326, 2008.

[14] F. Yu, S. Park, Y. Tian, M. Jin, and S. Kim, “Efficient Hole Detour Scheme for Geographic Routing in Wireless Sensor Networks,” Proc. of the 68th Annual IEEE Vehicular Technology Conference, pp. 153-157, 2008.

[15] Q. Fang, J. Gao, and L. Guibas, “Locating and bypassing routing holes in sensor networks,” Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM’04), pp. 2458-2468, 2004.

[16] B. H. Wellenhof, H. Lichtenegger, J. Collins, "Global Positioning System: Theory and Practice," fifth ed., Springer Verlag, 2001.

[17] J. Li, J. Jannotti, D.S.J.D. Couto, D. R. Karger and R. Morris, “A scalable location service for geographic ad-hoc routing,” Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom’03), pp. 120-130, 2003.

[18] A. Savvides, C. Han and M. Strivastava, “Dynamic fine-grained localization in ad-hoc networks of sensors,” Proc. of

the 6th Annual International Conference on Mobile Computing and Networking (MobiCom'01), pp. 166-179, 2001.

[19] I. F. Akylidiz, W. Su, Y. Sankarasubramanizm, and E. Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine, vol. 40, issue. 8, pp. 102-116, 2002.

[20] M. I. Brownfield, "Energy-efficient Wireless Sensor Network MAC Protocol," 2006.

[21] Chipcon. CC2420 Data Sheet. http://www.chipcon.com/.

[22] B. Latre, P. D. Mil, I. Moerman, B. Dhoedt, and P. Demeester, "Throughput and Delay Analysis of Unslotted IEEE 802.15.4," Journal of Networks, vol. 1, issue. 1, pp. 20-28, 2006.

ABOUT THE AUTHORS:

Hyunseung Choo received the MS in Computer Science from the University of Texas at Dallas and the Ph.D. from the University of Texas at Arlington, in 1990 and 1996, respectively. Since 1998, he has been with the School of Information and Communication Engineering, Sungkyunkwan University (Korea) and is now Professor and Director of the Convergence Research Institute. Since 2005, he has been the Director of the Intelligent HCI Convergence Research Center supported by the Ministry of Knowledge Economy (Korea). He has published over 300 papers in international journals and refereed conferences. His research interests include embedded networking, mobile computing, and clouds. Dr. Choo has been Editor-in-Chief of the Journal of Korean Society for Internet Information for three years and Journal Editor of Journal of Communications and Networks, ACM Transactions on Internet Technology, Journal of Supercomputing, and Founding Editor of Transactions on Internet and Information Systems since 2010. He is a member of the ACM, IEEE, and IEICE.

Moonshik Choi received the B.S degree in Computer Engineering from Ajou University and received the M.S. degree in Information and Communication Engineering from Sungkyunkwan University, Korea, in 2011. He is currently working as an engineer in LSIS, Korea. His research interests include wireless sensor networks, routing protocol, mobile computing, and wireless communication.

Minhan Shon received the M.S. degree in Computer Science from the Sungkyunkwan University(SKKU) in 2004. He is a Ph.D. candidate in Computer Science at SKKU. He is currently working at Intelligent HCI Convergence Research Center. His research interests include clustering, routing, localization, and duty-cycle algorithms in wireless sensor networks.

Dongsoo S. Kim received the M.S. degree in Computer Science from the University of Texas at Dallas, TX, USA, in 1994, and the Ph.D. degree in Computer Science and Engineering from the University of Minnesota, Minneapolis, MN, USA, in 1998. Dr. Kim worked as a research scientist for Electronics and Telecommunications Research Institute from 1986 to 1992, and as a project manager for Megaxess Inc. from 1998 to 2000. In 2000, he joined the Department of Electrical and Computer Engineering, Indiana Univ. Purdue Univ. Indianapolis, USA. He is currently an Associate Professor of the Department and a Director of Center for Sensor and Ubiquitous Networking, IUPUI. His research includes switch networks, optical switches, network survivability, protection switching, network planning, QoS provisioning in the Internet, mobile ad-hoc networks, mobility modeling, sensor networks, and power-aware routing.

Protein Secondary Structures Prediction based on Evolutionary Computation

Alfonso E. Márquez Chamorro
School of Engineering
Pablo de Olavide University
Seville, Spain
amarcha@upo.es

Federico Divina
School of Engineering
Pablo de Olavide University
Seville, Spain
fdiv@upo.es

Jesús S. Aguilar-Ruiz
School of Engineering
Pablo de Olavide University
Seville, Spain
aguilar@upo.es

ABSTRACT

In this paper we propose an approach based on evolutionary computation for the prediction of secondary protein structure motifs. The prediction model consists of a set of rules that predict both the beginning and the end of the regions corresponding to a secondary structure state conformation (α-helix or β-strand). The prediction is based on a set of specific amino acid physical-chemical properties. In addition we also propose a statistical study regarding the propensities of each pair of amino acids in capping regions of α-helix and β-strand. Experimental results confirm the validity of our proposal.

Keywords

Protein Secondary Structure Prediction, α-helix, β-strand, β-sheet, Evolutionary Computation.

1. INTRODUCTION

The Protein Secondary Structure Prediction (PSSP) consists in predicting the location of α-helices, β-sheets and turns within a sequence of amino acids without any knowledge of the tertiary structure of the protein.

PSSP has received much attention lately, since knowledge of the location of the elements in secondary structure could be used by approximation algorithms to obtain the tertiary structure of the protein. Being able to predict, from the amino acid sequence, how a protein will fold, is one of the main open problems in computational biology, and have important applications, e.g., in the development of new drugs.

Repetitive motifs appear in a secondary structure, and the most common kind of motif is α-helices. An α-helix is a dextro-helical structure, with about 3.6 amino acids per turn. Such structure is held together by hydrogen bonds. In particular the amino group of amino acid n provides a hydrogen bond with the carbonyl group of the amino acid n + 4. Another common structure is represented by the β-sheet. β-sheets are characterized by theirs flattened and extended shape, and have a maximum number of hydrogen bonds between peptides that provide stability to the structure. Several peptide chains (β-strands), which are held together with hydrogen bonds in a zig-zag, constitute a β-sheet motif. The lamellar structure formed proportionate flexibility but no elasticity. The adjacent chains of a β-sheet can be targeted in the same direction (parallel β-sheet) or opposite direction (antiparallel β-sheet).

Several methods were applied to the PSSP problem. These methods can be divided into two categories: statistical and soft computing approaches. Statistical methods are based on the calculation of amino acid probabilities to belong to a given secondary structure motif [5, 12, 17]. On the other hand, soft computing methods provide processing capabilities that can be used in order to solve the problem of PSSP. Such methods are characterized by the fact that they are tolerant of imprecision, uncertainty, partial truth, and approximation. The most popular soft computing paradigms applied to PSSP are: artificial neural networks (ANNs) [19, 18, 8], nearest neighbors [11, 21] and support vector machines (SVMs) [23, 4]. Some soft computing methods used in this problem are focused on determining contact maps (distances) between amino acids residues of a protein sequence. When a contact map is defined, proteins can be fold and the tertiary structure can be determined.

In this paper, we propose a strategy based on evolutionary computation, to predict α-helices and β-sheets from sequences of amino acids. Evolutionary algorithms (EAs) are adaptive methods that can be used to solve optimization problems. We believe that EAs are good candidate for tackling this problem. In fact, PSSP can be seen as a search problem, where the search space is represented by all the possible folding rules. Such a space is very complex, and has huge size. EAs have proven to be particularly good in this kind of domains, due to their search ability and their capability of escaping from local optima.

In our proposal, the prediction is made ab initio, i.e., without any known protein structure as a starting template for the search. The prediction model will consist of rules that predict both the beginning and the end of the regions corresponding to a α-helix or a β-strand. With this we want to overcome the limitation of existing methods that typically fail at predicting motifs boundaries [25]. In particular, β-sheet determination is more difficult to predict than α-helix [9].

Other methods based on evolutionary computation have been applied to secondary structure prediction. For instance, in [6], an EA using a torsion angle representation was proposed, and [22] introduced an approach based on lattice models.

In this paper, we also propose a statistical analysis of each pair of amino acids that are located in the beginning and the end of the α-helix or β-sheets sequences. From this study, we can extract information that is useful in order to predict secondary structures.

The rest of paper is organized as follow. In the next section we propose the statistical analysis of the sequences used in this paper. In section 3, we discuss our proposal to predict protein secondary structure motifs. Section 4 provides the experimentation and the obtained results. Finally, in the last section, we draw some conclusions and analyze possible future works.

2. STATISTICAL ANALYSIS

As already mentioned, the aim of this paper is to propose a method for identifying the beginning and the end of motifs in sequence of amino acids. Figure 1 shows an example of a sub-sequence of amino acids relative to a secondary structure, e.g., an α-helix. Each amino acid in the sequence is identified by its position, and amino acids between N1 and C1 form the α-helix. It follows that amino acids in positions N-cap and C-cap are those that immediately precede or follow the beginning or the end of the structure, respectively.

. . .	N_{CAP}	N_1	N_2	N_3	...	C_3	C_2	C_1	C_{CAP}	. . .

α-helix

Figure 1: Relevant positions in an α-helix and a β- strand.

The analysis proposed in this section is aimed at studying the different propensities of each pair of amino acids in the studied positions, i.e, N-cap, N1, C-cap and C1. Knowing the propensities of each pair of amino acids at these positions would allow us to extract useful information about the properties of those amino acid that are located in the beginnings or ends of the different secondary structure motifs.

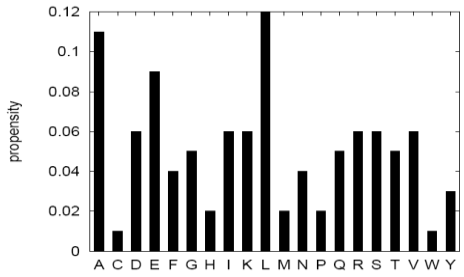


Figure 2: Amino acid propensities in helix sequences.

The data set used in this paper includes 163,461 α-helix and 216,390 β-strand sequences with a total of 2,177,854 and 1,606,246 amino acids respectively. These sequences were extracted using the DSSP program [15] from 12,860 non-redundant protein sequences taken from PDB and sharing less than 30% sequence identity. To the best of our knowledge, no other approaches have used such a high number of secondary structure states sequences for a similar study. Before the cited analysis, we have also performed several studies to analyze certain aspects of the data set. Figure 2 show a chart with the propensities for each amino acid to belong to an α-helix. It can be noticed that A (Alanine) and L (Leucine) show the higher probabilities. It is worth mentioning that these two amino acids are nonpolar and have neutral charge.

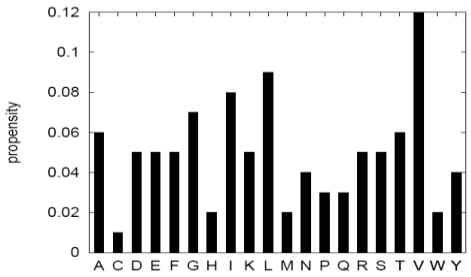


Figure 3: Amino acid propensities in beta sequences.

On the other hand, Figure 3 proposes a graph relative to the amino acid probabilities of belonging to a β-strand. In this case, V (Valine), I (Isoleucine) and L (Leucine) have a probability of 12, 9 and 8%, respectively. It is interesting to notice that also in this case these amino acids have two characteristics in common, nonpolarity and neutral charge. Figure 4 proposes a diagram that represents the ratios aimed at studying the most frequent length of an α-helix sequence. We can observe that the most common length is 6, and that after this peak the graph almost follows a normal distribution with center 13. From this study we have discovered that the average size of α-helix sequences is 13.46 residues. Figure 5 shows a diagram with the representation of the proportions for each size of β-strand sequences. Also in this case the graphs follows a normal distribution, were the average size of β-strands sequences is 7.49 residues.

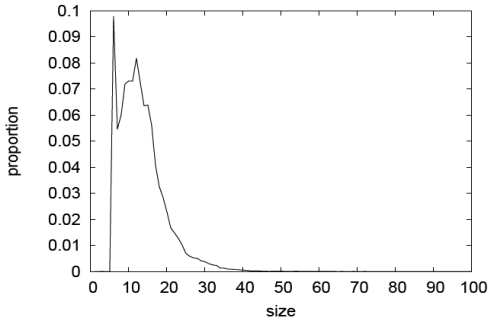


Figure 4: Representation of alpha helix sequences sizes.

Table 1 gives us more insights on propensities of each amino acid to belong to either N-cap or C-cap positions in α-helix and β-strand sequences. These probabilities are computed over the total of appearances of each amino acid. For the α-helix sequences, amino acids D, N, P and S (polar amino acids), present the higher probabilities in the case of N-Cap, while for the for C-cap position G (Glycine) is by far the most probable. As far as β-strand sequences are concerned, amino acids D, G and P (small amino acids), have the highest probabilities to appear in N -Cap positions. Amino acids D, G and N show the higher propensities for C-cap position.

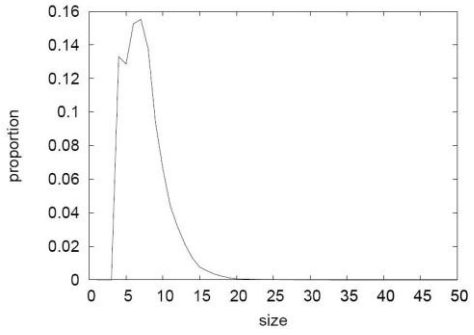


Figure 5: Representation of beta strand sequence sizes.

Table 1: Amino acid propensities for C-cap an N-cap positions in alpha and beta sequences.

	Alpha-helix			Beta-strands	
AA	***NCap***	***CCap***	***AA***	***NCap***	***CCap***
A	0,35	0,85	A	0,98	1,07
C	1,19	1,36	C	0,74	1,04
D	**2,71**	0,80	D	**1,90**	**1,87**
E	0,48	0,64	E	1,13	1,06
F	0,50	1,04	F	0,56	0,61
G	2,00	**3,46**	G	**1,87**	**1,71**
H	1,35	1,46	H	0,99	1,03
I	0,29	0,51	I	0,39	0,49
K	0,48	1,00	K	1,37	0,98
L	0,31	0,82	L	0,51	0,83
M	0,45	0,94	M	0,94	0,87
N	**2,40**	1,78	N	1,72	**1,85**
P	**2,95**	0,00	P	**2,70**	1,45
Q	0,49	0,99	Q	1,18	0,88
R	0,53	0,95	R	1,14	0,88
S	**2,68**	1,25	S	1,04	1,32
T	**2,46**	0,74	T	0,84	1,06
V	0,33	0,56	V	0,40	0,50
W	0,50	0,58	W	0,74	0,61
Y	0,62	1,01	Y	0,63	0,64

We have computed the global propensities for each pair of amino acids in N-cap, N1 positions and C1, C-cap positions (start and end of either a helix or a sheet). In this analysis, we have used the following formula:

$$P^{g}_{X_i Y_{i+1}} = \frac{n^{helix}_{X_i Y_{i+1}}}{\sum_{AB} n^{helix}_{A_i B_{i+1}}} \Big/ \frac{n^{total}_{XY}}{\sum_{AB} n^{total}_{AB}}$$

where $P_{Xi\,Yi+1}$ is the global propensity for a XY amino acid pair, $X_i\,Y_{i+1}$ represents a pair of amino acids in a helix or sheet capping position, and $A_i\,B_{i+1}$ represents a pair of amino acids in any consecutive position in a protein sequence. This equation computes the relative frequency of the amino acid pair XY at positions i, i+1 (N-cap, N1 or C1, C-cap) in helices or strands and the relative frequency of the pair in the total protein set. In order to analyze the propensity of a pair of amino acids to be in a certain position of either a helix or a strand, we used equation 1 and built four propensity matrices, shown in Figures 6, 7, 8 and 9. In these matrices, rows represent the N-cap or C-cap position and columns are relative to either the N1 or the C1 position. Each propensity is represented by a different color. A cell with blue color represents a high likelihood for this pair. On the other hand, a red cell represents a low propensity.

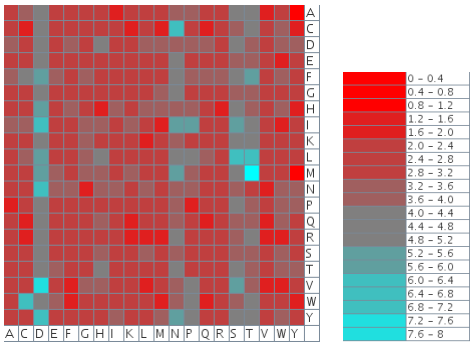


Figure 6: Propensity matrix for N-cap, N1 helix positions.

Figure 6 shows the propensity matrix for N-cap, N1 positions of the helices. From the analysis of this matrix, we can conclude that amino acids D, N ,S and T (Aspartic acid, Asparagine, Serine and Threonine, respectively) are the most likely to occupy the N1 of a helix, while there are no specific amino acids for the N-cap position. This means that every amino acid could hold such a position. By further analyzing these amino acids, we have observed that all these amino acids have a polar side chain. Moreover, the most frequent pair in an α-helix start is M, T (Methionine and Threonine).

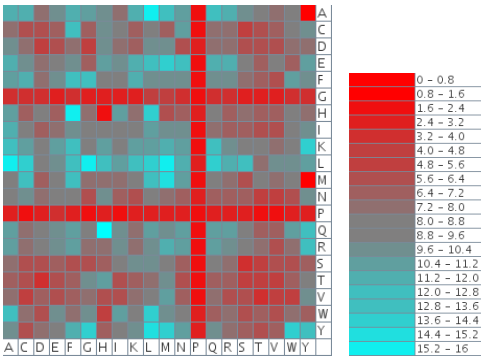


Figure 7: Propensity matrix for N-cap, N1 helix positions.

Figure 7 shows the propensity matrix for C1, C-cap positions of the helices. In this case the amino acid P (Proline) has the lowest propensity in both C1 and C-cap positions and the G amino acid (Glycine) at C1 position. The most frequent pair in an α-helix end is Q, H (Glutamine and Histidine).

Figure 8 shows the matrix for N cap, N1 positions of β-strands. From the analysis of the matrix, we can conclude that the amino acids that are most likely to appear in N1 position are G, P, N and D (Glycine, Proline, Asparagine and Aspartic acid respectively). Also in this case, these amino acids show a common characteristic: they have a small residue. Also in this case no specific conclusion can be drawn for the N-cap position. The most frequent pair in a β-strand start is P, V (Proline and Valine).

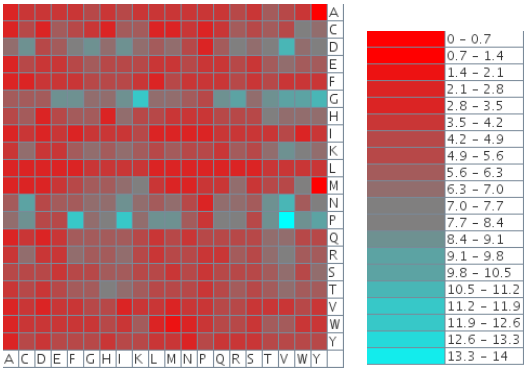


Figure 8: Propensity matrix for Ncap-N1 strand positions.

Figure 9 report the propensity matrix for C1, C-cap positions of a β-strand. In this case the amino acids N, D and P (Proline) have the lowest propensity to be in position C1, while amino acids I, V and Y (Isoleucine, Valine and Tyrosine respectively) shows the highest propensity for this position. In this case, these amino acids are all hydrophobic residues. No particular conclusion can be derived for the C-cap position. The most frequent pair in a β-strand end is Y, C (Tyrosine and Cysteine).

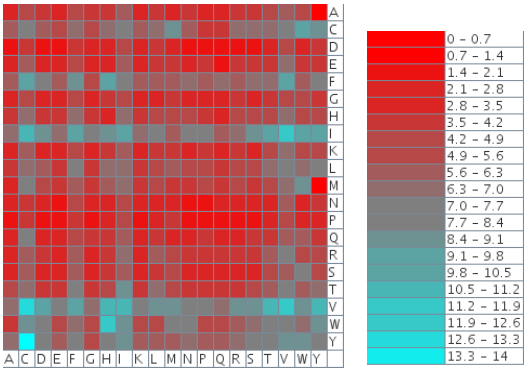


Figure 9: Propensity matrix for C1-Ccap strand positions.

3. MATERIALS AND METHODS

In this section, we present our proposal for the identification of α-helices and β-strands within a sequence of amino acids. α-helix and β-strand are a subsequence of amino acids.

The aim of this paper is to propose an EA capable of finding a set of rules that can be used in order to predict whether or not a particular amino acid lies in either position N-cap or C-cap, for both α-helices and β-sheets. With such a predicting model, we could then precisely identify the beginning and the end of the considered secondary structures.

The experimental procedure followed in this paper is shown in Figure 10. During the data acquisition stage, the α-helix and β-strand sequences are extracted from the Protein Data Bank (PDB) [3], as described in section 2. Later, these sequences are used for training our evolutionary algorithm, which will generate a set of rules representing the predictive model. As already mentioned, these rules establish if a particular amino acid is relative to either a N-cap or a C-cap position, so if the amino acid precede or follow the begin or the end of a structure. In order to test our algorithm, we apply these rules to a set of known protein sequences, thus used as test set.

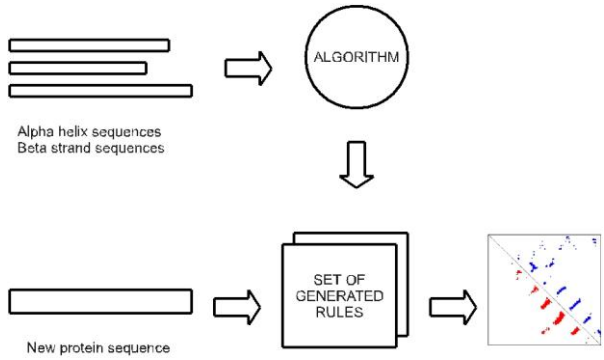


Figure 10: Experimental and prediction procedure.

3.1 Encoding

As already stated, the prediction model proposed by our algorithm will be based on a set of amino acid properties. In particular, we consider three properties:

Hydrophobicity For this property, we use Kyte-Doolitle hydropathy profile [14] for the hydrophobicity representation. In this way the hydrophobicity spectrum is discretized into a set of well defined intervals.

Polarity The Grantham's profile [13] was used for the polarity representation.

Charge Klein's scale [16] for net charge codification. We represent an amino acid with negative charge, according the Klein's scale, with 1, a positive charge with 1 and a neutral charge with 0.

Notice that the profile values of each amino acid are normalized to a range of between 1 and 1 for hydrophobicity and polarity.

Each individual of the population represents a rule, and will consist of window of two amino acids. For each amino acid, the three above mentioned properties will be represented. An individual may represent either the beginning or the end of an α-helix or a β-sheet (N-cap, N1 or C1, C-cap positions).

Figure 11 proposes an example of individual, where positions P_1, P_2, P_1' and P_2' represent the hydrophobicity ranges of the first and second amino acid of the window, respectively. P_1 and P_2 are real numbers which determine a hydrophobicity range for the amino acid in that position. Positions P_3, P_4, P_3' and P_4' represent the polarity intervals according to Grant scale of the first and second amino acid respectively. Also in these cases, these values are real numbers, which determine a polarity range for the amino acid in that position. Positions P_5 and P_5' represent the net charge property values of the two amino acids. These two positions may assume three different integer values: 1 for a negative charge, 0 for neutral charge and 1 for a positive charge.

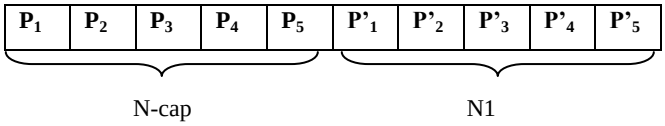


Figure 11: Example of chromosome codification for a beginning of an α-helix or a β-strand.

3.2 Fitness Function

The aim of the algorithm is to find both general and precise rules for identifying helices and sheets. To this aim, we have chosen as fitness of individuals the F-measure, which is given by the following formula:

$$F = 2\frac{Recall \cdot Precision}{Recall + Precision}$$

The higher the fitness, the better the individual. Recall represents the proportion of training examples that matches this rule. Precision represents the error rate.

In the literature, it has been proved that α-helices and β-sheets are characterized by some properties of the amino acids in positions N-cap, N1 or C1, C-cap. In order to increase the effectiveness of our proposal, we consider some of these properties. In particular, in [20] it has been demonstrated that there are molecules with asymmetrical distributions of charge in the limits of an α-helix. This means that the residues in limits of the helix are polar. Results obtained by our algorithm confirm this observation. Moreover, in [7, 10], it has been proven that many helices present a positive charge in its last turn and a negative charge at its first turn.

On the other hand, we also consider some specifications for the β β-sheet capping prediction. It has been demonstrated that hydrophobic amino acids have a high propensity to be at N1 and C1 positions (especially V, I, Y and W) in a β-sheet. In addition, many strands present a negative charge in C-cap and a positive charge in N-cap positions [9].

We increase the score of those individuals that fulfill one requirement in a 50%, and in a 100% for those individuals that present the two properties.

3.3 Genetic Operators

Individuals are selected with a roulette wheel mechanism. A roulette wheel is built, where the sector associated to each individual of the population is proportional its fitness. Individuals with higher fitness have more probability of being selected, since their sector is wider.

Elitism is also applied, i.e., the best individual always survives to the next generation.

Uniform crossover is used in order to generate offsprings. Crossover is applied with a 1.0 probability. All the offsprings are obtained by crossover except the one with best score which was copied without any change (elitism). Mutation is applied with a probability of 0.5. If mutation is applied, one gene of the individual is randomly selected, and its value is increased or decreased by 0.01. If the selected gene is relative to the charge of the amino acid, then its value is randomly changed to one of the other two allowed possibilities. After that an individual has been mutated, it is checked for validity, i.e., its values are within the ranges allowed for each property: [1, 1] for hydrophobicity and polarity and 1, 0 or 1 for the net charge. If the encoded rule is not valid, then the mutation is discarded.

The initial population is randomly initiated. For the experiments proposed in this paper, the population size is set to 100. After having evaluated the initial population, the first generation is created. If the fitness of the best individual does not increase over twenty generation, the algorithm is stopped and a solution is provided.

We evolve four populations separately: one population contains individuals that encode rules identifying the beginning of an α-helix, a second population contains individuals representing rules identifying the end of the helix. A third population contains individuals that encode rules identifying the beginning of a β-sheet, and the last population contains individuals representing rules for the end of a β-sheet. At the end of the evolutionary process, the best individuals from each population are extracted, and together they form the proposed solution.

In the following we outline the main solution adopted for the EA proposed. In particular, we discuss the various solutions concerning the fitness, the representation and the genetic operators used.

4. EXPERIMENTS AND DISCUSSION

In this section, we present the experimentation performed in order to assess the validity of our proposal. The prediction of protein secondary structure is obtained from amino acid sequences. For this reason, we need to obtain a set of known protein sequences. We obtain the sequences from the PDB site [3], where the information regarding secondary structures is also provided. However this information will be used only for testing our predicting model.

As explained in section 3, a set of 12,830 non-homologous and non-redundant proteins with a homology lower than 30% were obtained from PDB, using the PDB Advanced Search [2]. We have only selected the structures which contain protein chains and not DNA or RNA chains. The complete list of the 12,830 PDB protein identifiers can be downloaded in [1]. The DSSP program [15] was used in order to extract the secondary structure relative to α-helix and beta-sheet states of each protein based on the atomic coordinates in the PDB file. Once we have located the motifs in the protein sequence, we extract the amino acids from N-cap to C-cap positions of the helix or sheet (figure 1), which are the amino acids that are in relevant positions in an α-helix or beta-sheet. From these sequences, we have randomly selected a subset of 5,000 α-helix and 5,000 β-strand sequences with a minimum size of four residues. These sequences were extracted from a subset of proteins sequences with length less than 150 residues. Coils and no-motifs protein sequences are included as negative examples.

In order to validate the obtained results, a 10-fold cross-validation has been applied. The data set is divided into 10 subsets, and the holdout method is repeated 10 times. Each time, one of the 10 subsets is used as the test set and the other 9 subsets are put together to form a training set. Then the average result across all 10 trials is computed. A model is obtained for each fold. This model consists of a set of rules that identify beginnings and ends of an α-helix or of a β-strand respectively.

For each fold, we compute the following measures:

Recall represents the percentage of correctly identified positive cases. In our case, Recall indicates what percentage of beginnings or ends of motifs have been correctly identified.

Precision is a measure to evaluate the false positive rate. Precision reflects the number of real predicted examples.

Specificity, or True Negative Rate, measures the percentage of correctly identified negative cases. In this case, Specificity reflects what percentage of no beginnings or ends of motifs have been correctly identified.

Accuracy is the proportion of true results in the population.

The optimal number of rules necessary for the prediction is unknown. For this reason, we performed experiments with a different number of iterations of the algorithm, more specifically from 10 to 40. Notice that after each iteration a set of rules is provided. The more iterations of the algorithm, the more rules will be incorporated in the final prediction model.

In the experiments proposed in this paper, we used the following parameters for the EA. The population size is set to 100. Crossover and mutation probabilities are set to 1.0 and 0.5, respectively. The maximum number generations is set to 100. These parameters were established after a set of preliminary tests.

Table 2 and Table 3 show the obtained results for the helix capping prediction algorithms (starts and ends of helices). In particular, the first column provide the number of iterations of the algorithm, and the rest of the columns report the average recall, specificity, precision and accuracy. Standard deviation is also reported. It can be noticed that for the α-helix capping prediction, the algorithm obtained extremely high accuracy, with an average of 0.99. The average recall is about 0.64, in C-cap and about 0.62 in N-cap prediction. Precision shows a low rate of error in the prediction with an average of about 0.69. All the measures vary weakly depending on the number of iterations. The results become more or less stable after 20 executions for all the measures. Previous works achieve an average recall of 30 38% in N-cap prediction [24]. Our approach improved these results.

Table 2: Average results for the prediction of the beginning of α-helices obtained for different number of iterations. Standard deviation is reported between brackets.

It.	*Recall μ±σ*	*Spec. μ±σ*	*Prec. μ±σ*	*Accuracy*
10	0.604±0:103	0.993±0.001	0.693±0.024	0.992±0.002
20	0.635±0.096	0.991±0.002	0.687±0.023	0.993±0.002
30	0.638±0.066	0.993±0.000	0.692±0.012	0.992±0.001
40	0.623 ±0.055	0.995±0.000	0.732±0.010	0.992±0.001

Table 3: Average results for the prediction of the end of α-helices obtained for different number of iterations. Standard deviation is reported between brackets.

It.	*Recall μ±σ*	*Spec. μ±σ*	*Prec. μ±σ*	*Accuracy*
10	0.633±0:160	0.993±0.002	0.665±0.022	0.992±0.003
20	0.643±0:196	0.993±0.003	0.694±0.049	0.993±0.004
30	0.656±0.066	0.992±0.002	0.668±0.031	0.992±0.003
40	0.634 ±0:101	0.992±0.001	0.640±0.013	0.992±0.002

Results relative to the prediction of β-strands capping are reported in table 4 and table 5. These results are slightly less accurate than those relative to the helix prediction. The main difference can be noticed in the results obtained for the N-cap and C-cap recall, with an average recall of about 0.18 in N-cap, and about 0.52 in C-cap prediction. So, in the case of the N-cap prediction, the result is much lower than in the case of N-cap prediction of α-helices. The precision is about 0.59, in N-cap and about 0.68 in C-cap prediction. High levels of accuracy and specificity are shown in both cases. Unlike α-helices [24], to the best of our knowledge, there are not previous results reported in the literature for the β-sheet capping prediction.

Table 4: Average results for the prediction of the beginning of β-strands obtained for different number of iterations. Standard deviation is reported between brackets.

It.	*Recall μ±σ*	*Spec. μ±σ*	*Prec. μ±σ*	*Accuracy*
10	0.151±0.083	0.995±0.001	0.671±0.039	0.978±0.002
20	0.163±0.040	0.988±0.001	0.596±0.018	0.996±0.001
30	0.198±0.015	0.994±0.000	0.551±0.019	0.975±0.000
40	0.187 ±0.055	0.995±0.001	0.565±0.044	0.975±0.002

Table 5: Average results for the prediction of the end of β-strands obtained for different number of iterations. Standard deviation is reported between brackets.

It.	*Recall μ±σ*	*Spec. μ±σ*	*Prec. μ±σ*	*Accuracy*
10	0.489±0.101	0.990±0.001	0.615±0.014	0.984±0.003
20	0.524±0.040	0.991±0.001	0.663±0.008	0.988±0.001
30	0.516±0.014	0.994±0.000	0.760±0.019	0.985±0.000
40	0.586 ±0.059	0.993±0.001	0.728±0.019	0.987±0.002

Figure 12 shows an example of rule discovered by our algorithm. In particular this rule is relative to the beginning of an α-helix. If we inspect this rule, we can see that the hydrophobicity value for the amino acid in the N-cap position is between 0.52 and 0.92, the polarity value lies between 1.0 and 0.93 and neutral charge (0.0). Therefore, this amino acid could be L (Lysine) or F (Phenylalanine), which fulfills these features according to the cited scales. As it can be noticed, the rules that compose the prediction model provided by our algorithm are easily interpretable. We believe that this is an important factor, since this would facilitate the interpretation of results by an expert in the field.

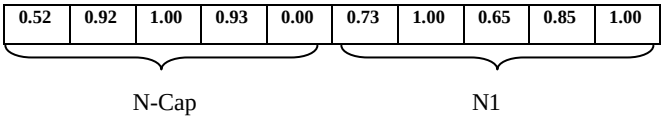


Figure 12: Example of a resulting rule for a beginning of an α-helix.

In conclusion, we can say that the proposed algorithm obtained satisfactory results. Moreover, the algorithm has been tested using a high number of sequences (5,000 helix and 5,000 strand sequences). We believe that this represent an important factor. In fact, the number of protein sequences available increase by the day, and thus, having a method that is scalable would be very important.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an evolutionary algorithm for α-helix and β-strand capping prediction from sequences of amino acid. The prediction is based on three amino acids properties, i.e., hydrophobicity, polarity and net charge. Moreover, some particular characteristic of these motifs are considered in order to improve the search process performed by the algorithm.

We have performed a statistical analysis aimed at discovering the amino acid propensities in capping positions in 163,461 α-helices and 216,390 β-sheets extracted from PDB using the DSSP program. We have computed the probability, for each pair of possible amino acids, to appear in both N-cap and N1 positions and C1, C-cap positions. This study provided us with useful information for the prediction of secondary structure. In fact, this information could be used for modifying the fitness function, improving in this way the evolutionary search. A study of each single amino acid has been also developed in each position. From this study, we could individuate which amino acid is more probable to appear in one of the positions taken into consideration.

In order to test the validity of the proposed algorithm, we performed a set of experiments using 5,000 α-helix and 5,000 β-strand sequences. These sequences were extracted from a protein data set from Protein Data Bank. In particular, we considered 12; 830 non-redundant and non-homologous protein with a homology rate lower than 30%. To the best of our knowledge, no other approaches have used such a high number of sequences in α-helix capping regions prediction. Results obtained on the prediction of α-helices are very encouraging and in particular, the accuracy characterizing the prediction models obtained is very high independently from the number of rules generated. As far as the experiments on the prediction of β-sheets, we have not found other results in the literature to contrast our results. However, also in this case, the accuracy obtained is satisfactory, even if the results are slightly worse than those obtained for the α-helices.

Future works will be focused on the analysis of different properties to be included in the fitness function, with the aim of increasing the quality of the prediction model. For example we are planning to incorporate the residue size, which has a significant relevance according to our statistical study. We will also expand the number of residues in the window of amino acids.

Furthermore, we are studying the possibility of incorporating a local search phase in the algorithm that will help to improve individuals. We also intend to extend our experimentation to other dataset of protein sequences.

6. REFERENCES

[1] Complete list of pdb protein identifiers used in this article. http://www.upo.es/eps/marquez/proteins.txt.

[2] Protein data bank advanced search http://www.pdb.org/pdb/search/advSearch.do.

[3] Protein data bank web. http://www.wwpdb.org.

[4] J. Cheng and P. Baldi. Improved residue contact prediction using support vector machines and a large feature set. Bioinformatics, 8, 113, 2007.

[5] P. Chou and G. Fasman. Prediction of protein conformation. Biochemistry, 13(2), 222–245, 1974.

[6] Y. Cui, R. Chen, and W. Hung. Protein folding simulation with genetic algorithm and supersecondary structure constraints. Proteins: Structure, Function and Genetics, 31, 247–257, 1998.

[7] Doig and B. R.L. N- and c-capping preferences for all 20 amino acids in alpha-helical peptides. Protein Science, 4(7), 1325–1336, 1995.

[8] P. Fariselli and R. Casadio. A neural network based predictor of residue contacts in proteins. Protein Engineering, 12, 15–21, 1999.

[9] F. Farzadfard, N. Gharaei, H. Pezeshk, and S. Marashi. Beta-sheet capping: Signals that initiate and terminate beta-sheet formation. J. Structural Biology, 161, 101–110, 2008.

[10] N. Fonseca, R. Camacho, and A. Magalhaes. Amino acid pairing at the n- and c-termini of helical segments in proteins. Proteins, 70, 188–196, 2007.

[11] D. Frishman and P. Argos. Incorporation of non-local interactions in protein secondary structure prediction from the amino acid sequence. Protein Engineering, 9, 133–142, 1996.

[12] J. Garnier, D. Osguthorpe, and B. Robson. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. J. Mol. Biol., 120, 97–120, 1978.

[13] R. Grantham. Amino acid difference formula to help explain protein evolution. J. J. Mol. Bio., 185, 862–864, 1974.

[14] K. J. and R. Doolittle. A simple method for displaying the hydropathic character of a protein. J. J. Mol. Bio., 157, 105–132, 1982.

[15] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers, 22(12), 2577–2637, 1983.

[16] P. Klein, M. Kanehisa, and C. DeLisi. Prediction of protein function from sequence properties: Discriminant analysis of a data base. Biochim. Biophys., 787, 221–226, 1984.

[17] V. Lim. Algorithms for prediction of a-helical and b-structural regions in globular proteins. J. Mol. Biol., 88, 857–872, 1974.

[18] L. McGullun, K. Bryson, and D. Jones. The psipred protein structure prediction server. Bioinformatics, 16, 404–405, 2000.

[19] N. Qian and T. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. J. Mol. Biol., 202, 865–884, 1988.

[20] J. Richardson and D. Richardson. Amino acid preferences for specific locations at the ends of alpha helices. Science, 240, 1648–1652, 1998.

[21] Salamov and V. Solovyev. Protein secondary structure prediction using local alignments. J. Mol. Biol., 268, 31–36, 1997.

[22] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. Biochim. Biophys., 231, 75–81, 1993.

[23] J Ward, L. McGullun, B. Buxton, and D. Jone. Secondary structure prediction with support vector machines. Bioinformatics, 13, 1650–1655, 2003.

[24] C. Wilson, P. Boardman, A. Doig, and S. Hubbard. Improved prediction for n-termini of alpha-helices using empirical information. Proteins, 57(2), 322–330, 2004.

[25] C. Wilson, S. Hubbard, and A. Doig. A critical assessment of the secondary structure prediction of alpha-helices and their n-termini in proteins. Protein Eng., 15, 545–554, 2002.

ABOUT THE AUTHORS:

Alfonso E. Márquez Chamorro received the degree in Computer Science and the MS degree in Computer Science and Telecommunications, from the Universidad Autónoma of Madrid, Spain, in 2007 and 2009 respectively. He is a PhD young researcher at the Universidad Pablo de Olavide, Sevilla, Spain from 2008. His research interests focus mainly on Machine Learning, Data Mining and Evolutionary Computation applied to Bioinformatics and Proteomics.

Federico Divina graduated in Computer Science at the Ca'Foscari University of Venice, Italy. In 2004 he received his PhD degree with a dissertation on the use of hybrid Evolutionary Computation for Inductive Logic Programming at the Department of Computer Science of the Free University of Amsterdam, The Netherlands. He was a post-doc at the university of Tilburg from 2004 to 2006, when then he moved to the Pablo de Olavide University where he is actually an assistant professor. His research interests include Evolutionary Computation, Machine Learning, Bioinformatics and Artificial Societies.

Jesus S. Aguilar–Ruiz received the B.Sc. degree in 1992, the M.Sc. degree in 1997, and the Ph.D. degree in 2001, all in computer science, from the University of Seville, Spain. He is an Associate Professor in Computer Science at Pablo de Olavide University, Seville, Spain. He has been member of the Programm Committee of several international conferences, and reviewer for relevant journals. His areas of research interest include evolutionary computation, data mining and bioinformatics. http://www.upo.es/eps/aguilar

Automatic authoring of interactive multimedia documents via media-oriented operators

Diogo Santana Martins, Didier Augusto Vega-Oliveros, and
Maria da Graça Campos Pimentel
Universidade de São Paulo
São Carlos, SP – Brazil
{diogosm, davo, mgp}@icmc.usp.br

ABSTRACT
When capturing multimedia records of collaborative activities (e.g. lectures, meetings, etc.), later access to the captured activity is usually provided by a linear video comprising the contents of the exchanged media. Such alternative impairs the review of the activity to the extent that the user only counts on the traditional timeline-based video controls. In scenarios in which automated tools generate interactive multimedia documents as a result of capturing an activity, the literature reports the use of ink-based and audio-based operators that allow the identification of points of interaction in the resulting document. In previous work, we defined a taxonomy of media-based operators in order to take into account user interactions with boards and videos, and extended the audio and ink-based operators with action-based alternatives. In this paper, the applicability of the operators is demonstrated in the context of the automatic generation of multimedia documents for interactive TV. Results from the user evaluation suggest that the operators provide means for faster review of a session when compared to a linear video.

Categories and Subject Descriptors
H.5.1 [**Information Interfaces and Presentations**]: Multimedia Information Systems – *Audio Video.*

General Terms
Documentation, Human Factors.

Keywords
Interactive Video, Document Engineering, Authoring.

1. INTRODUCTION
Capture and access (C&A) has been defined as "the task of preserving a record of some live experience that is then reviewed at some point in the future" [1]. A typical domain for C&A applications has been the recording and the review of collaborative work sessions such as distance education lectures and meetings. In those events, synchronous communication tools enable collaboration by exchanging text, images, documents, audio or video streams. In some scenarios, it is paramount that collaborative synchronous sessions be recorded for later review. Some of the most important reasons for reviewing a recorded session include keeping accurate records, revisiting portions of the session which were misunderstood, obtaining proofs and recalling certain ideas [12]. Particularly in the case of webconferencing tools, the approach usually adopted for recording a session is to generate a linear video with the content of the exchanged media. Such approach makes reviewing the session a linear and time consuming process, especially if the user only counts on traditional VCR-like operators (e.g. play, pause, forward, rewind) available in media players.

Researches in C&A systems [24] and smart meetings rooms [31] have been concerned with the development of better methods for reviewing sessions. These researches generally tackle the issue of developing indices so that users can jump to relevant points of interest within the session. Such non-linear access is mediated by specialized tools called browsers (commonly regarded as meeting browsers or meeting players). Features for building indices range from intentional user annotations to events derived directly from media elements via content-based analysis [19]. Browsers for recorded sessions are generally focused on specific types of indexed features, such as audio, video, discourse and artifacts [30].

Current approaches for developing meeting browsers have their drawbacks. These browsers are commonly built using ad-hoc development frameworks that entail tight coupling among the capture environment and the browser tool [5], thus reducing reuse of the same tool to access sessions gathered from different environments (e.g. meeting rooms and webconferencing systems). Moreover, these browsers are usually specialized toward one or two types of indices [26], impairing the richness of random-access operators that users can apply while reviewing the session.

In earlier research we reported the opportunity of exploiting operators which, by modeling the user interaction associated with pen-based devices such as electronic whiteboards, allow the review of the ink-based session as a document [9]. The approach was extended to allow browsers to be built through the automatic generation of an interactive multimedia document (iMMD) from the captured session involving multiple media [26]. The generated iMMD is enriched with several types of timestamped media-based operators — e.g. ink-based interactions [8] and audio events [25] — called Interactors as a generalization of "operators based on the interaction of a user with some media": the result is an interactive multimedia document with points of interest presented on a timeline, for instance. By means of document transformations, an XML-based interchange document is converted into an iMMD that synchronizes the streams of captured media and provides timeline-based access operations for browsing.

Inspired by the effectiveness of the *Interactors* [26] model, we presented the definition of new operators that take into account user interactions with boards, text messages and videos, and extended the audio and ink-based operators with action-based alternatives [29]. In this paper, we demonstrate an instantiation of the extended model considering the architectural and document engineering issues for generating interactive multimedia

documents in a declarative language (Nested Context Language (NCL) [23]) with procedural objects in Lua [13]. In order to evaluate the model and its instantiation, we conducted a user study considering data from a captured environment in use. The results of this experiment, which considered a timeframe of one month between recording and review, suggest that the documents generated by the operators can assist users in reviewing a session faster when compared to a linear document.

The remaining of this paper is organized as follows: Section 2 reports theoretical background on indexing and browsing of captured sessions. Related work is discussed in Section 2. Section 3 reviews the Interactors model. In Section 4 the model is instantiated for the automatic generation of multimedia documents for iTV. A proof-of-concept prototype is shown in Section 5. Evaluation results are reported in Section 6. Section 7 summarizes this paper's contributions and points out future research efforts.

2. INDEXING AND BROWSING OF CAPTURED SESSIONS

A classification of indices for captured media is proposed by Minneman et al. [19] who categorize indices into four broad classes: intentional annotations, performed explicitly by participants while the meeting is happening; side-effect indices, produced by capturing user-media or user-equipment interactions, such as slide changes and microphone activation; derived indices, automatically obtained by content-based analysis, for instance to identify speakers or detect moments of user-user interaction [17]; and post-hoc indices, consisting of user-media interactions performed during review of the meeting recordings. An extension of this taxonomy is proposed by Geyer et al. [15] who define online and offline indices, built during and after the meeting respectively; they define also explicit and derived indices, built by users and obtained from analysis of media elements, respectively. Similar developments have been proposed by Bouamrane and Luz [5] who formalize user-media interactions yielding indices at production-time or at consumption-time.

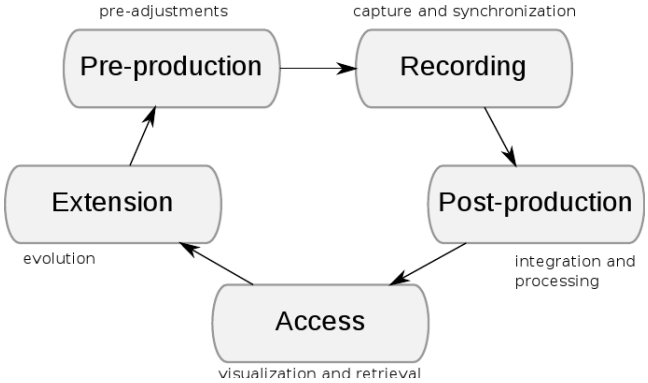


Figure 1: Lifecicle of captured media. Adapted from Abowd et al. [2] and Pimentel et al. [22]

From these categorizations we notice that central dimensions are the types of indexed interactions (user-media, user-equipment or user-user) and the moments at which these interactions take place. In regard to moments for capturing interactions, Abowd et al. [2] and later Pimentel et al. [22] propose a model of the lifecycle of captured media (Figure 1). Each phase of the lifecycle presents opportunities to build indices based on different types of interactions. In the pre-production phase offline intentional annotations can be applied to perform content segmentation, for instance. During recording, online annotations and side-effect indices are commonly built [6]. Offline indexing can occur in the post-production phase, such as the generation of derived indices [4]. Review-time explicit indices obtained from user-media interaction can be built while the user is accessing captured media elements (by means of annotations [21] or discrimination of moments [24], for instance). Such user-media interactions can be used to enrich and generate new versions of the original media elements [8] [9] in the extension phase.

Meeting browsers exploit indices to assist users through access interfaces to navigate recordings. Whittaker et al. [30] categorize meeting browsers according to the types of indices available. Audio-based browsers generally focus on events such as speaker turns, pause detection and emphasis; video browsers exploit keyframes and observable participant behavior; artifact browsers enable navigation by means of user-media interaction such as slide changes, ink-based notes and document sharing; and discourse-based browsers concentrate on navigation of speech transcripts, named entities and perceivable emotions. Tucker and Whittaker [26] point out several limitations of meeting browsers, among them impaired browsing on devices with limited resources (such as mobile devices and TV set-top boxes) and filtered presentation (browsing through summarized versions of the captured media).

When contextualized to research in indexing and browsing recordings, the Interactors model enables the definition of mechanisms to automatically generate document-based browsers by means of several types of indices: online side-effect indices (slide transitions and chat messages), intentional annotations (ink-based interactions), and offline derived indices (observable participant behavior from audio). Such indices are used to provide an interface focused on navigation of artifacts and audio events via timeline-based visualizations. Our document-centric approach tackles challenges for meeting browsers: in particular, the approach enables the efficient review of meeting recordings via a constrained device such as a TV set-top box. Moreover, adopting structured multimedia documents opens up benefits for content enrichment, filtering and sharing, as advocated by Cesar et al.[10].

3. RELATED WORK

Although multimedia researchers have identified several authoring paradigms [7], a recurrent limitation regarding research on meeting browsers has been the focus on navigation of a single type of medium (e.g. audio, video, whiteboards, etc.). For instance, Ehlen et al. [13] describe an interface for timeline-based meeting review and user feedback that concerns only navigation of speech transcripts via events of interest (e.g. decisions, minutes). Similar limitations are observable in video-only browsers: Yu et al. [32] present a system to navigate segmented video recordings through events of user-user interaction (e.g. propose, request information, acknowledge, etc.). Another video-only approach is reported by Behera et al. [4], which provides a browser to query a repository of SMIL documents generated from recordings of projected slides: using the SMIL browser, users can navigate segments of the retrieved documents. Video segmentation according to scene changes in projected slides is also tackled in TalkMiner [3] system, which performs post-hoc indexing of webcasts and provides a web-based interface to navigate the recordings. Indexing of whiteboard content is reported by Branhan et al. [6] who explore the opportunity of

organizing content captured from ambient collaborative boards according to contextual metadata (e.g. dates, locations, participants) collected with low effort in instrumented environments. Branhan et al. [6] also advocate for flexible interfaces that can be accessed in heterogeneous devices, which have already been experimented in collaborative scenarios promoting the social sharing of video [11]. In a complementary effort, Zsombori et al. [33] propose the use of video narratives techniques to support the automatic authoring of video from user generated content, and present a system that allows combining content captured with devices from (many) users. Mirri et al. [20] have also identified the opportunity for users to be able to provide alternatives to a multimedia document, and present a system which allows users to add alternative contents to the original multimedia.

In environments that can record multimodal information from heterogeneous media sources, for instance in the context of webconferencing systems and smart meetings rooms, it is desirable that the captured information can be accessed in an integrated manner, preferably with browsers that exploit cross-indexing mechanisms between the several streams of multimedia information and the metadata. This requirement is not fully tackled by researches that focus on a single type of medium (as reported before in the case of audio, video and boards), hindering their generalization to environments that coordinate groups of synchronous collaboration tools. In face of this limitation, we devise the opportunity of defining operators that provide a foundation for building multimodal browsers that orchestrate several streams of information in a common framework. In the next sections we delineate the core issues of the Interactors model designed with the purpose of tackling these requirements.

4. INTERACTORS MODEL

In this section we review and extend the Interactors model of operators to generate interactive multimedia documents (iMMD) from captured media [26] [29]. The model allows the development of browsers through automatic generation of an interactive multimedia document (iMMD) from the captured session involving multiple media [26].

DEFINITION 1 (INTERACTION EVENT). *An* interaction event *is a kind of metadata generated as a consequence of the many interactions among users, equipment and media that occur in a captured experience.*

Typical classes of interaction events that occur in a session are user-media interactions (e.g. slide changes, annotations, etc.), user-user interactions (e.g. turn taking in discussions, eye gaze, etc.) and user-equipment interactions (e.g. microphone activation, laser pointers, etc.). In the context of capture and access systems, when a live experience (e.g., a meeting) is captured, the data are recorded as temporal streams of multimedia information alongside their interaction events, all of them synchronized in a session that is the abstraction of a multimedia container.

DEFINITION 2 (SESSION). *A session is a non-empty set of media streams, synchronized and co-related, over which at least one kind of interaction event can be associated. A captured session describes j media elements and X_j logged interaction events per medium.*

We generalize the set of interaction events associated with a specific type of media as a media operator, as henceforth defined.

DEFINITION 3 (INTERACTOR). *An Interactor is an operator that is applicable to a specific type of medium and consists in a set with at least one interaction event. We define a set of interaction events as a mapping TL(a,b) where $a \leq j$ is a captured media element and b is a logged Interactor. By definition, each Interactor can map an unlimited number of interaction events. Thus for a specific media element i, the mapping TL(i,b), $b \leq 0$, represents the full set of interaction events for the media element i. Additionally, there is the constraint that an Interactor may not be applicable to every type of medium (e.g. the color attribute of an ink stroke is not applicable to audio)*

The taxonomy of Interactors defines four broad classes of features related to the user-media interaction [9] [26]:

a) time, given that each interaction event is timestamped to the start of a capture session;
b) attributes, related to media features collected during the capture, such as color and thickness (for ink strokes) as well as pitch and noise (for audio), and so on;
c) action, given that a user can perform several actions over the media element, such as drawing, erasing, muting, etc.;
d) position, which considers the boundary limits of the interaction event when occurred over a surface, e.g, cartesian coordinates of an ink stroke.

While experimenting with Interactors, we noticed that time, rather than being characterized as an isolated category, can be used in conjunction with the other categories of features as well. Based on this premise, the original set of Interactors has been extended in order to consider two new requirements:

i) the need for a list of time moments to be returned by an operator;
ii) the need to include the time interval in which the media are to be processed.

In this context, we define a time moment as an instant in the annotation session and a time interval as a segment delimited by two time moments. The following sections include the operators that have been defined to fulfill these requirements.

4.1 Inkteractors

Inkteractors are a special type of Interactors obtained by processing features of user-ink interactions — which is common when information is captured from meetings, classroom or museums, for instance, by means of pen-based devices such as electronic whiteboards or tablets. These operators can be applied to ink strokes so as to allow the generation and playback of documents containing alternative views of the original ink-based interaction process. In the listing below, the original specifications of some Inkteractors have been updated to tackle timing requirements.

Time-based. Used to filter or expand the media elements based solely on time constraints.

- TimeSlice(time *StartTime*, time *EndTime*): returns a list of the ink strokes generated in the specified time interval – this can be used to generate an image which aggregates the returned strokes, for instance.

Attribute-based. Considers attributes of pen strokes, such as color, thickness, type of stroke (free-form ink, geometric shape, etc.).

- ChangeOnAttribute(attribute *A*, time *StartTime*, time *EndTime*): returns a list of time moments when ink strokes were changed according to attribute *A* within the time interval defined by *StartTime* and *EndTime* – this can be used to generate an index to a timeline corresponding to a change in the color of the stroke, for instance;
- FilterByAttributeValue(attribute *A*, value *V*, time *StartTime*, time *EndTime*): returns a list of time moments when ink strokes were changed so that attribute *A* is equal to value *V*, within the time interval defined by *StartTime* and *EndTime* — this can be used, for instance, to generate an index to a timeline corresponding to the time moments when a specific color was used.

Action-based. While interacting with the capture system, a user can perform several actions over the pen strokes, such as drawing, erasing, changing color and so on. The history of such actions is kept together with the stroke representation, as well as with author who performed each one of them.

- *ChangeOnAuthor*(*time* StartTime,*time* EndTime): returns a list of time moments when there was a change in the author of the strokes, within the time interval defined by *StartTime* and *EndTime* – this can be used to generate an index to a timeline corresponding to a change in the author of a stroke, which may be of interest in distributed systems in which several users may draw strokes on a common surface;
- *FilterByAuthor*(*id* ID,*time* StartTime,*time* EndTime): returns a list of time moments when there was a change in the author of the strokes so that the author is identified by *ID*, within the given time interval — this can be used to generate an index to a timeline corresponding to the time moments when the author started producing strokes after someone else.

Position-based. Drawing surface and strokes are represented as a set of points in cartesian coordinates. Boundary limits, i.e., minimum and maximum values on both X and Y axes are also recorded for each stroke.

- ChangeOnArea(coord *X*, coord *Y*, time *StartTime*, time *EndTime*): returns a list of time moments when there was a change in the specified area;
- FilterByArea(coord *X*, coord *Y*, time *StartTime*, time *EndTime*): returns a list of ink strokes drawn in the given area during the specified time interval.

4.2 AudioInteractors

AudioInteractors are a special type of Interactors obtained by content-based analysis of user speech files captured in a session. AudioInteractors have been categorized as time-based and attribute-based [28]. In the listing below, the specification of the operators has been updated to reflect the new requirements.

Time-based. Obtained by detecting patterns on the digital audio file (during the post-production phase). They can be used to identify time moments or time intervals in the speech when a specific voice behavior applies.

- silenceMoments(time *Tmin*, time *StartTime*, time *EndTime*): returns a list of time moments when there were no voices for at least *Tmin* units of time during the time interval defined by *StartTime* and *EndTime*.
- spokenMoments(time *Tmin*, time *StartTime*, time *EndTime*): returns a list of time moments before which someone has spoken for at least *Tmin* units of time during the time interval defined by *StartTime* and *EndTime*.

Attribute-based. There are digital audio attributes (e.g. frequency, pitch, noise, amplitude) that can be exploited to detect points of interest within the media element. Attribute-based operators include the following, also found in an updated version as detailed earlier.

- voiceIncrease(time *StartTime*, time *EndTime*): returns a list of time moments when there was a consistent increase in speech volume in the given time interval.
- conversation(time *StartTime*, time *EndTime*): in reference to the given interval *T* in the audio file, returns the potential number of participants who spoke during this interval.
- outstandingMoments(time *StartTime*, time *EndTime*): return a list of time moments when there were outstanding moments in the media element. Outstanding moments are those when several people are talking at the same time with consistent increase in the volume of their voices.

Various content-based methods can be employed to derive AudioInteractors. As an example, the computation of the time-based operator *SilenceMoments()* using wavelet transformations [16], in particular the *Haar transform*, has been detailed elsewhere [28]. An important category of AudioInteractors extended in previous work is the one associated with a user explicitly activating the audio mute function:

Action-based. Obtained by capturing moments in which a user activated the mute function of the microphone; examples are:

- enterAudioMute(time *StartTime*, time *EndTime*): returns a list of time moments, within the given time interval, when the mute function was activated.
- exitAudioMute(time *StartTime*, time *EndTime*): same as above, but with the mute function deactivated.

It is very common that users exchange text-based messages and interact with board-like surfaces to present slides during collaborative activities. Targeting at these requirements, the following sections define Interactors that tackle these issues, namely *TextInteractors* to encompass interactions via text messages and *BoardInteractors* to encompass interactions with board-like surfaces. Additionally, *VideoInteractors* are also defined to encompass user-image interactions.

4.3 TextInteractors

TextInteractors are a special type of Interactors obtained by identifying user-user interaction by means of textual (typed) messages: the messages can, for instance, be exchanged in a special Chat window which provides the message exchange service. These are also divided into two categories:

Attribute-based. Many attributes of text messages can be collected during capture, including font type and color.

- ChangeOnAttribute(attribute *A*, time *StartTime*, time *EndTime*): returns a list of time moments when the text message were changed according to attribute *A* within the time interval defined by *StartTime* and *EndTime* – this can

be used, for instance, to generate indexes to a timeline corresponding to a change in the given attribute (e.g., the time moments when there was a change of the font color of the text message);

- FilterByAttributeValue(attribute *A*, value *V*, time *StartTime*, time *EndTime*): returns a list of time moments when text messages were changed so that attribute *A* is equal to value *V*, within the time interval defined by *StartTime* and *EndTime* — this can be used, for instance, to generate an index to a timeline corresponding to the time moments when the text color was set to blue.

Time-based. Obtained by detecting intervals of message exchange or their inexistence.

- silenceMoments(time *Tmin*, time *StartTime*, time *EndTime*): returns a list of time moments when there were no messages exchanged for more than *T* (*T*>*Tmin*) units of time, during the time interval defined by *StartTime* and *EndTime*.
- textMoments(time *Tmin*, time *StartTime*, time *EndTime*): returns a list of time moments before which someone has typed messages for at least *Tmin* units of time during the time interval defined by *StartTime* and *EndTime*.

4.4 BoardInteractors

There are many situations in which users make use of board-like surfaces to deliver slide show presentations in meetings and lectures, for example. Some web conferencing systems offer board surfaces to present slides for discussion by the group; Webcast applications may use boards for the presenters to deliver their talks using slides and audio, for instance. There are also applications in which the slide show may be the only information presented, such as Slideshare[1], for example. Given the wide use of boards independently of ink-based annotation, we observed the need for defining operators directly associated with boards. These are also divided into two categories:

Time-based. Obtained by monitoring user-image interaction in the capture phase. The operators can be used to build a timeline with time moments corresponding to when one particular slide was presented.

- ChangeBoard(time *StartTime*, time *EndTime*): returns a list of time moments when there was a change of slide within the specified time interval – this can be used to generate a corresponding timeline;
- IdleBoard(time *T*, time *StartTime*, time *EndTime*): returns a list of time moments when there was no change of slide for at least *t* seconds within the specified time interval.

Atribute-based. There are attributes from the slides (whether a slide has text, image, animation, etc.) which can be used to detect slides of interest.

- ChangeOnAttribute(attribute *A*, time *StartTime*, time *EndTime*): returns a list of time moments when there was a change according to attribute *A* within the time interval defined by *StartTime* and *EndTime* – this can be used to generate an index to a timeline corresponding to a change in the attribute (e.g., the time moments when there was a change from text-based slide to a slide containing an animation);
- FilterByAttributeValue(attribute *A*, value *V*, time *StartTime*, time *EndTime*): returns a list of time moments when slides were changed so that attribute *A* is equal to value *V*, within the time interval defined by *StartTime* and *EndTime* — this can be used, for instance, to generate an index to a timeline corresponding to the time moments when the attribute contains the given value (e.g., the time moments when the slide contained an animation).

4.5 VideoInteractors

In lecture webcasts, such as those available in MIT OpenCourseware and Google Tech Talks, slides are captured by recording a video of the projected content. In order to enable a proper retrieval of the content, systems such as Talkminer [3] analyze the captured video by segmenting it in keyframes that represent changes in the presented slides. Another recurrent concern is the segmentation of the captured video according to changes in the scenes (for instance, the switch of cameras to change the focus from the slides to the lecturer and vice versa). Aiming at such scenarios, a *VideoInteractor* is an operator obtained by identifying user interactions with streams of visual content: this content may be, for instance, a set of slides, photos or a video. These are also divided into the following categories:

Time-based. Obtained by monitoring user-image interaction in the capture phase. The operators can be used to select moments or intervals when an image was reviewed.

- blankMoments(time *Tmin*): moments when no image was presented for *T* (*T*>*Tmin*) units of time.
- imageMoments(time *Tmin*): returns the moments before which a user was presented with an image for *T* (*T*>*Tmin*) units of time.
- imageIntervals(time *Tmin*): returns the start moment of time intervals in which a user was presented with an image for *T* (*T*>*Tmin*) units of time.

Attribute-based. There are attributes from the image (type, size, etc.) that can be exploited to detect images of interest.

- imageType(string *Type*): returns moments when an image of the given type was presented (e.g. PNG or JPG)
- imageSize(int *Size*): returns moments when an image of the given size was presented.
- cutMoments(): return the moments when there was a cut in the target video due to change of cameras
- others: one can also define operators based on features of images such as edges and corners (for JPEG), and level of alpha channel (for PNG).

5. AUTOMATIC GENERATION OF INTERACTIVE MULTIMEDIA DOCUMENTS

In this section we present the most important document engineering issues regarding an instantiation of the Interactors model. For demonstration purposes, we instantiate the model considering that the automatically generated document is reviewed in interactive TV clients whose primary interaction mechanism is via remote controls. As such the generated iMMD follows design guidelines and interaction mechanisms specially suited to these devices. It is worth stressing that the Interactors

[1]http://www.slideshare.net

model is not restricted to such circumstances, being also applicable for interactive documents on the Web and mobile devices, for instance.

In Figure 2 we present an architecture that illustrates the applicability of the Interactors model in the lifecycle of captured sessions for iTV. As can be noticed in Figure 2, the session data captured by the environment is exported into a data interchange document comprising links to the several media elements and logged interactions for each media element. The exported interchange document feeds the document generation component which transforms it into an iMMD. After these steps, the automatically generated iMMD can be transmitted (e.g. via TV broadcast or return channel) to client set-top boxes and be reviewed by users in interactive TV clients.

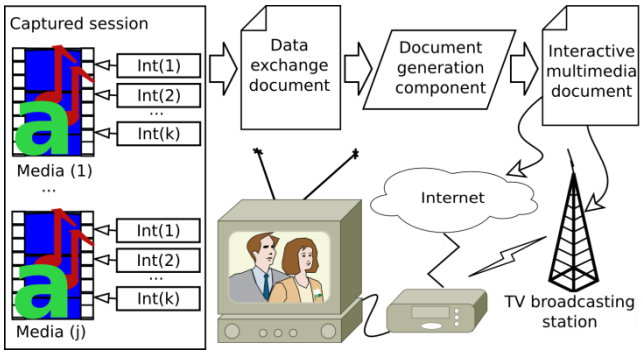


Figure 2: Instantiation of the Interactors model for iTV documents

The generated iMMD is intended to be generated from a XML-based interchange document that aggregates the captured media alongside logged interaction events (e.g. slide changes, ink-based events, chat messages). Adopting a XML-based interchange document enables the proposed model to be instantiated to different capture environments, regardless of particularities of their implementations.

In the following sections it is detailed the main issues of this architecture: Section 0 details the data exchange document. The document generation component is explained is Section 0. Section 0 reports the characteristics of the generated iMMD, including its declarative and procedural features.

5.1 Data exchange document

The data exchange model standardizes:

i) the description of the several media elements comprising a session; and

ii) the description of the logged interaction events captured by the environment.

This model is defined by means of a XML Schema and can be used by different environments to export captured data, enforcing loose coupling among the underlying session storage format and the generated iMMD. Moreover, a structured document facilitates transformation-based approaches to automatically generate iMMDs.

A data exchange document (Figure 3) has a primary element player which comprises several media elements captured by the meeting environment. For illustration purposes, the document in Figure 3 describes slides, chat conversations, video and audio (lines 3-12, 14-22, 24-30, 32-38, respectively); but it is worth stressing that the underlying XML schema foresees other types of medium not represented in this document. Each media element has attributes to specify its source, which may be local or remote. A media element is local when a bundle is retrieved from the capture environment, containing the data exchange document and all captured media; remote media is relevant when the exchange document is obtained as response from a web service, for instance, thus requiring that the media be retrieved later by their URLs.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <player author="..." >
3    <slideSet>
4      <local path="player/slides/">
5        <interactor description="slide change">
6          <image file="1.png" begin_ss="58" end_ss="76"/>
7          <image file="2.png" begin_ss="76" end_ss="94"/>
8          ...
9          <image file="23.png" begin_ss="425" end_ss="425"/>
10       </interactor>
11     </local>
12   </slideSet>
13   ...
14   <chatSession>
15     <local path="player/chatLog.html">
16       <interactor description="someone wrote">
17         <chat id="chat0" begin_ss="54" end_ss="56"/>
18         ...
19         <chat id="chat13" begin_ss="438" end_ss="441"/>
20       </interactor>
21     </local>
22   </chatSession>
23   ...
24   <video>
25     <server url="http://...">
26       <interactor description="participant interview">
27         <area begin_ss="10" end_ss="385"/>
28       </interactor>
29     </server>
30   </video>
31   ...
32   <audio>
33     <local filePath="file://...">
34       <interactor description="participant interview">
35         <area begin_ss="0" end_ss="385"/>
36       </interactor>
37     </local>
38   </audio>
39     ...
40 </player>
```

Figure 3: Excerpt illustrating the global structure of the data exchange document

Additionally, each media element has a set of Interactors associated with it. For instance, slides can have a series of Interactors of type slide change associated with them (lines 5-9), while chat sessions can have Interactors of type someone wrote (lines 17-19). For each Interactor, the schema provides specific identification and timing attributes. Interactors related to continuous media elements (such as streams of audio and video) are identified by filename and timed by their start and end moments. On the other hand, Interactors related to discrete media elements (captured from whiteboards, chats, photographic cameras, scanners, etc.) are identified by filename or surrogate and are timed by the period between two capture events (attributes begin_ss and end_ss): for instance, the period among two slide

changes or two chat messages. All these time markings are relative to the beginning of the session and are used for synchronization purposes when the iMMD is generated.

5.2 Document generation component

In order to create an iMMD from a captured session, the document generation component performs transformations on the exported data interchange document. This component clusters the Interactors by type (e.g. slide changes, ink annotations, chat conversations, etc.) and for each cluster it builds a main timeline, decorated with its Interactors. The generated timelines are assembled in the iMMD with the media elements. Consequently, the Interactors act as operators that enable users to navigate points of interest — such navigation is possible due to activation of a procedural document (more details on section 0) which guarantees that all media elements are synchronized when an operator is issued.

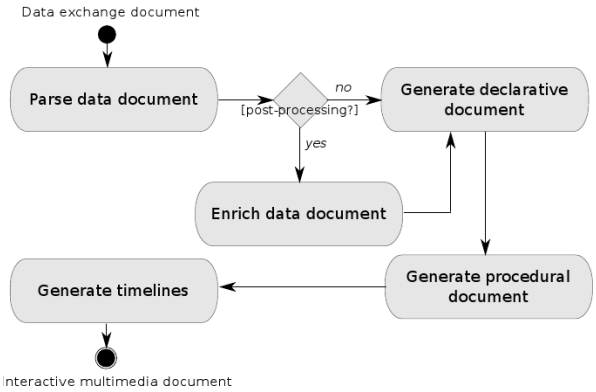


Figure 4: Activity model for the document generation component

The document generation component (Figure 4) first performs parsing of the data exchange model. If postprocessing is available (for example to build offline derived indices) the resulting DOM is enriched with additional Interactors. The next activity is concerned with the generation of a declarative multimedia document which acts as the user interface for reviewing the session. This declarative document describes the attributes of the media elements (such as medium type, source location, etc.), position and size of each media element on the screen, inter-media connections and synchronization according to the Interactors. Additionally, the declarative document includes the timeline that is generated in a later activity. In particular, this timeline is the element connecting the declarative document with the procedural document generated afterward.

It is worth mentioning that enabling enrichment of the data document opens up possibilities for exploring Interactors in the access and extension phases also. Once review-time Interactors for media editing and bookmarking are provided, the data document can be enriched with user-media interactions that could yield the generation of new, user-tailored versions of the original session.

5.3 Interactive multimedia document

The interactive multimedia document is the element that allows a user to access and to review a captured session. The automatically generated iMMD is composed of two distinct documents: a declarative document and a procedural document. In order to enforce strong separation among the declarative and procedural documents, a MVC (Model-View-Controller) approach has been adopted (Figure 5).

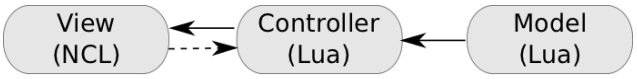


Figure 5: Global structure of the iMMD. Continuous lines are intra-document notifications and dashed lines are user-initiated events.

Built in NCL, the declarative document is responsible exclusively for rendering and synchronizing the media as well as configuring the layout of the multimedia presentation. Built in Lua, we have defined two procedural documents: *i*) the timeline model, which maps navigation events (e.g. user-initiated events, such as pressing a remote control button) to interaction events (e.g. points of interest) within the media; and *ii)* the timeline controller, which handles navigation events and queries the timeline model to update the state of the presentation.

Structurally, the declarative document defines a set of anchors that represent all available operators contained in the document timeline (Figure 6). All discrete media such as slides and chat messages are grouped in chronological order into a single NCL object (lines 11-18) which connects and orchestrates their synchronization. In the case of continuous media, such as audio and video, a separate anchor is defined (lines 3-9) for each media. Additionally, an anchor is defined for the timeline (lines 19-25).

```
1  <ncl xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
2      ...
3      <media descriptor="video2_Descriptor" id="video2small"
4                  src="ds3.avi">
5          <area begin="2s" id="v2t2s"/>
6          <area begin="4s" id="v2t4s"/>
7          ...
8          <area begin="384s" id="v2t384s"/>
9      </media>
10     ...
11     <media descriptor="stop_Descriptor" id="btnStop"
12                 src="stop24.gif" type="image/gif">
13         <!-- sync discrete media -->
14         <area begin="2s" id="bt2s"/>
15         <area begin="4s" id="bt4s"/>
16         ...
17         <area begin="384s" id="bt384s"/>
18     </media>
19     <media descriptor="timeLine_Descriptor" id="timeLine"
20                 src="tl.lua" type="application/x-ginga-NCLua">
21         <area id="t2s"/>
22         <area id="t4s"/>
23         ...
24         <area id="t384s"/>
25     </media>
26     ...
27 </ncl>
```

Figure 6: Excerpt of the automatically declarative document focusing on anchors

Imported by the declarative document (lines 19-20), the procedural document takes care of handling navigation events issued by the user. Upon receiving these events, this object

operates over the declarative document to jump to the requested point of interest. This action causes the document to be played back via a signal (Lua event) that activates one of the available anchors in the declarative document.

The mappings established by the timeline model (Figure 7) are registered in the structure iEvent – which contains information about the interaction events – by means of records storing the moment (instant) of the point of interest, navigation event (button) issued by the user, normalized offset position (dist), in dots or pixels, of the point of interest in the visual representation of the timeline and a sequential identifier (definition) for the point of interest.

```
/* FIRST STRUCTURE */
local tam = E   //number of Interactive Events
local frame={}
frame[1]={instant='t1s',button='RED', dist= 0, definition= 1}
...
frame[E]={instant='tTs', button='B', dist= X, definition= D}

/* SECOND STRUCTURE */
local mTam = N //number of Interactors
local menu={}
menu[1]={text='Midia(1)_Interactor(1)', button='RED',
definition= 1}
...
menu[N]={text='Midia(J)_Interactor(K)', button='B',  definition=
D}
```

Figure 7: Excerpt of the automatically generated procedural document focusing in the timeline model

The total number of Interactors is stored by the variable mTam, being $mTam \leq eTam$ because each Interactor is a non-empty set of interaction events. The structure menu defines the underlying data structure used for generating the interactive timeline to navigate the session. Each record of this structure is logically linked to the iEvent structure by a pair (*B*,*D*) where *B* is a button and *D* is the identifier of an interaction event.

Both data structures are employed by the timeline controller to activate specific anchors of the declarative document in response to a navigation event issued by the user. The timeline controller gets the information of this structures using the `getEvent(pos)` and `getMenu(pos)` methods, defined by the timeline model. The `getEvent(pos)` returns the corresponding interactive event at the `iEvent` structure, similarly to what the `getMenu(pos)` method does relative to the menu structure. The main methods used by the timeline controller are:

- `Redraw`: takes care of updating the visual appearance of the timeline in the declarative document. Its responsibilities include the placement of marks in the timeline corresponding to the interaction events;
- `Stop()` and `Start()`: these functions are responsible for triggering the resynchronization of an specific anchor in the declarative document;
- `Handler(evt)`: receives as a parameter a user navigation event (issued via remote control) and reacts according to a mapping that relates an event to its corresponding action. Beyond controlling the access to specific points of interest, this function concerns also other interface-related issues;
- `openMenu()` and `closeMenu()`: receives events for showing/hiding the decorated timeline when a button in the remote control is pressed.

In summary, the timeline controller is responsible for receiving events from the remote control, querying the timeline model for the anchor that corresponds to the point of interest and, in response, activating the returned anchor in the declarative document to update the presentation state.

6. CASE STUDY

DiGaE (*Distributed Gathering Environment*) is a capture environment for collaborative meetings that can be used either in instrumented rooms or in webconferencing mode. When used for webconferencing purposes, a special configuration called DiGaE Home provides a web-based tool to capture audio and video streams from the desktop computer, as well as other communication tools such as instant text-based messaging and synchronous software-based whiteboard.

We employed the DiGaE Home tool in a meeting with a remote lecturer and a group of attendants that used whiteboard software to load a set of slides and make ink-based annotations, a chat session to provide text-based message exchange, and video+audio conferencing. User-board and user-ink interactions as well as textual messages and the audio+video files were recorded. The dynamics of the meeting required that attendants took turns with respect to talking, asking, drawing and typing text.

Six volunteers participated in the experiment: five (the audience) were located in an instrumented meeting room and one (the presenter) in a remote location. The participants were college students recruited from the computer science program of a university. The participants were native speakers of Portuguese with knowledge of English. All participants had previous experience browsing the Web and using TV remote controls. Before the meeting started, participants had a brief introduction to the capture environment, the interface elements and the topic of the meeting. The purpose of the meeting was the presentation of a topic to the room attendants. The participants of the meeting room could also discuss, ask questions, comment and contribute with the dynamics of the meeting.

Table 1: Listing of Interactors used in the prototype.

Category	Operator	Attributes
Inkteractors	FilterByAuthor	author
AudioInteractors	spokenMoments	-
TextInteractors	filterByAttributeValue	authors
BoardInteractors	filterByAttributeValue	slides titles
BoardInteractors	changeBoard	-

After the meeting ended, the recorded session was processed, exported and transformed into an iMMD in NCLua. The Interactors applied in this document are listed in Table 1. Provided that the enriched document must be transformed into an appropriate interactive multimedia format for playback, in Figures 8 and 9 we show the use of an iMMD in NCL (declarative) [23] with Lua (procedural) [13].

Figure 8 depicts the layout of the final iMMD: its playback is controlled by a player which offers several interaction options by means of a keyboard when the playback occurs in a computer-based platform, for instance, or a remote control when the playback occurs in a TV-based platform. The interface contains: *1)* the whiteboard region; *2)* the video region; *3)* the chat region; and *4)* the timeline region.

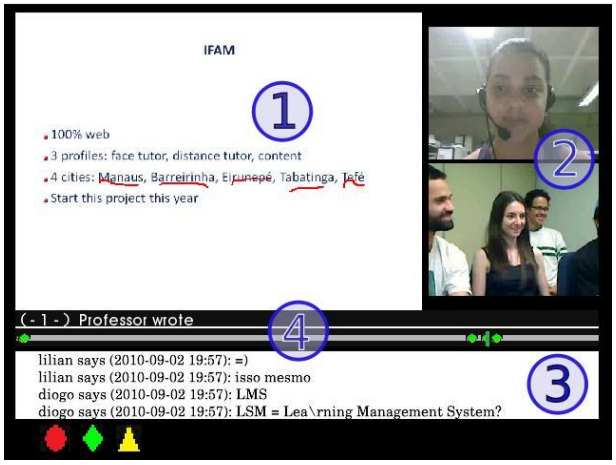


Figure 8: Structure of the final interactive multimedia document illustrating the timeline decorated with text-based Interactors.

In Figure 8, the color buttons in the bottom left corner (red circle, green diamond and yellow triangle) can be used to toggle media-specific Interactors to decorate the timeline. The timeline enables users to navigate to different points of interest within media elements. For example, the decorated timeline of Figure 8 shows the Interactor events related to the filterByAttributeValue() TextInteractor, with author as attribute, after pressing the green button, and it indicates time moments in which there were professor-text interactions during the meeting.

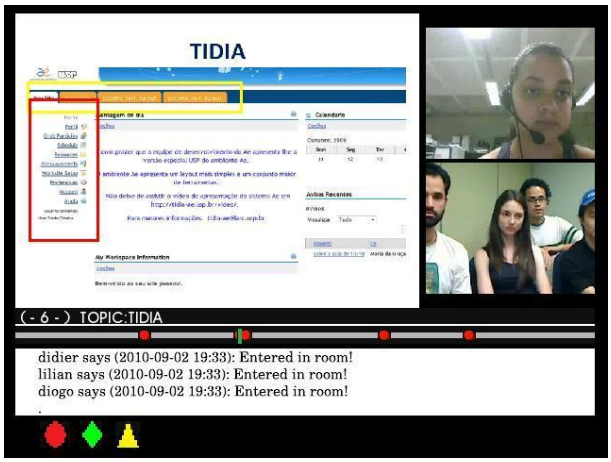


Figure 9: Screenshot illustrating the timeline decorated with ink-based Interactors.

Additionally, a label is included above the timeline to indicate the specific Interactor that is active. If other Interactors are available for the same media element, both up/down navigation keys, and also a numerical code, can be used to select them. In Figure 9, the red button and then the up/down keys were used to activate the ChangeOnAttribute() BoardInteractor, regarding the text title as attribute; it is clear that the presentation updates the label and the corresponding points of interest in the timeline. In order to navigate to a particular point of interest, users employ the right/left navigation buttons in the playback interface to focus on the desired point, and press the *ok* (or *enter*) button when finished.

7. EVALUATION WITH USERS

This experiment consisted on reviewing a recorded session considering the scenario of webconferences. The evaluation was performed with four of the six participants which were present in the recorded proof-of-concept scenario, one month after the session was recorded. The participants received a brief introduction to Interactors, the main interface elements and a list of five simulated search requests about the meeting. The search requests were formulated regarding information about the discourse of the speaker, the content of the questions issued by the audience during the session, the themes of the conversations among participants and information from the slides.

The experiment was designed so that users had to answer each request by navigating the iMMD timeline enriched with the Interactors menu. Additionally, we employed the think aloud protocol so that user decision criteria regarding the navigation of the interface could be analyzed. Users were instructed to speak about their actions over the interface and the reasons of the interaction decisions they made. At the end of the test, each user participated in an interview regarding his understanding of the concept and usefulness of the tool.

The majority of the users felt comfortable with the possibility of navigating points of interest in the recorded session and acknowledged about the usefulness of the Interactors menu, except for some shortcomings. In the beginning of the experiment, when users were not very aware of the possibilities of the tool, they had trouble to associate a logical operator to its meaning. This result revealed learnability problems in the interface. All users could easily recognize how to navigate in the interaction events of the timeline and into the Interactors menu. However, most people observed that for browsing between operators of the same type, it was necessary to roll over the menu to find the best Interactor for the search request. This behavior may have led, in 44% of the requests, participants to choose the first operator which allowed achieving the desired point of interest and not the best one, given the similarity of its meaning in the context of the request. In the experiment, users were able to create logical queries involving up to two Interactors. Finally, users appreciated the ability to find points of interest quickly.

With these results we could notice that, having an interval of one month between the capture of the experience and the review of the recorded session, many participants did not recall all information discussed in the meeting. But on the other hand participants remembered specific contextual events that took place in the session, such as the overall visual content of a slide, the moments when some participant asked a question, the moments when some participant sent a text message, among others. We suggest that such contextual events have been stored in the episodic memory of the participants and can be exploited as cues to recall details of past experiences, as advocated by Kim et al. (2006). Furthermore we noticed that Interactors have helped users exploit such events during the review of the session.

8. CONCLUSIONS AND FUTURE WORK

Building upon a taxonomy of media-oriented operators defined in previous work [29], in this paper we augmented these results by demonstrating the applicability of the Interactors model for two complementary situations: *i)* the automatic authoring of interactive multimedia documents; and *ii)* the efficient review of recorded sessions by users. Regarding the automatic generation of interactive multimedia documents, we stressed the most important document engineering issues for tackling this problem. Among them, we mention the problem of interchanging data among heterogeneous systems, namely a web-based capture environment and a TV-based access/playback environment, for which the solution we presented builds upon structured document-based interchange and transformation mechanisms. Another automatic authoring issue that was tackled concerns how to automatically provide means for users to interactively watch a recorded session. For this issue, we demonstrated that metadata collected with low effort through the multimedia production process can be a good alternative for navigating multimedia recordings, provided that such metadata is combined via properly defined operators.

Regarding the utility of Interactors for end-users, we conducted an evaluation with a proof-of-concept prototype using multimedia information captured in a webconferencing tool and generated interactive multimedia documents enhanced with Interactors. Exploiting specifically the effectiveness of the operators when reviewing recorded sessions, we analyzed if the operators could help people revisit previous recorded experiences more efficiently. For this purpose, the experiment considered a timeframe of one month among the recording and the review sessions. Our evaluation results suggest that Interactors are effective means to assist the review of captured collaborative sessions by navigating points of interest. Even though we noticed problems related to the selection of the operators in the interface, in general, users acknowledged about the easiness and effectiveness to review the session using points of interest. More importantly, the results suggested that the operators can help users to revisit previous (recorded) experiences more efficiently when compared to the use of non-interactive videos. In face of these results, formalization of new operators and extension of automatic authoring to the Web environment are planned as far as future work is concerned.

Acknowledgments. We thank FAPESP, CNPq, CAPES, FINEP, MCT and the users, specialists and anonymous reviewers for the many important suggestions.

9. References

[1] G. Abowd, E. Mynatt, and T. Rodden. The human experience [of ubiquitous computing]. *IEEE Pervasive Computing*, 1(1):48–57, 2002.

[2] G. D. Abowd, C. G. Atkeson, J. Brotherton, T. Enqvist, P. Gulley, and J. LeMon. Investigating the capture, integration and access problem of ubiquitous computing in an educational setting. In *ACM CHI'98*, pages 440–447, 1998.

[3] J. Adcock, M. Cooper, L. Denoue, H. Pirsiavash, and L. A. Rowe. Talkminer: a lecture webcast search engine. In *ACM MM '10*, pages 241–250, 2010.

[4] A. Behera, D. Lalanne, and R. Ingold. DocMIR: An automatic document-based indexing system for meeting retrieval. *Multimedia Tools and Applications*, 37(2): 135–167, 2007.

[5] M.-M. Bouamrane and S. Luz. Meeting browsing. *Multimedia Systems*, 12(4-5):439–457, October 2006.

[6] S. Branham, G. Golovchinsky, S. Carter, and J. T. Biehl. Let's go from the whiteboard: supporting transitions in work through whiteboard capture and reuse. In *ACM CHI'10*, pages 75–84, 2010.

[7] D. C. A. Bulterman and L. Hardman. Structured multimedia authoring. *ACM Trans. Multimedia Comput. Commun. Appl.*, 1(1):89-109, 2005.

[8] R. G. Cattelan, C. Teixeira, R. Goularte, and M. D. Pimentel. Watch-and-comment as a paradigm toward ubiquitous interactive video editing. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4 (4):1–24, 2008a.

[9] R. G. Cattelan, C. Teixeira, H. Ribas, E. Munson, and M. Pimentel. Inkteractors: interacting with digital ink. In *ACM SAC'08*, pages 1246–1251, 2008b.

[10] P. Cesar, D. C. A. Bulterman, and A. J. Jansen. Benefits of structured multimedia documents in IDTV. In *ACM DOCENG 2006*, pages 176–178, 2006.

[11] P. Cesar, D. C. A. Bulterman, J. Jansen, D. Geerts, H. Knoche, and W. Seager. Fragment, tag, enrich, and send: Enhancing social sharing of video. *ACM Trans. Multimedia Comput. Commun. Appl*. 5(3):1-27, 2009.

[12] M. Czerwinski, D. W. Gage, J. Gemmell, C. C. Marshall, M. A. Pérez-Quiñones, M. M. Skeels, and T. Catarci. Digital memories in an era of ubiquitous computing and abundant storage. *Comm. ACM*, 49(1):44, 2006.

[13] R. de Mello Brandão, G. de Souza Filho, C. Batista, and L. Gomes Soares. Extended Features for the Ginga-NCL Environment: Introducing the LuaTV API. In *Computer Communications and Networks (ICCCN)*, pages 1 –6, 2010.

[14] P. Ehlen, M. Purver, J. Niekrasz, K. Lee, and S. Peters. Meeting adjourned: off-line learning interfaces for automatic meeting understanding. In *ACM IUI '08*, pages 276–284, 2008.

[15] W. Geyer, H. Richter, and G. D. Abowd. Towards a Smarter Meeting Record-Capture and Access of Meetings Revisited. *Multimedia Tools and Applications*, 27(3): 393–410, 2005.

[16] R. C. Guido, J. F. W. Slaets, R. Köberle, L. O. B. Almeida, and J. C. Pereira. A new technique to construct a wavelet transform matching a specified signal with applications to digital, real time, spike, and overlap pattern recognition. *Digital Signal Processing*, 16(1):24–44, 2006.

[17] S. Junuzovic, R. Hegde, Z. Zhang, P. A. Chou, Z. Liu, and C. Zhang. Requirements and recommendations for an enhanced meeting viewing experience. In *Proc. of the 16th ACM international conference on Multimedia*, pages 539–548, 2008.

[18] J. Kim, H. Kim, and K. Park. Towards optimal navigation through video content on interactive tv. *Interacting with Computers*, 18(4):723 – 746, 2006.

[19] S. Minneman, S. Harrison, B. Janssen, G. Kurtenbach, T. Moran, I. Smith, and B. van Melle. A confederation of

tools for capturing and accessing collaborative activity. In *ACM MULTIMEDIA'95*, pages 523–534, 1995.

[20] S. Mirri, L. A. Muratori, M. Roccetti, and P. Salomoni. The Directors' cut: a solution to collaborative multimedia management. *Multimedia Tools Appl*. 53(1):319-344, 2011.

[21] K. Ntalianis, A. Doulamis, N. Tsapatsoulis, and N. Doulamis. Human action annotation, modeling and analysis based on implicit user interaction. *Multimedia Tools and Applications*, (online first), October 2009.

[22] M. G. C. Pimentel, Y. Ishiguro, B. Kerimbaev, G. Abowd, and M. Guzdial. Supporting educational activities through dynamic web interfaces. *Interacting with Computers*, pages 353–374, 2001.

[23] L. F. G. Soares, R. F. Rodrigues, R. Cerqueira, and S. D. J. Barbosa. Variable handling in time-based XML declarative languages. In *ACM SAC '09*, pages 1821–1828. ACM, 2009.

[24] C. A. C. Teixeira, G. B. Freitas, and M. Pimentel. Distributed discrimination of media moments and media intervals: a watch-and-comment approach. In *ACM SAC'10*, pages 1929–1935, 2010.

[25] K. N. Truong and G. R. Hayes. Ubiquitous computing for capture and access. *Found. Trends Hum.-Comput. Interact.*, 2(2): 95–171, 2009.

[26] S. Tucker and S. Whittaker. Accessing Multimodal Meeting Data: Systems, Problems and Possibilities. In *Proc. Work. Machine Learning for Multimodal Interaction*, pages 1–11, 2004.

[27] D. A. Vega-Oliveros, D. S. Martins, and M. G. C. Pimentel. "This conversation will be recorded": automatically generating interactive multimedia documents from captured media. In *ACM DOCENG '10*, 2010a.

[28] D. A. Vega-Oliveros, D. S. Martins, and M. G. C. Pimentel. Interactors: operators to automatically generate interactive multimedia documents from captured media. In *Proc. Brazilian Symposium on Multimedia and the Web*, 2010b.

[29] D. A. Vega-Oliveros, D. S. Martins, and M. G. C. Pimentel. Media-oriented operators for authoring interactive multimedia documents generated from capture sessions. In *ACM SAC '11*, volume 2, pages 1267–1272, 2011.

[30] S. Whittaker, S. Tucker, K. Swampillai, and R. Laban. Design and evaluation of systems to support interaction capture and retrieval. *Personal and Ubiquitous Computing*, 12(3): 197–221, 2007.

[31] Z. Yu and Y. Nakamura. Smart meeting systems: A survey of state-of-the-art and open issues. *ACM Computing Surveys*, 42(2):1–20, 2010.

[32] Z. Yu, Z. Yu, H. Aoyama, M. Ozeki, and Y. Nakamura. Social interaction detection and browsing in meetings. In *Proc. Intl. Conf. Ubiquitous Computing*, pages 40–41, 2008.

[33] V. Zsombori, M. Frantzis, R. L. Guimaraes, M. F. Ursu, P. Cesar, I. Kegel, R. Craigie, and D. C. A. Bulterman. Automatic generation of video narratives from shared UGC. In *Proceedings of the ACM Conference on Hypertext and Hypermedia (HT '11)*, pages 325-334, 2011.

ABOUT THE AUTHORS:

Diogo Santana Martins received his B.Sc. and M.Sc. degrees in Computer Science from Universidade Federal de São Carlos (UFSCar), Brazil. Currently he is a Ph.D. candidate in Interactive Web and Multimedia Systems at the Universidade de São Paulo, São Carlos, Brazil. The author is experienced on research in information retrieval and personalization in electronic learning environments; currently he investigates context aware computing methods for indexing and browsing multimedia recordings.

Didier Vega-Oliveros is a Technical Leader in the department of research and development of Semantic and Multimodal products at Invit Innovative Business, Brazil. He received his master's degree in Computer Science from Institute of Mathematics and Computer Science (ICMC) at University of São Paulo in 2011 and his B.Sc in Computer Engineering from National University of Colombia in 2008. His fields of interests encompass Ubiquitous Computing, Interactive TV, Human Computer Interfaces and Context and Situation Awareness.

Maria da Graça Campos Pimentel is a Full Professor in the Computer Science Department at the Universidade de São Paulo, São Carlos, Brazil, where she is the head of the Intermidia Laboratory. Dr. Pimentel received her Ph.D. in Computer Science in 1994 from the University of Kent, UK. Her research interests include multimedia systems, ubiquitous computing and Interactive TV. She is a member of the ACM, the IEEE, and the Brazilian Computer Society (SBC).

Efficient Page Caching Algorithm with Prediction and Migration for a Hybrid Main Memory

Hyunchul Seok, Youngwoo Park, Ki-Woong Park, and Kyu Ho Park
KAIST
Daejeon, Korea
{hcseok, ywpark, woongbak}@core.kaist.ac.kr and kpark@ee.kaist.ac.kr

ABSTRACT
Emerging next generation memories, NVRAMs, such as Phase-change RAM (PRAM), Ferroelectric RAM (FRAM), and Magnetic RAM (MRAM) are rapidly becoming promising candidates for large scale main memory because of their high density and low power consumption. Many researchers have attempted to construct a main memory with NVRAMs, in order to make up for the limits of NVRAMs. However, we find that the preexisting page caching algorithms, such as LRU, LIRS, and CLOCK-Pro, are often sub-optimal for NVRAMs due to its DRAM-oriented design including uniform access latency and unlimited endurance. Consequently, the algorithms cannot be directly adapted to the hybrid main memory architecture with PRAM.

To mitigate this design limitation, we propose a new page caching algorithm for the hybrid main memory. It is designed to overcome the long latency and low endurance of PRAM. On the basis of the LRU replacement algorithm, we propose a prediction of page access pattern and migration schemes to maintain write-bound access pages to DRAM. The experiment results have convinced us that our page caching algorithm minimizes the number of the write access of PRAM while maintaining the cache hit ratio. The results show that we can reduce the total write access count by a maximum of 52.9% and the consumed energy by 19.9%. Therefore, we can enhance the average page cache performance and reduce the endurance problem in the hybrid main memory.

Categories and Subject Descriptors
B.3.2 [Memory Structures]: Design Styles Cache memories; D.4.2 [Operating Systems]: Storage Management - *Allocation /deallocation strategies, Main memory*

General Terms
Page cache algorithm

Keywords
Page Cache, Hybrid Main Memory, PRAM, Migration

1. INTRODUCTION

The main memory of today's computer systems is showing significant change with the emergence of next generation types of memory, NVRAMs, such as Phase-change RAM (PRAM), Ferroelectric RAM (FRAM), and Magnetic RAM (MRAM) [20][24][14]. Several studies have observed that DRAM based main memory, while it significantly increases the computing power, also greatly increases the cost of a computer system [4]. Therefore, many trials have been made to replace DRAM. One such trial attempted to use hybrid memory by including NVRAMs.

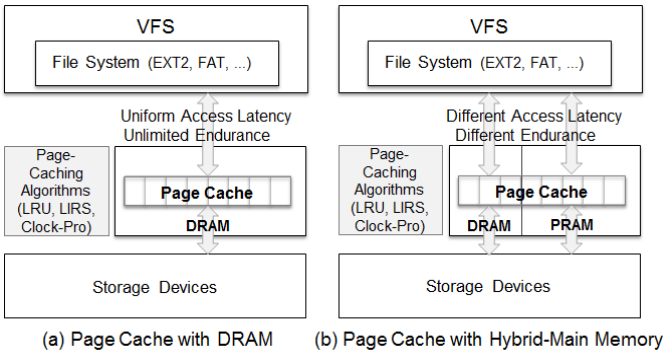


Figure 1. Page caching system architecture

Among these memories, PRAM is the most promising candidate for uses in large scale main memory because of its high density and low power consumption [20]. While DRAM stores each bit of data in a separate capacitor, PRAM uses the phase of the material to represent each bit. PRAM density is expected to be much greater than that of DRAM (about four times). Also, PRAM is a non-volatile memory because the phase of the material does not change after power-off. It has negligible leakage energy regardless of the size of the memory. Table 1 shows the properties of PRAM compared to those of DRAM [17]. Though PRAM has attractive features such as high density, non-volatility, low idle power, and good scalability, the access latency of PRAM is still not comparable to DRAM latency. PRAM also has a worn-out problem caused by limited write endurance. In previous research, the hybrid main memory approaches of DRAM and PRAM have been adopted to make up for the latency and endurance limits of PRAM [10][20][18]. Those researchers proposed several hardware and software schemes to manage the hybrid main memory.

On the other hand, in modern computer systems, a large part of main memory is used as a page cache to hide disk access latency, as can be seen in Figure 1(a). Many page caching algorithms such as LRU, LIRS [12], and CLOCK-Pro [19] have been developed and show good performance for current DRAM based main memory. However, such previous page caching algorithms only considered the main memory with uniform access latency and unlimited endurance. They cannot be directly adapted to the

hybrid main memory architecture with DRAM and PRAM, as shown in Figure 1(b).

This paper is an extended work related to our previous work in [22]. The previous work mainly focused on a page caching algorithm design to enhance the average page cache performance and to reduce the endurance problem. In this paper, attempts are made to integrate a prediction and page migration mechanisms as the key primitive for additional performance enhancement.

The main objective of this work is to propose a new page caching algorithm for the hybrid main memory. The algorithm is designed to overcome the long latency and endurance problem of PRAM. On the basis of conventional cache replacement algorithms, we propose a prediction of the page access pattern by page monitoring and migration schemes to move write-bound access pages to DRAM. In particular, we present analytic metrics for PRAM endurance, energy, and latency, and illustrate that existing page caching algorithms such as LRU, LIRS, and CLOCK-Pro are suboptimal in the hybrid main memory. We present improved algorithms for enhancing the page cache performance on the hybrid main memory. It minimizes the write access of PRAM while maintaining the cache hit ratio. Therefore, we can enhance the average page cache performance and reduce the endurance problem in the hybrid main memory. The remainder of the paper is organized as follows. In Section 2, we briefly summarize the conventional page caching algorithms and their limits when used in hybrid memory. In Section 3 we present the design of our algorithm and describe the prediction algorithm and migration strategy. The performance evaluation results are given in Section 4, and we briefly discuss future work in Section 5. Section 6 concludes this paper.

2. BACKGROUND

A proper page caching algorithm can have a significant effect on improving performance of I/O by hiding the long latency of disks. Many studies have been undertaken to make efficient page cache algorithms. However, most page cache replacement algorithms have been designed for memory with uniform access latency and unlimited endurance. In this paper, we evaluate the page caching algorithms for hybrid memory, which has different read and write latencies and different endurance. We propose a new page caching algorithm that includes prediction and migration schemes for efficiently hiding the bad effects derived from the different properties of hybrid memory and compare the results. Although we implement our algorithm based on the LRU replacement algorithm, our technique can be utilized with the conventional page caching algorithm and can improve the performance of page caching in hybrid memory.

LRU (Least Recently Used) has been widely used as a cache management and page cache replacement algorithm [6][2][8][9] [11]. When the cache is full and a miss occurs, this algorithm selects the page where it is in the LRU position as a victim. One of the advantages is that it is very simple to implement this algorithm. The algorithm also has constant time and space overhead. But it has some disadvantages. It is known that LRU shows the best performance on the workloads of Stack Depth Distribution (SDD) [6]. However, LRU cannot operate well with an access pattern with weak locality, such as sequential scans, a cyclic pattern with slightly larger than cache size.

LIRS (Low Inter-Reference Recency Set) was proposed by Jiang and Zhang in 2002 in order to solve the problems of LRU [12].

Table 1. Properties of PRAM

Attributes	DRAM	PRAM
Non-volatility	No	Yes
Cost/TB	Highest (~4x PRAM)	Low
Read Latency	50ns	50-100ns
Write Latency	20-50ns	~ 1us
Read Energy	~ 0.1nJ/b	~ 0.1nJ/b
Write Energy	~ 0.1nJ/b	~ 0.5nJ/b
Idle Power	~ 1.3W/GB	~ 0.05W
Endurance	∞	10^8 for write

This algorithm uses Inter-Reference Recency (IRR) to determine which page should be replaced. The objectives of LIRS are both to effectively address the limits of LRU and to retain the low overhead of LRU. As the history information of each page, IRR is defined as the number of other pages accessed between two consecutive references to the page. The algorithm assumes that if the IRR of a page is large, the next IRR of the page is likely to be large. Therefore, LIRS selects pages with large IRRs for replacement.

CLOCK-Pro was proposed by Jiang, Chen, and Zhang in 2005 [19]. It is based on CLOCK which is a simple approximation of the LRU replacement algorithm [7][5]. CLOCK can reduce the overheads of the LRU algorithm, which are related to the overhead of moving a page to the MRU position on every page hit [3]. The objectives of CLOCK-Pro are to obtain a low computational requirement and to remove the disadvantages, which are the same disadvantages as those of LRU with the workload of weak locality. Therefore, CLOCK-Pro was designed to include the way in which LIRS and CLOCK work.

In addition to the replacement algorithms mentioned above, there are many caching algorithms, most of which have the goal of improving the performance problems of LRU. In studies of such methods, FBR [21], LRU-2 [16], 2Q [13], LRFU [15], and MQ [25] were proposed and researchers tried to combine "recency" (LRU) and "frequency" (LFU) in order to compensate for the disadvantages of LRU. These disadvantages are derived from the access patterns such as the sequential stream and a cyclic pattern with larger than cache size. However, these methods cannot consider the physically different properties of hybrid memory. For example, PRAM has about ten times larger write latency than DRAM does, as can be seen in Table 1. Therefore, if some pages are frequently accessed by write accesses and these pages are in PRAM memory, the overall performance will degrade. In such a case, we can improve the performance if the page cache algorithm can make a decision to put the write-bound pages into DRAM rather than into PRAM. In this case, the performance is affected by the position of the pages.

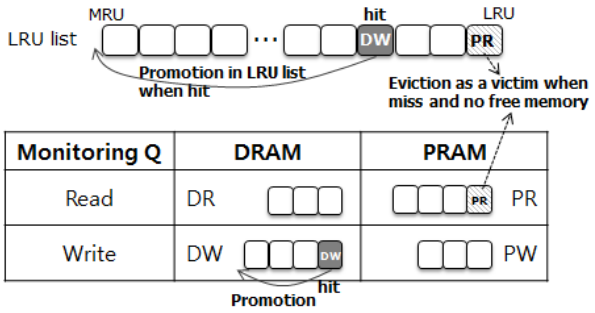


Figure 2. Overview of page caching algorithm

3. PAGE CACHING ALGORITHM FOR HYBRID MAIN MEMORY

In this section, we describe the design of a new page caching algorithm for the hybrid main memory; this design consists of DRAM and PRAM. Although the conventional cache algorithms show good performance, they cannot be directly adapted to the hybrid main memory. Because the hybrid main memory uses two different types of memories, it is important to consider their properties in order to maximize the performance and efficiency. As can be seen in Table 1, PRAM has a very long latency compared to DRAM when writing a page. If the page cache algorithm causes a lot of writes to PRAM, the average page cache performance gets worse. In addition, PRAM has low endurance compared to DRAM. If there are many write accesses, PRAM will be worn-out quickly. Consequently, the page cache algorithm on the hybrid main memory should be designed to overcome the long latency and endurance problem of PRAM. To solve these problems, we propose a new page caching algorithm with prediction of page access pattern and migration schemes.

3.1 Basic Operation

To satisfy the requirements, we designed the new page caching algorithm according to the conventional cache algorithm. We add a prediction of page access pattern and migration schemes. Although any conventional cache replacement algorithm can be adapted, we chose the LRU replacement algorithm, which is very simple but works well. In order to monitor the pages and to adapt the migration scheme, we use four monitoring queues, which consist of a DRAM read queue, a DRAM write queue, a PRAM read queue, and a PRAM write queue, as shown in Figure 2. When one page block is accessed, it is retained into both the LRU list and one of the four queues by its access pattern and the memory type where it is located.

Figure 2 shows the basic operation of our algorithm. When a page fault occurs, we put the page into the MRU position in the LRU list with the LRU replacement algorithm. In addition, we put the page into one of the monitoring queues according to the page access type. For example, if the page's access request is read and a selected memory page is on DRAM, we put the page into the DRAM read queue. If there is no free memory when a miss occurs, the least recently used page block is selected as a victim page, which also follows the LRU replacement algorithm. At the same time, we evict the page related to the victim page. For example, if a type of the victim page is read cache and resides in PRAM, we eliminate the page from the PRAM read queue by evicting the page in LRU list. When the page hits, the page in both the LRU list and the monitoring queue is promoted, as can be seen in Figure 2.

Algorithm 1 : Page Migration Function

input: page address and request type
W_{cur}: Current weight value
Tr_{mig}: Threshold value for determining migration
Tr_q: Threshold value for determining movement between read and write queues

```
Calculate W_cur
if page in PRAM then
    if W_cur ≥ Tr_mig then
        migrate the page to DRAM
    else if W_cur ≥ Tr_q and page in read queue then
        the page moves to write queue
    else if W_cur ≤ −Tr_q and page in write queue then
        the page moves to read queue
    end if
else if page in DRAM then
    if W_cur ≤ −Tr_mig then
        migrate the page to PRAM
    else if W_cur ≤ −Tr_q and page in write queue then
        the page moves to read queue
    else if W_cur ≥ Tr_q and page in read queue then
        the page moves to write queue
    end if
end if
```

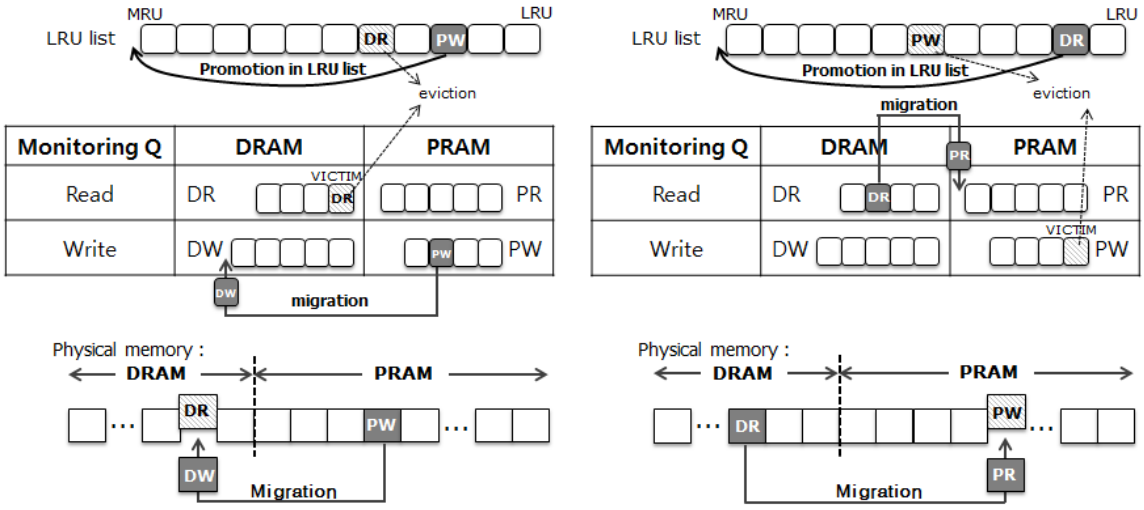
3.2 Prediction and Page Migration

The write-bound pages on PRAM cause performance degradation and worn-out problem because of PRAM's long latency and low endurance. To solve these problems, we need to move the write-bound pages from PRAM to DRAM. Additionally, we must move the read-bound pages from DRAM to PRAM because we have to gather the read-bound pages to PRAM. In order to effect the efficient migration of pages, we have to know which pages are write-bound and which pages are read-bound. For prediction of the access pattern, we calculate the weighting values, which indicate how close the values are to write-bound or read-bound. By monitoring the request type of the page requests, the weighting value can be calculated by using a moving average with weight $\alpha \in [0,1]$ as follows:

$$W_{cur} = \alpha W_{prev} + (1-\alpha)RT \quad (1)$$

, where RT means the requested type of the page; its value is 1 if the page request is write and -1 if the page request is read. When a page fault occurs, the page is inserted into both the LRU list and one of the monitoring queues. We selected the value of α as 0.5. We will explain in detail in section 4.2.1.

There are two migration cases, as shown in Figure 3: one is the migration of write-bound pages from PRAM to DRAM; the other is the migration of read-bound pages from DRAM to PRAM. To determine when migration occurs, we calculate the W_{cur} value at every request. According to equation 1, this value is increased



(a) Migration of the write-bound page

(b) Migration of the read-bound page

Figure 3. Migration of cached pages between DRAM and PRAM

when write requests occur and is decreased at every read request. Algorithm 1 shows how the cached pages are migrated. When a page request hits, W_{cur} of the hit page is calculated by equation 1 and we determine whether migration occurs. For example, if the write access is hit on a page in the PRAM write queue, PW, as shown in Figure 3(a), and its weighting value is over Tr_{mig}, this page will migrate to the DRAM write queue. Similarly, as can be seen in 3(b), if a page in the DRAM read queue, DR, is hit by a read request and its W_{cur} value is under $-Tr_{mig}$, this page will migrate to the PRAM read queue. We use two threshold values for determining the migration and the movement between read and write queues in the same memory. Tr_{mig} is the threshold value for determining whether a page is migrated and Tr_q is the threshold value for determining movement between read and write queues.

Figure 3(a) shows an example of migration from PRAM to DRAM. If there is no free space in DRAM, we have to select a victim page on DRAM. In this case, we select the victim page from the bottom of the DRAM read queue and remove it from DRAM. The write-bound page in PRAM is moved to the DRAM where the victim page was located. In the DRAM write queue, this page is put into the top of the queue. If there is no element in the DRAM read queue when we find a victim page for migration, we choose a victim page from the bottom of the DRAM write queue, which means that the victim page is the least recently used. The migration of read-bound pages is similar to the migration of write-bound pages, as shown in Figure 3(b). To select a victim page, we select the bottom page of the PRAM write queue first, and if there are no pages in the PRAM write queue, we choose the bottom page of the PRAM read queue. When we remove a victim page for migration, we just remove it from the memory, the LRU list, and the monitoring queue. The reason why we do not change the pages between the victim page for migration and the page that will migrate is that such an action would cause an additional write on PRAM.

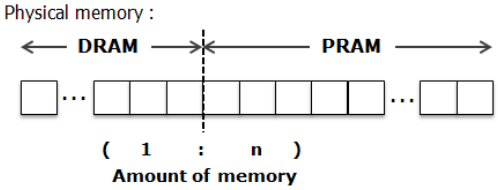


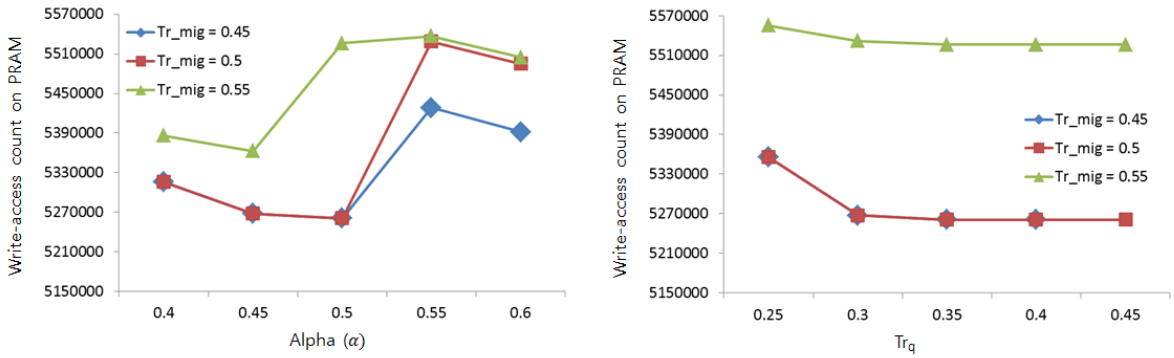
Figure 4. Composition of the hybrid main memory

4. EXPERIMENT

In order to evaluate the proposed page cache algorithm for the hybrid main memory, we used a trace-driven simulation. We implemented our algorithm and designed the hybrid main memory architecture. In this section, we present the performance results of our hybrid cache algorithm. First, we demonstrate the overall experimental environment. We then describe the evaluation results in terms of hit ratio, write access count on PRAM, and energy consumption.

4.1 Experiment Setup

In order to evaluate the performance of our page cache scheme, we have to define the hybrid main memory. We assume that the hybrid main memory consists of DRAM and PRAM, which are divided by a memory address. The memory which has the low memory address is DRAM and the high section is allocated to PRAM, as shown in Figure 4. PRAM density is generally expected to be higher than that of DRAM [17][23][20], so that we allocate larger amount of memory to PRAM. Because PRAM density is expected to be four times higher than that of DRAM [17][23], we mainly select that the PRAM-to-DRAM ratio is four. However, this density value can be varied in several researches [20] so that we additionally evaluated out page caching algorithm with various PRAM-to-DRAM ratios.



(a) Results when Tr_q is 0.35

(b) Results when Alpha (α) is 0.5

Figure 6. The experiment results with various parameters when the memory size is 2000 on Financial1 workload

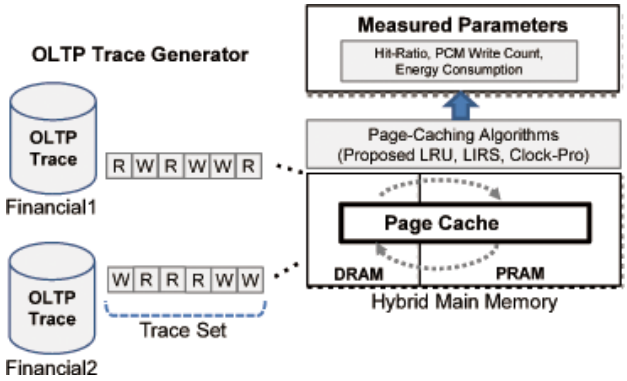


Figure 5. Trace-driven simulation for evaluation of page-caching algorithms

To evaluate the performance characteristics of the proposed page caching algorithm, we constructed a trace-driven simulation environment and coupled it to the OLTP traces. The objective of the simulator is to evaluate the performance of the page caching algorithms with regard to a real hybrid main memory and practical workload. Figure 5 shows the overall simulation environment.

As the workloads for the performance evaluation, we exploited an Online Transaction Processing (OLTP) application I/O, which has high throughput and is insert/update-intensive. We use two workload traces from the OLTP of two large financial institutions. These traces were made available courtesy of Ken Bates from HP, Bruce McNutt from IBM, and the Storage Performance Council [1]. These two traces are related to requests to a storage and suitable to test the page caching algorithm. The first financial workload has 9 million traces, and among them, about 80% of traces are write accesses. The second financial workload has 5 million traces and about 19% of traces are write accesses, as shown in Table 2. By using two financial workloads, we can test the performance with the cases of write-intensive and read-intensive workloads.

4.2 Evaluation Results

Table 2. A summary of the workloads used in this paper

Trace Name	Number of Requests	Ratio of Write-access pages
Financial1	9156833	80.59%
Financial2	5436256	18.95%

In this section, we show the trace-based simulation results in terms of hit ratio, write access count on PRAM and consumed energy in order to show the performance of our page caching algorithm. We compared the experimental results for our algorithm with those of the conventional page caching algorithms such as LRU, LIRS, and CLOCK-Pro.

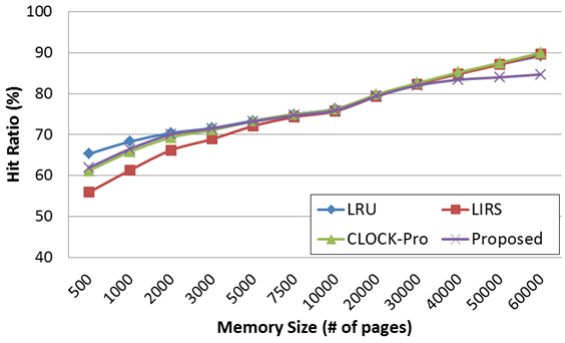
4.2.1 Parameters used in evaluation

In order to predict a page's access pattern, we use equation 1, which includes one parameter, α. In addition, we use two threshold values to determine the time when migration occurs and when a page cache changes its status between read-bound and write-bound pages. The two values are Tr_{mig} and Tr_q, explained in section 3.2. Before we test our page cache algorithm, we have to determine the values of these parameters in order to minimize the total access counts on PRAM.

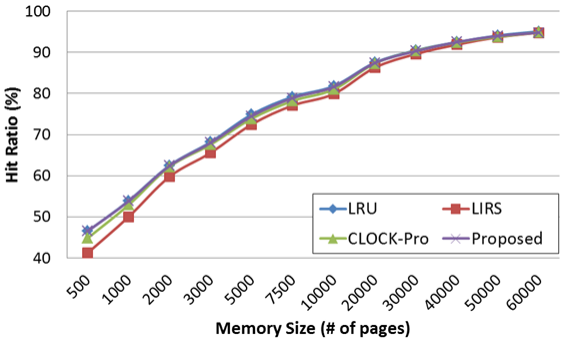
We chose the parameter values through experiments. Figure 6 shows the total number of write accesses on PRAM with the *Financial1* workload when we change α, Tr_{mig}, and Tr_q values. Figure 6(a) shows the total access count on PRAM with various α and Tr_{mig} when Tr_q is fixed at 0.35. From Figure 6(a), the evaluation results convince us that the write-access count is minimized when α is 0.5 and when Tr_{mig} is 0.45 or 0.5. Therefore, we select α as 0.5. Figure 6(b) shows the results when α is 0.5. From this graph, we can know that the results are the smallest when Tr_q is larger than 0.3. Finally, we select the values of Tr_{mig} and Tr_q as 0.5 and 0.35, respectively.

4.2.2 *Hit ratio*

The first experiment measures the hit ratio, which is important in determining the performance of the page caching algorithm. We compared the hit ratio of our page caching algorithm to the hit ratios of the conventional page caching algorithms. We tested it with many sizes of main memory. The results are shown in Figure 7.

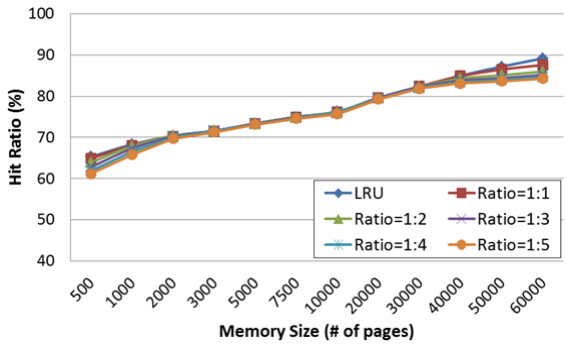


(a) *Financial1*

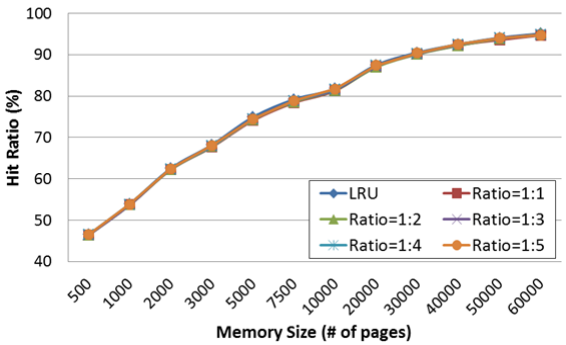


(b) *Financial2*

Figure 7. Hit ratio of proposed algorithm and conventional algorithms on financial workloads



(a) *Financial1*



(b) *Financial2*

Figure 8. Hit ratio of proposed algorithm with different PRAM-to-DRAM ratio on financial workloads

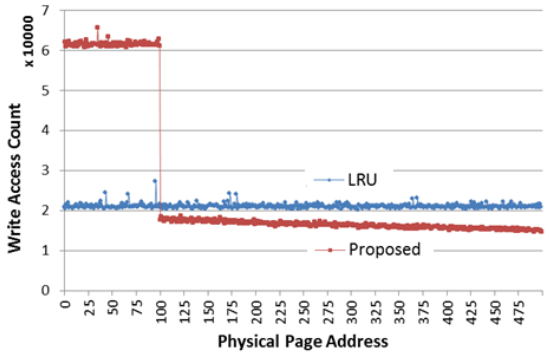
From the results, it can be seen that the LRU replacement algorithm shows the highest hit ratio through the whole range of memory sizes. While the hit ratio of our algorithm is lower than that of LRU, the results show that the hit ratio is similar to those of the LRU, LIRS, and CLOCK-Pro algorithms. Actually, we designed our algorithm on the basis of the LRU replacement algorithm and added the prediction and migration schemes. We expected the hit ratio of our algorithm to be similar to that of the LRU replacement algorithm. Because we designed the selected victim page for migration to simply be eliminated, as explained in section 3.2, it is possible that the page faults will occur more. Consequently, the hit ratio is lower than that of the LRU. Although the migration scheme causes a degradation of the hit ratio, the hit ratio is still larger than that of LIRS and CLOCK-Pro for small sizes of the main memory.

The next experiment measures the hit ratio with varying the ratio of the size of PRAM to the size of DRAM, as shown in Figure 8. We compare the hit ratio of the proposed scheme with the hit ratio of LRU replacement algorithm. In overall region, the hit ratio of the proposed scheme shows a similar hit ratio in comparison with LRU. In case of the results of *Financial1*, when the size of memory is set to 500 pages, the hit ratio of proposed scheme is decreased from 64.9% to 61.2% as the size of PRAM increases. This can be seen in Figure 8(a). It is due to the size reduction of DRAM, which leads to hit ratio degradation by the victim process when many migrations of write-bound pages occur. When the size of memory is set to 60,000 pages, the experiment result also shows a gap between hit ratio of proposed scheme and the hit ratio of LRU. The main reason is that even infrequently accessed pages can be remained to LRU list as the size of DRAM increases. On the other hand, the above pages in the proposed scheme can be frequently selected as victim so that the pages can be evicted from LRU list, which leads to more cache miss. In case of the results of *Financial2*, the hit ratio at 500-page size of a memory is barely changed. With read-intensive workload like *Financial2*, hit ratio degradation by the victim process of a read-bound page does not affect largely because the size of PRAM is larger than DRAM.

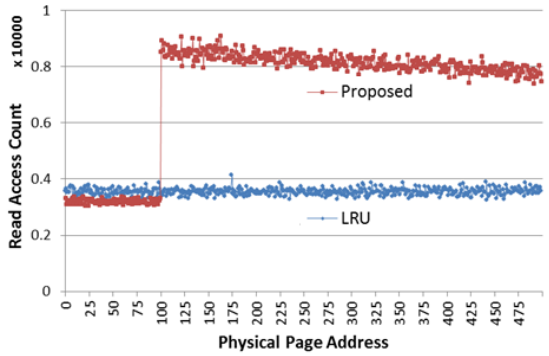
4.2.3 *Write access count on PRAM*

The write access count on PRAM is important because it is related to the total latency of the page cache and the lifetime of PRAM. PRAM has a very long latency compared to DRAM, so the performance of page caches degrades if many write pages hit on PRAM. In addition, PRAM wears out more quickly because of its

low endurance. Therefore, in order to use a page cache on the hybrid main memory of DRAM and PRAM, a page caching algorithm must be able to reduce the write access count on PRAM.



(a) Write access distribution



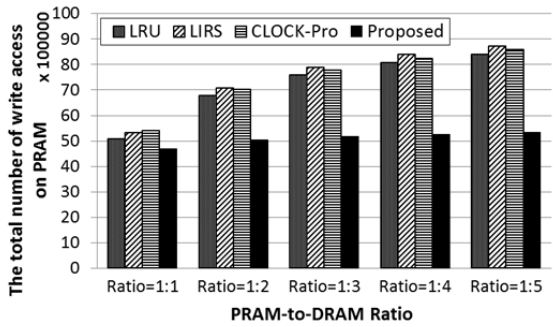
(b) Read access distribution

Figure 9. Write and read access distribution in the hybrid main memory

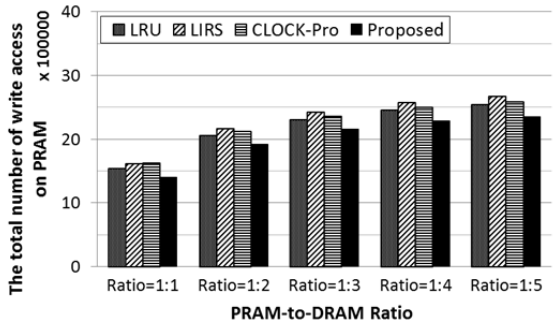
To show that the proposed scheme reduces the write access count on PRAM, and conversely increases the read access count on PRAM, an experiment was designed by profiling the value of the read/write access distribution when the start address was varied from 0 to 500. In this experiment, it was assumed that the first 100 blocks would be mapped to DRAM, and the other would be mapped to PRAM. As can be seen in Figure 9(a), the minimized write access distribution is suitable for PRAM characteristics. Conversely, as shown in Figure 9(b) the maximized read access distribution implies that the read-bound data is migrated to the PRAM region, and so, in comparison with the LRU scheme, the proposed page-caching algorithm is more suitable for a hybrid main memory system.

In this experiment, we evaluated the write access count on PRAM and compared it with the results of the conventional algorithms. During the simulation, we counted read and write accesses of DRAM and PRAM. Figure 10 shows the total number of write accesses on PRAM with workloads, denoted as *financial1* and *financial2*. We measured the count number with several memory sizes. The memory size is the total size of DRAM and PRAM. In Figure 10, the number of accesses for the four algorithms decreases as the memory size grows. Because the number of faults is decreased with the increase of the size of memory, the total number of write accesses on PRAM decreases when the memory size increases.

When using our algorithm, we can know that the total number of



(a) *Financial1*



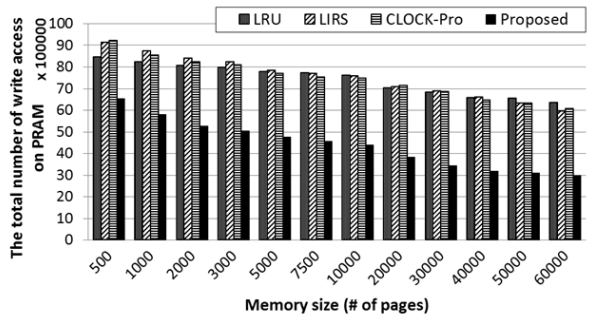
(b) *Financial2*

Figure 11. The total write access count with different PRAM-to-DRAM ratio when the memory size is 2000 on financial workloads

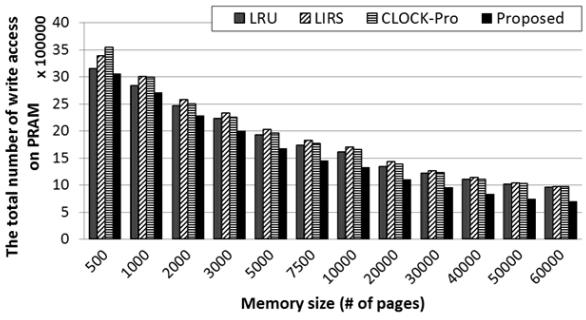
write accesses is reduced compared to that for the conventional page caching algorithm. We can reduce the total write access count on PRAM by 34.8% for *financial1* and 6.97% for *financial2* when we use the hybrid main memory with 2000-page sizes. We can reduce the total write access count by a maximum of 52.9% for *financial1* and 27.8% for *financial2*. The reason why the gain of *financial2* is smaller than that of *financial1* is that the workload of *financial2* has a small number of write accesses and we can obtain performance improvement by reducing the write access request on PRAM by using migration.

The next experiment shows the total write access count on PRAM when the ratio of the size of PRAM to the size of DRAM varies. Figure 11 shows the results when the total size of hybrid memory is 2000 pages. Because the size of PRAM is increased with the increasing the ratio of PRAM to DRAM, total write counts are increased with both two workloads. However, in case of *financial1*, the proposed algorithm can maintain the total number of write access in all cases while the results of other algorithms are increased, shown in Figure 11(a). Therefore, the proposed algorithm can efficiently decrease the total write access number with write-intensive workload like *financial1*. In case of *financial2*, although the total number of write access on PRAM is increase when using our algorithm, it can reduce the write counts

compared to other algorithms. We also evaluated the total number of write access with all cases of memory sizes and we find that the patterns of the total number of write access are the same through all cases of the PRAM-to-DRAM ratio.

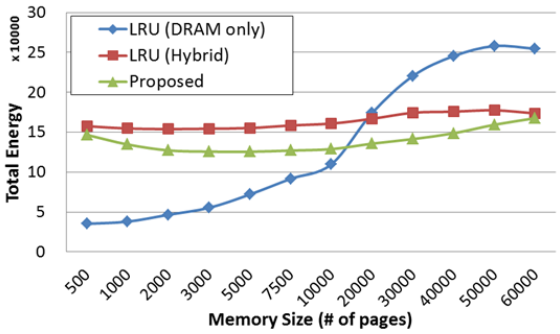


(a) *Financial1*

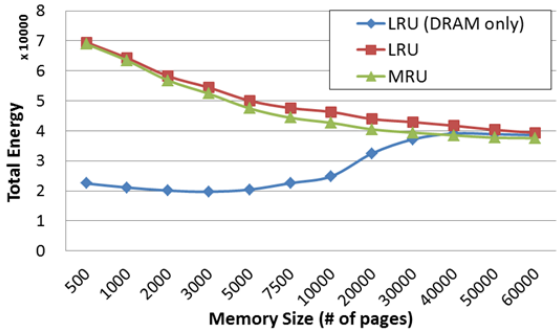


(b) *Financial2*

Figure 10. The total write access count of PRAM on financial workloads

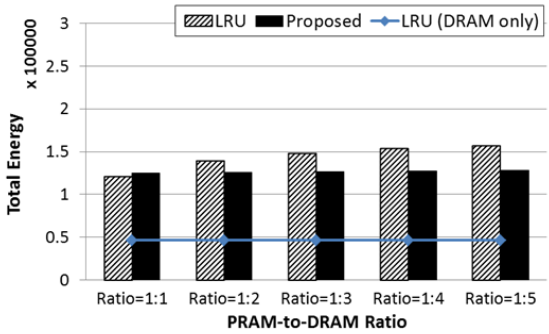


(a) *Financial1*

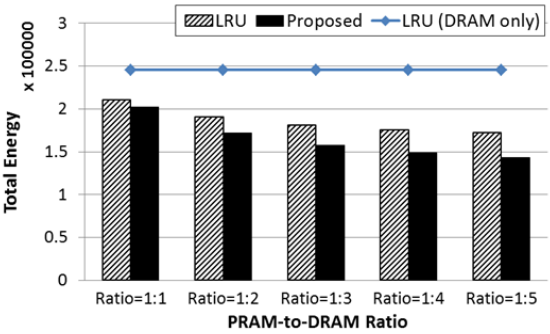


(b) *Financial2*

Figure 12. Consumed energy on memory of proposed algorithm and conventional algorithms on finalcial workloads



(a) 2000-page size of hybrid memory



(b) 40000-page size of hybrid memory

Figure 13. Consumed energy on the hybrid memory of proposed algorithm and LRU algorithm with different PRAM-to-DRAM ratios on finalcial1 workload

4.2.4 Total Energy

We measured the total energy, which consists of operation energy and idle energy. Operation energy is the energy when memory accesses occur. During the evaluation, we can obtain the total access count for the various memory sizes. By using measured count values, we can calculate the operation energy with the read and write energy of both DRAM and PRAM, as can be seen in table 1. The idle energy is the energy continuously consumed by memory devices even if they do not operate. Idle energy can be calculated according to the operation time and the idle power of each memory. We also measured the operation time, which is the time for running the workload. Figure 12 shows the consumed energy of a memory when using the proposed algorithm and the LRU algorithm. In the case of the LRU algorithm, we use hybrid memory and DRAM-only memory for comparison. DRAM shows

low read and write energy but has large idle power compared to PRAM. Therefore, by increasing the size of the memory, the power consumed by DRAM can be increased, as shown in Figure 12. From this result, we can use a page cache on the hybrid main memory in order to reduce the consumed energy if we use the memory with over 20000-page size. If we consider the results with the hybrid main memory, the proposed page caching algorithm uses lower energy than LRU does. Normally, PRAM write energy is large, but we can reduce the write counts on PRAM so that we can reduce the total energy. We can maximally reduce the total consumed energy by 19.9% for *financial1*.

Figure 13 shows the total energy which a memory consumes with various ratios of PRAM to DRAM, in which we selected two memory sizes. In case of 2000-page size which is lower than 20000-page size, the consumed energy by only DRAM with LRU is the lowest, as can be seen in Figure 13(a). As increasing the ratio of PRAM to DRAM, total energy consumed by memory with both LRU and proposed algorithms tend to increase. It is because the total number of write access on PRAM is increasing as increasing the ratio of PRAM to DRAM, as shown in Figure 11(a). However, the idle power of DRAM cannot affect the total energy because a size of DRAM is still small. In case of 40000-page size, DRAM size is large so that its consumed power occupies most of the total energy. Therefore, as increasing the ratio of PRAM to DRAM, total energy consumed by memory is decreasing even if the write-access counts on PRAM are increased, as shown in Figure 13(b). When comparing the consumed energies by the proposed and LRU algorithms, the proposed algorithm uses the lower power compared to the power by LRU through the all cases of PRAM-to-DRAM ratios because the proposed algorithm can reduce the total access counts on PRAM.

5. FUTURE WORK

As to future work, we need to evaluate our algorithm with more workloads. In our evaluation, we used only two workloads, which show the write-bound and read-bound features. Therefore, we have to explore our algorithm with various characteristics such as sequential read/write patterns or mixed write and read patterns.

We also used a trace-driven simulator to evaluate our algorithm with traces. For future work, we will apply our algorithm to the page cache algorithm of the Linux kernel code and evaluate it with benchmark programs like SPEC. In order to implement our algorithm, we will have to emulate PRAM because PRAM is not currently available. By analyzing the read/write time for PRAM, we can emulate PRAM with DRAM, albeit with software delay.

6. CONCLUSIONS

We propose a new page caching algorithm for a hybrid main memory. The hybrid main memory is organized by heterogeneous types of memories, which have different properties of the access latency, density, and endurance. In modern computer systems, a large amount of main memory is used as a page cache to hide disk access latency. Because the conventional page caching algorithms only deal with uniform access latency and endurance, a new page caching algorithm is needed. When using PRAM to make a hybrid main memory, the important things to consider are the long write latency and the low endurance of PRAM. Therefore, we proposed a page caching algorithm with page monitoring and migration schemes to keep read-bound access pages in PRAM and write-bound access pages in DRAM.

The experimental results show that out algorithm can reduce the total number of write accesses by a maximum of 52.9%. Our algorithm shows that a hit ratio is similar to the hit ratio of the conventional page caching algorithms, such as LRU, LIRS, and CLOCK-Pro. Our algorithm minimizes the write access of PRAM while maintaining the cache hit ratio. Therefore, we can enhance the average page cache performance and reduce the endurance problem in hybrid main memory. In addition, we can reduce the consumed energy by a maximum of 19.9%.

7. ACKNOWLEDGMENTS

The work presented in this paper was supported by MKE (Ministry of Knowledge Economy, Republic of Korea), Project No. 10035231-2010-01.

8. REFERENCES

[1] UMass TraceRepository, *http://traces.cs.umass.edu/index.php/Storage/Storage*.

[2] Aven, E. G. Coffmann, and I. Kogan. *Stochastic Analysis of Computer Storage*. Amsterdam: Reidel, 1987.

[3] S. Bansal and D. S. Modha. CAR: Clock with Adaptive Replacement. *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, 2004.

[4] L. A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, December 2007.

[5] R. W. Carr and J. L. Hennessy. WSClock - A Simple and Effective Algorithm for Virtual Memory Management. *Proceedings of the eighteh ACM symposium on Operating systems principles (SOSP'81)*, pages 87–95, 1981.

[6] E. G. Coffman and P. J. Denning. *Operating Systems Theory*. Prentice-Hall, 1973.

[7] F. J. Corbato. A Paging Experiment with the Multics System. *MIT Project MAC Report MAC-M-384*, May 1968.

[8] Dan and D. Towsley. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. *Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, pages 143–152, April 1990.

[9] P. J. Denning. The Working Set Model for Program Behavior. *Communications of the ACM*, 11(5):323–333, May 1968.

[10] G. Dhiman, R. Ayoub, and T. Rosing. Pdram:a hybrid pram and dram main memory system. *In Proceedings of the 46th Annual Design Automation Conference*, July 2009.

[11] E. Horowitz, D. D. Sleator, and R. E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28(2):202–208, February 1985.

[12] S. Jiang and X. Zhang. Lirs: An efficient low inter-reference recency set replacement policy to improve buffer cache performance. *In Marina Del Rey*, pages 31.42. ACM Press, 2002.

[13] T. Johnson and D. Shasha. 2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm. *Proceedings of the 20th VLDB Conference*, pages 297.306, 1994.

[14] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting Phase Change Memory as a Scalable DRAM Alternative. *Proceedings of the 36th annual international symposium on Computer architecture*, June 2009.

[15] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim. LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies. *IEEE Transactions on Computers*, 50(12):1352.1361, December 2001.

[16] E. J. O'Neil, P. E. O'Neil, and G. Weikum. The LRU-K Page Replacement Algorithm for Database Disk Buffering. *Proceedings of the ACM SIGMOD international Conference on Management of data*, pages 297.306, May 1993.

[17] K. H. Park, Y. Park, W. Hwang, and K.-W. Park. Mn-mate: Resource management of manycores with dram and nonvolatile memories. *12th IEEE International Conference on HPCC*, September 2010.

[18] Y. Park, S. K. Park, and K. H. Park. Linux kernel support to exploit phase change memory. *Linux Symposium*, July 2010.

[19] S. J. Performance and S. Jiang. Clock-pro: An effective improvement of the clock replacement. In *Proceedings of USENIX Annual Technical Conference*, 2005.

[20] M. K. Qureshi, V. Srinivassan, and J. A. Rivers. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, June 2009.

[21] J. T. Robinson and M. V. Devarakonda. Data Cache Management Using Frequency-Based Replacement. *Proceedings of ACM SIGMETRICS conference*, pages 134.142, 1990.

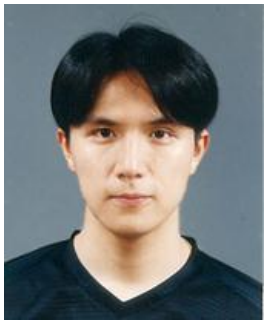
[22] H. Seok, Y. Park, and K. H. Park. Migration Based Page Caching Algorithm for a Hybrid Main Memory of DRAM and PRAM. *Proceedings of the 2011 ACM Symposium on Applied Computing*, March 2011.

[23] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie. Hybrid Cache Architecture with Disparate Memory Technologies. *Proceedings of the 36th annual international symposium on Computer architecture*, June 2009.

[24] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology. *Proceedings of the 36th annual international symposium on Computer architecture*, June 2009.

[25] Y. Zhou and J. F. Philbin. The Multi-Queue Replacement Algorithm for Second Level Buffer Caches. *Proceedings of the USENIX Annual Technical Conference*, pages 91–104, June 2001.

ABOUT THE AUTHORS:



Hyunchul Seok received the BS degree in Electrical Engineering from Pohang University of Science and Technology (POSTECH) in 2001 and the MS degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2008. He is currently working toward the PhD degree in the division of electrical engineering at KAIST. His research interests include storage systems, memory systems, and embedded systems.

Youngwoo Park received the BS and MS degrees in the division of electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2004 and 2006, respectively. He is currently working toward the PhD degree in the division of electrical engineering at KAIST. His research interests include storage systems, flash file systems, and embedded systems. He is a member of the IEEE and the IEEE Computer Society.

Ki-Woong Park received the BS degree in computer science from Yonsei University in 2005 and the MS degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2007. He is currently working toward the PhD degree in the Division of Electrical Engineering at KAIST. His research interests include cloud computing systems, security protocol, and network security. . He received a 2009–2010 Microsoft Graduate Research Fellowship. He is a member of the ACM, IEEE and the IEEE Computer Society.

Kyu Ho Park received the BS degree in electrical engineering from Seoul National University, Korea in 1973, the MS degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 1975, and the DrIng degree in electrical engineering from the University de Paris XI, France in 1983. He has been a Professor of the Division of Electrical Engineering at KAIST since 1983. He was a president of the Korea Institute of Next Generation Computing for the period 2005-2006. His research interests include computer architectures, storage systems, cloud computing, and parallel processing. He is a member of ACM and IEEE.

Template-based autonomous navigation and obstacle avoidance in urban environments

Jefferson R. Souza, Daniel O. Sales, Patrick Y. Shinzato,
Fernando S. Osorio and Denis F. Wolf
Mobile Robotics Laboratory, University of Sao Paulo (USP)
Av. Trabalhador Sao-Carlense, 400 – P.O. Box 668 – 13.560-970, Sao Carlos, Brazil
{jrsouza, dsales, shinzato, fosorio, denis}@icmc.usp.br

ABSTRACT
Autonomous navigation is a fundamental task in mobile robotics. In the last years, several approaches have been addressing the autonomous navigation in outdoor environments. Lately it has also been extended to robotic vehicles in urban environments. This paper presents a vehicle control system capable of learning behaviors based on examples from human driver and analyzing different levels of memory of the templates, which are an important capability to autonomous vehicle drive. Our approach is based on image processing, template matching classification, finite state machine, and template memory. The proposed system allows training an image segmentation algorithm and a neural network to work with levels of memory of the templates in order to identify navigable and non-navigable regions. As an output, it generates the steering control and speed for the Intelligent Robotic Car for Autonomous Navigation (CaRINA). Several experimental tests have been carried out under different environmental conditions to evaluate the proposed techniques.

Categories and Subject Descriptors
I.2.9 [**Artificial Intelligence: Robotics**]: Autonomous Vehicles.

General Terms
Algorithms, Performance, Design, Experimentation.

Keywords
Robotic Vehicles Navigation, Obstacles Avoidance, Template Matching, FSM and Urban Environments.

1. INTRODUCTION
Human driver errors are a major cause of accidents on roads. Frequently people get injured or even die due to road traffic accidents (RTA). Also, bad road and weather conditions increase the risk of RTA. Autonomous vehicles could provide safer conditions in roads for individual or collective use. They also could increase efficiency in freight transportation and provide some degree of independence to people unable to drive.

Research in mobile robotics has reached significant progress in the last 10 years. Part of them focus on autonomous navigation, which is a fundamental task in the area [19]. Lately, several works have been improving on navigation in outdoor environments. Competitions like DARPA Challenges [6] and ELROB [3] have been pushing the state of the art in autonomous vehicle control.

Figure 1. CaRINA test platform.

The most relevant results obtained in such competitions combine information obtained from a large number of complex sensors. Some approaches use five (or more) laser range finders, video cameras, radar, differential GPS, and inertial measurement units [6], [12]. Although there are several interesting applications for such technology, the cost of such systems is very high, which is certainly prohibitive to commercial applications.

In this paper we propose a vision-based navigation approach for urban environments, based on a low cost platform.

Our system uses a single camera to acquire data from the environment. It detects the navigable regions (roads), estimates the more appropriate maneuver, acquires and trains different levels of memory of the templates that should be done in order to keep the vehicle in a safe path, and finally, control steering and acceleration of the vehicle. Figure 1 presents our CaRINA test platform.

Our approach is based on two Artificial Neural Networks (ANNs). The first one identifies navigable regions in which a template-based algorithm classifies the image and identifies the action that should be taken by CaRINA. The images are acquired and then processed using ANNs that identifies the road ahead of the vehicle. After that, a Finite State Machine (FSM) is used to filter some input noise and reduce classification and/or control errors. In this work noise is considered as variations in the road color,

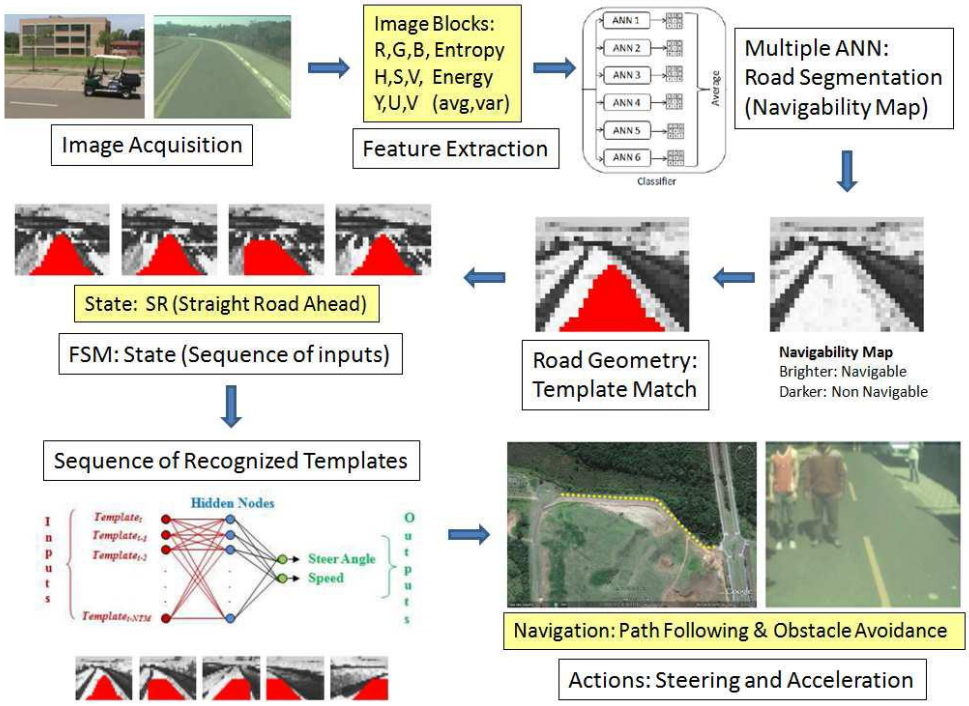


Figure 2. General outline of the autonomous navigation system.

such as dirt road (mud or dust), shadows, and depressions. After obtaining the current state (template), which is the input of a new ANN that works with levels of memory of the templates (LMT). This ANN aims to learn the driver's behavior, providing smoother steering and levels of speed in the same way as the driver. We considered six levels of template memory on the ANN searching to obtain the topology which provides more reliable results.

A sequence of states associated with an action is learned by the second ANN. States comes from situations identified by the Templates and FSM, and are mapped into steering angle and vehicle speed. Figure 2 shows a general outline of this system.

2. RELATED WORKS

Autonomous Land Vehicle in a Neural Network (ALVINN) [13] is an ANN based navigation system that calculates a steer angle to keep an autonomous vehicle in the road limits. In this work, the gray-scale levels of a 30 x 32 image were used as the input of an ANN. In order to improve training, the original road image and steering were generated, allowing ALVINN to learn how to navigate in new roads. The disadvantages of this work are the low resolution of a 30 x 32 image (gray-scale levels) and the high computational time. The architecture has 960 input units fully connected to the hidden layer to 4 units, also fully connected to 30 units in output layer. Regarding that issue, this problem requires real time decisions therefore this topology is not efficient.

Later, the EUREKA project Prometheus [7] for road-following was successfully performed using image based solutions, which provided trucks with an automatic driving system to reproduce drivers in repetitious long driving situations. The system also included a function to warn the driver in dangerous situations. A limitation of this project was an excessive number of heuristics created by the authors to limit the false alarms caused by shadows or discontinuities in the color of the road surface.

In the work [8], an outdoor mobile robot learns avoiding collisions by observing a human driver, the test platform is a vehicle equipped with sensors (laser, GPS and IMU) that produce a map of the local environment. This approach presents a method for automatically learning the parameters of the control model. The input to the control model is a goal point and a set of obstacles defined as points in the x-y plane of the vehicle. Three different approaches (Random, Genetic Algorithm (GA) and Simulated Annealing (SA)) were used in the learning step. The GA and Random performed similarly well. SA was worse than the other techniques. Also the authors compared the randomly learned parameter set with the principal component analysis, and observed than the random generalized well. The issue of this approach is the dependency of goal point and the set of obstacles, and all the parameters were limited to a fixed range.

Chan et al. [5] shows an Intelligent Speed Adaptation and Steering Control that allows the vehicle to anticipate and negotiate curves safely. This system uses Generic Self-Organizing Fuzzy Neural Network (GenSoFNN-Yager) which includes the Yager inference scheme [11]. GenSoFNN-Yager has as main feature their ability to induce from low-level perceptual information in form of fuzzy if-then rules. Results show the robustness of the system in learning from example human driving negotiating new unseen roads. The autonomous driver demonstrates that anticipation is not always sufficient. Moreover, large variations in the distribution of the rule were observed, which imply a high complexity of the system.

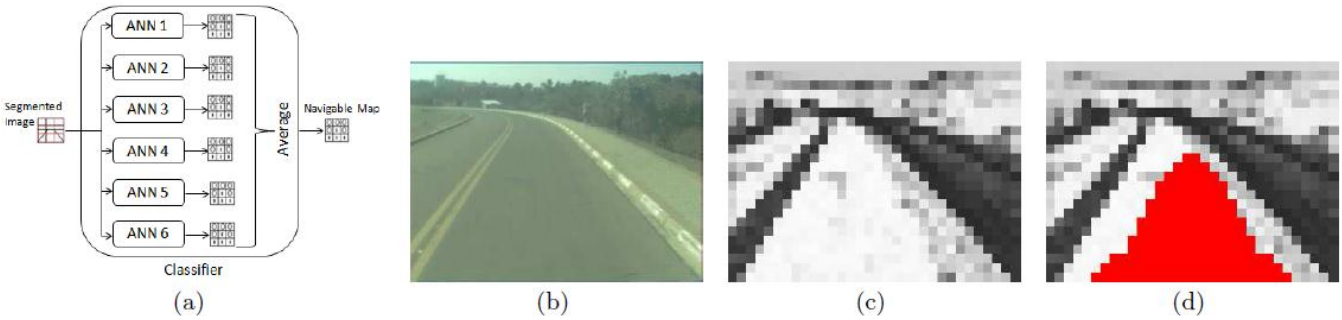


Figure 3. Classifier structure (a), real image (b), image processing step (c) and template matching (d).

Shihavuddin et al. [14] approaches the path map generation of an unknown environment using a proposed trapezoidal approximation of road boundary. At first, a blind map of the unknown environment is generated in computer, and then the image of the unknown environment is captured by the vehicle and sent to the computer using a RF transmitter module. After that, the image is pre-processed and the road boundaries are detected using the trapezoidal approximation algorithm. During this process, the vehicle operates independently avoiding the obstacles. The issue of this approach is the dependency of the camera tilt angle, because the vehicle moves through of the trapezium and reaches the next approximated trapezium having a previously tilt angle.

The work [10] focus on the task of lane following, where a robot-car learns anticipatory driving from a human and visual sensory data. During the learning steps the robot associates visual information with human actions. This information is derived from the street lane boundary that is detected in each image in real-time. In this work two modules were used, a reactive controller (RC) and a planner, where the former generates maps short-term information to a single steering control value and the latter generates action plans, i.e. sequences for steering and speed control. The final steering command is a combination of planner and RC output. The advantages of this approach are react to upcoming events, cope with short lacks of sensory information, and use these plans for making predictions about its own state, which is useful for higher-level planning. Despite many advantages, due to the inertia of the robot it is less visible than what could be expected from the plotted signal. Also the system is not able to predict future states.

Stein and Santos [18], proposes a method to compute the steering of an autonomous robot, moving in a road-like environment. The proposed system used ANNs to learn behaviors based on examples from human driver, replicating and sometimes even improving human-like behaviors. To validate the created ANNs, real tests were performed and the robot successfully completed several laps of the test circuit showing good capacities for recovery and for generalization with relatively small training data.

Markelic et al. [9], proposes a system that learns driving skills based on a human teacher. Driving School (DRIVSCO) is implemented as a multi-threaded, parallel CPU/GPU architecture in a real car and trained with real driving data to generate steering and acceleration control for road following. Besides, it uses an algorithm for detecting independently moving objects (IMOs) for spotting obstacles with stereo camera. A predicted action sequence is compared to the driver actions and a warning is issued if they are differing too much (assistance system). The IMO detection algorithm is more general in the sense that it will respond not only to cars, but to any sufficiently large (11 x 11 pixels) moving object. The steering prediction is very close to the human signal, but the acceleration is less reliable.

3. PROPOSED METHOD

Our approach is composed by 4 steps. In the first step an image is obtained and the road is identified using ANNs (Figure 3 (c)). In the second step, a template matching algorithm is used to identify the geometry of the road ahead of the vehicle (straight line, soft turn, or hard turn). In the third step, a FSM is used to filter noisy inputs and any classification error. Finally, a template memory is used in order to define the action that the vehicle should take to keep on road. These steps will be described in the next sub-sections.

3.1 Image Processing Step

We adopted the proposed method of Shinzato [15], which proposes to use ANNs to be applied into a road identification task. Based on the results, a system composed by six Multilayer Perceptron (MLP) ANNs was proposed to identify the navigable regions in urban environments (Figure 3 (a)). The real image can be seen on Figure 3 (b). The result of this ANNs output combination is a navigability map (Figure 3 (c)). The brighter blocks are the more likely area to be considered navigable. This step divides an image into blocks of pixels and evaluates them as single units. The advantage of this approach is that one can train the ANNs to identify different types of navigable and non-navigable regions (e.g. pavemented and non-pavemented roads).

Initially, the image processing step divides the image into blocks of pixels and evaluates then as single units. Several features are calculated for each block, such as: pixel attributes like red, green, blue (RGB) average, image entropy and others features obtained from this collection of pixels (region block). In the grouping step, a frame with (*M*x*N*) pixels resolution was sliced in groups with (*K*x*K*) pixels. Suppose an image represented by a matrix I of size (*M*x*N*). The element *I(m,n)* corresponds to the pixel in row m and column n of image, where *(0 <= m < M)* and *(0 <= n < N)*. Therefore, group *G(i,j)* contains all the pixels *I(m,n)* such that *((i*K) <= m < ((i*K)+K))* and *((j *K) <= n < ((j *K)+K))*. This strategy has been used to reduce the amount of data, allowing faster processing.

Once a block is processed, its attributes are used as inputs of the ANNs. The ANNs are used to classify the blocks considering their attributes (output 0 to non-navigable and 1 to navigable). Each

ANN contains an input layer with the neurons according to the image input features (see Table 1), one hidden layer with five neurons, and the output layer has only one neuron (binary classification). However, after the training step, the ANN returns real values between 0 and 1, as outputs. This real value can be interpreted as the classification certainty degree of one specific block. The main difference between the six ANNs is the set of image attributes used as input. All these sets of attributes (see Table 1) are calculated during the block-segmentation of the image. The choice of these attributes was based on the results presented in the work [15].

Table 2. Input attributes of the ANNs (R, G, B = red, green, blue components; H, S, V = hue, saturation, value components; Y, U, V = Luminance, average = av, normalized = norm, entropy = ent, energy = en and variance = var).

ANNs	Input attributes
ANN1	U av, V av, B norm av, H ent, G norm en and H av
ANN2	V av, H ent, G norm en, G av, U av, R av, H av, B norm av, G norm av and Y ent
ANN3	U av, B norm av, V av, B var, S av, H av, G norm av and G norm ent
ANN4	U av, V av, B norm av, H ent, G norm en and H av
ANN5	V av, H ent, G norm en, G av, U av, R av, H av, B norm av, G norm av and Y ent
ANN6	U av, B norm av, V av, B var, S av, H av, G norm av and G norm ent

After obtaining the six outputs of the ANNs referring to each block, the classifier calculates the average of these values to compose a single final output value. These values representing each block obtained from the original image form together the navigability map matrix (Figure 3(c)). This matrix is used to locate the most likely navigable region. It is important to mention that the ANN is previously trained using supervised examples of navigable and non-navigable regions selected by the user one time on an initial image frame. After, the trained ANN is integrated into the vehicle control system and used as the main source of information to the autonomous navigation control system.

3.2 Template Matching Step

After obtaining the ANN classification, 7 different road templates are placed over the image in order to identify the road geometry (some examples of these templates are presented in Figures 6, 7 and 8). One of them identifies a straight road ahead, two identify a straight road in the sideways, two identify soft turns, and two identify hard turns (e.g. a straight road ahead Figure 3 (d)). Each template is composed by a mask of 1s and 0s as proposed in [17]. The value of each mask is multiplied by the correspondent value into the navigability matrix (values obtained from the ANN classification of the correspondent blocks of the image). The total score for each template is the sum of products. The template that obtains the higher score is selected as the best match of the road geometry. Only one template can obtain a high score, because we use probabilities as the decision criteria.

3.3 Finite State Machine Step

The FSM uses the result of the template matching step as input, which carries out a classification for the road detected in each captured frame. This classification is defined by the template which best fits the matrix and its position. The developed FSM is composed by 5 states (straight road, soft turns left and right, and hard turns left and right) as shown on Table 2.

Table 2. States of the FSM used for the experimental tests.

States	Associated Action
SR	Keep steering wheel at center position
SLT	Smoothly turn the steering wheel to left
SRT	Smoothly turn the steering wheel to right
LT	Fully turn the steering wheel to left
RT	Fully turn the steering wheel to right

Figure 4 shows a partial scheme of the developed FSM. Transition 'a' represents classification of an input as a transition in direction of State 2, and 'b' classification as a transition in direction of State 1. Figure 4 represents a state change of state 1 to state 2. For example, 'a' represents a straight road state and 'b' soft turns left. To change the state in the FSM there must happen three consecutive equal states. In this work, we use the FSM with only 2 intermediate transitions between the states and have produced reasonable results. Detailed information can be seen in [17].

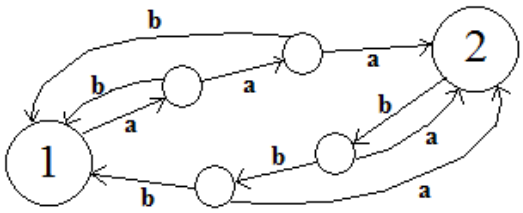


Figure 4. Transition between 2 states with 2 intermediate states.

The full FSM was designed with 5 states, which is obviously more complex to represent, but the partial scheme presented in Figure 4 still remains representative of the full FSM scheme. A transition between the current states to any other state occurs only after detecting a sequence of 'n + 1' identical inputs leading to the new state, where 'n' is the established number of intermediate states. The number of intermediate states varies according to the noise level in the images and navigability matrix representing the road, and also depends of the frame rate. In a system based on a high image acquisition frame rate, more misclassified inputs could be generated per time unit, so more intermediate states can be needed to discard some of these bad/noisy inputs. Environments with low level of input noise are associated to a smaller number of intermediate states.

3.4 Template Memory Step

After obtaining the current state (template) by FSM, this current template is used as input in the template memory step. In this step, the levels of memory of the templates are stored in a queue, as $\{Template_t, Template_{t-1}, Template_{t-2}, \ldots, Template_{t-NTM}\}$. In this work, the $Template_t$ represents the current template, $Template_{t-1}$ the previous template, $Template_{t-2}$ one template before the previous. This is done successively, until the number of template memory (NTM) is reached, where t represents the time

In this step, a second ANN is used differently of the ANNs used in the image processing step. The basic network structure (Figure

5) used is a feed-forward MLP, the activation function of the hidden neurons is the sigmoid function and the ANN learning is the resilient backpropagation (RPROP). The inputs are represented by templates memory and the outputs are the steer angle and speed.

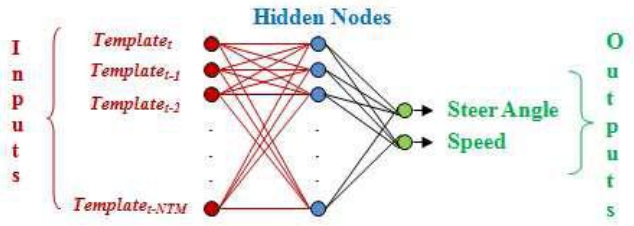


Figure 5. Structure of the second ANN used to generate steering and acceleration commands.

4. EXPERIMENTAL RESULTS

The experiments were performed using CaRINA (Figure 1), an electric vehicle capable of autonomous navigation in a urban road, equipped with a VIDERE DSG video camera, a ROBOTEQ AX2580 motor controller for steering control, an ARDUINO DUEMILANOVE is a microcontroller board used for vehicle speed control and a GARMIN 18X-5Hz GPS. The GPS was used only to register the log of the vehicle trajectories, and it was not used as input of the system. The image acquisition resolution was set to (320 x 240) pixels. The ANNs of the image processing step were executed using Fast Artificial Neural Network (FANN) [1], which is a free open source library which implements MLP ANNs in C, and the ANN in the template memory step was used Stuttgart Neural Network Simulator (SNNS) [2], which is also a software simulator for ANNs. For the development of the image acquisition, image processing and template matching algorithm, we used the OpenCV [4] library.

The results are divided into three scenarios. The first one shows an urban area without obstacles using only the steering control without template memory. The second scenario shows an urban area with obstacles (e.g. people, traffic cones) using the steering and speed control without template memory. The third scenario shows an urban area without obstacles, but uses the supervised learning (vehicle control system capable of learn behaviors based on examples obtained from human driver).

The results of the first scenario are described in two experiments performed with our vehicle[2]. In the first one, the vehicle should follow the road pavement (asphalt surface), including straight, soft turn and hard turn path segments. In the second experiment the vehicle should navigate into a narrow straight path segment, following a surface composed by a red pavement (red bricks). Both trajectories can be seen in Figure 10 (yellow lane - 1st Experiment and red lane - 2nd Experiment). The first experiment is the longer path, and the second one is the shorter and narrower straight path, both described in terms of their absolute GPS coordinates. In both cases the vehicle was able to keep itself in the road successfully, following the road path defined by the region previously indicated and trained as the navigable surface.

Figure 10. GPS trajectories (Scenario 1).

In order to obtain a quantitative analysis of the system, 30 representative frames of the defined states for this problem have been manually classified and compared to the outputs of the proposed algorithms. As shown in Table 3, satisfactory results were obtained (67.4% overall correct classification result). Most errors are due to small variations in the classification, and occurred between neighbor states/templates: left turn (LT) and soft left turn (SLT); right turn (RT) and soft right turn (SRT).

Table 3. Steering command results from Scenario 1 (first experiment).

States	SR	LT	RT	SLT	SRT
SR	**93.4%**	0.0%	0.0%	6.6%	0.0%
LT	20.0%	**53.4%**	0.0%	26.6%	0.0%
RT	13.3%	0.0%	**83.4%**	0.0%	3.3%
SLT	46.6%	0.0%	0.0%	**53.4%**	0.0%
SRT	46.6%	0.0%	0.0%	0.0%	**53.4%**

Figure 6 shows an example of the proposed identification method that resulted in 15 total state transitions (ST) detections (considering only the present frame), during a total sequence of 60 frames. The proposed FSM reduced these 15 transitions to only 4 effective ST. One example of a wrong frame discarded by FSM is showed in Figure 6 (c), where the current state was kept. Even when traversing a crosswalk the vehicle successfully keep moving in the correct direction.

Figure 7 presents an example of straight, soft left and right turns, and hard left and right turns being accurately identified by our system, with successfully navigation inside the safe region defined by the pavement paths (navigable regions). Figure 8 shows an example of a straight path being accurately identified by our system, with successfully navigation following the path defined by the red bricks surface.

[2] Experiment videos available in the Internet (Scenario 1):
1st. - http://www.youtube.com/watch?v=boJ_jRmtclU/
2nd. - http://www.youtube.com/watch?v=aJd31XoL6vA/

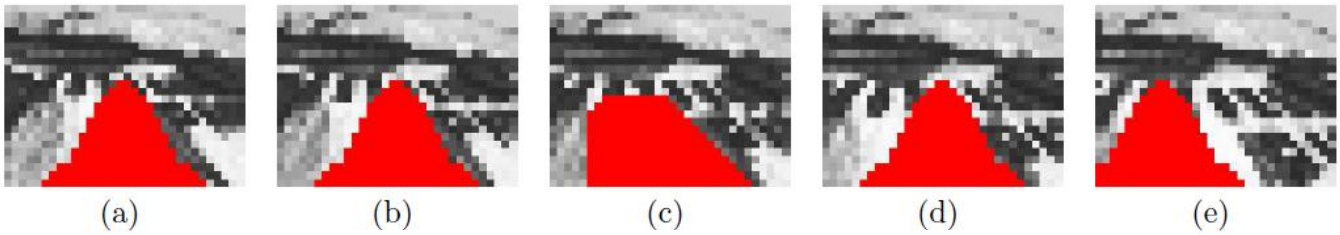


Figure 6. Road identification. (a), (b), (d), and (e) are straight road. (c) is soft left turn.

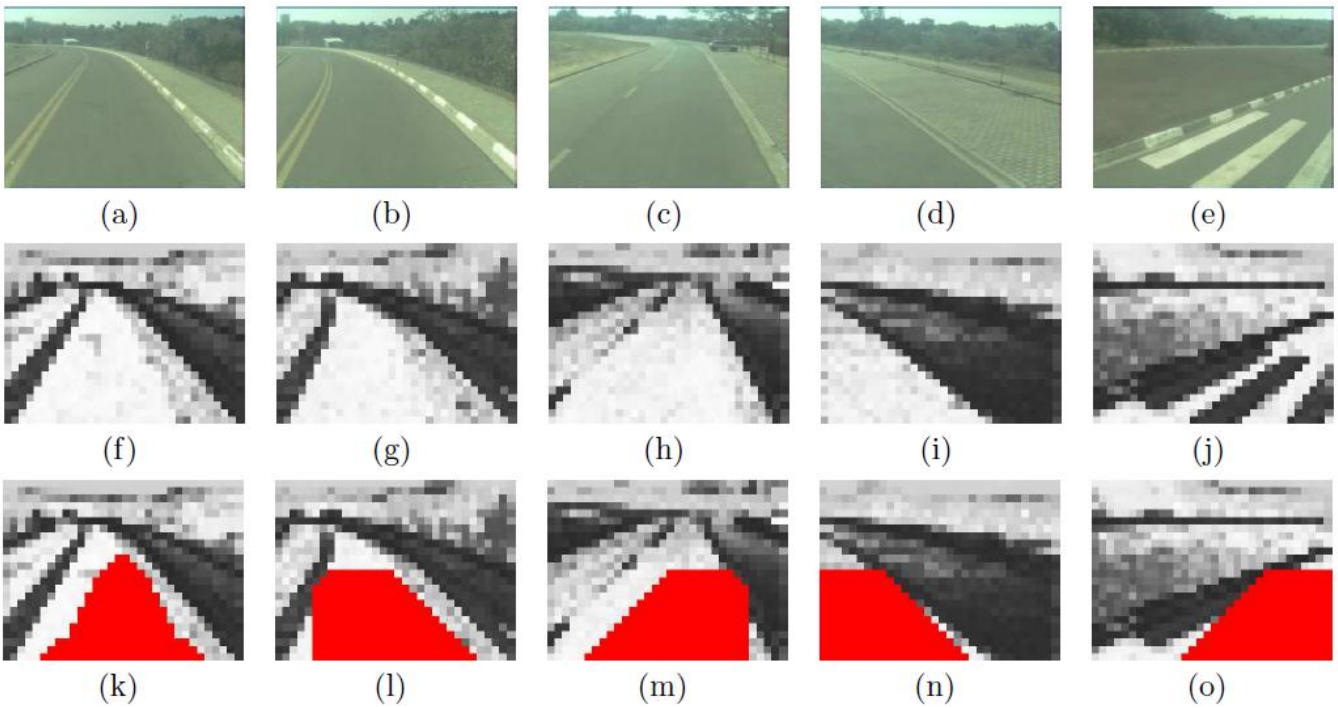


Figure 7. Experimental results of Scenario 1 (first experiment). Original Image (a), (b), (c), (d) and (e). Classification by ANNs (f), (g), (h), (i) and (j). Command output (k), (l), (m), (n) and (o).

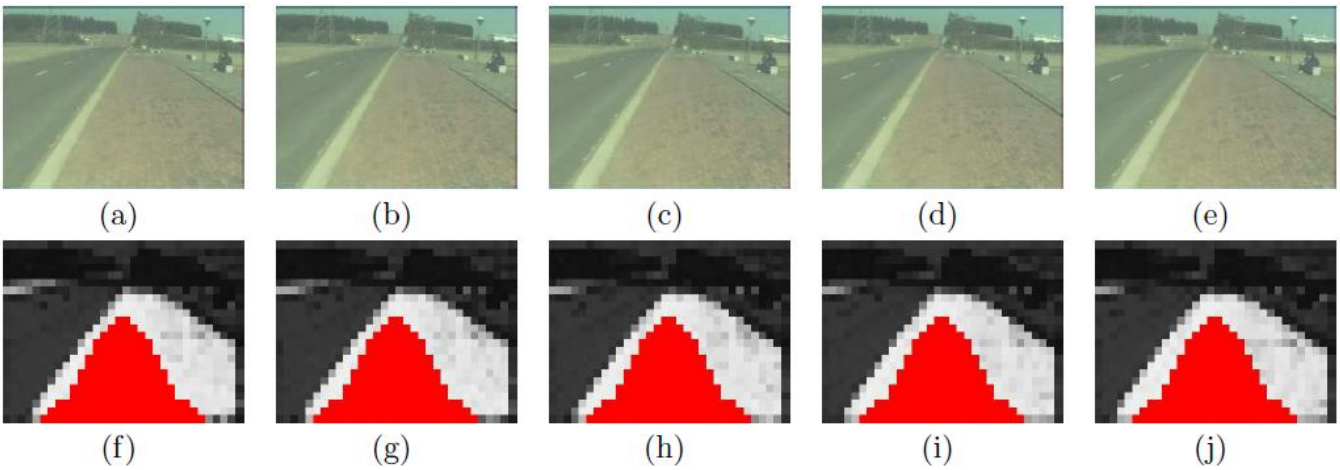


Figure 8. Experimental results of Scenario 1 (second experiment). Original Image (a), (b), (c), (d) and (e). Classification by ANNs (f), (g), (h), (i) and (j).

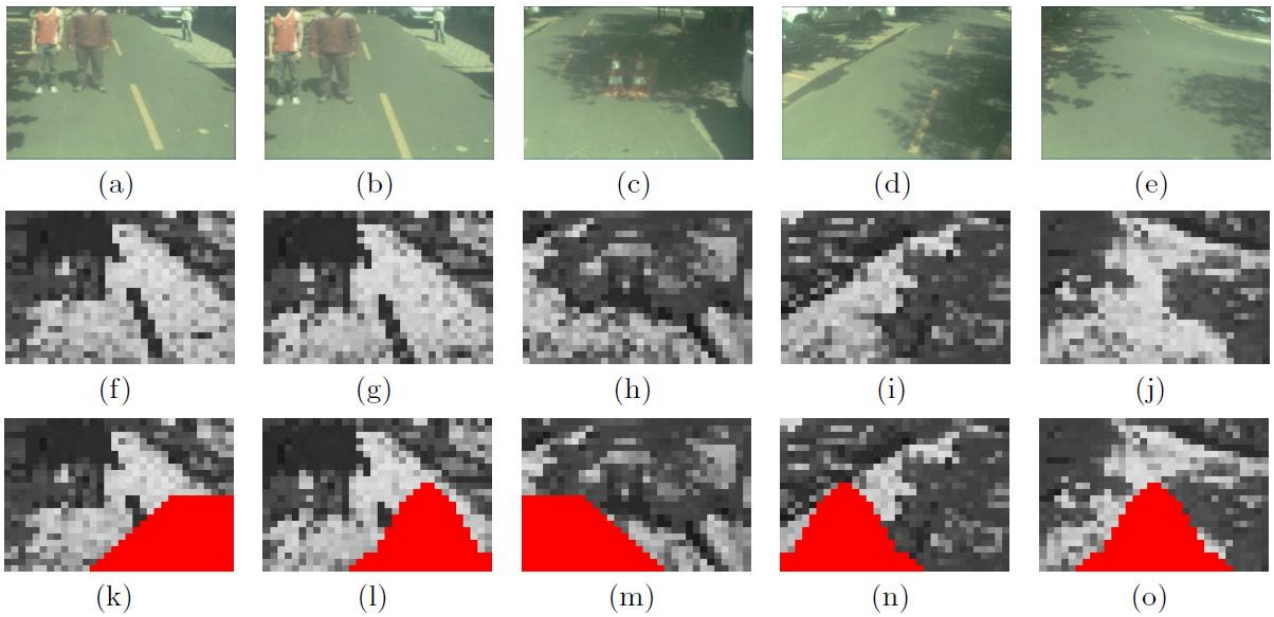


Figure 9. Experimental results of Scenario 2. Original Image (a), (b), (c), (d) and (e). Classification by ANNs (f), (g), (h), (i) and (j). Command output (k), (l), (m), (n) and (o)

The results of the second scenario were performed with the CaRINA platform[3]. The vehicle should keep on the urban road, spotting of obstacles (e.g. people, traffic cones), including straight, soft turn and hard turn path segments. The GPS trajectories using the vehicle can be seen in Figure 11.

Figure 11. GPS trajectory (Scenario 2).

Figure 9 shows in details the experiment of scenario 2, including an example of straight, soft left and right turns, and hard left and right turns being identified by our system, performing successfully the navigation inside the safe region defined by the urban path and the obstacles avoidance (Figure 9 (a), (b) and (c)).

[3] Experiment video available in the Internet (Scenario 2): http://www.youtube.com/watch?v=U2rRzNUPU8Y

The results of the third scenario also were performed with the CaRINA platform [16]. The vehicle should keep on the urban environment (road), replicating the human behavior based on examples from human driver. The GPS trajectory can be seen in Figure 12.

Figure 12. GPS trajectory (Scenario 3).

Table 4 shows the obtained values of the performed path by CaRINA for supervised learning. In Table 4, we analyze six LMT, which represent the architecture of the second ANN used in our proposed system. The numbers of LMT column represents the values that we tested randomly in order to develop a well-defined architecture. Half, Equal and Double shows the different architectures, for example, LMT = 3, where changes occur in the number of neurons in the intermediate layer of a MLP architecture RPROP (a learning heuristic for supervised learning in feed-

forward ANNs), tested the architectures: 3-1-2 (Half), 3-3-2 (Equal) and 3-6-2 (Double) (half the size of the ANN input layer equal the size of the ANN input layer double the size (in neurons) of the ANN input layer), obtaining the values of mean squared error (MSE), epoch (optimal point of generalization (OPG), i.e., minimum training error and capacity of maximum generalization) and the number of best ANN (because were initialized 5 different random seeds).

Table 4. Best results from MSE (M), Epoch (E) and ANN (A) for the six levels of template memory.

LMT	Half			Equal			Double		
	M	E	A	M	E	A	M	E	A
3	45.599	25	4	45.837	40	1	45.847	55	3
5	51.441	45	3	51.404	25	2	51.362	25	5
8	43.601	15	1	45.067	60	2	44.381	15	1
10	48.969	15	3	51.334	55	4	49.431	10	5
15	43.481	60	5	42.941	15	5	**42.233**	20	4
20	52.133	20	4	52.253	45	1	52.268	50	2

After the analysis of data on Table 4, the architecture 15-30-2 showed the lowest MSE for the epoch 20.

Table 5 presents the largest initial values of MSE for all LMT, based on Table 4 that shows the lowest values of MSE. Analyzing the best ANNs learned, reducing the MSE until the OPG. The architectures (3-1-2, 5-2-2, 8-4-2, 10-5-2, 15-7-2 and 20-40-2) showed the largest initial values of the MSE in order a reduction of high MSE.

Table 5. Best initial values of MSE.

MSE			
LMT	**Half**	**Equal**	**Double**
3	**1060.730**	929.475	762.547
5	**1004.350**	825.228	606.680
8	**858.441**	662.772	463.422
10	**847.994**	622.916	472.747
15	**714.645**	467.323	693.573
20	603.171	463.863	**1385.200**

Table 6 shows the percentage of the MSE for all architectures of the second ANN proposed based on Tables 4 and 5. The best architecture (15-30-2) is shown on Table 6, presenting the approximately error 5.9%. The lowest MSE of the Table 6 is defined as being the architecture 20-10-2 (error 3.7%), but after analyzing the data on Table 4 of this architecture, showed a high MSE (52.133) compared to other architectures. Therefore, the architecture 15-30-2 is the best.

Table 6. MSE results for LMTs.

MSE			
LMT	**Half**	**Equal**	**Double**
3	4.298%	4.321%	4.322%
5	5.121%	5.118%	5.114%
8	5.079%	5.250%	5.170%
10	5.774%	6.053%	5.829%
15	6.084%	6.008%	5.909%
20	**3.763%**	3.772%	3.773%

The Figures 13 and 14 illustrate the steer angle and speed of CaRINA using the architecture 15-30-2, showing the comparison between human driver and the ANN learned. Small oscillations are present in the data ANN learned, since the FSM used in the proposed system maintains the current state during 2 intermediate transitions, resulting in a linearity of the data (the vehicle keeps on the road with steer and constants speed).

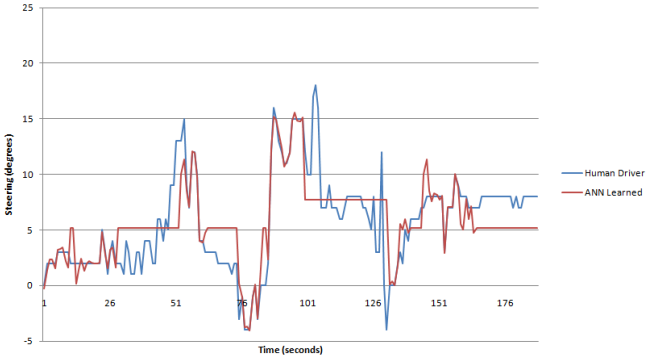


Figure 13. CaRINA steering – training data.

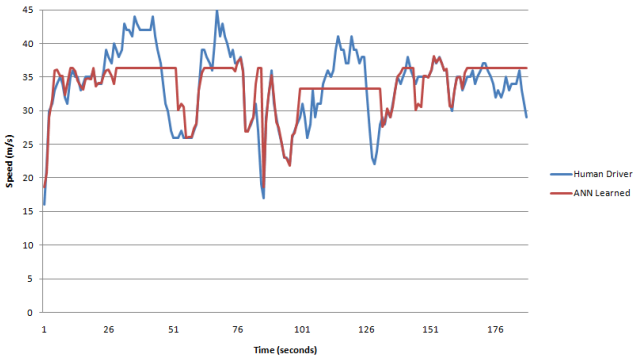


Figure 14. CaRINA speed – training data.

The l[illegible] obtained of the second ANN used in our proposed system. The architecture 15-30-2 was used in our experiments, showing the steer angle and speed for the test data.

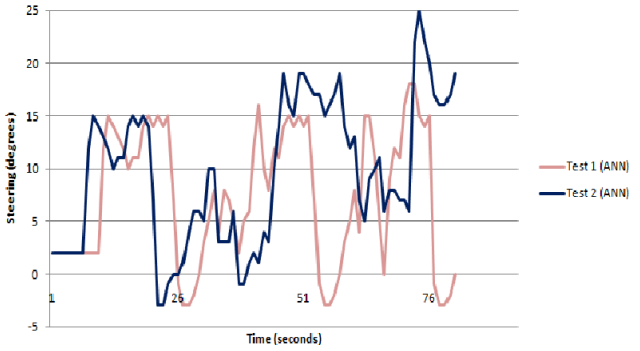


Figure 15. CaRINA steering – results obtained.

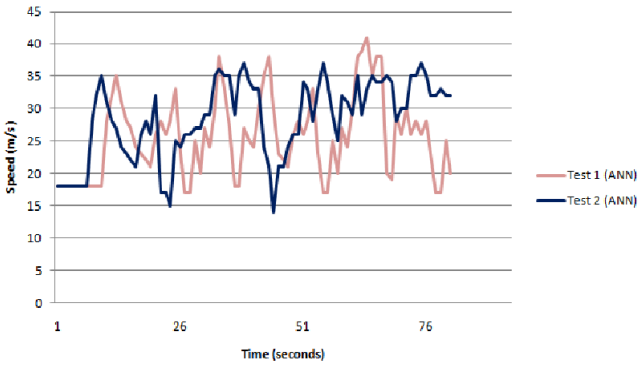


Figure 16. CaRINA speed – results obtained.

Figure 15 illustrates the steering of the ANN trained with the RPROP algorithm. It shows a better performance, with little oscillation as shown in Test 1 (the degree of steering between -5 and 15 as learned by ANN). The main difference when compared with Test 2 is small oscillation during the straight line used by CaRINA and it is quite similar the learned by ANN. Unlike Test 2, the degree of steering is more than 15 in some parts.

The speed performance is shown on Figure 16. In this case, the speed for Test 1 was obtained by ANN, showing a better performance when compared to Test 2, because it showed a better response to the learning of ANN and also generalized the data to navigate the vehicle on a safe path on the road unlike Test 2.

Experimental tests showed that the Test 1 generated as ANN output a better performance in the robot (CaRINA) behavior, as shown on Figures 15 and 16 (see Video[4]).

5. CONCLUSION AND FUTURE WORKS

Autonomous vehicle navigation is an important task in mobile robotics. This paper presented a vision-based navigation system which can be trained to identify the road and navigable regions using ANNs, template matching classification, FSM and a template memory. Our approach was evaluated using an Electrical Vehicle (CaRINA) tested in urban road following experiments. The vehicle was able to navigate autonomously in this environments executing a road following task, avoiding obstacles (e.g. people, traffic cones), and replicating the human behavior. Our quantitative analysis also obtained satisfactory results for the learning of ANNs with the respective architectures (LMT).

As future works, we plan to evaluate other classification methods and decision making algorithms. We also are planning to integrate camera and LIDAR laser in order to better deal with bumps and depressions in the road.

6. ACKNOWLEDGMENTS

The authors acknowledge the support granted by CNPq and FAPESP to the INCT-SEC (National Institute of Science and Technology - Critical Embedded Systems - Brazil), FAPESP doctoral grant (process 2009/11614-4), and Gustavo Buzogany for the technical contributions.

7. REFERENCES

[1] Fast artificial neural network library (fann), 2010. http://leenissen.dk/fann/, Access on 02 June.

[2] Stuttgart neural network simulator (snns), 2010. www.ra.cs.uni-tuebingen.de/SNNS/, Access 20 Nov.

[3] European land-robot (elrob), 2011. http://www.elrob.org/, Access on 18 May.

[4] Bradski, G. and Kaehler, A. Learning OpenCV: Computer Vision with the OpenCV Library. 2008.

[5] Chan, M., Partouche, D. and Pasquier, M. An intelligent driving system for automatically anticipating and negotiating road curves. International Conference on Intelligent Robots and Systems, pages 117–122, 2007.

[6] Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S. and Bradski, G. Self-supervised monocular road detection in desert terrain. In G. Sukhatme, S. Schaal, W. Burgard, and D. Fox, editors, In Proceedings of the Robotics Science and Systems Conference, 2006.

[7] Graefe, V. Vision for intelligent road vehicles. IEEE Symp. of Intell. Vehicles, pages 135–140, 1993.

[8] Hamner, B., Scherer, S. and Singh, S. Learning to drive among obstacles. IROS, pages 2663–2669, 2006.

[9] Markelic, I., Kjaer-Nielsen, A., Pauwels, K., Jensen, L. B. W., Chumerin, N., Vidugiriene, A., Tamosiunaite, M., Rotter, A., Hulle, M. V., Kruger, N. and Worgotter, F. The driving school system: Learning automated basic driving skills from a teacher in a real car. Transaction on Intelligent Transportation Systems, 2011.

[10] Markelic, I., Kulvicius, T., Tamosiunaite, M. and Worgotter, F. Anticipatory driving for a robot-car based on supervised learning. In Lecture Notes in Computer Science: Anticipatory Behavior in Adaptive Learning Systems, pages 267–282, 2009.

[11] Oentaryo, R. J. and Pasquier, M. Gensofnn-yager: A novel hippocampus-like learning memory system realizing yager inference. In International Joint Conference on Neural Networks, pages 1684–1691.

[12] Petrovskaya, A. and Thrun, S. Model based vehicle tracking in urban environments. IEEE International Conference on Robotics and Automation, 2009.

[4] Experiment video available in the Internet (Scenario 3): http://www.youtube.com/watch?v=SxdEY4JavAo/

[13] Pomerleau, D. A. ALVINN: An Autonomous Land Vehicle In a Neural Network. Advances In Neural Information Processing Systems, 1989.

[14] Shihavuddin, A. S. M., Ahmed, K., Munir, M. S. and Ahmed, K. R. Road boundary detection by a remote vehicle using radon transform for path map generation of an unknown area. International Journal of Computer Science and Network Security, 8(8):64–69, 2008.

[15] Shinzato, P. Y. and Wolf, D. F. A road following approach using artificial neural networks combinations. Journal of Intelligent and Robotic Systems, 62(3):527–546, 2010.

[16] Souza, J. R., Pessin, G., Shinzato, P. Y., Osorio, F. S. and Wolf, D. F. Vision-based autonomous navigation using neural networks and templates in urban environments. First Brazilian Conference on Critical Embedded Systems, pages 55–60, 2011.

[17] Souza, J. R., Sales, D. O., Shinzato, P. Y., Osorio, F. S. and Wolf, D. F. Template-based autonomous navigation in urban environments. In 26th ACM Symposium on Applied Computing, pages 1376–1381, 2011.

[18] Stein, P. S. and Santos, V. Visual guidance of an autonomous robot using machine learning. 7th IFAC Symposium on Intelligent Autonomous Vehicles, 2010.

[19] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L. E., Koelen, C., Markey, C., Rummel, C., Van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and Mahoney, P. Stanley, the robot that won the darpa grand challenge. Journal of Field Robotics, 2006.

ABOUT THE AUTHORS:

Jefferson Rodrigo de Souza is a PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He obtained his MSc degree in Computer Science at the Federal University of Pernambuco in 2010. At ICMC/USP, he has been working in machine learning techniques for mobile robots and autonomous vehicles. More specifically, autonomous driving based on learning human behavior. His current research interests are Mobile Robotics, Machine Learning, and Hybrid Intelligent System.

Daniel Oliva Sales is an MSc student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He has been working in autonomous navigation, finite state machines, robotics and machine learning. His current research interests are Mobile Robotics, Adaptation of Finite State Machines, and Neural Networks.

Patrick Yuri Shinzato is a PhD student in the Institute of Mathematics and Computer Science at the University of Sao Paulo. He obtained a M.S. in Computer Science in 2010 from the University of Sao Paulo. At ICMC/USP, he has been working in computer vision, machine learning and neural networks. His current research interests are Computational Vision, Artificial Intelligence, Mobile Robotics and Sensors Fusion.

Fernando Santos Osorio is an Assistant Professor in the Department of Computer Systems at the University of Sao Paulo (ICMC/USP). He obtained his PhD degree in Computer Science ("Informatique") at the INPG-IMAG ("Institut National Politechnique de Grenoble" - France) in 1998. Currently he is Co-Director of the Mobile Robotics Laboratory at ICMC/USP (LRM Lab.). His current research interests are Intelligent Robots, Machine Learning, Computer Vision, Pattern Recognition and Virtual Simulation. He has published an extensive number of journal and conference papers, focused in the above cited research areas.

Denis Fernando Wolf is an Assistant Professor in the Department of Computer Systems at the University of Sao Paulo ICMC/USP). He obtained his PhD degree in Computer Science at the University of Southern California - USC in 2006. Currently he is Co-Director of the Mobile Robotics Laboratory at ICMC/USP. His current research interests are Mobile Robotics, Machine Learning, Computer Vision, and Embedded Systems. He has published over 50 journal and conference papers over the last years.

Accelerating Large Semantic Web Databases by Parallel Join Computations of SPARQL Queries

Jinghua Groppe
Institute of Information Systems (IFIS),
University of Lübeck
Ratzeburger Allee 160
D-23538 Lübeck, Germany
+49 451 500 5705
groppej@ifis.uni-luebeck.de

Sven Groppe
Institute of Information Systems (IFIS),
University of Lübeck
Ratzeburger Allee 160
D-23538 Lübeck, Germany
+49 451 500 5706
groppe@ifis.uni-luebeck.de

ABSTRACT
While a number of optimizing techniques have been developed to efficiently process increasing large Semantic Web databases, these optimization approaches have not fully leveraged the powerful computation capability of modern computers. Today's multi-core computers promise an enormous performance boost by providing a parallel computing platform. Although the parallel relational database systems have been well built, parallel query computing in Semantic Web databases have not extensively been studied. In this work, we develop the parallel algorithms for join computations of SPARQL queries. Our performance study shows that the parallel computation of SPARQL queries significantly speeds up querying large Semantic Web databases.

Categories and Subject Descriptors
H.2.4 [**Database Management**]: Systems – *Parallel databases.*

Keywords
Semantic Web, SPARQL, RDF, Parallel Database.

1. INTRODUCTION
The Semantic Web databases are becoming increasingly large and a number of approaches and techniques have been suggested for efficiently managing and querying large-scale Semantic Web databases. However, current algorithms are implemented for sequential computation, and do not fully leverage the computing capabilities of current standard multi-core computers. Therefore, the querying performance of the Semantic Web databases can be further improved by maximally employing the capabilities of multi-core computers, i.e., parallel processing of SPARQL queries.

A parallel database system improves performance through parallelization of various operations, such as building indexes and evaluating queries. While centralized database systems allow parallelism between transactions (multi-user synchronization), parallel database systems additionally use parallelism between queries inside a transaction, between the operators and within individual operators of a query.

Ideally, the speedup from parallelization would be linear – doubling the number of processing units should halve the runtime. However, very few parallel approaches can achieve such ideal speedup. Since the existence of non-parallelizable parts, most of them have a near-linear speed-up for small numbers of processing units, and the speedup does not become larger than a certain constant value for large numbers of processing units. The potential speed-up of an algorithm on a parallel computing platform is governed by Amdahl's law [1]. The Amdahl's law discloses that a small portion of the problem, which cannot be parallelized, will limit the overall speed-up available from parallelization by a constant value. For this reason, parallel computing is only useful for either small numbers of processors, or highly parallelizable problems. We will see that SPARQL query evaluation is highly parallelizable.

Two major techniques of parallelism used in parallel database systems are *pipelined* parallelism and *partitioned* parallelism [5]. By streaming the output of one operator into the input of another operator, the two operators can work in series giving pipelined parallelism. N operators executed using pipelined parallelism can achieve a potential speedup of N. By partitioning the input data into multiple parts, an operator can be run in parallel using the multiple processing units, with each working on a part of the data. This partitioned data and execution provides partitioned parallelism.

Parallelism can be obtained from conventional sequential algorithms by using *split* and *merge* operators. The proven and efficient techniques (e.g. [20]) developed for centralized database systems can be leveraged in a parallel system enhanced with the split and merge operators. New issues that need to be addressed in parallel query processing include data partitioning and parallel join computation. The strategies of data partitioning contain (multi-dimensional) range partitioning, round-robin partitioning and hash partitioning [9]. A large number of optimizing techniques for parallel join processing of relational queries have been developed, e.g. [3], [4], [18], [19], [25], [26], [30] and [31], most of which focus on parallel hash-join algorithms.

Parallel computing and parallel relational databases have been employed for many years, and a number of efficient techniques have been developed, which can be leveraged for parallel processing of SPARQL. However, optimizing techniques for parallel relational databases do not specialize on the triple model of RDF and triple patterns of SPARQL queries. In this paper, we develop a parallel Semantic Web database engine based on the RDF- and SPARQL-specific properties. We focus on parallelization approaches for standard hardware with multiple processing cores and common memory and shared disks.

The contributions of this paper are

- parallel join computations especially for
  - hash joins using a distribution thread, and
  - merge joins using partitioned input,
- an approach to computing operands in parallel, and

- an experimental evaluation, which shows the performance benefits of parallel data structures and algorithms in Semantic Web databases.

This paper is an extended version of [11]. It has been extended by describing different standard join algorithms (see Section 3.1), the types of parallelism (see Section 3.2), Amdahl's law (see Section 3.3), and further implementation details (see Section 10.1) like maintaining histogram indices for fast histogram construction even for very large Semantic Web datasets (see Section 10.1.1).

2. SEMANTIC WEB DATA AND QUERIES: RDF AND SPARQL

The Semantic Web uses RDF [2] as its data format and SPARQL [24] as the basic query language of RDF data.

The core element of RDF data is the *RDF triple*, and a set of RDF triples is called an *RDF graph*.

Definition 1 (*RDF triple*): Assume there are pairwise disjoint infinite sets I, B and L, where I represents the set of IRIs [6], B the set of blank nodes and L the set of literals. We call a triple $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ an *RDF triple*, where s represents the subject, p the predicate and o the object of the RDF triple. We call an element of $I \cup B \cup L$ an *RDF term*.

SPARQL queries are evaluated on *RDF graphs*, and select data based on the matching of graph patterns of SPARQL. The core component of SPARQL graph patterns is a set of triple patterns *s p o*. *s p o* corresponds to the subject (*s*), predicate (*p*) and object (*o*) of an RDF triple, but they can be variables as well as RDF terms. Within a SPARQL query, the user specifies the known RDF terms of triples and leaves the unknown ones as variables in triple patterns. The same variables can occur in multiple triple patterns and thus imply joins. A triple pattern matches a subset of the RDF data, where the RDF terms in the triple pattern correspond to the ones in the RDF data.

SPARQL queries are ideally suited to parallel execution, thanks to the properties and semantics of SPARQL:

- The basic construct of SPARQL queries is a triple pattern. Each triple pattern is applied on the RDF graph independently of other triple patterns.
- Each triple pattern applied to the RDF graph returns a bag of solutions (a solution consists of a set of variables with their bound values).
- Joins of different triple patterns are independent of each other.

Therefore, SPARQL queries consist of uniform operations applied to uniform streams of data. Each operator produces a new solution, so the operators can be composed into highly parallel dataflow graphs. SPARQL queries expose a very high proportion of parallelization, and thus can significantly benefit from parallel processing.

Table 3 presents the example RDF data Records.rdf with 3 triples, and an SPARQL query DLCRecords.sparql for this RDF data. DLCRecords.sparql consists of a SELECT clause and a WHERE clause. The SELECT clause identifies the variables to appear in the query results, i.e., the variable bindings of ?a and ?c. The WHERE clause contains two triple patterns, which identify the constraints on the input RDF data. The triple pattern ?a <records> ?c matches the first two triples of Records.rdf, and thus its result is {(?a=<ID1>, ?c=<ID6>),(?a=<ID2>, ?c=<ID5>)}. The triple pattern ?a <origin> <DLC> matches the last triple of Records.rdf, and thus its result is {(?a=<ID2>)}. The two triple patterns impose a join over the common variable ?a, and their result is hence {(?a=<ID2>, ?c=<ID5>)}, which is the final query result.

Table 3. Example RDF Data and SPARQL Query

Records.rdf	DLCRecords.sparql
<ID1> <records> <ID6> <ID2> <records> <ID5> <ID2> <origin> <DLC>	SELECT ?a ?c WHERE { ?a <records> ?c. ?a <origin> <DLC>.}

3. BASICS OF SEQUENTIAL JOIN COMPUTATION AND PARALLELISM

3.1 Join Algorithms

We describe the merge join, the index and the disk-based hash join algorithm in the following subsections. Common database literature [10] regards the merge join as fastest (sequential) join algorithm whenever the input data is already sorted, the index join as fastest join algorithm whenever one input is indexed, and the disk-based hash join as fastest join algorithm for large unsorted and un-indexed input.

3.1.1 Merge Join

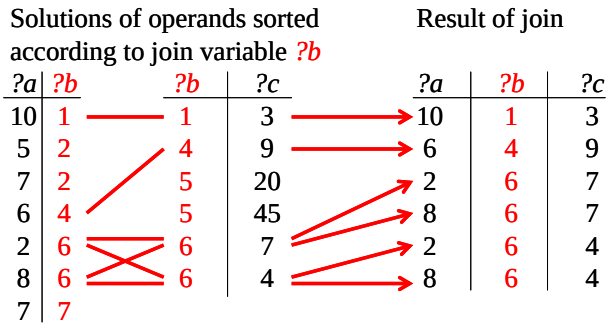


Figure 1: Merge join example

Let V_L be the set of bound variables of the left operand of a join, and V_R be the set of bound variables of the right operand. We call the variables in $V_L \cap V_R$ the *join variables*. The merge join requires the solutions of its operands sorted in the same way according to its join variables. For an example of the merge join algorithm applied to the solutions of its operands, see **Figure 1**. The merge join algorithm first reads the solutions from both operands and checks if they are joinable (i.e., the bound values of the join variables are the same). If they are *not* joinable, the next solution of the operand with the smaller values is read, because we are sure that the remaining solutions of the other operand are equal or larger values according to the sort criterion and thus cannot be joined with the smaller value. In the case that they are joinable, the merge join algorithm reads the next solutions from both operands until the bound values of the join variables differ from the first read solution. All combinations of the read solutions with the same bound values of the join variables of both operands must now be joined and returned.

3.1.2 Index Join

The index join utilizes a given index of one of its join operands to optimize join processing: The index join iterates through the solutions of one join operand and searches for relevant join partners, i.e. solutions with the same bound values for the join variables, from the other join operand by using its index. An operand typically provides an index like a B^+-tree if it is an index scan operator for retrieving the result of a triple pattern.

If both operands of a join are not index scans, but their solutions fit into main memory, then it is also efficient to first index the results of one operand *R* using an in-memory hash table, and then apply the index join using the just created index. For indexing the solutions of *R*, a hash function over the join variables must be used, which maps bound values of the join variables in a solution to an integer, and such that the join partners for the solutions can be found within one index access. This type of index join is often also called *in-memory hash join*.

3.1.3 Disk-Based Hash Join

Like the in-memory hash join, we use again hash functions over the join variables. The disk-based hash join first distributes the input data into smaller partitions using hash functions over the join variables. After the partitions become small enough, a general-purpose join algorithm is used to join corresponding partitions of both operands.

3.2 Types of Parallelism

Different types of parallelisms can be used during query processing (see **Figure 2**). We will describe some of these types in more detail in following paragraphs:

1. A transaction contains several queries and updates of a database application. Transactions typically conform to the ACID properties of databases:
   - Atomicity: A transaction should be processed atomic, i.e., the effects of all its queries and updates are visible to other transactions or none of them.
   - Consistency: The database should be left in a consistent state before and after the transaction.
   - Isolation: The transaction should be processed isolated, i.e., the effect of the transaction should be the same as when all transactions are sequentially processed.
   - Durability: The effect of a successful transaction must be durable even after system crashes, damages of storage systems, or other erroneous soft- or hardware.
2. As a transaction contains several queries and updates, these queries and updates can be processed in parallel if they do not dependant on each other: If an update inserts, deletes or modifies a triple, which influences the result of another query in the transaction, then the update and query must be processed in the order they occur in the transaction. The transaction in **Figure 2** contains an insertion of the triple *dc:journal1 rdf:type bench:Article*, which is matched by the triple pattern *?a rdf:type bench:Article* of the first query, such that the first query must be processed before the insertion. However, the last query does not contain any triple pattern matching the triple *dc:journal1 rdf:type bench:Article*, and therefore the last query can be processed in parallel to the insertion and also in parallel to the first query of the transaction.
3. A query (and also an update) of a transaction is transformed into an operatorgraph consisting of several operators to be executed by the database system. Two forms of parallelisms can be applied here: The first form applies operators independent from each other in parallel. The second uses pipelining, where already computed intermediate results are transferred to subsequent operators as early as possible, which already process this intermediate result further leading to a smaller memory footprint and to saving i/o accesses.
4. Many operators itself can use parallel algorithms to determine their results. Prominent examples of such operators are sorting and join operators.

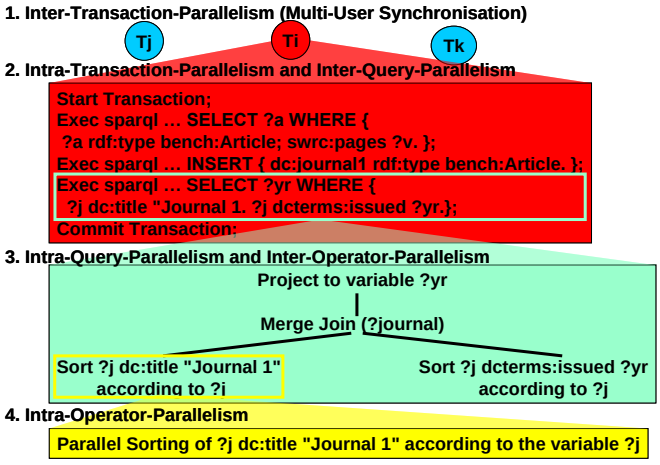


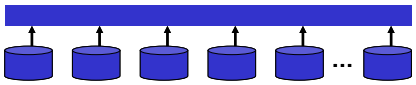
Figure 2: Types of parallelisms in query processing

For data parallelism, one tries to distribute the data into different disjoint fragments, such that operations like sorting or joins can be done in parallel in the different fragments. The extent of parallelism can be chosen dependant on the size of data, i.e., larger data can be distributed into more fragments, such that more computers can be used to process the data in parallel leading to scalable solutions. Furthermore, data parallelism can be combined with pipelining.

There are two forms of I/O parallelism (see **Figure 3**):

- The access parallelism uses different I/O devices like hard disks to process one job. Access parallelism is important for data parallelism and the distributed fragments should be therefore stored on different I/O devices.
- During job parallelism independent jobs are processed on different I/O devices in parallel. Applying job parallelism is important for inter-transaction parallelism and inter-query parallelism.

(I) Intra-I/O-Parellelism (Access Parallelism)



(II) Inter-I/O-Parallelism (Job Parallelism)

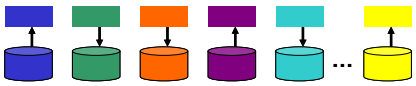


Figure 3: Types of I/O parallelisms

3.3 Amdahl's Law

The batch speedup determines the response time improvement of parallel processing. The batch speedup for N computers is defined to be the response time when using one computer (and sequential algorithms) divided by the response time when using N computers, parallel algorithms and the same data. Ideal would be a doubled batch speedup when using two times more computers (see **Figure 4**). A linear improvement would be still fine, because one could determine the number of computers needed to obtain any response time one wants. However, experiments show that in typical cases the batch speedup does not increase or only slightly increases after a certain upper limit has been reached. In many cases the batch speedup even decreases for more computers.

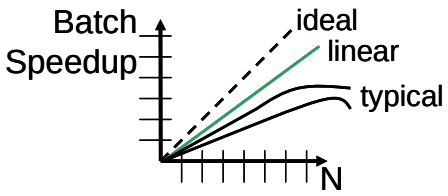


Figure 4: Batch speedup dependant on the number N of computers

The reasons for the limits of scalability are startup and termination overhead of the parallel algorithms, inferences when accessing the logical and physical resources, overloads of individual computers, lock conflicts, limited partitioning possibilities of data, transactions and queries, and skew in the execution times of subtasks.

Amdahl's law [1] now can determine an upper limit for the batch speedup, if the fraction of the execution time of the non-optimized, sequential part of the algorithm is known in relation to the overall execution time (see **Figure 5**). Using this formula, one can determine a maximal batch speedup of 20 if the sequential fraction is only 5%.

$$\text{Batch Speedup} = \frac{1}{(1-F_{opt}) + \frac{F_{opt}}{S_{opt}}}$$

F_{opt} = Fraction of the optimized (parallelized) component ($0 \le F_{opt} \le 1$)
S_{opt} = Speedup for the optimized (parallelized) component

Figure 5: Formula for batch speedup

3.4 Parallel Monitors and Bounded Buffers

An important concept in parallel computing is parallel monitors. A parallel monitor is a high level synchronization concept and is introduced in [17]. It is a program module for concurrent programming with common storage, and has entry procedures, which are called by threads. The monitor guarantees mutual exclusion of calls of entry procedures: at most one thread executes an entry procedure of the monitor at any time. Condition variables may be defined in the monitor and used within entry procedures for condition synchronization.

An example of a parallel monitor is a bounded buffer. Mainly, a bounded buffer has two operations, *put* to store an element in the bounded buffer and *get* to retrieve an element from the bounded buffer. The bounded buffer has a specific constant limit for the number of stored elements. If a thread tries to put an element into a full bounded buffer, then the bounded buffer forces the calling thread to sleep until another thread calls *get* to retrieve one element. A *get* on an empty bounded buffer causes the calling thread to sleep until another thread puts one element inside.

The bounded buffers are typically used in producer/consumer patterns, where several threads produce elements and other threads consume these elements. The advantage of bounded buffers in comparison to unbounded buffers is that the usage of the main memory is bounded, i.e. consumer threads cannot store more elements than allowed by the main memory, and thus avoiding the problem of out-of-memory errors. Therefore, we use bounded buffers for the communication between threads for the parallel join computation.

4. PARALLEL JOIN USING A DISTRIBUTION THREAD

When we use several threads for join computation (see **Figure 6**) and the input is not partitioned yet, we have to partition the input data among these join threads using e.g. (multi-dimensional) range partitioning or hash partitioning.

The input data of a join must be partitioned according to the join variables in the following way: the solutions (of two join operands) with the same values bound to the join variables are distributed to the same thread. This ensures (a) that each join thread only involves the data in its own bounded buffers, and (b) that the overall join result is correct. Hash partitioning uses a hash function over the values of the join variables to determine to which join thread an input element is distributed. Hash partitioning promises good distributions and efficient partitioning computation. We hence use the hash partitioning for most parallel join processing.

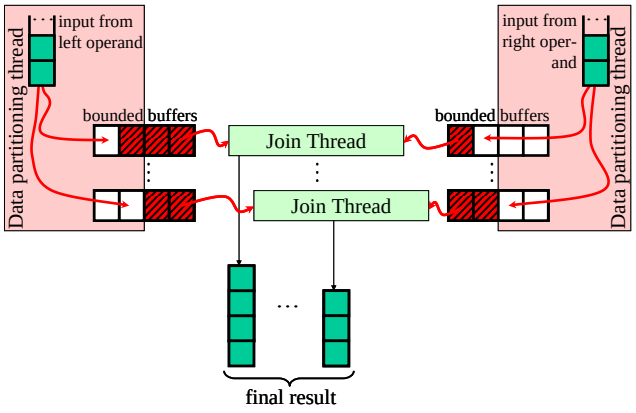


Figure 6: Parallel join computation

For each operand of joins, we use a *data partitioning thread* to distribute the data into the bounded buffers of join threads (see

Figure 6). The join thread reads data from its two bounded buffers and performs a sequential join algorithm. If one join thread has finished its computation, then its result can be processed by succeeding operators (without waiting for the other join threads). This approach is the fastest one of parallelizing join algorithms like the hash join and the index join.

When processing large data, the joins like the disk-based hash join and index join involve complex operations during joining. Parallelizing these join algorithms, even plus the overhead of data partitioning, still can speed up join processing.

5. PARALLEL MERGE JOIN USING PARTITIONED INPUT

Merge joins require their input to be sorted. Merge joins on already sorted data are very fast and cheap in terms of I/O and CPU costs. The benefit from parallelizing merge joins can not compensate the overhead of data partitioning even for large input data and large join results. The processing performance benefits from the parallel computation of merge joins, if the input is already partitioned.

The merge join operator typically follows either the triple pattern operator in an operator graph, or the operators like filters, which do not destroy the partitions of the input. Furthermore, the output of a merge join with such partitioned input is again partitioned, such that succeeding merge joins can use partitioned input as well. In order to generate a partitioned input for the merge join, we can retrieve the result of triple patterns using range partitioning (see Figure 7). Range partitioning in comparison to hash partitioning has the advantage that the data in all the ranges can be read and processed in parallel.

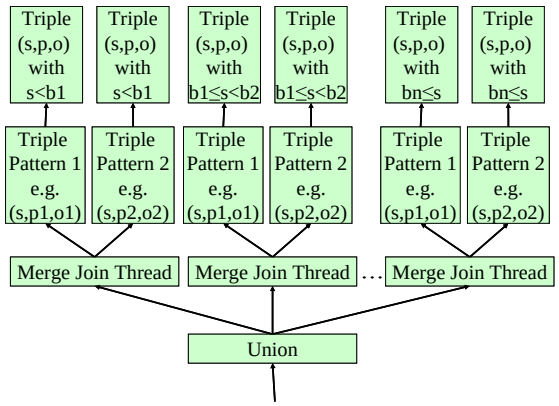


Figure 7: Parallel merge join using range partitioning

SPARQL engines for large-scale data, like *Hexastore* [29] and *RDF3X* ([21] and [22]), use six indices corresponding to six collation orders SPO, SOP, PSO, POS, OSP and OPS to manage RDF triples. As an explanation, the collation order SPO defines the subject (S) as the primary, the predicate (P) as the secondary and the object (O) as the tertiary sort criterion. Depending on which positions in a triple pattern contain RDF terms (e.g., the subject and the object), one of the indices (e.g., SOP) is used to efficiently retrieve the data by using a prefix search. Using these collation orders, many joins can be computed using the fast merge join approach over sorted data. RDF3X employs just B^+-trees for each collation order and prefix searches, and thus gaining a simpler and faster index structure than [29].

Employing B+-trees as the indices for each collation order has another advantage: The results of retrieving B^+-trees can be very easily partitioned according to the range information in the histograms of triple patterns. For each kind of triple patterns, an equi-depth histogram [23] is constructed (see Section 10.1.1) during the query optimization phase of our SPARQL engine. Among other information, each interval in this histogram contains the range and the number of triples allocated in this interval. We use special histogram indices to fast compute histograms over large datasets.

Figure 8 describes how to get the partitioned results of a triple pattern using range partitioning: We assume that the data in the ranges [(3, 2, 1), (3, 4, 7)] will be distributed to the first partition and the data in [(3, 5, 5), (3, 7, 4)] to the second partition. Two partitioning threads can perform the range partitioning in parallel: One first searches for the border (3, 2, 1) in the B^+-tree and then follows the chain of leafs until the border (3, 4, 7), and the retrieved data belong to the first partition; another starts searching from the border (3, 5, 5) until the border (3, 7, 3), and retrieves the data for the second partition.

All approaches described so far for parallel joins apply also for the computation of OPTIONAL constructs. They are left outer-joins and can hence be analogously parallelized.

6. PARALLEL COMPUTATION OF OPERANDS

Another way to parallelize joins is to process their operands in parallel. In order to parallelize the processing of join operands, we use two *operand threads* (see **Figure 9**). An operand thread computes its operand and puts the result into its bounded buffer.

Join approaches like hash joins first read in the whole data of one operand and afterwards start reading from the other operand. For such joins, the parallelism is not high, depending on the size of the bounded buffer. For the joins like merge joins, which synchronously process the operands' data, two operand threads can work in parallel.

However, advanced techniques like sideways information passing (SIP) [22] cannot be applied to parallel computation of operands, as using SIP to compute the next solution of one operand relies on the current solution of the other operand.

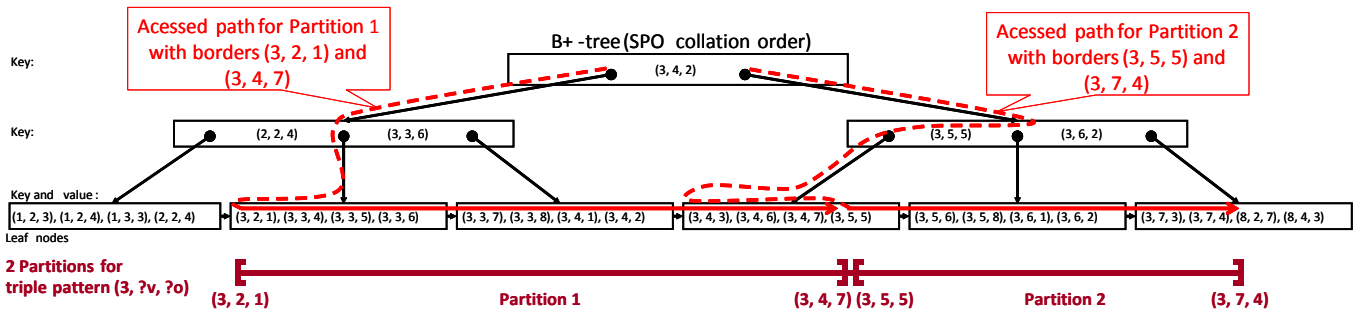


Figure 8: B+-tree accesses for partitioned input. Integer ids are used instead of RDF terms as components of the indexed triples.

The experiments show that the parallel computation of operands speeds up the evaluation of some queries especially if the processing of operands involves complex computations, but the previously discussed approaches for parallel joins are superior.

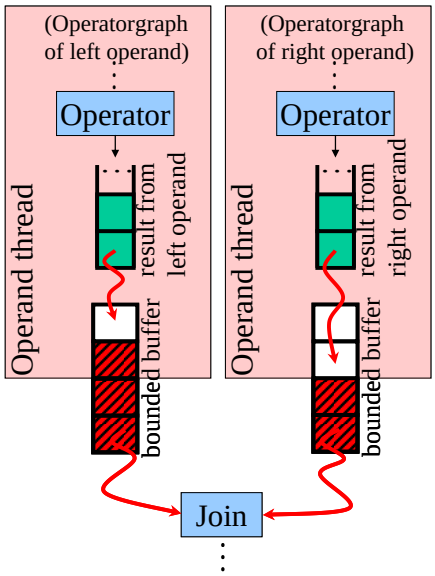


Figure 9: Parallel computation of operands

7. PERFORMANCE EVALUATION

We study the performance benefits for parallelizing Semantic Web database engines. In these experiments, we focus on the index approach *RDF3X* in [21] and [22], since it is similar to Hexastore in [29], but uses a simpler and faster index structure than [29]. We compare the pure *RDF3X* approach, i.e., using sequential join algorithms, with several parallel versions of the *RDF3X* approach: *PHJ RDF3X* is the *RDF3X* approach using a parallel hash join algorithm with 8 join threads and distribution threads; *PMJ RDF3X* uses parallel merge join algorithms with 8 merge join threads with partitioned input; *PO RDF3X* computes the operands of the last hash join in parallel. Combinations like *PMJ PO RDF3X* combine together several parallel approaches like *PMJ RDF3X* and *PO RDF3X*.

The original RDF3X prototype ([21] and [22]) has several limitations:

- It does not support full SPARQL 1.0
- It only supports very simple filter expressions.
- It neglects prefixes in predicates. This improves the performance, but leads to information loss. For example, the two different triples (<a>, <www.**film**/title>, "Ratatouille") and (<a>, <www.**game**/title>, "Ratatouille") are stored as one triple (<a>, <title>, "Ratatouille") in the original RDF3X prototype.
- It does not support datatypes, and this leads to e.g., that the two identical integer values `+2` and `2` are treated as two different strings.
- It supports only in-memory hash joins, i.e., if the input data of hash joins cannot fit into memory, RDF3X cannot process the hash joins.

In order to lift these limitations and avoid problems resulting from not supported features of the original RDF3X prototype, we have re-implemented the RDF3X approach. Our re-implementation successfully runs all the W3C test cases [7], which contain over 200 queries. As we will show later, the execution times of our re-implementation are similar to, and often outperform those of the original RDF3X prototype, compared with results of [22], although we have used Java as programming language and the original RDF3X prototype is implemented in C++. An online demonstration of our implementations is publicly available (see [15] and [13]).

The test system for the performance analysis uses an Intel Core 2 Quad CPU Q9400 computers, each with 2.66 Gigahertz, 4 Gigabytes main memory, Windows XP Professional (32 bit) and Java 1.6. We have run the experiments ten times and present the average execution times and the standard deviation of the sample.

In order to build indices faster over the very large dataset, index constructions are performed in a cluster with a Dual Quad Core Intel CPU X5550 computer with 2.67 Gigahertz, 6 Gigabytes main memory, Windows XP Professional (x64 Edition) and Java 1.6 64 bit, and 6 Intel Core 2 Quad CPU Q9400 computers, each with 2.66 Gigahertz, 4 Gigabytes main memory, Windows XP Professional (32 bit) and Java 1.6.

We use the large-scale dataset Billion Triples Challenge (BTC) [27] and corresponding queries. Three kinds of indices are constructed over the BTC dataset: 2 dictionary indices for mapping between RDF terms and integer-ids; 6 SPARQL queries; 6 histogram indices for fast generating histograms of triple patterns.

We have imported *all* of over 830 million distinct triples of the Billion Triples Challenge. In comparison, the performance analysis in [22] used only a subset of it.

The queries (BTC 1 to BTC 8) used by [22] return very small intermediate and final results, which can be processed directly in memory. For these queries, the parallel approaches often do not show benefits and are dominated by their overhead. Therefore, we also use several additional queries (EBTC 1 to EBTC 8) with

bigger cardinalities (see the Appendix). **Table 4** presents the processing times by the different approaches.

Although the queries BTC 1 to BTC 8 are designed to retrieve very small results and thus are favorable to the RDF3X using sequential algorithms, except for the query BTC 7, our parallel approaches have similar performance as the sequential one. Our parallel approaches significantly outperform the sequential approach for the queries BTC 4 (PMJ RDF3X being the fastest), BTC 5 (PH PMJ RDF3X being the fastest) and BTC 6 (PO RDF3X being the fastest). For the EBTC queries, PHJ RDF3X is the fastest approach when evaluating EBTC 1, EBTC2, EBTC4 and EBTC 7, PO RDF3X is the fastest approach for EBTC 5, and PO PHJ RDF3X is the fastest one for EBTC 6. Parallel join algorithms work well whenever join results are large: If a merge join has a large result, PMJ RDF3X or PH PMJ RDF3X belongs to the fastest ones. If a hash join has a large result, PHJ RDF3X is the fastest.

The time for index construction for the BTC dataset with over 830 million distinct triples was 30 hours. The space consumption is 31.1 Gigabytes for the dictionary indices, 30.8 Gigabytes for the evaluation indices and 30.8 Gigabytes for the histogram indices.

Table 4. Evaluation times (in seconds) for BTC Data

Query	RDF3X	PHJ RDF3X	PMJ RDF3X	PH PMJ RDF3X	PO RDF3X	PO PHJ RDF3X	Size of results
BTC 1	0.047 ± 0.014	0.047 ± 0.026	0.047 ± 0.015	0.047 ± 0.033	0.045 ± 0.013	0.046 ± 0.028	2
BTC 2	0.042 ± 0.016	0.046 ± 0.01	0.119 ± 0.134	0.11 ± 0.13	0.12 ± 0.012	0.12 ± 0.01	48
BTC 3	0.452 ± 0.082	0.436 ± 0. 085	0.656 ± 0.047	0.74 ± 0.05	0.454 ± 0.105	0.48 ± 0.04	34
BTC 4	44.9 ± 0.1	45.99 ± 0.13	**31.98 ± 2.99**	33.8 ± 0.9	46.1 ± 0.2	44.86 ± 0.12	3
BTC 5	3.58 ± 0.12	3.52 ± 0.04	3 ± 0.06	**2.99 ± 0.1**	5.2 ± 0.2	5.15 ± 0.13	0
BTC 6	1.61 ± 0.03	1.65 ± 0.08	0.959 ± 0.564	0.76 ± 0.04	**0.63 ± 0.05**	0.76 ± 0.1	1
BTC 7	0.629 ± 0.077	6.33 ± 0.07	6.96 ± 2.74	13.7 ± 0.7	170 ± 1	173 ± 1	0
BTC 8	0.381 ± 0.070	0.36 ± 0.05	0.88 ± 0.09	0.87 ± 0.07	0.52 ± 0.07	0.52 ± 0.08	0
EBTC1	153.2 ± 3.7	**105.9 ± 1.7**	153.16 ± 3.38	113.44 ± 3.08	149 ± 2	139 ± 2	798553
EBTC2	6.05 ± 0.41	**4.41 ± 0.71**	6.58 ± 0.34	4.69 ± 0.14	6.58 ± 0.7	5.6 ± 0.9	1206
EBTC3	2.08 ± 0.05	3.12 ± 0.072	2.13 ± 0.13	2.13 ± 0.17	1.97 ± 0.13	3.3 ± 0.12	57
EBTC4	1883 ± 32	**1241 ± 26**	1844 ± 18	1297 ± 21	1834 ± 21	1850 ± 37	134588
EBTC5	136.9 ± 4.1	145 ± 20	134 ± 3	160 ± 29	**129 ± 3**	137 ± 18	3856586
EBTC6	2810 ± 24	1736 ± 85	2649 ± 39	1799 ± 60	2677 ± 9	**1555 ± 38**	58849326
EBTC7	6.4 ± 0.34	**4.87 ± 0.85**	6.8 ± 0.62	5.37 ± 0.61	7.4 ± 0.9	5.33 ± 0.84	18
EBTC8	2.15 ± 0.07	3.01 ± 0.07	2.10 ± 0.06	3.07 ± 0.012	2.06 ± 0.13	3.35 ± 0.16	57

7.1 Performance Gains and Loss

Several main factors contribute to the gains and loss of performance from parallelizing Semantic Web database engines:

(a) PHJ RDF3X performs best whenever hash joins have large results.

(b) Whenever merge joins have to process large input and have large results, then PMJ RDF3X or PH PMJ RDF3X outperforms the other sequential and parallel approaches.

(c) The overhead of parallel processing like data partitioning will dominate if involved data is small in size.

(d) Advanced techniques like sideways information passing (SIP) [22] cannot be applied to some parallel computations.

Therefore it is the task of the query optimizer to estimate the sizes of the join results and choose the proper join algorithms.

8. SUMMARY AND CONCLUSIONS

Since the disappearing of single-core computers, parallel computing has become the dominant paradigm in computer architectures. Therefore, developing parallel programs to fully employ the computing capabilities of multi-core computers is under the necessity. This is especially important for time-consuming processing like querying growingly large Semantic Web databases. In this work, we develop a parallel SPARQL engine, especially focusing on the parallelism of join computations. We propose different parallel join approaches in order to maximally gain from parallel computing.

Our experimental results show that parallel join computation outperforms sequential join processing for large join results; otherwise the parallel overhead compensates the performance improvements through parallelization. Therefore, the query optimizer must decide when to apply parallel join approaches. The proper application of parallel computation can significantly speed up querying very large Semantic Web databases.

9. REFERENCES

[1] G. Amdahl: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. *AFIPS Conference Proceedings*, (30), pp. 483-485, 1967.

[2] D. Beckett (editor), RDF/XML Syntax Specification (Revised), *W3C Recommendation*, 10th February 2004.

[3] H. Boral, W. Alexander, L. Clay, G. Copeland, S. Danforth, M. Franklin, B. Hart, M. Smith, P. Valduriez: Prototyping Budda: a highly parallel database system. *IEEE KDE*, 1990.

[4] D. J. DeWitt, R. H. Gerber, G. Graefe, M. L. Heytens, K. B. Kumar. GAMMA - A High Performance Dataflow Database Machine, *VLDB*, 1986.

[5] D. DeWitt, J. Gray: Parallel Database Systems: The Future of High Performance Database Systems. *Communications of ACM*, 35(6):85-98, June 1992.

[6] M. Dürst , M. Suignard, Internationalized Resource Identifiers (IRIs), http://www.ietf.org/rfc/rfc3987.txt, *W3C Memo*, 2005.

[7] L. Feigenbaum (editor), DAWG Testcases, http://www.w3.org/2001/sw/DataAccess/tests/r2, 2008.

[8] Friend, E. H., Sorting on Electronic Computer Systems, *Journal of the ACM*, 3 (3), 1956.

[9] G. Graefe, Query evaluation techniques for large database. *ACM Computing Surveys* 25, 2 (June), 73-170, 1993.

[10] S. Groppe, Data Management and Query Processing in Semantic Web Databases, *Springer*, 2011. ISBN 978-3-642-19356-9

[11] J. Groppe, S. Groppe, Parallelizing Join Computations of SPARQL Queries for Large Semantic Web Databases, *26th Symposium On Applied Computing (ACM SAC 2011)*, TaiChung, Taiwan, 2011.

[12] J. Groppe, S. Groppe, S. Ebers and V. Linnemann, Efficient Processing of SPARQL Joins in Memory by Dynamically Restricting Triple Patterns, *ACM SAC*, 2009.

[13] J. Groppe, S. Groppe, A. Schleifer, Volker Linnemann, LuposDate: A Semantic Web Database System, *ACM CIKM*, Hong Kong, China, 2009.

[14] S. Groppe, J. Groppe, External Sorting for Index Construction of Large Semantic Web Databases, *ACM SAC*, 2010.

[15] S. Groppe, J. Groppe, LUPOSDATE Demonstration, http://www.ifis.uni-luebeck.de/index.php?id=luposdate-demo, 2009.

[16] A. Harth, J. Umbrich, A. Hogan, S. Decker. YARS2: A Federated Repository for Querying Graph Structured Data from the Web, *ISWC*, 2007.

[17] C. A. R. Hoare: Monitors: An Operating System Structuring Concept. *Commun. ACM* 17(10): 549-557, 1974.

[18] M. Kitsuregawa, H. Tanaka, T. Motooka: Application of Hash to Data Base Machine and Its Architecture, *New Generation Computing*, Vol. 1, No. 1, 1983.

[19] M. Kitsuregawa, Y. Ogawa: A New Parallel Hash Join Method with Robustness for Data Skew in Super Database Computer (SDC), *VLDB*, Melbourne, Australia, 1990.

[20] P. Mishra, M. Eich, Join processing in relational databases, *ACM Computing Surveys* 24, 1, 63-113. 1992.

[21] T. Neumann, G. Weikum, RDF3X: a RISCstyle Engine for RDF, *VLDB*, Auckland, New Zealand, 2008.

[22] T. Neumann, G. Weikum, Scalable join processing on very large RDF graphs, *SIGMOD*, 2009.

[23] G. Piatetsky-Shapiro, C. Connell, Accurate Estimation of the Number of Tuples Satisfying a Condition. *SIGMOD*, 1984.

[24] E. Prud'hommeaux, A. Seaborne, SPARQL Query Language for RDF, *W3C Recommendation*, 2008.

[25] D. Schneider, D. DeWitt: A Performance Evaluation of Four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment, *SIGMOD*, Portland, 1989.

[26] D. Schneider, D. DeWitt, Tradeoffs in Processing Complex Join Queries via Hashing in Multiprocessor Database Machines, *VLDB*, Melbourne, Australia, 1990.

[27] Semantic web challenge 2009. billion triples track. http://challenge.semanticweb.org/.

[28] M.-E. Vidal, E. Ruckhaus, T. Lampo, A. Martinez, J. Sierra and A. Polleres, Efficiently Joining Group Patterns in SPARQL Queries, *ESWC*, 2010.

[29] C. Weiss, P. Karras, A. Bernstein, Hexastore: Sextuple Indexing for Semantic Web Data Management, *VLDB*, 2008.

[30] J.L. Wolf, D.M. Dias, P.S. Yu: An Effective Algorithm for Parallelizing Sort-Merge Joins in the Presence of Data Skew, *DPDS*, 1990.

[31] H.J. Zeller, J. Gray, Adaptive Hash Joins for a Multiprogramming Environment, *VLDB*, Australia, 1990.

10. APPENDIX

This appendix contains the implementation details of our query engine. Furthermore, this Appendix presents the additional queries for the Billion Triples Challenge (BTC), used in our performance study in Section 7.

10.1 Implementation

We have integrated our technique of fast sorting into the index approaches RDF3X ([21] and [22]) and Hexastore [29].

More details about query evaluation and data management in Semantic Web databases with special focus to our LUPOSDATE implementations can be found in [10].

10.1.1 Building Indices

Most index approaches use B^+-trees to store the RDF triples (e.g. [29], [21] and [22]). B^+-trees can be built very efficiently from a sorted list of data by avoiding expensive node splitting. Among a number of sorting algorithms, *merge sort* scales well for very large data and performs especially well for external sorting. Therefore, we use the merge sort technique to sort data for constructing indices efficiently. In the merge sort algorithm, we use a sort&merge-heap [14] with replacement selection [8] in order to increase the size of the initial runs. We create and maintain three different types of indices:

Dictionary indices: One dictionary index maps RDF literals into integer ids; one translates integer ids back into RDF literals. When storing RDF literals in the dictionaries, we use difference encoding in order to save space: we determine common left substrings of the current and previously stored strings and store only the length of the common left substring together with the remaining right substring of the current string. Furthermore, after transforming id-values of query results back to RDF literals, we cache the literals with their ids together in order to avoid multiple materializations. We use the strategy of least recently used (LRU) caches to further improve the performance of these materializations.

Evaluation indices: These indices are used for the evaluation of SPARQL queries. They are constructed from sorted id triples according to the 6 collation orders of RDF. The id triples are obtained by using the dictionary index from strings to ids.

Histogram indices: Our plan generator uses equi-depth histograms [23] for result cardinality estimations of triple patterns and for calculating the overall cost of a plan. A histogram is created for a triple pattern and a specific variable of it. Each interval in the histogram contains the number of the triples allocated in this interval, and the numbers of distinct values. In order to speed up the generation of equi-depth histograms, we use an additional special B^+-tree for each collation order. In each inner node of this special B^+-tree, we additionally store the number of triples and the number of distinct subjects, predicates and objects as well as three bits indicating if the subject, predicate or object of the first triple *F* in the subtree is different from the triple before *F*.

Using this special B$^+$-tree and especially its additional information, we can very quickly find the corresponding information related to a triple pattern by not only passing leaf nodes, but also jumping over whole subtrees. Therefore, a histogram for the triple pattern can be constructed very efficiently. This is also shown by our experimental results, where histogram computations take only one or few seconds in most cases and up to approx. 1 minute in rare cases. Without using histogram indices, histogram computations often take hours or even days in rare cases. **Figure 10** illustrates how to construct the histogram for the variable `?v` and the triple pattern `(3, ?v, ?o)` from a histogram index.

Once a histogram has been calculated, it is stored in a separate index, and can be reused for the triple patterns with the same RDF terms and variables at the same positions, but independent from the names of variables. Since these additional B$^+$-trees are only needed for efficiently computing histograms, updates of these B$^+$-trees can be delayed to the times with low workload.

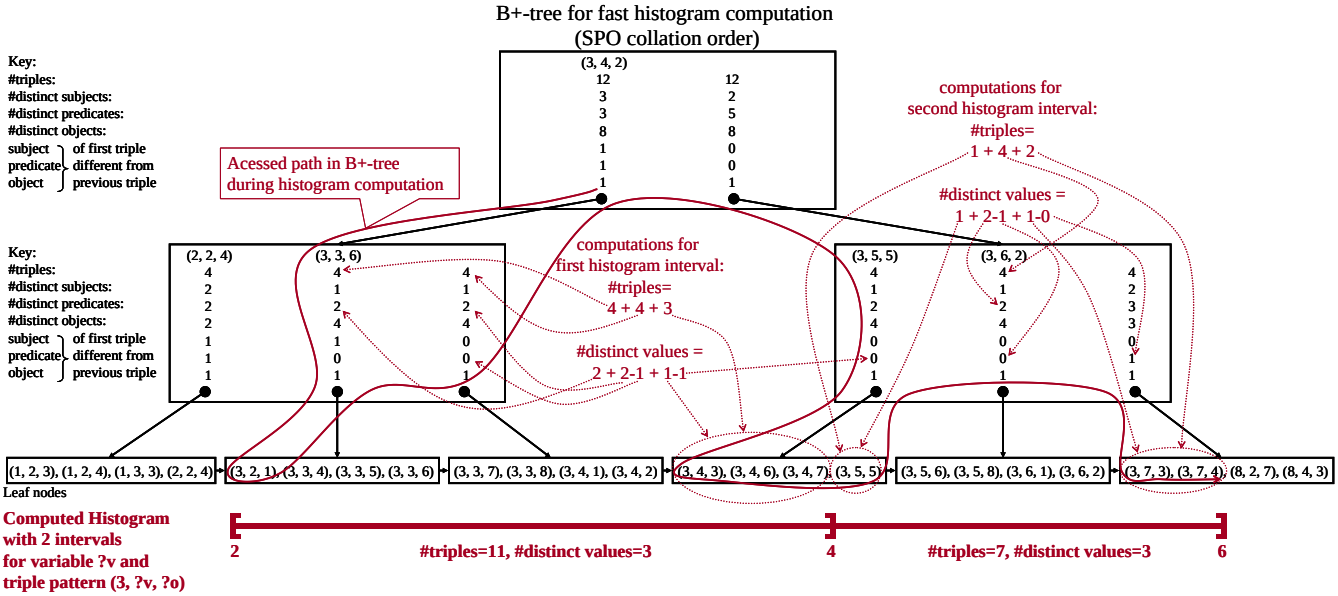


Figure 10: Fast construction of the histogram for the variable ?v and the triple pattern (3, ?v, ?o) using a histogram index

10.1.2 Constructing Execution Plan

Join orders determine the sizes of intermediate results (e.g., [4]), and are an important factor, which influences the performance of query processing. Therefore, a good execution plan depends much on join ordering.

We employ the technique of dynamic programming for generating the execution plan with the optimal join order, i.e., our query optimizer composes a new best solution by considering the best solutions of its subproblems. In more detail, for a set *S* of triple patterns, our query optimizer builds every possible disjoint subsets *S1* and *S2*, the union of which is *S*, i.e., $S = S1 \cup S2$. For each of these subsets *S1* and *S2*, our query optimizer looks up the best solutions for their join orderings, which has been computed earlier, and calculates a new cost for this solution. Among these solutions, our query optimizer chooses the ones with the minimal costs for the overall solution for *S*. Our query optimizer uses equi-depth histograms of triple patterns for calculating the cost of individual joins, and the overall cost of a concrete plan.

Since merge joins without additional sorting phases are very cheap, our query optimizer first chooses such merge joins. If a query contains many triple patterns, we split the set of triple patterns into two sets according to either Cartesian products (first choice), membership of the largest star-shaped joins (second choice) or paths (third choice). The join orders of these subsets are then optimized separately. If there is only one huge star-shaped join or one huge path join, this join is optimized greedy by first joining those triple patterns, the results of which have the smallest cardinality. This strategy scales well for a number of triple patterns and can generate near optimal plans. In our experiments, we have used this strategy for more than 7 triple patterns, which occurs very seldom in real-world queries.

We also integrate important logical optimization rules like pushing filter and variable and constant propagations into our query optimizer. All these logical optimization rules can reduce the sizes of intermediate results.

10.1.3 Differences to the original RDF3X prototype

In RDF data, typed literals like integer values of XML Schema can have several representations, e.g. `2` and `+2`. During query processing, they are treated as identical integer values. Therefore, they should be assigned with the same id. Otherwise, some processing, like the computation of join, might create wrong results. However, the W3C test cases [7] show that the query results must contain the original representation. Similar remarks also apply to language-tagged literals. For example, "`Text`"`@DE` and "`Text`"`@de` are treated as identical values, but the original representation must be maintained for the final result. Thus, we additionally store an extra id, which refers to the original representation in the index, if necessary. Different representations of identical values are only possible for typed literals and language-tagged literals, which can only occur in objects of

triples. Therefore, we only need to additionally store an extra id for objects if the original representation differs from the indexed one. In comparison, the original RDF3X prototype does not support datatypes, and considers e.g. the same integer values 2 and +2 to be two different literals.

The original RDF3X prototype ([21] and [22]) supports additional special aggregated indices for fast processing a special kind of queries. For example, the two triples (<a>, <b>, <C>) and (<a>, <b>, <D>) are aggregated as (<a>, <b>, 2) in the aggregated SP index in order to represent 2 triples with a subject <a> and a predicate <b> (but with different objects). The SP index can be used to fast retrieve the result of a triple pattern <a> ?p ?o, if the variable ?o is neither further used nor occurs in the final result. Considering that such cases occur seldom in real world and the additional costs for maintaining the aggregated indices, we do not implement the aggregated indices.

10.2 Additional BTC Queries

Table 5 presents the query operations performed by the original RDF3X approach.

This section in the appendix presents the additional queries for the Billion Triples Challenge (BTC), used in our performance study in Section 7.

Table 5. Operations by RDF3X for the BTC queries

Query	# merge joins	# hash joins	DISTINCT
BTC 1	3	0	
BTC 2	3	0	
BTC 3	4	0	
BTC 4	5	1	
BTC 5	2	1	√
BTC 6	2	2	√
BTC 7	4	3	√
BTC 8	3	1	
EBTC 1	1	1	
EBTC 2	2	1	
EBTC 3	1	1	
EBTC 4	1	1	
EBTC 5	2	1	
EBTC 6	4	1	
EBTC 7	2	1	√
EBTC 8	1	1	√

We assume that all BTC queries define the namespaces rdf, rdfs, foaf, dbpedia, purl, sioc, skos, atom , geoont, geocountry and geopos by

```
PREFIX rdf:
 <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:
     <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/property/>
PREFIX purl: <http://purl.org/dc/elements/1.1/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
PREFIX atom: <http://www.w3.org/2005/>
PREFIX geoont: <http://www.geonames.org/ontology#>
PREFIX gecountry:
   <http://www.geonames.org/countries/#>
PREFIX geopos:
   <http://www.w3.org/2003/01/geo/wgs84_pos#>.
```

Query EBTC 1:

```
select * where { ?y foaf:maker ?m.
?x foaf:depiction ?d;
   dbpedia:hasPhotoCollection ?y.}
```

Query EBTC 2:

```
select * where { ?l purl:title ?t.
?x purl:title "Wimbledon_College_of_Art";
   sioc:has_creator ?c; sioc:links_to ?l.}
```

Query EBTC 3:

```
select * where { ?x skos:subject
<http://dbpedia.org/resource/Category:1960_in_Form
ula_One>; dbpedia:wikilink ?l.?l foaf:name ?n.}
```

Query EBTC 4:

```
select * where {
?m purl:title ?t.?x foaf:made ?m; foaf:nick ?i.}
```

Query EBTC 5:

```
select * where {
?x atom:Atomuri ?u; atom:Atomname ?n.
?y atom:Atomname ?n; atom:Atomemail ?m.}
```

Query EBTC 6:

```
select * where {
?x geoont:name ?n1; geopos:lat ?l;
   geoont:inCountry geocountry:DE.
?y geoont:name ?n2; geopos:lat ?l;
   geoont:inCountry geocountry:DE.}
```

Query EBTC 7:

```
select distinct ?t where { ?l purl:title ?t.
?x purl:title "Wimbledon_College_of_Art";
   sioc:has_creator ?c; sioc:links_to ?l.}
```

Query EBTC 8:

```
select distinct ?n where { ?x skos:subject
<http://dbpedia.org/resource/Category:1960_in_Form
ula_One>; dbpedia:wikilink ?l. ?l foaf:name ?n.}
```

ABOUT THE AUTHORS:

Jinghua Groppe earned her Bachelor degree in Computer Science and Applications from the Beijing Polytechnic University, her Master degree in Computer Science from the University of Amsterdam and her Doctor degree from the University of Lübeck Germany. She worked as Software Engineer in the Chinese Academy of Launch Vehicle Technology/China Aerospace Corporation and in many European and DFG-projects. Her research interests include Semantic Web, internet programming, query optimization, satisfiability and containment, parallel and distributed systems, cloud computing and XML.

Sven Groppe earned his diploma degree in Informatik (Computer Science) in 2002 and his Doctor degree in 2005 from the University of Paderborn. He earned his habilitation degree in 2011 from the University of Lübeck. He worked in the European projects B2B-ECOM, MEMPHIS, ASG and TripCom. He was a member of the DAWG W3C Working Group, which developed SPARQL. He was the project leader of the DFG project LUPOSDATE. His research interests include Semantic Web, query and rule processing and optimization, cloud computing, data visualization, visual query languages and embedded languages.