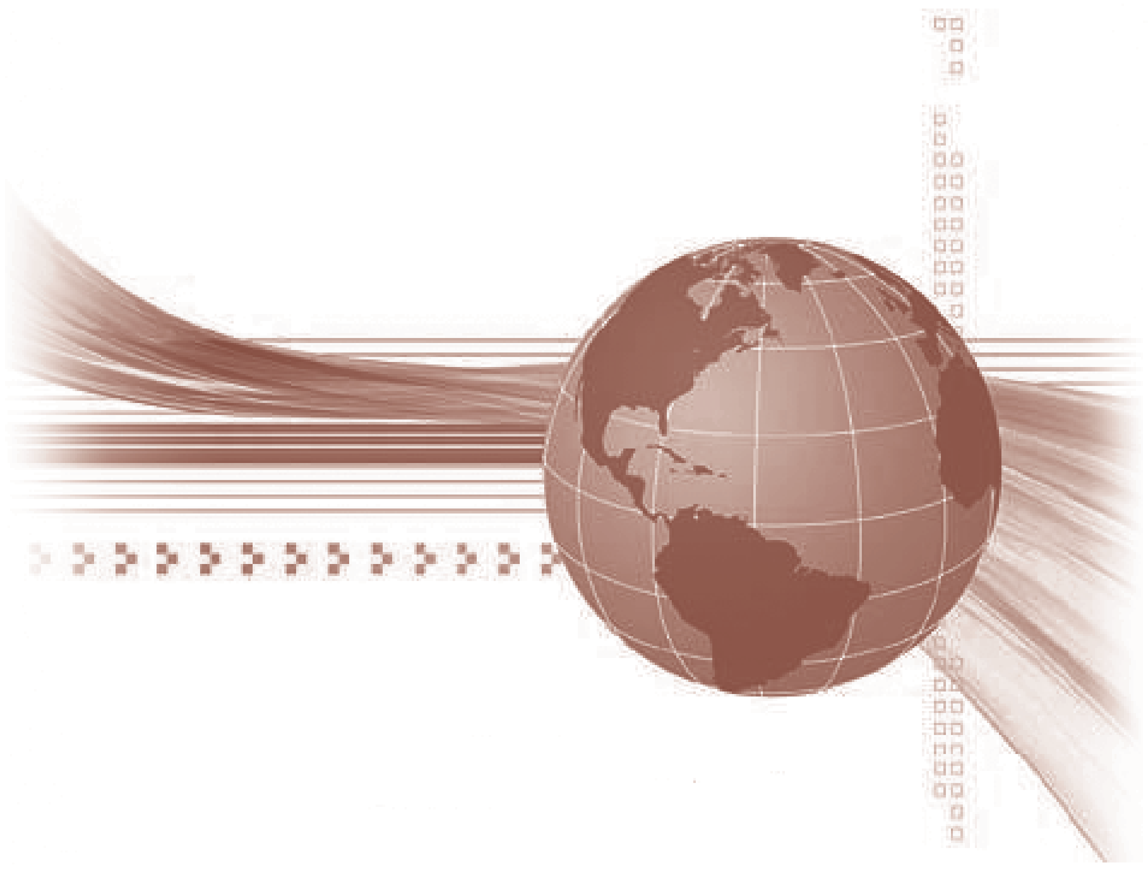
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

4/2012

YEAR Volume 57 (LVII) 2012
MONTH DECEMBER
ISSUE 4

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

4

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

GENERATING PERT NETWORK WITH TEMPORAL CONSTRAINTS

NASSER EDDINE MOUHOUB[(1)] AND SAMIR AKROUF[(2)]

Abstract. A scheduling problem is organizing in time a set of activities, so as to satisfy a set of constraints and optimize the result. The temporal constraint modifies the project scheduling, therefore in loses its characteristics. Our objective is to solve this problem by finding the various types of temporal constraints then modeling them by using graphs. Furthermore we apply a technique for transforming an AoN graph (Activities on Nodes) which is unique and contains a significant number of arcs. This graph is not preferred by practitioners of project management. We transform the AoN graph into an AoA graph (Activities on Arcs) which contains fewer arcs and is preferred by practitioners of project management. In this paper we present some concepts of line graphs and an illustrative example of the proposed method.

1. INTRODUCTION

In project scheduling problems, operational monitoring activities are very important. The project manager draws up the schedule by using graphs. The drawing of AoN (Activities on Nodes) graph also called potential graph or French graph is easy because of its uniqueness despite the large number of arcs it generates. Besides the AoA (Activities on Arcs) graph also called PERT network or American graph is more difficult because of the dummy arcs it generates. However, practitioners prefer to work with the AoA graph because it is easy to read; each activity is represented by an arc. Specialists who insist on using the AoA graph have a number of arguments to justify their choice. This is why according to Fink et al. [1], it is more concise. Furthermore, Hendrickson et al. [2] explains that it is close to the famous Gantt diagram. According to Cohen et al. [3], the structure of the PERT network is much more suitable for certain analytical techniques and optimization formulations.

Received by the editors: June 3, 2012.

2010 *Mathematics Subject Classification.* 90B35, 90B10.

1998 *CR Categories and Descriptors.* G.2.2 [**Discrete Mathematics**]: Graph Theory – *Graph algorithms*.

Key words and phrases. AoA network, AoN network, PERT network, Project scheduling, Temporal constraint.

However, the major disadvantage of this method is in the existence of dummy arcs (see figure 3. (a) and (b)). Their number is likely to be significantly high especially if the size of the network is too large, thus the AoA graph is not unique. In this paper, we focus on finding a method to move from a simple graph (AoN graph) to an AoA graph that will be correct and will respect the scheduling table taking into account prior and temporal constraints. This method will be a draft for the construction of an algorithm that will achieve the transition from the AoN graph to the AoA graph taking into account the temporal constraints.

2. THE PROJECT SCHEDULING

The constraints to which are subjected the various activities, and contributing to the realization of the project, are of various types. We distinguish the potential constraints, disjunctive and cumulative constraints. The potential constraints are the following:

- The constraints of anteriority according to which an activity j cannot start before an activity i is finished, for example, the construction of the pillars follows the foundations
- Temporal constraints which means that a given activity i cannot begin before an imposed date, or that it can be completed after an imposed date.

The problem of scheduling with only potential constraints is called project scheduling problem. Lacomme and al. Paper [4] presents the two conventions which are used in practice for displaying project networks:

2.1. **Activity on node graph (AoN).**

Each activity is represented by a node in the network. A precedence relationship between two activities is represented by an arc or link between the two (see Figure 1). This graph is called the Activity on Node graph (AoN graph).

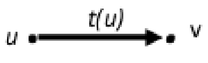


FIGURE 1. The activity u, with duration $t(u)$, precedes the activity v.

2.2. **Activity on arc graph (AoA).** Each activity is represented by an arc in the network. If activity u must precede activity v, there is an arc from u to v. Thus, the nodes represent events or "milestones" (e.g., "finished activity u") like in Figure 2. This graph is called the Activity on Arc graph (AoA graph) or PERT graph.

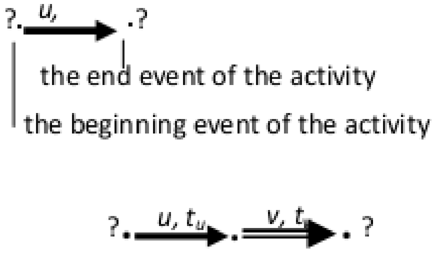

?. u, .?
the end event of the activity
the beginning event of the activity
?. u, t_u . v, t_v . ?

FIGURE 2. The activity u precedes the activity v in PERT graph.

The representation of Table 1 (Figure 3. (a)) in PERT graph, is false; to correct it we introduce an additional activity of duration 0 which does not influence over the total duration of the project. This activity is called a dummy activity. We then modify the table (see Table 2) of scheduling and the PERT graph (figure 3. (b)) the drawing will be easy.
The introduction of the dummy activities gives the possibility to solve certain situations and raise ambiguities. They do not take in consideration any material or financial mean [5].

Code	Predecessors
c	a,b
d	b

TABLE 1. An-under table of precedence of c, d.

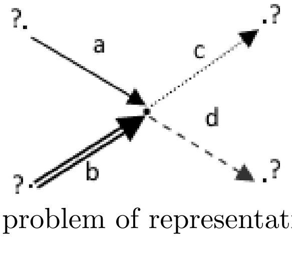


FIGURE 3.(a). The problem of representation in PERT graph.

Code	Predecessors
c	a,f
d	b
f	b

TABLE 2. The new under-table of precedence of C, D and F.

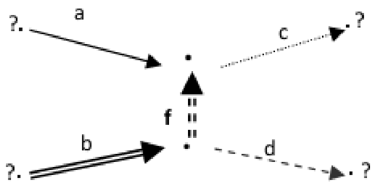


FIGURE 3. (b). Introduction of the dummy activity f and the representation in PERT graph.

For more details for these two methods and their differences, the reader can refer to [6], [7] and [8]. The study of this field is not only in order to facilitate the task to experts, but also for theoretical interests, these are always renewed by the researchers. We can remind that the durations are not mentioned on the different graphs. These durations can be uncertain. For this precise case, there are more details in [9], [10] and [11].

3. THE LINE GRAPH OF GRAPH

Let $G = (X, U)$ a simple or multiple digraph. We build starting from G a graph or line graph noted $L(G)$, called line graph or line digraph of G as follows: The arcs of $L(G)$ are in bijective mapping with the nodes of G for simplicity reasons; we give the same name to the arcs of G and the corresponding nodes of $L(G)$. Two nodes u and v of $L(G)$ are connected by an arc of u towards v if and only if the arcs u and v of G are such as the final end of u matches with the initial end of v, i.e. $T(u) = I(v)$ [12] (see Figure 4).

3.1. **Example.** Let G the following directed acyclic graph be (Figure 4): By definition, any directed graph G admits a unique line graph $L(G)$. On the other hand, two non isomorphs directed acyclic graphs can have the same line graph.

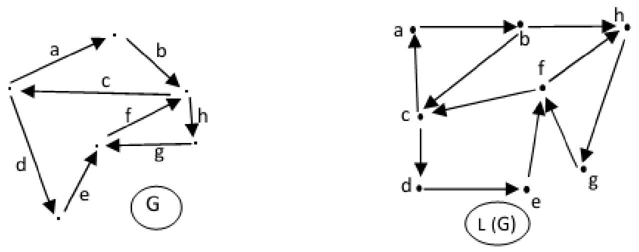


FIGURE 4. A graph G and his line graph $L(G)$.

3.2. **The opposite problem.** We suppose the following opposite problem: Being given a directed acyclic graph H, is it the line graph of any directed acyclic graph? In other words, does there exist a graph G such as $L(G)$ is isomorphs with H, where $H = L(G)$?

G admits a configuration "Z" if G contains four nodes a, b, c and d such as if (a, c), (b, c) and (b, d) are arcs of G, then (a, d) is not an arc of G. With an only aim of simplicity, one will give the name of bar of "Z" to arc (b, c) (see Figure 5) [13].

Configuration "Z" appears when two nodes have common successors and no common successors or by symmetry when two nodes have common predecessors and no common predecessors.

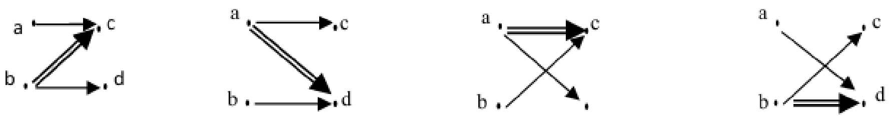


FIGURE 5. The configuration "Z" and his forms.

3.3. **Theorem.** The line graphs have been studied but we will present, in this section, the features in which we are interested and obtained from [12]. H is the line graph of a directed acyclic graph if:

- H does not contain any "Z" configuration.
- Arcs of H can be partitioned in a complete bipartite $B_i = (X_i, Y_i), i = 1, ..., m,$ such as $X_i \cap X_j = \emptyset$ and $Y_i \cap Y_j = \emptyset, \forall i \neq j$.

The bipartite B_i of H are then in a bijection with the nodes also noted B_i which are neither sources nor well, two nodes B_i and B_j of G being connected by an arc from B_i towards B_j if and only if the complete bipartite B_i and B_j of H are such as $Y_i \cap X_j = \emptyset$ (figure 6).

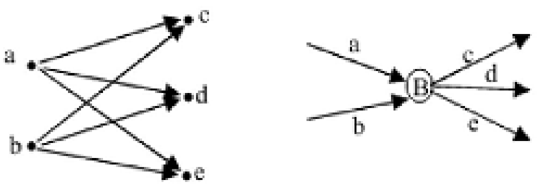


FIGURE 6. A complete bipartite B of G and the star of G.

- *H* does not contain any configuration "Z" and any pair of nodes having common successors has all their common successors.
- Any pair of nodes having common predecessors has all their common predecessors. For more details on this theorem, the reader can refer to [12].

Thus, *H* is not the line graph of any directed acyclic graph if and only if there is a pair of nodes having common successors and no common successors or common predecessors and no common predecessors [5].

4. GENERATING AOA GRAPH

Because of the facility of the use of AoA graph, we must concentrate our efforts on the study of the possibility of transforming the AoN graph (a significant number of arcs) to AoA graph (a reduced number of arcs).
So, we want to know how to transform the graph H (which is an AoN graph) in order to get a new graph which is the line graph (AoA graph). According to [5], the difficulty which arises is to know if H does contain "Z" configurations or not? If it does not, it is a line graph and the transformation is immediate (as in Figure 7).

Code	Predecessors
b_1	$a_1, ..., a_m$
.	..
b_n	$a_1, ..., a_m$

TABLE 3. The sub-table of anteriorities.

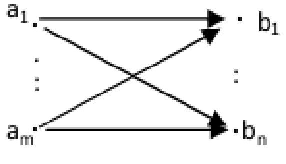


FIGURE 7. (a): The complete bipartite *B* in the AoN graph.

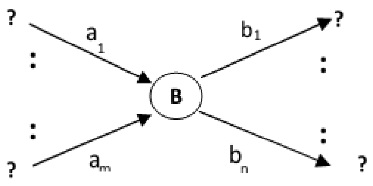


FIGURE 7. (b): The star B of the corresponding AoA graph.

But if it contains “Z” configurations, we have to eliminate the bare from each “Z” preserving the constraints of succession. We then introduce, in the AoN graph, a dummy arc f in every “Z” (Figure 8). The introduction of the

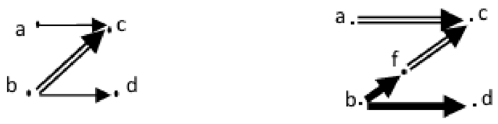


FIGURE 8. “Z” configuration, his corresponding transformation in AoN graph and the partition of the complete bipartite.

dummy arcs aims to eliminate all the “Z” configurations from the AoN graph, the constraints remain unchanged. We should recall that the dummy arcs are not necessary in the AoN graph but are introduced only to build AoA graph. For more details on this transformation from AoN graph to AoA, the reader can refer to [5] and [13].

5. THE TEMPORAL CONSTRAINTS

The temporal constraint is a time allocation constraint. She comes from imperative management constraints such as the supply availability or time delivery, etc.
It specifies the time interval (or semi-interval) during which it is possible to perform or carry out an activity. These constraints are often due to availability of stakeholders (human resources): for example a company which produces frames can only intervene between June 15 and August 31 [1].

The temporal constraint affects the project scheduling and changes. He no longer has the characteristics of the project scheduling. The problem therefore, is to find a way or a technique to normalize the situation and bring it back to the project scheduling. In the following, we will propose an original method which allows us to model the temporal constraints and include them in project scheduling.

We can classify the most important temporal constraints into six types and by adding the precedence constraint they become seven:

C1: Activity A starts t time units before the work begins.
C2: Activity A can only start t time units after the beginning of work.
C3: Activity B must start t time units after the end of activity A.
C4: Activity B starts a fraction of time a/b after the start of activity A (a < b).
C5: Activity B must start t time after the start of activity A ($t < t_A$).
C6: Activity A must start before time t.
C7: Activity B must immediately follow the activity A.

5.1. **Modeling temporal constraints.** In project scheduling, which is a particularly in an AoN graph, incident arcs outside a node (that is to say an activity) have the same value.

The presence of temporal constraints in the graph AoN violates this property, which makes solving the project scheduling impossible. Calculating dates and critical path research ... also become impossible.

Modeling by using graphs can solve this problem. We will present in the following a new technique that allows handling such constraints.

The Figure 9 gives the unique representation of these constraints in the graph AoN. It is clear that the values on the arcs incident to a node outside are different (see the example in Figure 11. (a)). Here we leave the project scheduling.

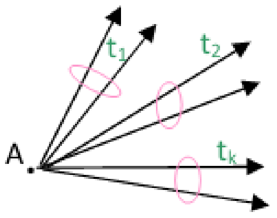


FIGURE 9. Main temporal constraints in AoN graph.

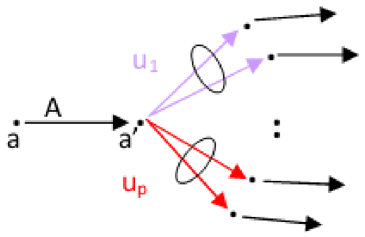


FIGURE 10. (a). Representation of (C2) and (C3) constraints in AoA graph.

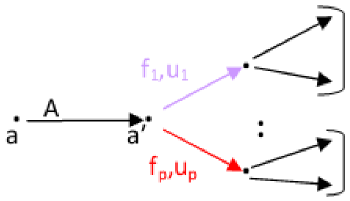


FIGURE 10. (b). Another Representation of (C2) and (C3) constraints in AoA graph with less activities than in (a).

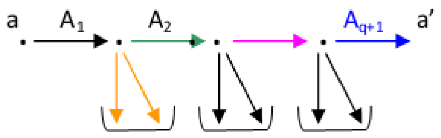


FIGURE 10. (c). Representation of (C4) and (C5) constraints in AoA graph.

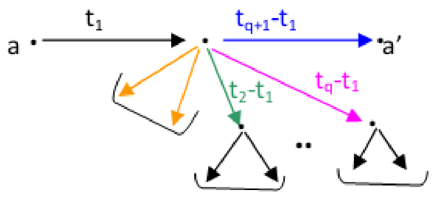


FIGURE 10. (d). Representation of (C4) and (C5) constraints in AoA graph with the same number of activities as in (c).

Figure 10. (a) shows that in the AoA graph, each activity coming after the activity A, has its own dummy arc ui. This representation is poor because the number of dummy arcs may be very important, which clutters the graph.

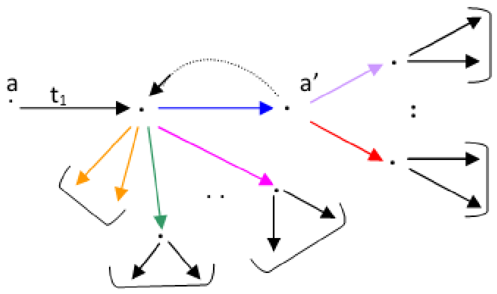


FIGURE 10. (e). type of constraints in PERT graph combining (c) and (d).

A better representation (Figure 10. (b)) consists in gathering several dummy arcs succeeding the real activity A and which have the same value in a single dummy activity. Note that the dummy arc in this context is not of zero duration. She is introduced to solve this problem and introduce the time constraints in the project scheduling.

For constraints of type (C4) and (C5), we notice that both starts after the beginning of activity A. Representation in AoA graph implies the segmentation of A into several tasks, in the general case ($A = A_1 + A_2 + ... + A_{q+1}$). Two models of these constraints are possible (figure 10 (c). and 10. (d)). We note that the representation of figure 10. (d) is more convenient.

Finally, we can combine the figure 10. (b) and 10. (d) keeping in mind the idea of minimizing dummy arcs.

In conclusion, to arrive to figure 10. (e) we must modify in the AoN graph, the arcs incident outside a vertex and who do not have the same value, by introducing dummy arcs of length 0 in order to partition the complete bipartite graph with AoN graph. All these combinations lead us to the changes made in the two graphs AoA and AoN respectively (figure 11)

Correspondence between the representations of temporal constraints in AoN graph and in AoA graph is our goal, we modify figure 9 in figure 11. (a) and figure 10. (e) in figure 11. (b).

The introduction of activities $f_2, ..., f_k$ of times $t_2 - t_1, ..., t_k - t_1$ has the advantage of giving the same value to the arcs of the same initial node in the graph AoN. There is no difficulty to verify that the arcs of the graph (Figure 11. (a)) are partitioned into a complete bipartite graph and that is the associate graph of the graph in Figure 11. (b).

For example, Let A be an activity of duration 5 time units. Suppose that: A precedes B,

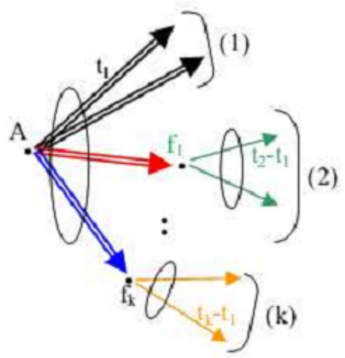


FIGURE 11. (a). modification in AoN graph.

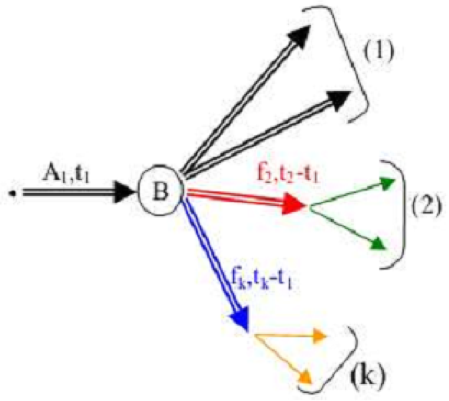


FIGURE 11. (b). Activity A is subdivised as (A_1, f_1) in AoN graph. Arcs of the same initial node have the same value.

B1 and B2 can not start a unit of time after the start of activity A,
B3 and B4 begin only 4 time units after the start of A
B5 can not start until A is 3/4 finished,
B6 and B7 begin only 6 time units after the end of A.
In AoN graph, let us draw the arcs leaving the node A (Fig.12.):

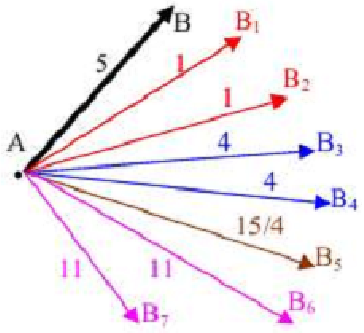


FIGURE 12. (a). No modification in AoN graph.

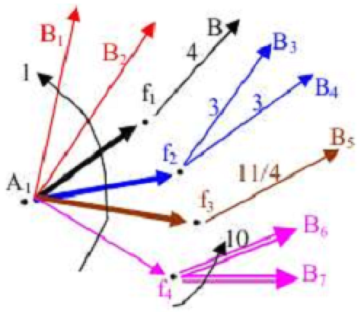


FIGURE 12. (b). is subdivised as (A1, f1) in AoN graph. Arcs of the same initial node have the same value.

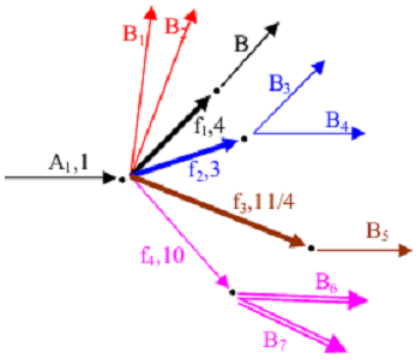


FIGURE 12. (c). No modification in AoN graph.

To illustrate what we have seen since the beginning of this paper and to construct AoA graph from AoN graph taking into account temporal constraints, we consider the following example:

5.2. **Example.** The Table 4 gives the precedence constraints.

Activity	Description	Predecessors	Duratiion
A	Site clearing	6	-
B	Removal of trees	5	-
C	General excavation	8	-
D	Grading general area	4	A
E	Excavation for trenches	3	A, B, C
F	Placing formwork and reinforcement for concrete	9	C
G	Installing sewer lines	2	D, F
H	Installing other utilities	8	E, F
I	Pouring concrete	5	E, F

TABLE 4. Precedence relations and durations for a nine activity project example.

Temporal constraints are:

- *B* can only start 3 time units after the beginning of the work.
- *C* can begin only after 7 time units the work begins.
- *E* begins when *C* is executed to 3/4
- *G* starts 4 units of time after the end of E.

The graphs in Figure 13. (a, b, c, d) show the changes in the AoN graph, then the AoA graph construction:

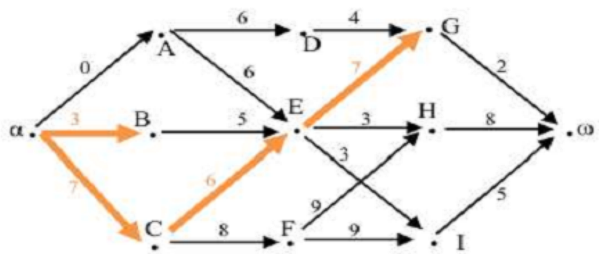


FIGURE 13. (a). graph from the schedule table (see Table 4.). Arcs in bold represents temporal constraints.

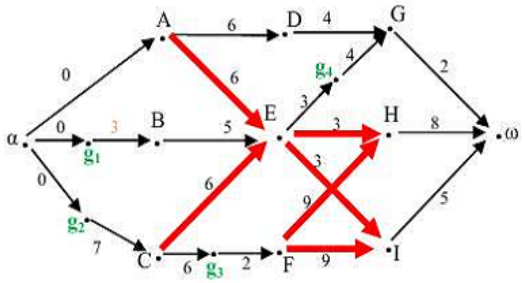


FIGURE 13. (b). AoN graph whose arcs have the same initial node have the same value. The dummy arcs from temporal constraints g_i: activities α, C, E are divided in two activities. "Z" bars are in bold..

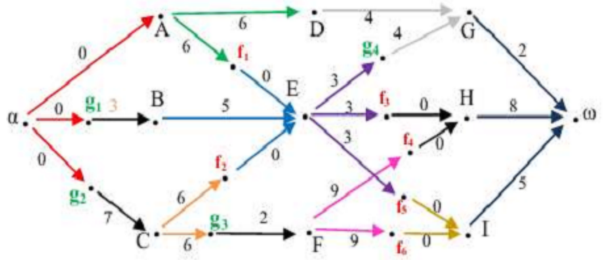


FIGURE 13. (c). The AoN graph with no "Z" configuration and whose nodes are reorganized into levels. We can verify that the arcs can be partitioned into complete bipartite.

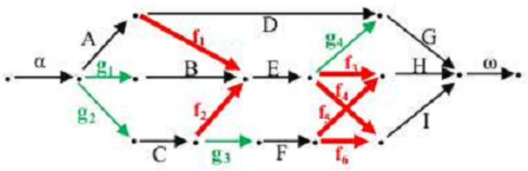


FIGURE 13. (d). AoA graph of Table 4. Activities duration not included (the fi "in bold" have duration zero).

5.3. **Discussion.** The algorithm inspired from this method finishes since the loop is carried out only when there is a 'Z' sub graph or a temporal constraint. The number of 'Z' in AoN graph is known and finite. Also, the number of temporal constraints is known. The rest of the algorithm is a succession of simple instructions. The complexity of the algorithm is polynomial $(O(n^4))$.

6. CONCLUSION

This work has introduced graphs associates in project scheduling problems, with or without the presence of 'Z' in the graph for AoN, for AoA graph construction. He also used the modeling of temporal constraints that can be included in the project scheduling when the resolution becomes easier thus the calculation of dates at the earliest, at the latest, free margins, the critical path, etc becomes possible by applying Bellman algorithm.

This work opens up perspectives, such as searching the minimal PERT graph network in terms of dummy arcs is NP-hard or in terms of nodes. The project scheduling with limited resources can be viewed by using modeling with graphs.

References

[1] Fink, G.: Recherche oprationnelle et rseaux, Lavoisier, Paris,(2002).

[2] Henderckson, C., Project Management for Construction, Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA l52l3, Version 2.2,08.

[3] Cohen, Y., Sadeh, A.: A New Approach for Constructing and Generating AoA Networks, Journal of computer science, 1-1, 2007. *http://www.scientificjournals.org/journals2007/articles/1049.htm*

[4] Esquirol, P. and P. Lopez, P .: l'ordonnancement, ECONOMICA, Paris, France, ISBN 2-7178-3798-1, 1999.

[5] Mouhoub, N. E., Belouadah, H. and Boubetra A.: Algorithme de construction d'un graphe Pert partir d'un graphe des potentiels donn, STUDIA UNIV. BABES BOLYAI, INFORMATICA, Volume LI, Number 2, 2006.

[6] Bernard ROY, Algbre moderne et thorie des graphes, tome 2, Fascicule 3, Problmes d'ordonnancement et ensembles de potentiels sur un graphe, DUNOD, Paris, France, 1970.

[7] A. Haga, Tim O'keefe, Crashing PERT networks: a simulation approach, 4th International conference of the Academy of Business and Administrative Sciences, Quebec City, Canada, July 12-14, 2001.

[8] F. Bacchus and F. Kabanza, Planning for Temporally Extended Goals. Annals of Mathematics and Artificial Intelligence, 22(1-2) :5-27, 1998.

[9] P. Morris, N. Muscettola, and T. Vidal, Dynamic Control of Plans with Temporal Uncertainty, In Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI-01, pages 494-502. Morgan Kaufmann, 2001.

[10] I. Tsamardinos, T. Vidal, and M. E. Pollack, CTP: A New Constrained-based Formalism for Conditional Temporal Planning, Constraints journal, special issue on Planning, 8(4):365-388, 2003.

[11] Neal SAMPLE, Pedram KEYANI, Gio WIEDERHOLD, Scheduling Under Uncertainty: Planning for the Ubiquitous Grid, Computer Science Department, Stanford University, Stanford CA 94305, 2002.

[12] C. Heuchenne, Sur une certaine correspondance entre graphes, Bull. Soc. Roy. Sci. Liege, 743-753, 33, 1964.

[13] Mouhoub, N. E., Benhocine, A. and Belouadah, H.: A new method for constructing a minimal PERT network, (APM) Applied Mathematical modeling, Elsevier ISSN: 0307904X, Vol. 35, Issue: 9, 4575-4588, 2011.

[1] Department of Computer Science, Faculty of Mathematics and Informatics, Bordj Bou Arreridj University, El Anasser, 34030, Algeria
E-mail address: mouhoub_n@yahoo.fr

[2] Department of Computer Science, Faculty of Mathematics and Informatics, LMSE, Bordj Bou Arreridj University, El Anasser, 34030, Algeria
E-mail address: samir.akrouf@gmail.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

NETWORK ROUTING MODELLED BY GAME SEMANTICS

DANIEL MIHÁLYI, VALERIE NOVITZKÁ, PETER PRAZŇÁK,
AND PETER POPOVEC

Abstract. An important task in network routing is to find the best path for message passing through the computer networks. In our paper we show how network routing can be described by linear logic formulae and modeled as a game tree in terms of game semantics.

1. Introduction

One of the primary needs of routing in computer networks is to state the best path for delivering a message from a sender to a receiver with respect to a given network topology. Presently, information about available paths are dynamically actualized in every router and it is stored in dynamic routing tables. The best way is determined according to an actual situation in a network by executing appropriate algorithm returning a value of the least metrics computed from path cost, channel capacity and speed, eventually response time and other parameters. From the computer network's point of view, a choice of the best path is a mostly deterministic process depending on an actual network environment. Hovewer, a user does not have and he does not need any information about the environment in which his message is passed to a receiver, and the best path choice appears him as a non-deterministic process.

The aim of our paper is to propose a formal description of network routing using an appropriate logical system in such a way that it reflects both the user's and enviroment's point of view, respectively. Then we construct a model that is illustrative enough for the best path choice.

For the description of network routing, we choose the language of linear logic introduced by J. Y. Girard in [11]. In contrast to classical logical systems, linear logic can express the dynamics of processes, sequentiality of particular actions, and eventually, resource manipulation, namely space and time on

Received by the editors: June 10, 2012.

2010 *Mathematics Subject Classification.* 03F52, 91A40.

1998 *CR Categories and Descriptors.* F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic – *Computational logic*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *Games*.

Key words and phrases. game semantics, linear logic, network routing.

logical level. Linear logic has many applications in computer science. The processes described by formulae of linear logic can be modeled by Petri nets [18], Turing machines, special categories [21] or game semantics. There exist several logic programming languages based on linear logic, e.g. Lygon [15], LPP [4, 19], etc.

In our approach we exploit the special property of additive conjunction that can be understood variously from different points of view. The author of linear logic interpreted additive conjunction as a choice between alternative actions conditioned by an environment. Girault in his work [14] sees the sense of additive conjunction as a deterministic selection, i.e. a kind of deterministic choice in a situation, where several actions are available but only one of them can be performed. In our approach, we interpret this conjunction as a synthesis of both previous views for the designation of the best path. This logical connective expresses non-deterministic choice from the user's point of view and dependent choice from the network's point of view arising from actual environment.

The semantics of linear logic was formulated by various models: coherent spaces, phase spaces [12], symmetric monoidal closed categories [3], game semantics, etc. For our approach, we use the game semantics method that models linear formulae as polarized game trees.

Game semantics is based on the game theory formulated by John Von Neumann and Oscar Morgensen in 1944. Game theory [20] is a mathematical discipline enabling to model real situations by games and it is used mainly in economics. It helps to solve decision problems and to search winning strategies. Originally, game semantics was formulated by P. Lorentzen [17] for the justification of intuitionistic logic. Game semantics was formulated in 90-ies of 20. century and has found many applications in computer science. For instance, modeling of interactions [1], defining semantics of various logical systems and programming languages [8, 7], modeling and verification of software [2, 9], in hardware design [10], in linguistics and artificial intelligence.

Game semantics for linear logic was formulated and published by many authors, e.g. [1, 2, 5, 6, 8]. A structure used as a model of this method is a game tree which is a suitable structure for representing behavior of computer networks. A game tree is a directed graph without loops and its firm enables us to see the posssible winning strategy in finding the best path from a source to a destination.

In Section 2, we present a short overview of linear logic and we introduce the syntax of the fragment of linear logic suitable for our purposes. Section 3 introduces the basic notions and principles of game semantics together with an interpretation of linear logic connectives. In Section 4, we show how linear logic can be used for describing network routing by linear formulae and how its game semantics can serve for modeling network routing by game tree.

2. Linear logic

Logical systems play important role in computer science. Classical logics (propositional and predicate logics) have their well-established applications within description and proving of some program properties. They are often used in specification languages for stepwise refinement of specifications to implementation, in program verification and many other areas of computer science.

Non-classical logic, for instance temporal logic enables to state the truth of formulae depending on time. Further modal logics enrich proposition or predicate logics with the operators of the necessity and possibility (classical modal logic), knowledge and belief (epistemic logic), obligation, permission and forbiddance (deontic logic). None of these logical systems does not be able to describe dynamics of processes running in real world already on syntactic level. The first logical system enabling to describe dynamics of processes, their causality and resource handling is linear logic formulated by J. Y. Girard in 1987 [11, 12]. The subformulae of the classical logic's formulae may be either true or false. This property is statical, the truth value of a subformula is stable in the frame of whole formula. Therefore we can say that classical logic has statical nature, it is suitable for description of fixed situations, but not for the description of dynamics in real world processes. The formulae of linear logic can be regarded as resources that can be produced and consumed. This fact can be expressed by linear implication

$$\varphi \multimap \psi \tag{1}$$

where a resource φ is consumed after performing linear implication, which can be denoted as linear negation $\varphi^{\perp}$. A resource ψ is produced by performing linear implication (1). For instance, if we would like to travel by a train we must to have some amount of money φ for buying a ticket ψ. After an execution of this process $\varphi \multimap \psi$, our amount of money is consumed, $\varphi^{\perp}$, but we obtain a new resource, a ticket ψ.

Linear formulae can be regarded also as actions. In this case linear implication expresses causality. An action φ is a cause of an action ψ.

Linear logic introduces new logical connectives. In this paper we use a fragment of linear logic with the following syntax:

$$\varphi ::= \mathbf{0} \mid \mathbf{1} \mid \top \mid \perp \mid p \mid \varphi \otimes \psi \mid \varphi \multimap \psi \mid \varphi \,\&\, \psi \mid \varphi \oplus \psi \mid \varphi \parr \psi \mid \varphi^{\perp} \tag{2}$$

where

- p denotes atomic actions without internal structure;
- $\varphi \otimes \psi$ is multiplicative conjunction expressing that both actions φ and ψ are performed. The neutral element of this logical operation is the constant $\mathbf{1}$;
- $\varphi \multimap \psi$ is linear implication where action φ is a cause of an action ψ;
- $\varphi \,\&\, \psi$ is additive conjunction expressing external non-determinism. Only one action is executed depending on environment. The neutral element of this logical operation is the constant $\top$;
- $\varphi \oplus \psi$ is additive disjunction expressing also the execution of one action, but we do not determine which one will be performed. Additive disjunction expresses internal non-determinism, it is a dual logical operation to additive conjunction. The neutral element is the constant $\mathbf{0}$;
- $\varphi \parr \psi$ is multiplicative disjunction expressing: if φ is not performed then ψ is performed and vice versa;
- $\varphi^{\perp}$ denotes linear negation, it expresses that after performing an action φ the reaction $\varphi^{\perp}$ arises. Linear negation is involutive i.e. $\varphi^{\perp\perp} \equiv \varphi$.

3. Game semantics

In this section we formulate the semantics of our fragment of linear logic in terms of game semantics. We use dialogue games where two players participate: a proponent and an opponent. The notion game denotes a collection of rules how to play it. An actual performing of a game according these rules we call a game session. The aim of a game session is to find winning strategy i.e. a sequence of moves of proponent and opponent leading to success. Sometimes, a move of a proponent or an opponent is called a half move, if we are interesting only in winning strategy for a proponent. In the case of linear logic a proponent efforts to prove the truth of formula, an opponent efforts to deny the validity of this formula. A winning strategy is a sequence of moves leading to the success of proponent, i.e. to proving formula's validity.

Game semantic provides very useful graphical representation of a game session called game tree. A game tree is a pair

$$T = (V, A) \tag{3}$$

i.e. a directed graph consisting of a set of vertices V and a set of arrows A. A game tree for our fragment of linear logic is a digraph without loops. The vertices represent the positions in a game session and the arrows represent the moves of proponent and opponent. A game session is represented by a path from the root to the leaves in a game tree. We are looking for winning strategy, i.e. a path in game tree, which leads to the win of a proponent. The root of the game tree is the starting point of a game session. The leaves contain information about its success or non-success.

We formulate the game semantics of our fragment of linear logic as follows: every logical connective can be translated to a fragment of a game tree.

Additive conjunction $\varphi \mathbin{\&} \psi$ expresses dependent choice that can be interpreted as a half move of a proponent drawn by the dash lines in Figure 1. The vertice $\varphi \mathbin{\&} \psi$ has two sons reflecting the situation that only one of the actions φ and ψ will be executed. A decision, which of them will be performed depends on proponent or implies from the environment.

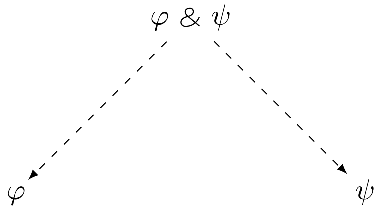


Figure 1. Interpretation of additive conjunction

In the case of additive disjunction a process can follow also only with one action but a proponent cannot predict with which one. Additive disjunction can be translated to the fragment of a game tree in Figure 2.

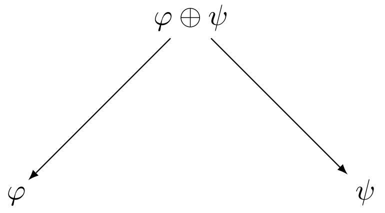


Figure 2. Interpretation of additive disjunction

To translate multiplicative conjunction $\varphi \otimes \psi$ to a fragment of a game tree is harder. Multiplicative conjunction expresses that both actions φ and ψ will be performed. Which of them will be performed as the first depends on a proponent's decision. Figure 3 reflects this idea.

Multiplicative disjunction $\varphi \parr \psi$ expresses a situation: if φ is not performed then ψ is performed or if ψ is not performed then φ is. A fragment of a game tree for multiplicative disjunction is in Figure 4.

$$\begin{tikzcd}
& \varphi \otimes \psi \arrow[dl, dashed] \arrow[dr, dashed] & \\
\varphi \arrow[d, dashed] & & \psi \arrow[d, dashed] \\
\psi & & \varphi
\end{tikzcd}$$

FIGURE 3. Interpretation of multiplicative conjunction

$$\begin{tikzcd}
& \varphi \parr \psi \arrow[dl] \arrow[dr] & \\
\varphi^{\perp} \arrow[d] & & \psi^{\perp} \arrow[d] \\
\psi & & \varphi
\end{tikzcd}$$

FIGURE 4. Interpretation of multiplicative disjunction

Linear implication $\varphi \multimap \psi$ expresses causality, i.e. an action φ is a cause of an action ψ. Following this idea, linear implication can be translated to the fragment of game tree in Figure 5.

$$\begin{tikzcd}
\varphi \arrow[d] \\
\psi
\end{tikzcd}$$

FIGURE 5. Interpretation of linear implication

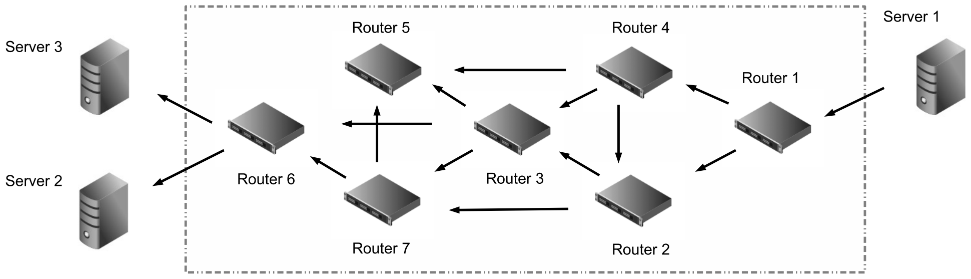


Figure 6. Sending a message trough a network

In the syntax of our fragment of linear logic we do not consider exponentials. The exponentials $!\varphi$ and $?\varphi$ enable to express inexhaustibility and potential inexhaustibility, respectively. Translation of these operators introduces loops into the game trees which become directed graphs with loops and they are not suitable for our approach.

4. Example: Network routing

In this section we present a simple example of network routing. We show how this problem can be formulated by linear formulae and modeled by game semantics.

We assume a network in Figure 6 consisting of seven routers denoted by $Router1, Router2, \ldots, Router7$ and three servers $Server1, Server2$ and $Server3$. We consider a case that $Server1$ sends a message trough this network to $Server2$. We are interesting in all possible paths from a source server to a target server. This situation is resolved at 3rd layer of OSI model - at network layer - where routing protocols are defined. Information about the possible paths in a network are stored in the routing tables that can be

- static routing tables or
- dynamic routing tables.

If the static routing tables are used, a message follows the path stated explicitly by network administrator. In a case of dynamic routing tables, the passing paths are actualized depending on an interconnection state between the adjacent routers. If some path is not accessible either of overloading or connection interruption, one of the other accessible paths is selected.

In the first step, we specify this problem using linear logic. We denote by

- $p_i, i = 1, \dots, 7$ express the i-th router in our network;
- $p_i \multimap p_j$ is linear implication expressing that a router p_i sends the message to the router p_j directly, i.e. p_i and p_j are neighbouring and interconnected;
- assuming dynamic routing tables, linear additive conjunction $p_i \,\&\, p_j$ expresses that a message proceeds either to the router p_i or p_j. That means both routers are accessible but a user cannot predict which path will be used. Form the network's environment a choice is made depending on a value of metrics.

From Figure 6 we see that several accessible paths through this network are possible. A message enters into a network through the router p_1 and outputs from this network through the router p_6. From the router p_1 a message can follow either to p_2 or to p_4. This move we can specify by a linear formula

$$p_1 \multimap (\ p_2 \,\&\, p_4\ ). \tag{4}$$

Similarly, we specify how a message can follow from any router in our network. The particular steps we can describe by the following linear formulae:

$$\begin{array}{l} p_2 \multimap (\ p_3 \,\&\, p_7\ ) \\ p_3 \multimap (\ p_5 \,\&\, p_6 \,\&\, p_7\ ) \\ p_4 \multimap (\ p_2 \,\&\, p_3 \,\&\, p_5\ ) \\ p_7 \multimap (\ p_5 \,\&\, p_6\ ) \end{array} \tag{5}$$

An accessible path in our network can be specified by linear implication, for instance

$$p_1 \multimap p_4 \multimap p_3 \multimap p_7 \multimap p_6 \tag{6}$$

We see that from the router p_5 there is no passing path to p_6, therefore we have no linear formula starting from p_5. For p_6 we cannot formulate linear implication, because a message leaves a network through this router and follows to the given server. From Figure 4 it is straightforward that every path ending in p_5 is dead end.

Using translation rules introduced in previous section, for every formula in (4) and (5) we can construct a game tree in Figure 7. The root of this game tree is an antecedent of linear implication p_1 with one son, the succedent of implication $p_2 \& p_4$. This vertice has two sons p_2 and p_4 representing dependent choice. Applying this consecution for every vertice we get a game tree where leaves are either p_6 or $\Box$. The path ending with p_6 is passing path, a sended message successfully passes through our network. The paths in game tree ending with $\Box$ indicate dead ends, i.e. these paths cannot be used for transporting

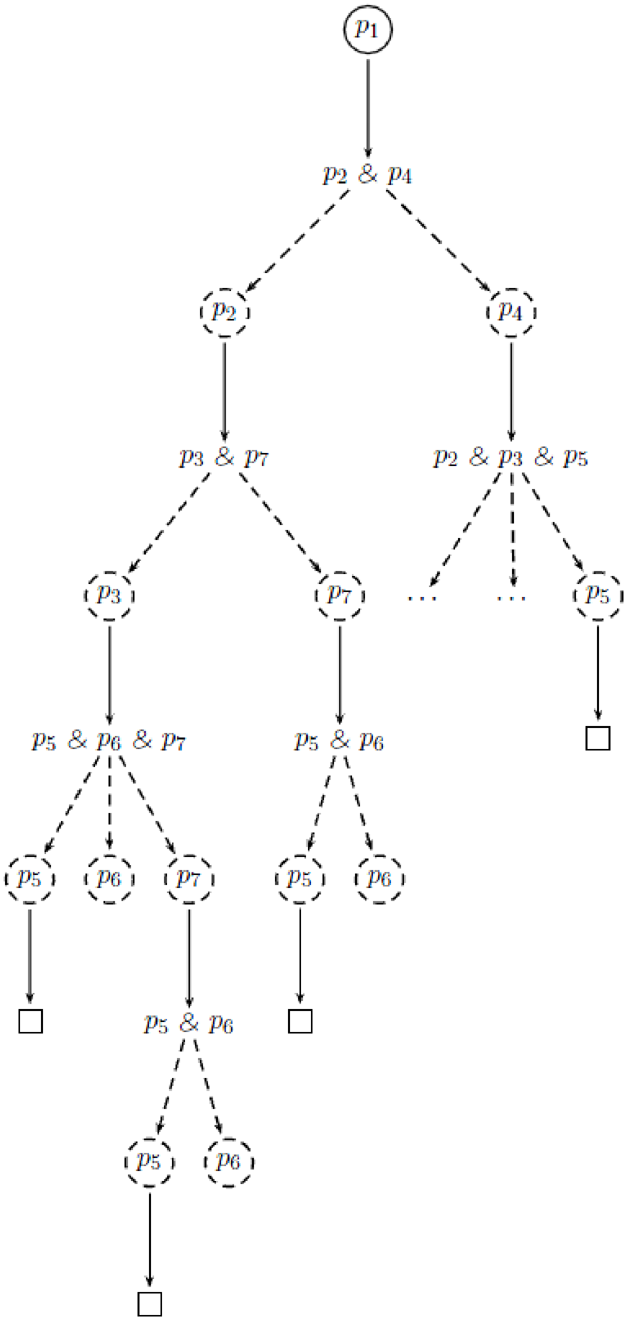


FIGURE 7. Game tree for network routing

a message through our network successfully. If we simplify a value of metrics as a number of hops, we obtain this information from a path's deep in a game tree. From Figure 7 we see that the shortest path is $p_1 \multimap p_2 \multimap p_3 \multimap p_6$ and it is the best path for routing in our network.

5. Conclusion

In our paper we present an illustration how network routing can be described by formulae of linear logic and modeled by game semantics. We define interpretation of linear connectives by the fragments of game tree. Our approach seems to be appropriate to specify dependent choice of the best network path from a game tree.

In our further research we would like to extend our approach by including exponentials to our fragment of linear logic expressing that some actions can be potentially used repetitiously. Another extension cane be done by labeling moves by metrics values computed by some version of Dijkstra algorithm. We hope that our approach enriched with exponentials can be suitable for specifying and modeling reliable program systems.

6. Acknowledgement

This work was supported by the Slovak Research and Development Agency under the contract No. APVV-0008-10 "Modelling, simulation and implementation of GPGPU-enabled architectures of high-throughput network security tools."

This work is the result of the project implementation: Center of Information and Communication Technologies for Knowledge Systems (ITMS project code: 26220120030) supported by the Research & Development Operational Program funded by the ERDF.

References

[1] Abramsky, S.: Semantics of Interaction: An Introduction to Game Semantics, In: Proceedings of the 1996 CLiCS Summer School, Isaac Newton Institute, P. Dybjer and A. Pitts, eds. (Cambridge University Press) 1997, pp.1-31

[2] Abramsky, S., Ghica, D., Ong, L., Murawski, A.: Applying Game Semantics to Compositional Software Verification, Proc. of the 10th Intern.Conf. Tools and Algorithms for the Construction and Analysis of Systems, Springer LNCS 2988, 2004, pp.421-435

[3] Ambler, S.,J.: First Order Linear Logic in Symmetric Monoidal Closed Categories, Research Report ECS-LFCS-92-194, University of Edinburgh, 1992

[4] Banbara M., Tamura N.: Compiling Resources in a Linear logic Programming Language, Proceedings of JICSLP'98 Implementation Technologies for Programming Languages Based on Logics, June 1998, pp. 32–45,

[5] Blass A.: Is Game Semantics Necessary?, In (Boerger, E., Gurevich, Y. and Meinke, K. eds.): 7th. Workshop Computer Science Logic, CSL'93, Springer LNCS 832, 1994, pp. 66–77,

[6] Blass A.: A Game Semantics for Linear logic, Annals of pure and Applied Logic, Vol 56, 1992, pp. 182–220,

[7] Curien P.-L.: Notes on Game Semantics, Techn.Rep.CNRS University of Paris, 2006

[8] Delande, O.: Symmetric Dialogue Games in the Proof Theory of Linear Logic, PhD. Thesis, École Politechnique 2009,

[9] Dimovski, A., Lazic, R.: Assume-Guarantee Software Verification Based on Game Semantics, Proc. on Formal Methods and Software Engineering, Springer LNCS 4260, Macao, 2006, pp. 529–548,

[10] Ghica D.,R.: Application of Game Semantics: From Program Analysis to Hardware Synthesis, School of Computer Science, University of Birmingham, 2009,

[11] Girard, J.-Y. : Linear Logic, Theoretical Computer Science, London Mathematical 50:1, pp. 1-102, 1987.

[12] Girard, J-Y., Lafont, Y.,Taylor, P.: Proof and Types, Cambridge University Press (Cambridge Tracts in Theoretical Computer Science, 7), 1990

[13] Girard, J-Y.: Linear logic: Its Syntax and Semantics, Cambridge University Press, 2003

[14] Girault C., Valk R.: Petri nets for systems engineering - a guide to modeling, verification, and applications, Springer, ISBN 3-540-41217-4, 2003, pp. 370–382,

[15] Harland J., Pyms D., Winnikof M.: Programming in Lygon and Overview, Algebraic Methodology and Software Technology, Springer 1996,

[16] Lafont, Y.,Streicher, T.: Games Semantics for Linear Logic, Logic in Computer Science (LICS 91), p. 43-50, IEEE Computer Society Press, 1991,

[17] Lorentzen P.: A Dialogue Criterium for Constructivity, In Infinistic methods, Warszaw, 1961, pp. 193–200

[18] Mihályi D., Novitzká V., Slodičák V.: From Petri Nets to Linear Logic, CSE'2008, Vysoké Tatry - Stará Lesná, 24. - 26. 9. 2008, Košice, Elfa, 2008, pp. 48-56, ISBN 978-80-8086-092-9,

[19] Miller D.: Overview of Linear Logic Programming, Linear Logic in Computer Science, Cambridge University Press, 2000,

[20] Peters, H.: Game theory - a Multi-leveled Approach, Springer-Verlag, 2008

[21] Slodičák V.: Some Useful Structures for Categorical Approach for Program Behavior, Journal of Information and Organisation Sciences, Vol. 35, No. 1, 2011, pp. 93–103

Technical University of Košice

E-mail address: `(Daniel.Mihalyi,Valerie.Novitzka)@tuke.sk, praznak@minv.sk,, popovec@fei.tuke.sk`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

ISSUES IN COLLECTIONS FRAMEWORK DESIGN

VIRGINIA NICULESCU, DANA LUPŞA, AND RADU LUPŞA

Abstract. A good framework/library can reduce the cost of developing an application. This study is an exploration of issues related to designing collections frameworks by analyzing the existing approaches and by emphasizing the fundamentals and the main desiderata of such developments. The corresponding theoretical concepts are analyzed, and for defining data structures an approach based on properties is discussed. Also, some design leading questions are specified in order to emphasize possible new development approaches.

1. Introduction

In the imperative programming setting data structures represent an old and a very important issue. So, different libraries and frameworks have been built in time, based on different programming paradigms.

Initially, the focus was on the structure of the data and on different strategies used for their representation into the memory. The behavior of such a structure was not strictly defined since anyway, the encapsulation of the data with the operations was not yet possible.

By introducing the concept of abstract data type [2], data structures were defined in a more accurate and formal way, by introducing well defined types. A step forward has been done on this subject with object oriented programming (OOP) - a higher order of abstraction being achieved.

Based on OOP we may not only define generic data structures by using type or parametric polymorphism, but also we can separate the definitions from the implementations by using interfaces [9]. Design patterns ([5], [7], [8]) moved things forward, and introduced more flexibility and reusability for data structures.

Genericity is another important issue related to the field of data structures. Any kind of data structure for a collection is formed by a number of elements which are usually of the same type. A specific collection has properties and behaviour which are not dependent on the type of its constitutive elements. So, generally, we may consider them as shape-dependent structures. When properties such as sorting are introduced they could became value-dependent structures.

Received by the editors: September 12, 2012.

2000 *Mathematics Subject Classification.* 68P05.

1998 *CR Categories and Descriptors.* E.1 [**Data**]: Data Structures; E.2 [**Data**]: Data Storage Representation .

Key words and phrases. data structures, collections frameworks, genericity, representation.

1.1. **Motivation and organization of this paper.** Our purpose is to investigate the possibility to create a good framework/library for working with collection data structures.

It is well known that a good framework can reduce the cost of developing an application by an order of magnitude because it lets you reuse both design and code. To consider the problem of software reuse in the moment of the design is not an easy task. A nice description of what this means is made in [10]: "Developing reusable frameworks cannot occur by simply setting down and thinking about the problem domain. No one has the insight to come up with the proper abstractions. Domain experts won't understand how to codify the abstractions that they have in their heads, and programmers won't understand the domain well enough to derive the abstractions. In fact, often there are abstractions that do not become apparent until a framework has been reused."

Collection class libraries have been criticized as being too unwieldy, too inflexible and generally difficult to use [16]. It has been pointed out, for example, that to provide for future flexibility the introduction of many incrementally different types is needed, but huge hierarchies are hard to understand and to use [11].

On short, a good framework is one that makes the programmers job easier and programs better [16]. Much of the engineering effort should go into the design of the class and the type system [11]. A clean, good design would keep things easy to implement and easy to use. In order to achieve this, our research investigates basic, fundamental properties of collection.

In order to make a fundamental analysis on the collections design we start with a short presentation of collections (Section 2). Section 3 collects some important research leading questions. We analyze existing solutions in Section 4 and Section 5. In Section 4, we present programming paradigms used by two existing solutions and emphasize their advantages and disadvantages. In Section 5, we examine some existing approaches for designing collections framework and compare them with our ideas.

2. Important issues regarding collections

2.1. **Collection and containers.** The term container in modern computer programming can actually refer to many things. A **container** is an object created to hold other objects. While the term container is used in C++ STL to denote collection containers, Java programmers use the term "collections" rather than "containers". We may consider that the meaning of "container" term is more related to the storage, and the term "collection" is more related to the represented concept.

In this paper, we are going to use the term collection. Exceptions are situations when we are presenting concepts or terminology used by C++ STL.

Java documentation [1] says that **a collection** is sometimes called a container and it is simply an object that groups multiple elements into a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data.

Collection	*Alternate names*	*Remarks*
Bag	multiset, collection	admit duplicate elements
Set		no duplicate elements
Sequence	list	elements are arranged in a strict linear order; order information has nothing in common with the elements themselves; it is provided as separate information within operations (when needed)
Stack		**specific:** insertions/extractions are made following a fixed (predefined) strategy: Last In First Out (LIFO);
Queue		**specific:** insertions/extractions are made following a fixed (predefined) strategy: First In First Out (FIFO);
DeQueue		**specific:** insertions/extractions can be made *to/from both ends*
Map	unique associative container, associative array, dictionary	elements are of type (key, value) ; keys are unique
Multimap	multiple associative container	elements are of type (key, value); keys are not unique

Table 1. The most used collections

We present in table 1 a list of most important collections that are known as widely used in programs, and that are provided by most collections frameworks. Collections alternate names and some important remarks are also presented in the table.

A study of collections framework design should pay special attention to its conceptual and logical foundations. A theoretical study of general collection properties is investigated in relation with STL [17] in the next section.

2.2. Collection container properties.

A property is a feature that could be added to a data structure and then could be removed; it is something that fundamentally characterized the data structure.

Starting from general collection container concepts presented in [17], we make an inventory with the main properties that can make the difference between containers.

- unordered - sequence
- multiple - unique
- simple - pair
- non-associative - associative

The first property can be considered as default. These are unordered, multiple, simple, non-associative. A bag is a collection defined by using only default properties.

We should note that associative property means that we have a pair collection and the first element in the pair is the key - that have special properties associated with it. For example, associative containers are supposed to offer fast access to elements based on keys. That imposes restrictions over the possible choices to implement the collection.

Sorted is a property can be added to many collections but, in this way, they became value-dependent structures. In this study, we are not going to include it, since we restrain the analysis only to shape-dependent structures.

Collection	*sequence*	*unique*	*key-ed (associative)*
Bag	-	-	-
Set	-	yes	-
Sequence	yes	-	-
Stack	used for operation specification	-	-
Queue	used for operation specification	-	-
DeQueue	used for operation specification	-	-
Map	-	yes	yes
Multimap	-	-	yes

TABLE 2. Collections and their properties

2.3. **Building collections by their properties.** In this section we are going to present collections mainly based on the properties enumerated before.

We remarked that associative property is discussed only when we have a pair collection. Associative collections support efficient retrieval of elements (values) based on keys. That is why we are going to use only one property that we name it *key-ed* instead of having pair and associative properties.

Table 2 defines the basic collections based on the considered properties.

Still, we may observe that a *set* can be also defined as unordered, unique and key-ed collection. That is: the element is its key, no value is used, and we ask for efficient retrieval of elements based on keys.

Note that some of the basic collections imply specific restrictions regarding operations. Examples are adding and extracting elements for stacks, queues, and deQueues. Another example is the element access operation for key-ed collections. They are not expressed by using properties. Starting from this observation we may consider an extension of the properties set by including stack or queue element access types as properties. But this kind of approach would lead not only to an exponential growth of number of properties combination, but also will add, as properties, restrictions that are usually defined at operations level.

3. What to consider when designing collections frameworks

In order to make a fundamental analysis on a collections framework design we may start by putting some important research leading questions:

- Which are the fundamentals that should be considered when designing collections framework? (In this paper, section 2 presents shortly collections concepts and their properties).
- Is a hierarchical approach appropriate for developing collections framework? (see section 4.1) Are the types that correspond to the main collections in relations of subtyping kind only? If not, what other relations should be considered?
- What are the solutions to assure genericity? Which is best? (see section 4.2)
- How certain levels of abstractness influence collections framework properties? (See section 4.3.) Which has to be the leading focus: the needed behavior or the performance?

Within each of the above questions, we also have in mind the next question:

- What are the important lessons to be learn from others experience?

4. Collections framework design: a short overview of two existing solutions

In this section we are going to analyze STL versus Java Collection Framework (JCF) ([12], [15]). Both JCF and C++ STL provide collections frameworks. There are a number of differences, some of them stem from the language features and philosophy. Some other differences are simply design choices and we are not going to present them here.

4.1. **Interfaces versus Concepts.**

To define collections, Java [1] uses a clean separation between interfaces: `Collection`, `List`, `Set`, `SortedSet`, `Map`, `SortedMap` and implementations: `ArrayList`, `LinkedList`, `HashSet`, `TreeSet`, `HashMap`, `TreeMap`.

On the other hand, STL containers are all concrete classes, with no interface-implementation hierarchy, in order to make them more efficient.

4.1.1. *Problems defining a hierarchy.*

Java tried to classify collections based on their properties and behaviour. But things are not straight forward and Java classifications suffered small modifications over time. For example, `Vector` became `ArrayList`, while `Vector` stayed only for compatibility and is deprecated in the current version.

Note that, even if Java language is based on only one hierarchy of classes (derived from `Object`), in JCF there is not only one class hierarchy: `Map` is not a `Collection`. In the same time, STL `map` and `unordered_map` are truly containers of key-value pair (`std::pair<K,V>`).

In [16] it is specified that Java `Map` doesn't extend `Collection` by design. Collection could be made to extend Map, but forcing this, it leads to an unnatural interface. If a `Map` would be a `Collection`, its elements should be key-value pairs, but this provides a very limited `Map` abstraction. There are two important problems: accessing a

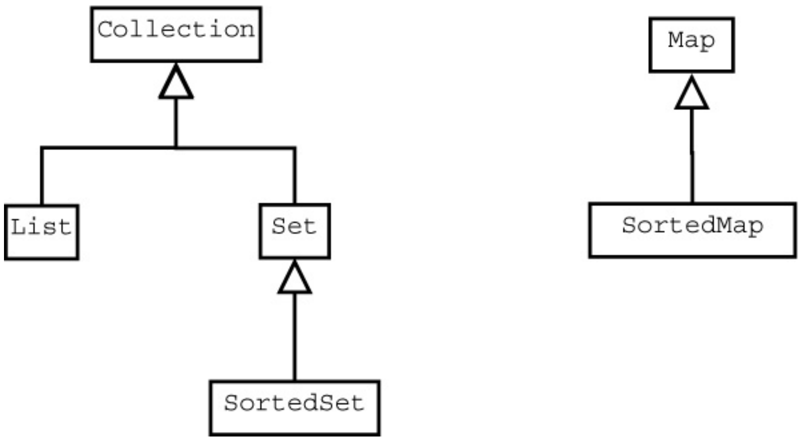


FIGURE 1. Java Collections Framework major interfaces.

value for a given key, and deletion of an entry for a given key, without knowing the value it maps to.

4.1.2. *Uniform naming scheme.* STL's containers have a uniform naming scheme, with identical names for functions with identical roles. Note that a set of member function names and arguments, together with their semantic, is called a *concept*, and a class that actually implements those functions is said to *implement the concept.* As an example, an STL container is required to have a pair of functions, `begin()` and `end()`, returning iterators to the first element and, respectively, just after the last element. This implicitly implies that there is an order between elements, so that we have a first and a last element. Although the concepts are not part of the language, but rather a convention between programmers, and there cannot exist a run-time polymorphism based on concepts, concepts allow a **compile-time polymorphism**: a function template can be parameterized on a type that implements the container concept; thus, that function can call `begin()` and `end()` on the argument of container type.

4.2. **Genericity.** Parametric genericity, initially represented in object oriented setting by source code reuse mechanism as C++ templates, became more and more popular and other object oriented languages as Java and C# enhanced their new versions with mechanisms that offer parametric types.

In C++ the template mechanism allows us not to create a single class, but to specify only once the pattern for creation of some classes that are different only by the type of some parameters. The template mechanism allows a very high degree of flexibility, but it is considered in some literature not a really parametric polymorphism mechanism since for each actual parameter a new class is created.

The mechanism which was included in Java since JDK 1.5 is considered more efficient since just one class is created for each parameterized class. Also, the mechanism

of parameterized Java classes allows bounded polymorphism – the specification of a certain behavior of parameters by interface implementation.
A similar mechanism is implemented in C# too.

A comparison between C++ templates and the extensions for generics of the C# and Java languages based on their suitability for scientific computing was done in [6]. These measurements suggest that both Java(TM) and C# generics introduce only little run time overhead when compared with non-parameterized implementations. With respect to scientific application, C# generics have the advantage of allowing value types (including builtin types) as parameters of generic classes and methods. Also, in [3] there is study about the performance of generics for scientific computing in various programming languages, based on a standard numeric benchmarks. The conclusion was that in current implementations of generics must be improved before they are used for efficiency-critical scientific applications.

The C++ templates mechanism is considered for implementing parametric polymorphism based on a "heterogeneous" approach. The "heterogeneous" approach constructs a special class for each different use of the type parameters. The compiled code is fast, but the object code could become bulky since we have many different versions of each class.

Java generics implement "homogenous" approach of the parametric polymorphism. Since is based on "type erasing" we have strong restrictions, and maybe the most important is represented by the impossibility of specifying static members for the generics.

4.3. **Level of Abstractness.** From the previous comparison we may conclude that Java collections design is based on the corresponding abstract definitions (Abstract Data Types), when STL design has more utilitarian focus. So we may conclude that the abstraction is higher for first one. If the goal would be to establish to which extend these approaches are the most appropriate and useful for the common developers, then the task would be very difficult. The well trained developers may consider that the STL approach offers rapidity and better safety. On the other hand JCF is much easier to learn and deal with.

A low level of programming focuses on performance and usually doesn't use an intermediary tool as a framework. A framework design should be leaded by the behavior but in the same time it should not ignore the performance and safety. The choice should consider the programmer needs.

5. Some other collections frameworks approaches

There are others collections frameworks as well. The Guava project contains several of Google's core libraries that they rely on in their Java-based projects: collections, caching, primitives support, concurrency libraries, common annotations, string processing, I/O, and so forth. Each of these tools really do get used every day by Googlers, in production services [14].

There are also extensions of existing frameworks. For example, utilities available in java.util.Collections (from JCF) are extended by fastutil by providing type-specific maps, sets, lists and queues with a small memory footprint and fast access and insertion. Fastutil came up as a necessity during the development of a web crawler, as they needed to manage structures with dozens of millions of items very efficiently [13].

Another way to model existing collections, is to reconsider the way they are defined. A collections framework based on set theory is Yet Another Collections Library (YACL) [18]. The project YACL consider a model in which Function extends Relation extends Set. Bags and Sequences extend Function. Hence a Function (equivalent to Sun's Map class/interface) is a type of Set. They build a theoretically sound collections library on the top of JCF Set: Set implements java.util.Set. All classes can be constructed from java.util collections and maps (where applicable).

In [4], the aim is to define a collection library for Java which uses multiple inheritance to offer a flexible framework for defining collection types rather than providing a complex exhaustive set of particular collection classes. They identify a small number of software engineering concepts relevant to the design of libraries of collections. They distinguish three basic orthogonal semantic properties of collections: ordering of elements, definition and handling of duplicate elements, definition of keys for efficient search. They use the next properties : order (ordered, sorted, userOrdered), duplicates (duplFree, duplIgnore, duplError) and search (searchable) that are intended to extend JCF. Particular collection types should be built by using derivation and by specifying their properties in terms of these basic types. For example, the interface type 'Bag' can be defined as:

```
interface Bag[ELEMENT] extends Collection[ELEMENT] {}
```

and the type 'List' as:

```
interface List[ELEMENT]
extends UserOrdered[ELEMENT], Bag[ELEMENT] {}
```

Because we want to get an clean design, our idea is not to extend other collections, but rather to redefine collections themselves in terms of their properties. We identified some concepts relevant to the design of collections framework. Among them, there are some that are present in [4]: UserOrdered as a property that we named Sequence (in opposition with unordered), Duplicates (that is in opposition with unique, and we named it multiple). The definition of List in [4] corresponds to our definition, but we avoid using bag and collections in the same time. We also tried to avoid using any reference to bag when specifying a list. We also considered one new property in our classification (key-ed) although it was not our purpose to make an inventory of some new and not yet existing properties, as we based our approach on STL concepts [17].

We reconsider the way collections are defined by considering small number of software engineering concepts relevant to the design of libraries of collections.

6. Conclusions

Since in the literature there are different classifications and definitions for the types corresponding to different collections, the existing implementation solutions - frameworks - are also very different. The authors of this paper are conscious of, and try to overview different initiatives. In this paper, an inventory of theoretical concepts is made and existing collections frameworks are compared. We have specified some design leading questions and for each of them we have done an analysis in specific sections of this paper. Their role is to emphasize possible new development approaches.

The goal of this paper is to to investigate possible approaches to the design a good framework for working with collection data structures, by analyzing the existing solutions and by emphasizing the fundamentals and the main desiderata of such developments.

References

[1] J. Bloch, *The Java Tutorial. Trail: Collections* `http://docs.oracle.com/javase/tutorial/collections/`.

[2] L. Cardelli, P. Wegner, *On understanding types, data abstraction, and polymorphism* ACM COMPUTING SURVEYS, (1985).

[3] L. Dragan, S.M. Watt, *Performance Analysis of Generics in Scientific Computing*, Proceedings of Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05), 2005, pp.93-100.

[4] M. Evered, G. Menger, J. L. Keedy, A. Schmolitzky, *A Useable Collection Framework for Java*, 16th IASTED Intl. Conf. on Applied Informatics, Garmisch Partenkirchen, 1998.

[5] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object Oriented Software*, Addison-Wesley, 1994.

[6] J. Gerlach, J. Kneis, *Generic programming for scientific computing in C++, Java, and C#*. Lecture Notes in Computer Science. Proceedings of International Workshop on Advanced Parallel Processing Technologies (APPT) Xiamen, China, 2003, pp.301-310.

[7] D. Nguyen, *Design Patterns for Data Structures*, SIGCSE Bulletin, 30, 1, March 1998, 336-340.

[8] V. Niculescu, G. Czibula, *Fundamental Data Structures and Algorithms. An Object-Oriented Perspective*, Casa Cartii de Stiinta, 2009 (in Romanian).

[9] V. Niculescu, *Storage Independence in Data Structures Implementation*, Studia Universitatis "Babes-Bolyai", Informatica, Special Issue, LVI(3), pp. 21-26, 2011.

[10] D. Roberts, R. Johnson, *Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks*, in Proceedings of the Third Conference on Pattern Languages and Programming, 1996, `http://st-www.cs.illinois.edu/users/droberts/evolve.html`

[11] C. Szypersky, S. Omohundro, S. Murer *Engineering a Programming Language: The Type and Class System of Sather*, in Programming Languages and System Architectures, ed. J. Gutknecht, Springer-Verlag, pp. 208-227, 1993.

[12] B. Stroustrup, M. Ellis, *The Annotated Reference C++ Manual*, Addison-Wesley, 1994.

[13] *fastutil: Fast & compact type-specific collections for Java*, `http://fastutil.dsi.unimi.it/`

[14] *Guava project*, `https://code.google.com/p/guava-libraries/`

[15] *Generic Java*, `http://download.oracle.com/javase/1.5.0/docs/guide/language/generics.html`

[16] *Java.The Collections Framework*, `http://download.oracle.com/javase/1.5.0/docs/guide/collections/`

[17] *STL Programmer's Guide*, `http://www.sgi.com/tech/stl/index.html`

[18] *YACL - Yet Another Collections Library*, `http://sourceforge.net/projects/zedlib`

Department of Computer Science, Babeş-Bolyai University, Kogalniceanu 1, 400084, Cluj-Napoca, Romania

E-mail address: {`vniculescu, dana, rlupsa`}`@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

CROSS-SITE SCRIPTING BROWSERS' PROTECTION ANALYSIS

ADRIANA NEAGOŞ AND SIMONA MOTOGNA

Abstract. The purpose of this paper is to present an overview upon the protection methods browsers provide against cross-site scripting, as an important security vulnerability. Time has imposed different measures and competitors on the browsers market had to keep track. The study has three goals: to take into consideration versions of Internet Explorer, Google Chrome, Mozilla Firefox and Opera, to use an established scoring system, the Common Vulnerability Scoring System, to measure certain vulnerabilities on each browser, and to develop a tool for web application flaws testing.

1. INTRODUCTION

The success or failure of an application is characterized by three factors: quality, time and cost. Software development focuses on delivering applications with minimal resources (people, software components and hardware) and quality is not always set as an important issue. If time and cost are related to the project management and can be changed on the spot, quality assurance is a complex process that should not be neglected. Several studies (NASA [21], IBM [13]) have led to an important conclusion: improving quality reduces development costs.

Last years brought us in front of an explosion of web applications. Desktop products are replaced by the three layer architecture of the server machine, the client machine and the network as delivery mechanism. In these conditions, software quality factors are changing their importance, and security becomes an important factor because of the transactions that are made and because of the sensitive data that is sent over the internet. In order to ensure security, people need an evaluation scale of a software application and in consequence,

Received by the editors: October 30, 2012.

2010 *Mathematics Subject Classification.* 68N30, 68M11.

1998 *CR Categories and Descriptors.* C.2.0 [**Computer-Communication Networks**]: General – *Security and protection*; C.4 [**Performance of Systems**]: *Measurement techniques*.

Key words and phrases. Cross Site Scripting, Security.

a lot of research, both in academia and industry, focuses on studying risks, vulnerabilities, and attacks against security. Open Web Application Security Project (OWASP) [22] is such an initiative, and maintains and updates a list of top 10 security risks.

Cross-Site Scripting (XSS) is the second as importance, and considered as having an average exploitability and a high degree of occurrence. There are a lot of cases in which automatic tools detect this risk in an easy way, but, however, there are some situations, generated by new technologies and browser characteristics, that make detection more difficult.

The purpose of this article is to present an in-depth analysis of detecting and preventing cross-site scripting vulnerabilities and their impact on software security factor. The rest of the paper is organized as follows: Section 2 presents some of the existing related work. Section 3 contains an analysis of XSS vulnerabilities and the main prevention methods. Section 4 focuses on the security policies proposed by different browsers and contains a case study, while the next section is dedicated to our evaluation of some XSS vulnerabilities. The proposed tool is presented in Section 6, and in the end we draw some conclusions and future directions of study.

2. RELATED WORK

A lot of research has been carried out in the field of XSS vulnerabilities. Most of them focus on studying pattern attacks, evaluating risks and proposing solutions to prevent them [14], [26], [9], but as far as we know there is no paper comparing XSS vulnerabilities from the browsers' point of view. Release notes gather a sum of new features, but do not always point out the security issues and the user does not have a proper overview of what the measures taken imply and how does the situation look like considering the rivals on the market.

Regarding Security Evaluation and Measurement, discussed in the last part of our paper there are several approaches. Besides the approaches carried out at major software companies, such Microsoft, IBM, Apple a.s.o., there are two important contributions to assessing security vulnerabilities and proposing metrics to evaluate their impact on software quality:

- Computer Emergency Center (CERT) at Carnegie Melon University with results in risk analysis, based on a tactical and on a systematic approach, and security measurement, that are integrated in IMAF (Integrated Measurement and Analysis Framework) [6].
- Common Vulnerability Scoring System (CVSS) [17] that developed a framework that supports scoring of security vulnerabilities.

3. ANALYSIS OF CROSS SITE SCRIPTING VULNERABILITIES

Cross-site scripting vulnerabilities were discovered back in 1996 when web pages became more interactive due to a new programming language, Javascript and were associated with the name of Netscape Communications, the most popular browser at that moment. Even if it were first reported as Web browser vulnerabilities, David Ross demonstrated in his paper "Script Injection" [10] that the problems may also come from the server side rather than from the client. The vulnerability was first named "HTML injection" and then referred with the acronym "CSS", but the confusion with Cascading Style Sheets determined in 2000 a new convention, "XSS".

People tended to underestimate the power of XSS, because it could not damage the operating system or exploit a database, but the attack on October 2005, when the first major XSS worm, called the Samy Worm spread on over a million of MySpace accounts rose the public attention.

Javascript, ActiveX, Silverlight, or Flash are lightweight programming languages used for a friendly user experience and for a more dynamic interaction with the application. They provide code that is executed by the browser on the client and is the way XSS is performed. Browsers execute this kind of code under sandboxing mechanism which means that only a set of operations should be performed, but even this protection is not enough. Michael Howard said that "All input is evil until proven otherwise. That's rule number one" [11], but in case of XSS not only inputs, but also outputs must be validated.

There are 3 types of XSS:

- *non-persistent or reflected*: is performed when there is no proper validation of user input through GET or POST requests and the response page is returned immediately and is spread generally by email via malicious urls;
- *persistent*: happens when the infected code is stored in the database and it is a regular threat to chat software or application including different posts. User does not access malicious links, just regular browsing can result into being robbed of information;
- *DOM-based*: results from dynamically-computed data, which means that the browser is manipulated to render DOM elements controlled by an attacker.

The following is an example of reflected XSS in ASP.NET web forms. We consider a form used in a registration page in which the user is required to input some data and then submit. The page validates the input on the server side and then returns a message. Suppose the input requires an email address, but this address should not have been submitted before, so this is why the validation is done on the server and not on the client. If the email address is not valid an error message will be returned. The form looks like this:

```
⟨form id = "form1" action = "#" runat = "server" method = "get"⟩
  ⟨div⟩
    ⟨div id = "errorDiv" runat = "server"⟩⟨/div⟩
    ⟨input id = "myinput" runat = "server"/⟩
    ⟨button onserverclick = "serverClick" runat = "server"⟩Submit⟨/button⟩
  ⟨/div⟩
⟨/form⟩
```

Now when the user clicks submit button, function *serverClick* is executed:

```
⟨script language = "C#" runat = "server"⟩
  void serverClick(Object sender, EventArgs e)
    {
      errorDiv.InnerHtml = "The following is not a valid email" +
        myinput.Value + "Please go to⟨ahref =' #'⟩Help⟨/a⟩";
    }
⟨/script⟩
```

If the user inputs some valid data everything works just fine, but XSS can be performed by sending the following input:

```
⟨div onmouseover =' alert()'⟩some valid input⟨/div⟩
```

Now that an attacker sees that the page is vulnerable he may change the javascript function executed on mouse over with some malicious code and send the URL asking for opinion on the received error. The web page seems trustful and the action is done on mouse over, so most of the users would not be suspicious receiving this.

We will use the same example for the stored XSS. Suppose the attacker has entered a valid email address, but he injected the following script in other input:

```
⟨script⟩window.location = "http : //maliciouspage.com"⟨/script⟩
```

In a normal scenario he should be redirected to a welcome page in which all the other users would be shown. Still, because of the script he inserted, script that was inserted in the database as he was careful to double the quote, the redirection page wouldn't be the welcome page, but his malicious site. Now, all the users after registration would be redirected to that page.

The example presented above is just a proof of concept. It is a personal ASP.NET web page, using Microsoft SQL Server 2008 R2 and tested on Internet Explorer 9.

Monitoring and improvement in preventing web applications against cross-site scripting follow the general steps used in avoiding vulnerabilities on web: code review, manual testing, automated testing and implementation of security policies on browsers, firewalls, each of them having pros and cons and assuring security to a certain extend.

3.1. **Code review.** There are two methods of reviewing code: a static analysis of the code when lines are reviewed without being executed or a dynamic analysis at the runtime. It is feasible for small and middle applications, but in case of complex products, it can be very laborious and time-consuming. It is difficult to verify all the tainted data if, for instance, code contains dynamic string-building techniques and predict the application's client state. Best practices advise white lists for the inputs and proper encoding for the outputs. Still, white lists are not always possible.

Any of these: Request.Params[”input”], Request[”input”], Request.Query String [”input”], Request.Cookies[”input”], Session[”input”], Application [”input”] can be sources of exploit and injection can be performed on tags like: ⟨ body⟩, ⟨frame⟩, ⟨img⟩, ⟨html⟩, ⟨layer⟩, ⟨link⟩ etc. or their attributes src, href, style.

Programming languages provide in-built functions that perform this kind of filters, but even Microsoft states regarding their ASP.NET method *ValidateRequest* that one should not rely only on this type of validation because unfortunately it is not 100 percent secure. Recent attacks prove this. Not only ASP.NET functions have security lacks. *parse_ url* is a function in PHP that verifies malformed urls. The function works correctly in most cases except if whitespaces are inserted. This was the vulnerability exploited on April 2011, on Facebook, when a malicious video was posted [25] or on CNN when urls inserted in ad networks were source of this attack. Other exploits were done also on The New York Times, on Twiter, e-Bay or on Fox News [15].

Microsoft offers an Anti-Cross Site Scripting Library [20] and OWASP advises programmers to use an API: ESAPI (The OWASP Enterprise Security API) [22] which is an open source web application security control library.

3.2. **Testing.** Testing any kind of input based vulnerability involves the same pattern: detection of a flaw, injection of data and confirmation of the exploit. A proxy can be used in order to intercept the HTTP traffic and TamperIE [4] for modifying the GET and POST requests.

Even if OWASP [22] classified it as easy detectable, we tend to disagree because of the variety of malicious strings: keyword ⟨script⟩ is not mandatory when code is inserted in body as attribute or event, quotes and double quotes can be alternated, whitespaces inserted or the statements can be encoded such that they are skipped by the filters such as:

$\rangle'''\rangle\langle script\rangle alert('XSS')\langle/script\rangle$

$\langle object\ type = text/htmldata ='' javascript : alert(([\mathrm{code}]);'' \rangle\langle/object\rangle$

$\langle body\ onload ='' javascript : alert(([\mathrm{code}])'' \rangle\langle/body\rangle$

There are several XSS Cheat Sheets to be used while testing, but the possibilities of injection seem infinite. Scanners and fuzzers were implemented to automate testing, Burp Suite [24], HP WebInspect [12], Acunetix [1] or Netsparker [16] are such applications. Mainly, they use the same mechanism: they insert a harmless string named XSS locator in the identified application data entries, scan the output and then perform the exploit in order to confirm the injection. The problem is that they test only using common scripts and technologies such as AJAX or JSON are not supported as they make this automation very difficult.

3.3. **Firewalls.** Runtime protection methods should also be taken into consideration, even if they affect the performance of the application. Web application firewalls(WAF's) monitor the communication across the network and therefore they inspect messages for Javascript and can enforce a set of rules in order to identify and to block XSS attacks that would not reach anymore the backend. Examples of such applications are Cisco ACEWeb Application Firewall [7], NetScaler App Firewall [8] or Barracuda Web Application Firewall [3]. Most WAF's implement the Intercepting Filter pattern or include one or more implementations in their overall architecture. One can also add filters to an application at deployment when implemented as a Web server plug-in or when activated dynamically within an application configuration.

Users should be also educated to avoid XSS exploits. Avoiding awkward links, paying attention to redirections or for instance turning off the HTTP TRACE can prevent the stealing of cookies.

Regardless the variety of prevention methods, new exploits continue to attack web applications and it's our duty to keep on protection against the known or unknown security flaws.

4. SECURITY POLICIES

We have intentionally omitted security policies from the above protection methods in order to provide a more in-depth analysis of this concept. XSS is the one security field that does not depend on the type of connection: encrypted or unencrypted, but is closely related to portability and mainly browser compatibility. Because it is rendered by different browsers, the display of a web page can be slightly different, and so its gate of access.

Same origin policy is called the policy adopted against browser-side languages that does not allow "access to most methods and properties across pages on different sites". This means that the sensitive information held about a certain user belongs and can be used only by the original site and cannot be accessed by other bad targeted sites. It is implemented by nearly each browser, but it does not guarantee complete security. In addition, modern browsers implemented several security policies that block an attacker to gain access on a client machine.

Firefox and Opera are known as relatively secure, while Internet Explorer (IE) is considered very vulnerable. A simple example is for instance when uploading text files through IE: if HTML content is inserted in the file, it doesn't treat it as plain text, but it interprets it as HTML. As this is not the most relevant example, we will try to make a fair analysis of the most used browsers, their evolution and the measures taken against XSS attacks.

Internet Explorer 6, Firefox 4 and Opera 9.5 introduced HttpOnly cookie attribute. It was intended to protect against retrieving information through document.cookie, but even with this, cookie information could be accessed through XMLHttpObject. Internet Explorer 7 made sure information was hidden and unavailable in the response header only if it was submitted from the same domain, but not the other browsers. "HttpOnly cookies don't make you immune from XSS cookie theft, but they raise the bar considerably" [2].

Fraud Protection was called the mechanism adopted by Opera 9.5 to enforce security. It was a tool powered with phishing information from Netcraft and PhishTank, and Malware protection from Haute Secure that provided automatic detection and warning of malicious sites and supported Extended Validation certificates.

Starting with Internet Explorer 8, a new very controversial browser filter for XSS has been introduced. It is similar to a proxy and the filtering is done using regular expressions. Still, Michael Brooks describes it as vulnerable and users claim that it also considers safe pages as potentially dangerous [5] and Google disables it by setting the X-XSS-Protection in the header to 0 or it can be turned off from the browser security tab. Similarly, Chrome 4 responded to the security features included in IE 8 and introduced a static analyzer which attempts to detect XSS, called Anti-XSS Filter. Adam Barth, a software engineer on the Chrome team, said "The XSS filter checks whether a script that's about to run on a Web page is also present in the request ... that's a strong indication that the Web server might have been tricked into reflecting the script"[1]. But even if it could have been considered a revolution in the security against XSS, methods to bypass the filter's protection and provide full disclosure appeared shortly.

Firefox 4 came with NoScript XSS Filter, an add-on that verifies the form of URLs. It reports if they have a suspicious content, but this without confirming the attack. It also added HTTP Strict Transport Security which informed the browser to automatically convert all attempts to access a certain site using HTTP to HTTPS requests instead. This was also supported by Chrome 4 and Opera 12 Beta, but not by Internet Explorer.

[1]from http://jetlib.com/news/tag/adam-barth/

In March 2011, together with the release of version 4, Firefox proposed the adoption of a new layer to enforce XSS protection called the Content Security Policy (CSP). This provides a way that helps the browser to differentiate between legitimate and malicious content. CSP requires that all JavaScript for a page are 1) loaded from an external file, and 2) served from an explicitly approved host. It means inline scripts are ignored unless they are defined in a white list. This framework is still not implemented yet on other browsers, but Microsoft claims that it will be a feature of Internet Explorer 10.

The methods stated above and adopted by the browsers are mainly applied against reflected XSS, as for the other two types of attack, it is not the duty of the browser to enforce protection.

4.1. **Case Study.** In the following lines, we will present a simple example of XSS attack against one of our web pages. It is an edit page that receives as parameter the ID of a document and then using this filtering retrieves information about that document and provides a way to modify the stored information (available at the url http://scs.ubbcluj.ro/ naie1000/phplab/editDoc.php). EditDoc.php?ID=1 is for example a valid request, but we have also tried inserting scripts. The simplest example

$\langle script\rangle alert(1)\langle /script\rangle$

was avoided in all browsers because of the position where ID parameter was inserted and because of the protection provided.

First script that was successfully executed was:

$'''-\rangle\langle /style\rangle\langle /script\rangle\langle script\rangle alert(0x000040)\langle /script\rangle$

So, let's see how different players on the browsers market react.

Internet Explorer provides a popup at the bottom of the page saying: "Internet Explorer has modified this page to help prevent cross-site scripting." and encodes characters in the url as follows:

$'\%22--\%3E\%3C/style\%3E\%3C/script\%3E3Cscript3E$
$alert(0x000040)\%3C/script\%3E$

Opera also makes this kind of encoding, but the url is not totally visible. The parameter is hidden and can be seen only if the user selects it on purpose. This way, an inexperienced user can easily be fooled and trapped in some attack of this kind. The alert appears on the page.

Firefox instead does no encoding and shows the alert without no validation.

Chrome also skips encoding, but the alert is not visible. The user is not informed about the blocked script as in the case of Internet Explorer, but it is protected against it.

To confirm the behavior we have tested using more different scripts and the four browsers reacted in the same way as above.

We used a harmless simple script just for example, but in the same way the alert is executed: any script requesting for instance document.cookie or redirecting the page to a malicious site can be inserted. We also used the default settings of each browser for a proper analysis of the vulnerability.

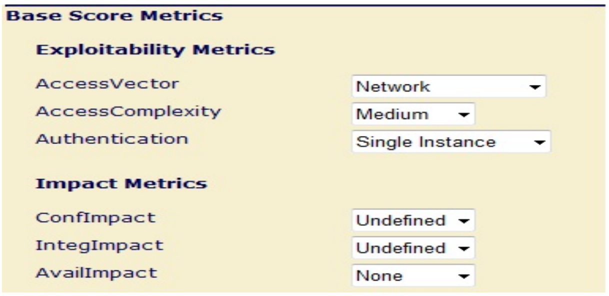
5. CVSS SCORES FOR XSS VULNERABILITIES

Our case study consists in computing the CVSS vulnerability scores for some XSS related vulnerabilities reported on the above mentioned browsers. CVSS or the Common Vulnerability Scoring System is an open framework that provides a numerical score by taking into consideration base, temporal or environmental properties of a certain vulnerability.

The computation is performed according to the formula given in [17] and the corresponding calculator [2]. Each of the three metric groups has its own characteristics and contains a set of metrics. Base metric group (Figure 1) describes the fundamental characteristics of vulnerabilities and is composed of the related exploit range, the attack complexity, the needed authentication level and the integrity, availability and confidentiality impact. Temporal metrics (Figure 2) are the metrics influenced by time passing, meaning the availability of exploit, the remediation level and the report of confidence. The environment (Figure 3) has also an impact when computing the score; environmental factors are the collateral damage potential, the target distribution and the confidentiality, integrity and availability requirement.

When talking about XSS exploits some of these metrics remain constant because of the type of this vulnerability. The related exploit range or the access vector is the network, because the attack is widely spread over the internet and the access complexity is medium. The majority of reports describe vulnerabilities on browsers having the standard, default configurations, but it is medium and not low, because the attacker is required to have some social engineering skills in order manipulate and fool custom users to access a certain page or click a specific button or link. In order to be considered successful, the attack has to gain information or control over the client machine and for this at least one instance of authentication is needed. The availability impact metric refers to the access of the attacker over the resources, meaning bandwidth, processor, disk space and his possibility to get a total shut-down of the affected resource. In case of an XSS exploit, we consider this availability impact to be zero; it is impossible from what we know until now for someone to access the resources using this type of vulnerability. We have intentionally skipped the confidentiality and integrity impact because they vary depending on the attack and we are focusing now on the constant metrics of XSS exploits.

[2]available at http://nvd.nist.gov/cvss.cfm?calculator&adv&version=2



Base Score Metrics

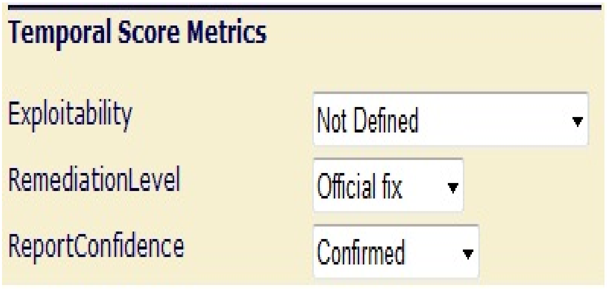
Exploitability Metrics

AccessVector	Network
AccessComplexity	Medium
Authentication	Single Instance

Impact Metrics

ConfImpact	Undefined
IntegImpact	Undefined
AvailImpact	None

FIGURE 1. Base scores metrics



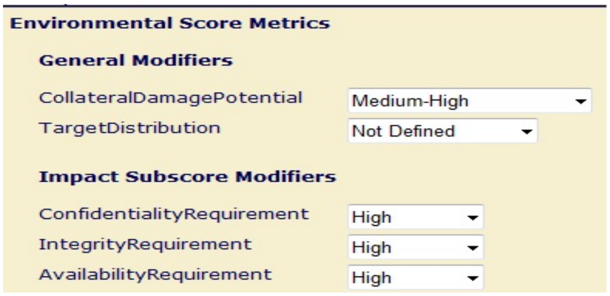
Temporal Score Metrics

Exploitability	Not Defined
RemediationLevel	Official fix
ReportConfidence	Confirmed

FIGURE 2. Temporal scores metrics

Moving to the temporal group, we will argue the chosen values to the metrics based on the vulnerabilities reported by Microsoft on their periodical security bulletins or by the other browsers in periodical advisories. We let the exploitability factor set to not defined, because officially they say that the exploitation code was not made public: ”Microsoft received information about this vulnerability through coordinated vulnerability disclosure.”, ”Microsoft had not received any information to indicate that this vulnerability had been publicly used to attack customers when this security bulletin was originally issued”, while the other browsers avoid to make this kind of statements naming only the person that reported the vulnerability. Moreover, all the vulnerabilities are confirmed and are reported only after an official fix is available.

The damage potential of an XSS vulnerability is according to OWASP moderate. We are not talking about a physical damage, but there can be significant loss of information. Last, but not least there are the impact requirement modifiers. Browsers are meant to be secure. Confidentiality, integrity and availability are the three features users require for safe browsing and financial transactions.



Environmental Score Metrics

General Modifiers

CollateralDamagePotential	Medium-High
TargetDistribution	Not Defined

Impact Subscore Modifiers

ConfidentialityRequirement	High
IntegrityRequirement	High
AvailabilityRequirement	High

FIGURE 3. Environmental scores metrics

The metrics that change depending on the XSS exploit are the confidentiality and integrity impact and the percentage of vulnerable systems. In order to see how these metrics differ we will consider four vulnerabilities.

URL Validation Vulnerability [18] is a critical vulnerability reported in February 2010 that appeared from incorrectly validated input. It provided the attacker access to the client machine with the same rights as the logged in user and if the attacker could reach administrative rights, he could install programs, read or change data. In this case, because remote code could be executed, the confidentiality and integrity impact is complete. Regarding the target distribution, which was Internet Explorer, it was reported as vulnerability on IE 7 and 8 which at that time covered 35.3 % of the market, so medium spread. The score in this case reaches 6.3.

Before the release of 10.63, Opera reported the existence of the following vulnerability: Reloads and redirects can allow spoofing and cross site scripting [23]. It allowed bypassing the same origin policy and execution of scripts in the wrong security context. One might also change configurations in the browser and gain access to the target computer. Opera, which then achieved a number of 2.1 %, described the vulnerability as critical, because the confidentiality and integrity impact was complete. Although the high impact, the flaw is rated with 2.1 because of the low spread.

ChromeHTML URI handler vulnerability is a vulnerability reported on Chrome on April 2009. It implied execution of arbitrary URI without any user interaction and could allow information leakage such as stealing of victim's cookies and directories and files disclosure, so confidentiality and integrity impact was just partial. It affected and was fixed on Chrome 1.0 targeted at that time with 4.9 % on the market. CVSS score in this case is 1.8.

March 13, 2012 brought to public's attention a vulnerability of the Content Security Policy implemented by Firefox. The vulnerability called XSS with multiple Content Security Policy headers [19] allows header injection into Firefox 11 (36.3 %). Because Firefox did not reported remote execution of code, confidentiality and integrity impact is considered partial and the score is around 5.5.

Target distribution is calculated taking into consideration data provided by http://www.w3schools.com/browsers/browsers_stats.asp, related to the date of the report.

The first remark is the importance of target distribution in calculation of score. If it slightly varies from low to medium the score increases with at least 2 points. The second observation is that the security policies adopted cannot face sophisticated attacks and can also provide security flaws.

These results can contribute to the evaluation of the business impact of XSS vulnerabilities. The browser-dependent risks must be carefully treated since they allow attackers to have end user privileges and to gain control of the applications. The computed scores confirm the OWASP evaluation and the position of cross site scripting vulnerabilities on their list [22].

6. Tool support

As you can probably imagine finding and testing security vulnerabilities is not an easy task. One has to be creative and determined in order to find all the entry points of an application and how they can bypass the validation methods added by the developer. Web Application Vulnerability Scanners are tools designed to automatically scan web applications for potential vulnerabilities. They perform a range of checks, such as field manipulation and input poisoning and enumerate the lacks found together with their target. Some of them also provide severity classification and general protection methods. Even if some claim differently, it seems that there is no security tool finding 100% of the vulnerabilities and avoiding false positive.

For testing purposes and in order to automate this process of finding web applications flaws and then try them on different browsers, we developed a security testing tool. *SecurityApp* is such an application. It is a software application designed for security vulnerabilities testing. It is a desktop solution that requires as input the URL of the desired web application and performs a set of tests in order to determine and confirm Cross-site scripting vulnerabilities. After running the tests, *SecurityApp* returns a report that displays the tests that failed, the application page, the parameter to which injection was performed and the injection string. It is an automated tool designed for testing, but it also allows manual checking. Users can add injection strings or can require testing of a specific page and parameters.

Briefly, the testing process follows the flow shown in Figure 4. The desktop application performs crawling base on the input URL. This means, it sends

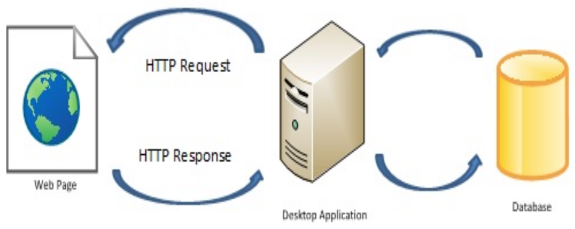


FIGURE 4. Testing process flow

several requests to the web application desired to be tested and analyzes the response by parsing the HTML code and looking for other accessible pages. After obtaining a new web page together with its GET and POST parameters, it initiates several attacks, injecting one of the parameters with strings retrieved from the database. The response is again parsed and analyzed in order to determine if it was successful or not.

Our work started by testing some of the available open-source or trials web security scanning tools in order to see the results on some web applications: ours or our colleagues and noticed that depending on the technology used the results were different, but also there was a huge difference between the performance, the vulnerability list and the initial configurations. One may be able to estimate the general capabilities of a scanner from the amount of REAL exposures that are located, the amount of exposures that are missing (false negatives) and from the amount of FALSE exposures (false positives) are identified as security exposures. The main focus is in not missing the present vulnerabilities so all the tested applications returned false positives or reported their occurrence as highly possible.

Because it addresses a certain type of vulnerability, *SecurityApp* performes an increased number of tests and tries to avoid the false positives by calculating a percentage of successful exploits for each parameter. The higher the percentage is, the more vulnerable that entry point is.

SecurityApp is a Windows forms application developed in .Net Framework 4. It performs crawling and cross-site scripting and it retrieves and stores data in a Microsoft SQL Server database. It consists of a set of modules used for different purposes: access to the database, models, user interface, crawling and XSS exploit.

Crawling is done based on the input given URL using the HTML Agility Pack library that allows reading and writing of DOM. It is an HTML parser supporting XPATH and XSLT for an easier interaction with the DOM objects.

Parsing is performed regardless the strict format of the HTML, so it is very tolerant to small errors in a malformed HTML response. Briefly, based on a string corresponding to an URL, it loads in the HtmlDocument the response received. The document is then used for manipulation. One might search for all DOM objects of some type or for all nodes having certain attributes. It may parse the children, check out for different values or add several nodes, attributes or events. XSLT is used for transforming XML documents and XPATH is used for addressing and referring parts of an XML document.

Considering these, we used HTML Agility Pack and XPATH to facilitate the parsing of the received response and collect all the referred links. We have checked for tags and forms. In case of a tag, we considered the *href* attribute and in case of forms the *action* attribute. If a form had no action attribute it meant the submit would be done to itself.

Before sending the request for the new page, we have checked that the page belonged to the domain. If for example, a web page had a link for Facebook authentication, Facebook application should not be tested. *SecurityApp* is meant for personal sites testing and not finding vulnerabilities over the internet.

Exploit operation for XSS is done using a web page and the injection scripts from the database. The application takes each parameter of the page and constructs a request for the page using an injection script and the parameters. Just one parameter is injected at a time in order to find the exact place of the page vulnerability. Parameters are sent via GET or POST and if not being exploited, they get the initial value or a default one. The strategy used is to construct scripts that can also bypass basic input sanitization:

- sanitize for apostrophe or for quotes, but not both
- avoid script tag ⟨SCRIPT⟩, but use ”javascript:” (specifier can also be used in HTML links)
- no use &, ⟨, ⟩ , # or ; characters
- no script tag ⟨SCRIPT⟩ or use of ”javascript” (successful when used in concatenations)
- encoded validation skip.

In order to confirm the vulnerability, the response page is analyzed. Each inserted script is expected to be found in the response, but in a certain place and having a certain format, if not the parameter must have been sanitized and the exploit was not successful.

The results are stored in the database for further processing in reports. The tool is in the early stages of development and have been design in order to offer an easy-to-use tool for testing the vulnerabilities for our own purposes.

Although several such tools exists, our application proved to be more efficent for our research purposes, due to its specificity.

7. CONCLUSIONS AND FUTURE WORK

The paper gives an overview of XSS vulnerabilities from a browser point of view. We studied the impact of URL Validation Vulnerability on Internet Explorer. Reloads and redirects can allow spoofing and cross site scripting on Opera, ChromeHTML URI handler vulnerability and XSS with multiple Content Security Policy headers vulnerability on Firefox. We have used the CVSS vulnerability scoring formula in order to measure the impact of these vulnerabilities on security and the obtained results confirm the OWASP analysis, for exploitability and impact.

It is our opinion that XSS vulnerabilities should be carefully treated and prevented. A combined protection approach involving writing secure code, proper testing and request validation made by the browser could eliminate most of the XSS attacks and improve significantly the security of each application.

As future direction of our study, we intend to analyze other forms of XSS vulnerabilities that are more difficult to perform and detect. We will continue testing ActiveX and Silverlight on Internet Explorer and Flash on the other browsers. We also intend to modify the default settings regarding security in each browser and observe the changes in behavior.

As part of our research we also develop a security testing tool that measures the vulnerability of an application to security attacks. This way we will automate the exploit mechanism and perform a larger number of tests.

References

[1] Acunetix, http://www.acunetix.com/
[2] J. Atwood, Coding horror, Protecting Your Cookies: HttpOnly, August 28, 2008
[3] Barracuda Networks, http://www.barracudanetworks.com/ns/products/web-site-firewall-overview.php
[4] BaydenSystems, http://www.bayden.com/tamperie/
[5] M. Brooks, Bypassing Internet Explorer's XSS Filter, Traps Of Gold-Defcon 2011, https://sitewat.ch/files/Bypassing%20Internet%20Explorer's%20XSS%20Filter.pdf
[6] CERT - Measuring Software Security Assurance - www.cert.org/research/2010research-report.pdf
[7] Cisco - ACE Web App Firewall http://www.cisco.com/en/US/products/ps9586/index.html
[8] Citrix NetScaler App Firewall http://www.citrix.com/English/ps2/products/product.asp?contentID=2312027
[9] G.A. Di Lucca , A.R. Fasolino, M. Mastoianni, P. Tramontana, Identifying cross site scripting vulnerabilities in Web applications, Proc. WSE 2004, pg. 71-80
[10] J. Grossman, S. Fogie, R. Hansen, XSS Attacks: Cross-site Scripting Exploits and Defense, Syngress, 2007

[11] M. Howard, D. LeBlanc, Writing Secure Code, Microsoft Press, 2003
[12] HP WebInspect, https://download.hpsmartupdate.com/webinspect/
[13] IBM Software, Reduce your cost of quality, http://www-01.ibm.com/software/rational/smb/quality/
[14] A. Klein, DOM Based Cross Site Scripting or XSS of the Third Kind. Web Application Security Consortium Articles, 4, 2005, http://www.webappsec.org/projects/articles/071105.shtml
[15] D. Lynch, XSS is fun!, October 20, 2011 http://davidlynch.org/blog/2011/10/xss-is-fun/
[16] Mavituna Security, http://www.mavitunasecurity.com/netsparker/
[17] P. Mell, K. Scarfone, S. Romanosky - A Complete Guide to the Common Vulnerability Scoring System Version 2.0, 2007, http://www.first.org/cvss/cvss-guide.pdf
[18] Microsoft - http://technet.microsoft.com/en-us/security/bulletin/MS10-002
[19] Mozilla Foundation Security Advisory, http://www.mozilla.org/security/announce/2012/mfsa2012-13.html
[20] MSDN - Security, Anti-Cross Site Scripting Library, http://msdn.microsoft.com/en-us/security/aa973814
[21] NASA - Open Source Summit 2011, http://www.nasa.gov/open/source/
[22] Open Web Application Security Project www.owasp.org
[23] Opera Support, http://www.opera.com/support/kb/view/973/
[24] PortSwingger Web Security, http://portswigger.net/burp/
[25] Social Hacking, Recent Facebook XSS Atacks Show Incresing Sophistication, April 21, 2011
[26] G. Wassermann, Static detection of cross-site scripting vulnerabilities, Proc. of ICSE 2008, pg.171-180

Babeş Bolyai University, Department of Computer Science, M. Kogălniceanu 1, 400084 Cluj-Napoca, Romania

E-mail address: `naie1000@scs.ubbcluj.ro, motogna@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

PHONEMES VERSUS GEOMETRIC PROPERTIES IN CLUSTERING OF POEMS

MIHAIELA LUPEA AND DOINA TĂTAR

Abstract. This paper discusses the comparison between two kinds of features in clustering of some literary poems by the same author, the Romanian poet, Mihai Eminescu. Using *Precision*, *Recall*, *Rand Index*, *Relative Precision* and *Purity* measures we conclude that the topics of poems are better characterized by the phonemes as features than by geometric properties (described by six indicators: $V/N; A; \Lambda; Var(\Lambda), Gini; Var(Gini)$) of the rank-frequency sequence of word forms.

1. Introduction

This paper discusses the comparison between two kinds of features in clustering of some literary poems by the same author, the Romanian poet Mihai Eminescu. The *Gold Standard* (GS) of evaluation is a manual one, which divides the set of the longest 45 Eminescu's poems (see Appendix A for correspondence numbers - titles) into five big clusters, topic (content) focused:

- Love-general stories (tales): {10; 48; 51; 62; 64; 87; 93; 104; 109; 110; 117; 120; 129};
- Love-personal stories: {8; 9; 13; 21; 38; 57; 68; 69; 94; 100; 143};
- Philosophy-tales wisdom: {25; 34; 52; 58; 61; 70; 90; 106; 126; 127; 130};
- Nature: {28; 91};
- History-patriotism: {6; 47; 49; 50; 74; 95; 123; 128}.

For clustering the poems, these are represented using the space vector method and the Algorithm of Agglomerative Hierarchical Clustering ([2, 4]). The complete-link similarity between two clusters and the cosine similarity measure between two vectors are applied.

Received by the editors: 20.11.2012.

2010 *Mathematics Subject Classification.* 68T50.

1998 *CR Categories and Descriptors.* I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Text analysis*.

Key words and phrases. clustering, Rand Index, purity, phonemes.

	same cluster *Meth*	different cluster *Meth*
same cluster *GS*	A	C
different cluster *GS*	B	D
$P_{Meth} = \frac{A}{A+B}$ $R_{Meth} = \frac{A}{A+C}$	$A + B = \sum_{i=1}^{K} C^2_{\|w_i\|}$ $A + C = \sum_{i=1}^{K} C^2_{\|w_i^{GS}\|}$	$D = C^2_{\|M\|} - (A + B + C)$

TABLE 1. *Precision* and *Recall* measures

For the first clustering we considered as features the phonemes, building phonemes vectors corresponding to the phonetic transcription of the poems. Such a vector for a poem has 31 components containing the relative frequencies of the Romanian phonemes in that poem, as it is presented in Section 3.

The features used in the second clustering are geometric properties of the rank-frequency sequence of word forms in poems, expressed by vectors containing six indicators: V/N; A; Λ ; Var(Λ); *Gini*; Var(*Gini*)), introduced in [1] and described in Section 3.

The first method of evaluation of the clusterings is by establishing classical *Precision* and *Recall* measures, as reported to the Gold Standard (GS) clustering. The second method is *Rand Index* ([2]) and a *Relative Precision* as inspired from *Rand Index* algorithm (Section 2.2). The third method is the calculus of *Purity* ([2, 3]), Section 2.3.

In all these cases (excepting *Rand Index*) the conclusion is that the best indicators are the phonemes. The reason for these results seems to be the fact the most indicators (introduced in [1]) are based on words, and the words consists of phonemes. So, the phonemes unify and refine the words function. However, in this paper we worked with only a part of the indicators introduced in [1].

2. EVALUATION OF CLUSTERING

2.1. **Precision and Recall.** Let M be a set of elements. Two clustering methods are applied to M obtaining the same number K of clusters:

- an arbitrary method *Meth*, providing the clusters: $w_1, \ldots, w_K$;
- a manual method, providing the gold standard *GS* clustering: $w_1^{GS}, \ldots, w_K^{GS}$.

For a comparison of these two clusterings Table 1 is built. In the table we use C_T^2 to denote the binomial coefficient indexed by T and 2.

	same cluster *Meth*	different cluster *Meth*
same cluster *GS*	A	C
different cluster *GS*	B	D
$RI = \frac{A+D}{A+B+C+D}$	$A+B = \sum_{i=1}^{K} C_{\lvert w_i \rvert}^{2}$ $A+C = \sum_{j=1}^{J} C_{\lvert C_j \rvert}^{2}$	$D = C_{\lvert M \rvert}^{2} - (A+B+C)$

TABLE 2. *Rand Index* measure

The value of A represents the number of pairs of elements from M with the property: if a pair belongs to the same cluster obtained with *Meth*, it belongs also to the same cluster of *GS*.

The significance and the values of *B, C, D* are defined in an analogous way, deductible from the positions in Table 1. $A+B$ represents the number of pairs of elements situated in the same cluster obtained using *Meth*, and $A+C$ represents the number of pairs of elements situated in the same cluster of *GS*.

Precision: $P_{Meth} = \frac{A}{A+B}$ counts how many of the determined cases by *Meth* are correct.

Recall: $R_{Meth} = \frac{A}{A+C}$ counts how many of the correct cases are determined by *Meth*.

2.2. **Rand Index.** The set M is partitioned by some objective observations in J classes: $C_1, \ldots, C_J$. An arbitrary clustering method *Meth* is applied to M obtaining K clusters: $w_1, \ldots, w_K$.

The measure **Rand Index** (*RI*) penalizes both the False positive pairs (B) and the False negative pairs (C) according to Table 2.

Using *Rand Index* measure, we could obtain a method of a direct comparison of two clusterings *R1* and *R2* with the same number of clusters.

Rand Index of clustering R1 relative to R2, denoted by $RI_{R1,R2}$ expresses how good the clustering *R2* is, when a cluster (of *R2*) is considered a class: a cluster is calculated by a more or less good method, a class is judged by some objective reasons, thus a partition in classes is more exact than a partition in clusters. A similar significance has $RI_{R2,R1}$, expressing the quality of clustering *R1*.

$RI_{R1,R2} \leq RI_{R2,R1}$ means a better quality of the clustering *R1* than of the clustering *R2* (with *RI* measure), when the same similarity measures of clustering are used in *R1* and *R2*.

The method could be applied also for the case of *Precision*, namely, *Relative Precisions*: $P_{R1,R2}$ and $P_{R2,R1}$ could be calculated. $P_{R1,R2} \leq P_{R2,R1}$ means a better quality of the clustering *R1* than that of the clustering *R2*.

2.3. **Purity.** Let us suppose that we have K clusters: $w_1, \ldots, w_K$ and J classes: $C_1, \ldots, C_J$ for a set M of elements. The purity of a cluster w_k is calculated as:

$$Purity(w_k) = max_j\{n_{kj}\}/|w_k|$$

where $n_{kj} = |w_k \cap C_j|$.

The index $j^*_k = \text{argmax}_j\ n_{kj}$ determines the majority class of the cluster w_k denoted by $C_{j^*_k}$.

The $Purity(w_k)$ is the number of elements provided by the majority class of the cluster w_k over the cardinal of the cluster. The higher the contribution of the majority class, the higher the purity of a cluster.

The *Purity* of a clustering is the weighted sum of the purities of all clusters:

$$Purity = \sum_{k=1}^{K} Purity(w_k) \times \text{weight}(w_k)$$

where $weight(w_k) = |w_k|/|M|$.

3. A case study - clustering of Eminescu's poems

In this section we apply the theory from the previous section using as M the set of the 45 longest poems of Eminescu (Appendix A). The poems are represented using the vector space method, where the vectors are:

(1) numeric vectors of 31 components containing the relative frequencies of the Romanian phonemes in the poem, describing the content of the poem in a phonetic manner. The phonemes correspond to the vowels (in number of 7), consonants (in number of 18) and 6 groups of letters('ce', 'ci', 'ge', 'gi', 'ch', 'gh'). The letter 'x' is decomposed in two phonemes [c]+[s].

For example, the statistics for the poem *Memento mori* (90) are:
- total phonemes number: 46433;
- vowels number: 21494;
- consonants number: 24939 (including the groups of letters);
- the vector of occurrences for all 31 phonemes(in this order: vowels, consonants, the groups of letters) is:
(3995, 5035, 4593, 1852, 3024, 1767, 1228, 512, 1614, 1827, 501, 385, 39, 70, 2491, 1431, 3370, 1341, 4072, 1831, 681, 2433, 362, 563, 402, 428, 262, 95, 103, 117, 9).

For the phoneme 'a', with 3995 occurences in the poem, its relative frequency in the category of vowels is computed as: 3995/21494 = **0.1859**.

The relative frequency in the category of consonants for the phoneme 'b', with 512 occurences in the poem is computed as: 512/24939 = **0.0205**.

The vector of phonemes for *Memento mori* is:
(**0.1859**, 0.2343, 0.2137, 0.0862, 0.1407, 0.0822, 0.0571, **0.0205**, 0.0647, 0.0733, 0.0201, 0.0154, 0.0016, 0.0028, 0.0999, 0.0574, 0.1351, 0.0538, 0.1633, 0.0734, 0.0273, 0.0976, 0.0145, 0.0226, 0.0161, 0.0172, 0.0105, 0.0038, 0.0041, 0.0047, 4.0E-4).

(2) numeric vectors of six components corresponding to some indicators: V/N; A; Λ ; Var(Λ); *Gini*; Var(*Gini*), which describe geometric properties of the rank-frequency sequence of word forms in poems ([1]).

The significance of the indicators is the following: V is the vocabulary size (words) of the text, N is the text length (the total number of words in the text), A (adjusted modulus) is an index of vocabulary richness.

As regarding Λ indicator, this is introduced as a normalization of L, the length of the arc beginning at $f(1)$ and ending at $f(V)$, $\Lambda = L/N * (Log(N)))$. *Gini*'s coefficient is connected with the cumulative relative frequencies which form an arc running from (0,0) and touching the bisector in (1,1). The magnitude of the area between the bisector and this arc yields *Gini*'s coefficient. The expressions for Var(Λ) and Var(*Gini*) are also introduced first time in ([1]).

For example, the vector for the poem *Memento mori* (90) is:
(0.365906068, 0.9311, 1.6175, 0.000068, 0.5717, 0.000033).

To obtain five clusters (like in *Gold Standard*) of Eminescu's poems we used the Algorithm of Agglomerative Hierarchical Clustering ([2, 4]) (or bottom-up clustering algorithm), the complete-link similarity between two clusters and the cosine similarity measure between two vectors.

In the bottom-up clustering algorithm we begin with a separate cluster for each poem and we continue by grouping the most similar clusters until we obtain a specific number of clusters (here five clusters).

For the cosine similarity measure between the vectors $V_1 = (a_1, a_2, ..., a_n)$ and $V_2 = (b_1, b_2, ..., b_n)$ the well known formula is used:

$$sim(V_1, V_2) = cos(V_1, V_2) = \frac{\sum_{i=1}^{i=n} a_i * b_i}{\sqrt{\sum_{i=1}^{i=n} a_i^2} \times \sqrt{\sum_{i=1}^{i=n} b_i^2}}$$

The complete-link similarity between two clusters $C1$ and $C2$ represents the similarity of two least similar members of the two clusters:

Precision	*Recall*	*Rand Index*	*Relative Precision*
$P_{R1} = 0.2997$	$R_{R1} = 0.5622$	$RI_{R1} = 0.6161$	$P_{R1,R2} = 0.2088$
$P_{R2} = 0.2392$	$R_{R2} = 0.2811$	$RI_{R2} = 0.6383$	$P_{R2,R1} = 0.3231$

TABLE 3. Measures for *R1* and *R2* clusterings

$$sim(C1, C2) = min\{sim(V_i, V_j)|V_i \in C1 \ and \ V_j \in C2\}.$$

The clustering $R1$ corresponds to the representation (1):

- w_1^{R1}:{8(2); 9(2); 21(2); 38(2); 57(2); 68(2); 69(2); 70(3); 94(2); 123(5)};
- w_2^{R1}:{6(5); 10(1); 13(2); 25(3); 34(3); 47(5); 48(1); 49(5); 50(5); 51(1); 52(3); 61(1); 62(1); 64(1); 74(5); 87(1); 90(3); 95(5); 109(1); 110(1); 117(1);126(3); 127(3); 128(5); 129(3); 130(2); 143(2)};
- w_3^{R1}:{28(4); 91(4); 93(1); 104(1); 120(1)};
- w_4^{R1}:{58(3); 100(2)};
- w_5^{R1}:{106(3)}.

The clustering $R2$ corresponds to the representation (2):

- w_1^{R2}: {52(3); 90(3)};
- w_2^{R2}: {6(5); 10(1); 87(1); 95(5); 109(1); 128(5); 130(3)};
- w_3^{R2}: {34(3); 58(3); 61(3); 91(4); 126(3); 129(1)};
- w_4^{R2}: {9(2); 13(2); 21(2); 25(3); 28(4); 48(1); 49(5); 50(5); 62(1); 64(1); 69(2); 93(1); 94(2); 104(1); 106(3); 110(1); 117(1); 127(3); 143(2)};
- w_5^{R2}:{8(2); 38(2); 47(5); 51(1); 57(2); 68(2); 70(3); 74(5); 100(2); 120(1); 123(5)}.

The numbers in brackets represent the manual assignation for the poems of one of the five clusters corresponding to *Gold Standard* clustering (see Introduction).

$R1$ and $R2$ are compared applying the measures of *Precision, Recall, Rand Index, Relative Precision*, and *Purity* and the results are reported in Table 3.

For computing the purities of *R1* and *R2* we consider that *GS* clustering represents the set of predefined classes. Table 4 contains the values of purities for all clusters of *R1* and $R2$, and also the overall purities for these clusterings.

The overall *Purity* for $R1$, $Purity_{R1} = 0.8 \times 0.22 + 0.37 \times 0.6 + 0.6 \times 0.11 + 0.5 \times 0.04 + 1 \times 0.02 = 0.504$.

clusters of *R1*	w_1^{R1}	w_2^{R1}	w_3^{R1}	w_4^{R1}	w_5^{R1}	
Purity	0.8	0.37	0.6	0.5	1	$Purity_{R1} = 0.504$

clusters of *R2*	w_1^{R2}	w_2^{R2}	w_3^{R2}	w_4^{R2}	w_5^{R2}	
Purity	1	0.42	0.66	0.36	0.45	$Purity_{R2} = 0.438$

TABLE 4. *Purity* measure for *R1* and *R2* clusterings

The overall *Purity* for $R2$, $Purity_{R2} = 1 \times 0.04 + 0.42 \times 0.15 + 0.66 \times 0.13 + 0.36 \times 0.42 + 0.45 \times 0.24 = 0.438$.

From Table 3 and Table 4 we can conclude:

(1) Both *Precision* and *Recall* are better in the case of *R1* clustering than in the case of *R2* clustering: $P_{R1} \geq P_{R2}$ and $R_{R1} \geq R_{R2}$.
(2) According to *Rand Index* measure the results are better for *R2* than for *R1*: $RI_{R1} \leq RI_{R2}$.
(3) In a direct comparison of *R1* and *R2* clusterings using *Relative Precision*, the quality of *R1* is better than that of *R2*: $P_{R1,R2} \leq P_{R2,R1}$.
(4) As $Purity_{R1} \geq Purity_{R2}$, we can say again that the clustering *R1* is of a better quality than $R2$.

4. CONCLUSIONS

In this paper *Precision, Recall, Rand Index, Relative Precision* and *Purity* evaluation measures are used to compare the impact of different features of poems in the topic-focused clustering of the 45 longest Eminescu's poems. Excepting *Rand Index*, all the other measures suggest that the phonemes as features characterize better the topic (content) of the poems than geometric properties (described by the indicators: $V/N; A; \Lambda; Var(\Lambda), Gini; Var(Gini)$) of the rank-frequency sequence of word forms.

REFERENCES

[1] Popescu, I.I., Čech, R., Altmann, G.: "The Lambda-structure of Texts"'. Studies in quantitative linguistics 10, RAM-Verlag, 2011.
[2] Manning, C., Raghavan, P., Schutze, H.: "Introduction to Information Retrieval", Cambridge University Press, 2008.
[3] Mihalcea, R., Radev, D.: "Graph-based Natural language Processing and Infromation Retrieval", Cambridge University Press, 2011.
[4] Tatar, D., Serban, G.: "Word clustering in QA systems", Studia Universitatis Babes-Bolyai, Seria Informatica 2003, 1, pp. 23–33.

Appendix A. The correspondence between numbers and titles of poems

No.	*Poem*	*No.*	*Poem*
(6)	Andrei Mureşanu	(8)	Aveam o muză
(9)	Basmul ce i l-aş. . .	(10)	Călin
(13)	Când crivăţul cu iarna...	(21)	Copii eram noi amândoi
(25)	Cugetările sărmanului. . .	(28)	Dacă treci râul Selenei
(34)	Demonism	(38)	Despărţire
(47)	Dumnezeu şi om	(48)	Eco
(49)	Egipetul	(50)	Epigonii
(51)	Făt-Frumos din tei	(52)	Feciorul de impărat fără. . .
(57)	Ghazel	(58)	Glossa
(61)	Impărat şi proletar	(62)	In căutarea Şeherezadei
(64)	Inger şi demon	(68)	Iubită dulce, o, mă lasă
(69)	Iubitei	(70)	Junii corupti
(74)	La moartea lui Heliade	(87)	Luceafărul
(90)	Memento mori	(91)	Miradoniz
(93)	Mitologicale	(94)	Mortua est!
(95)	Mureşanu	(100)	Nu mă-nţelegi
(104)	O călărire în zori	(106)	O, adevăr sublime...
(109)	Odin şi poetul	(110)	Ondina (Fantazie)
(117)	Povestea teiului	(120)	Pustnicul
(123)	Rugăciunea unui dac	(126)	Scrisoarea I
(127)	Scrisoarea II	(128)	Scrisoarea III
(129)	Scrisoarea IV	(130)	Scrisoarea V
(143)	Venere şi Madonă		

Babeş-Bolyai University, Department of Computer Science, M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: `lupea,dtatar@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

A STUDY ON USING REINFORCEMENT LEARNING FOR TEMPORAL ORDERING OF BIOLOGICAL SAMPLES

IULIANA M. BOCICOR

Abstract. The temporal ordering of biological samples, with the goal of retrieving the temporal evolution of dynamic biological processes, is an important problem within bioinformatics. As the general temporal ordering problem has been proven to be NP-complete, various approximation and heuristic methods are developed to approach it. Reinforcement Learning is an approach to machine intelligence in which an adaptive system can learn to behave in a certain way by receiving punishments or rewards for its chosen actions. This paper aims to investigate a reinforcement learning based approach to the temporal ordering problem and several variations to this approach, based on Q-Learning. The algorithms are experimentally evaluated on a time series gene expression data set and we provide analysis and comparisons of the obtained results.

1. Introduction

Reinforcement Learning [12] is an approach to machine intelligence in which an agent [11] can learn to behave in a certain way by receiving punishments or rewards for its chosen actions. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the highest reward by trying them. The reinforcement learning algorithms selectively retain the outputs that maximize the received reward over time.

The biological *temporal ordering (TO) problem* is formulated as the problem of constructing a sorted collection of multi-dimensional biological data, collection that reflects an accurate temporal evolution of a certain biological process. The final goal is to find certain patterns in the input data that

Received by the editors: December 1, 2012.

2010 *Mathematics Subject Classification.* 68P15, 68T05.

1998 *CR Categories and Descriptors.* I.2.6[**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.2.8[**Computing Methodologies**]: Problem Solving, Control Methods, and Search – *Heuristic methods*.

Key words and phrases. Bioinformatics, Temporal Ordering, Reinforcement Learning, Q-Learning.

vary over time and use them efficiently in order to be able to offer a proper characterization of the process in question.

In this paper we aim to investigate several variations to a reinforcement learning based approach for the TO problem, approach that we have previously introduced in [3]. The evaluations are made on a data set that was used in [3] and comparisons and analysis will be provided.

The rest of the paper is organized as follows. Section 2 introduces the biological TO problem, as well as an existing reinforcement learning based approach. A series of variations to this approach, more specifically to the underlying algorithm are presented in Section 3. Experimental evaluations, analysis and comparisons of the all the algorithms are given in Section 4. Section 5 outlines our conclusions and further work.

2. Background

In this section we will briefly present the TO problem, in a bioinformatics framework, then review some fundamental aspects related to the reinforcement learning based approach that we previously introduced for solving this problem [3].

2.1. **The Temporal Ordering Problem.**

The general *temporal ordering* problem has been tackled within multiple fields. In machine learning it is considered as important as the classification problem, given that, in certain cases, ordering a set of instances can provide more significant information than classifying them. The TO problem has been proven to be NP-complete [2], therefore various approximation and heuristic methods could be used to approach it.

Within the bioinformatics and computational biology framework, the TO problem refers to constructing a sorted collection of multi-dimensional biological data, collection that reflects an accurate temporal evolution of a certain biological process. A large part of the existing data is static, but biological processes are mostly dynamic. In order to be able to analyze and characterize these processes, scientists need dynamic information and one way to obtain this from static data is by inferring temporal orderings to this data. The TO problem is important within bioinformatics, as there are many practical applications for it, one of the most significant being in the field of cancer research, as cancer is inherently a dynamic disease.

2.2. **Reinforcement Learning based approach for the TO problem.**

Reinforcement Learning (RL) [7] is an approach to machine intelligence that combines two disciplines to solve successfully problems that neither discipline can address individually: *Dynamic programming* and *Supervised learning.* RL is a synonym of learning by interaction [9]. During learning, the adaptive system tries some actions (i.e., output values) on its environment, then it is reinforced by receiving a scalar evaluation (the reward) of its actions. The reinforcement learning algorithms selectively retain the outputs that maximize the received reward over time. In RL, the computer is simply given a goal to achieve and it learns how to achieve that goal by trial-and-error interactions with its environment.

In [3] we introduced a reinforcement learning based technique for identifying a temporal ordering of a series of multi-dimensional biological samples. Even though in the above mentioned work we refer strictly to gene expression data obtained from microarray experiments, the applicability of our method is more general and it can be used with different types of multi-dimensional biological data.

From a computational point of view, the TO problem was defined as the problem of generating a permutation that maximizes the overall similarity of the sequence of samples considered in the ordering [3]. The RL task associated to the TO problem consists in training the agent to find a path from the initial to a final state having the maximum associated overall similarity. During the training step of the learning process the learning agent determines its *optimal policy* in the environment, i.e. the mapping from states to actions that maximizes the sum of the received rewards. The equivalent *action configuration* is viewed as a permutation that gives the temporal ordering for the input samples. For training the TO agent [3] a Q-learning approach was used [12] and a new action selection mechanism was defined in order to guide the exploration of the search space [3]. After the training step of the agent has been completed, the solution learned by the agent, which indicates the recovered temporal ordering, is constructed starting from the initial state and following the *Greedy* mechanism. For more details about how the data was pre-processed, about the definitions of the state and action spaces, reward and transition functions or about the action selection mechanism, we refer the reader to [3].

3. Variations of the RL based approach

This section aims to present several variations we propose for the RL based approach introduced in [3], used to solve the biological TO problem. In [3] we introduced an action selection mechanism based on the ϵ-Greedy mechanism [12], which uses a look-ahead procedure, in order to better guide the learning agent through the search space. To investigate how the policy specifying the way in which a new action is chosen in each given state influences the accuracy of the recovered ordering, we firstly try a different action selection policy: the *softmax policy.* A second idea refers to using a different RL approach, one that combines Q-learning with an essential mechanism of RL: *eligibility traces.*

3.1. **The Softmax Action Selection Policy.** One key aspect of reinforcement learning is a trade-off between *exploitation* and *exploration* [13]. To accumulate a lot of reward the learning system must prefer the best experienced actions, however, it has to try (to experience) new actions in order to discover better action selection mechanisms for the future.

Several rules (policies) for choosing actions in order to make transitions among states during the learning process exist in the literature. The *greedy* policy implies that the learning agent chooses the highest-valued action in each state. An agent using this mechanism only exploits current knowledge to maximize its reward, but does not explore new states that could lead to higher long term rewards. A more effective method, which balances the exploration of new states with exploitation of current knowledge, is ϵ-Greedy [12]. It selects the greedy action with probability $1 - \epsilon$ and, in order to explore the environment, with probability ϵ it chooses an action at random, uniformly, not taking into consideration the action value estimates. Therefore, the main drawback of ϵ-Greedy is that the worst action is as likely to be chosen as the second best one.

A way to counter this disadvantage is to use a policy that chooses better actions more often. This is achieved by the *softmax* action selection policy [12], in which actions are ranked according to their value estimates and each action is chosen with a probability computed using its value. The greedy action will still have the highest probability. The most common softmax method uses a Gibbs, or Boltzmann distribution, where the probability of choosing action a in state s is (for a Q-learning approach):

$$\frac{e^{Q(s,a)/\tau}}{\sum_a e^{Q(s,a)/\tau}} \tag{1}$$

where τ is a positive parameter called *temperature*, which specifies how random actions should be chosen. For high values of the temperature all actions will be almost equiprobable. As the temperature is reduced, the actions that have higher value estimates are more likely to be selected and in the limit, as $\tau \to 0$, the best action is always chosen, this meaning that the softmax policy becomes the same as the greedy policy.

3.2. **Q-learning with eligibility traces.** Eligibility traces were firstly introduced in [5] and they are a basic mechanism used in RL for handling delay [10]. The idea is that each time a state is visited it is marked by a trace, which then gradually decays over time, exponentially, according to a decay parameter λ ($0 \leq \lambda \leq 1$) and to the discount rate parameter γ. The trace makes the state *eligible* for learning [10].

There are two types of possible implementations for eligibility traces:

- *Accumulating eligibility traces* - the trace increases each time a state is visited. States that are visited more recently and more often are assigned more credit. For a Q-learning approach, the accumulating trace is defined in the following way [10]:

$$e_{t+1}(s,a) = \begin{cases} \gamma\lambda e_t(s,a) + 1, & \textit{if} \;\; s = s_t \textit{ and } a = a_t \\ \gamma\lambda e_t(s,a), & \textit{otherwise} \end{cases} \tag{2}$$

  for all state-action pairs (s,a). Here $e_t(s,a)$ represents the eligibility trace of the state-action pair (s,a) at time t, s_t is the actual state and a_t the actual selected action at time t.

- *Replacing eligibility traces* - each time a state is visited its trace is reset to 1, disregarding the previous trace information. For a Q-learning approach the replacing trace for a state-action pair is [10]:

$$e_{t+1}(s,a) = \begin{cases} 1, & \textit{if} \;\; s = s_t \textit{ and } a = a_t \\ 0, & \textit{if} \;\; s = s_t \textit{ and } a \neq a_t \\ \gamma\lambda e_t(s,a), & \textit{otherwise} \end{cases} \tag{3}$$

  for all state-action pairs (s,a).

There are two approaches that combine Q-learning with eligibility traces: Watkins's $Q(\lambda)$ [14] and Peng's $Q(\lambda)$ [8]. As in this study we use only the former, we will briefly describe it in the following. In Q-learning the agent learns about the greedy policy, but usually, during training, it follows an exploratory policy (e.g. ϵ-greedy). Therefore, in learning about the greedy policy, the eligibility trace information can be used only as long as the greedy policy is followed. This means that eligibility traces are updated using Formula 2 or 3 (depending on the case) for the greedy actions, but the moment an exploratory action is taken the eligibility trace for the respective state-action pair is set to 0. The algorithm is given in Figure 1. We denote in the following by $Q(s,a)$ and $e(s,a)$ the Q-value estimate, respectively the eligibility trace value associated to the state s and action a, by α the learning rate, by γ the discount factor and by λ the decay parameter.

```
Repeat (for each episode)
  Select the initial state $s$ of the agent (as $s_1$).
  Choose action $a$ from $s$ using the given action selection mechanism.
  Repeat (for each step of the episode)
```

```
    Take action $a$, observe the reward $r(s,a)$ and the next state $s'$.
    Choose action $a'$ from $s'$ using the given action selection mechanism.
    $a^* \leftarrow argmax_b Q(s',b)$
    $\delta \leftarrow r(s,a) + \gamma \cdot Q(s',a^*) - Q(s,a)$
    Update $e(s,a)$ //$e(s,a) \leftarrow e(s,a) + 1$ or $e(s,a) \leftarrow 1$
    For all s,a:
      $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \delta \cdot e(s,a)$
      If $a' = a^*$
        $e(s,a) \leftarrow \gamma \cdot \lambda \cdot e(s,a)$
      else
        $e(s,a) \leftarrow 0$
    s ← s'
  until $s$ is terminal
Until the maximum number of episodes is reached or the $Q$-values do not change
```

FIGURE 1. Watkins's $Q(\lambda)$ algorithm [14].

Another possible implementation of $Q(\lambda)$, called *naive* $Q(\lambda)$ [12], would be the same as Watkins's algorithm, except that for an exploratory action the eligibility traces are not set to 0.

4. EXPERIMENTS

In this section we provide experimental evaluations of the algorithms described in Section 3. Several tests were made, for each Q-learning algorithm (*traditional Q-learning*, $Q(\lambda)$ and *naive* $Q(\lambda)$), using each type of eligibility trace (*accumulating* and *replacing*) and two different action selection policies (*one step look-ahead procedure* [3] and *softmax* action selection policy).

For the experiments we used a software framework that we have previously introduced for solving combinatorial optimization problems using reinforcement learning techniques [4].

4.1. **Case study.** The data set we used to test the performance of the different Q-learning based algorithms is a time series composed of gene expression data measuring the levels of expression of almost every yeast gene, at eight different time points, as yeast cells were affected by a given type of environmental change: dithiothrietol (DTT) exposure [6]. A time series is a collection of data resulted from a specific type of biological experiment: samples of tissues are extracted from the same individual at different and known moments in time, during the progression of the biological process. Thus, for a time series data set, the exact time of each sample is provided and the ordering is known. This data set was also used in [3] along with several different time series experiments for yeast or human cells and cancer gene expression data. As described in [3], in order to reduce the dimensionality of the input data (which is, usually, huge in the case of microarray experiments), we firstly pre-process the data by applying a statistical analysis, the final goal being the selection of those features (genes) that are most important for an accurate temporal ordering.

The algorithms are compared by examining the accuracy of the recovered orderings, by the number of epochs they need to achieve convergence and by the computational time. In [3] we introduced an evaluation measure, called *Samples Misplacement Degree (SMD)*, which, in our view, asseses the quality of a solution (ordering). We mention that smaller values for the SMD (smaller numbers of misplaced samples) indicate better orderings, the correct ordering having $SMD = 0$. Regarding the parameter setting, we remark that for all types of tests we used the following values: the discount factor for the future rewards is $\gamma = 0.95$; the learning rate is $\alpha = 0.8$; the number of training episodes is $7 \cdot 10^5$; for the tests using eligibility traces the decay parameter is $\lambda = 0.95$. For each of the two action selection policies, tests were made for different values of policy parameter (ϵ - in the case of the one step look-ahead procedure and τ - in the case of softmax): $\{0.1, 0.2, 0.5, 0.8, 0.9\}$. We mention that the experiments were carried out on a PC with an Intel Core i5-2400 Processor at 3.1 GHz (4 CPUs) with 8 GB of RAM.

4.2. **Comparative results.** In the following, we present the results obtained by each type Q-learning algorithm.

The *traditional Q-learning algorithm*, with no eligibility traces, proves a very good performance with both action selection policies. In the case of the *ϵ-Greedy based look-ahead procedure* [3], the optimal solution (the correct ordering $1, 2, 3, 4, 5, 6, 7, 8$) is obtained within very short amounts of time - less than 2 seconds, for all values of ϵ. During the first few epochs of the training process the algorithm obtains various orderings, depending on the value of the parameter ϵ, but it converges very soon to the optimal ordering, in less than 3000 training epochs, on average. This is illustrated in the first image of Figure 2, which depicts the overall similarity of the solutions obtained during the training process. We mention that the overall similarity of 97.051 corresponds to the correct ordering.

The *softmax action selection policy* also leads to the correct ordering, for all values of the temperature parameter, except for $\tau = 0.1$. In this case, the algorithm converges to a different ordering of the samples: $S = 8, 1, 2, 3, 4, 5, 6, 7$, having $SMD(S) = 3$. Still, we observe that the algorithm succesfully recovers

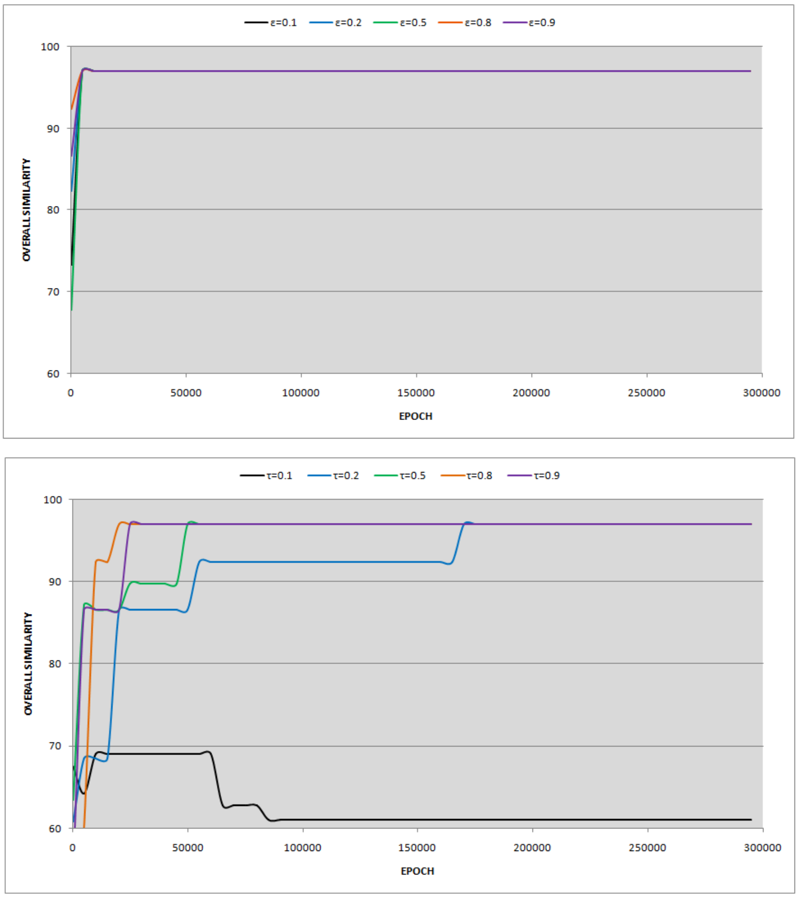


Figure 2. Q-Learning: the learning process.

the order of a subset of 7 samples, out of the set of 8. Another observation is that for the softmax policy the convergence is slower than with the other used policy, but as the temperature parameter increases the number of epochs needed to reach the solution decreases. This is illustrated in the second image of the Figure 2: for $\tau = 0.2$ the convergence is achieved after 170000 epochs, while for $\tau = 0.9$ only 20000 epochs are necessary. As for the computational time, we remark that even for smaller values of τ the solution is retrieved in less than 1 minute.

As soon as eligibility traces are introduced, the behaviour of the Q-learning algorithm changes radically: for certain values of ϵ or τ it does not converge at

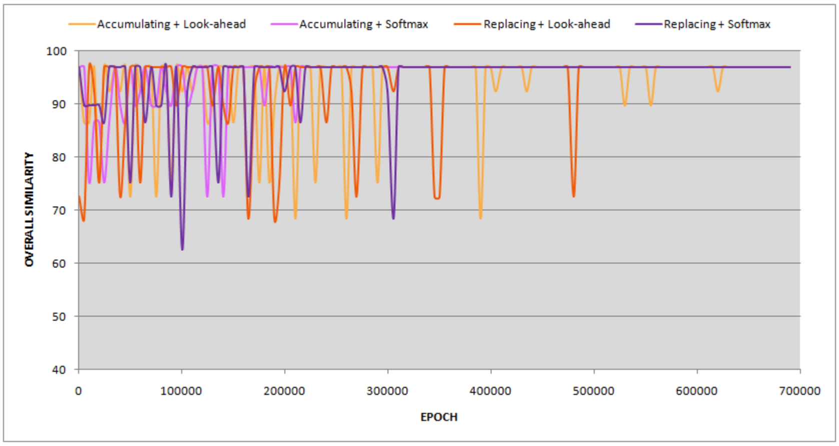


FIGURE 3. $Q(\lambda)$: the learning process for $\epsilon = 0.8$ and $\tau = 0.8$.

all, while for other values it retrieves the correct ordering, but within greater amounts of time.

Watkins's $Q(\lambda)$ [14] performs somewhat similarly for the two types of eligibility traces. In this case, the softmax policy achieves better convergence than the alternative. $Q(\lambda)$ in conjunction with *softmax* has the following behaviour: for $\tau = 0.1$ the algorithm converges to different orderings than the correct one, which have, however, values less than 4 of the SMD measure. As the temperature increases over 0.8 the algorithm slowly converges to the correct ordering, after 250000 training epochs. When used with the *look-ahead action selection procedure*, for small values of ϵ, it does not achieve convergence. But, for $\epsilon \geq 0.8$ it begins to converge to the correct solution after 250000 episodes, equivalently 140 seconds. Still, in some rare cases, there are individual epochs when it slightly deviates from the solution, but it soon returns to the maximum overall similarity ordering. These are illustrated in Figure 3, which shows the overall similarity of the solutions obtained during the training process, for both action selection strategies (using $\epsilon = 0.8$ and $\tau = 0.8$) and both types of traces. It can be observed that the algorithms using the softmax policy (represented in 2 different shades of purple) completely converge, while the ones using the look-ahead procedure (represented in light and dark orange) slighlty deviate from the solution once in a while.

In what concerns the *naive* $Q(\lambda)$, when run using *accumulating* eligibility traces and the *look-ahead policy* it does not converge for lower values of ϵ, but for $\epsilon \geq 0.8$ it soon converges to the correct solution, after 15000 episodes, in less than 3 seconds. In this case, the *softmax* selection mechanism leads to

	Accumulating traces		**Replacing traces**	
	Look-ahead	**Softmax**	**Look-ahead**	**Softmax**
$Q(\lambda)$	$\epsilon < 0.8 \Rightarrow$ div. $\epsilon \geq 0.8 \Rightarrow S$ ($>$ 250000 epochs)	$\tau = 0.1 \Rightarrow S'$ $\tau \geq 0.8 \Rightarrow S$ ($>$ 250000 epochs)	$\epsilon < 0.8 \Rightarrow$ div. $\tau \geq 0.8 \Rightarrow S$ ($>$ 250000 epochs)	$\tau = 0.1 \Rightarrow S'$ $\epsilon \geq 0.8 \Rightarrow S$ ($>$ 250000 epochs)
naive $Q(\lambda)$	$\epsilon < 0.8 \Rightarrow$ div. $\epsilon \geq 0.8 \Rightarrow$ S ($>$ 250000 epochs)	all values of $\tau \Rightarrow$ div.	all values of $\epsilon \Rightarrow$ div.	$\tau = 0.1 \Rightarrow S'$ $\tau > 0.1 \Rightarrow$ div.

TABLE 1. Results obtained by the $Q(\lambda)$ and naive $Q(\lambda)$ algorithms.

divergence for all values of the temperature parameter. For *replacing* traces, the situation is different. The algorithm does not converge for any of the tested values of ϵ, in the case of the *intelligent look-ahead procedure* and exhibits the same behaviour when combined with the *softmax policy*. Exception is the case when $\tau = 0.1$, when naive $Q(\lambda)$ converges to a different solution than the correct one and the convergence is achieved within less than 40 seconds.

The obtained results for $Q(\lambda)$ and naive $Q(\lambda)$ are synthesized in Table 1. Here S denotes the convergence to the correct ordering: $S = 1, 2, 3, 4, 5, 6, 7, 8$; S' denotes the convergence to a different ordering than the correct one and the abbreviation "div." is used to indicate divergence.

4.3. **Discussion.** We have experimented with three Q-learning based algorithms and two types of action selection policies in order to obtain results for the temporal ordering problem. The original model is a Q-learning algorithm which uses an intelligent ϵ-Greedy based look-ahead action selection mechanism [3]. The other two algorithms combine Q-learning with eligibility traces [5]. As action selection policies, we used the look-ahead procedure we introduced in [3], as well as the softmax selection policy [12].

The obtained results demonstrate that the traditional Q-learning algorithm performs better, for the considered problem, than the two algorithms that use Q-learning in conjunction with eligibility traces: Watkins's $Q(\lambda)$ [14] and naive $Q(\lambda)$ [12]. Both these algorithms are able to retrieve the correct solution, for certain values of the considered action selection policy parameters, but convergence is much slower than in the case of Q-learning without eligibility traces. A possible explanation for this behaviour would be the fact that in our representation of the environment the full set of states is never completely known and therefore eligibility traces can only be updated for a known subset of states. This leads us to the conclusion that, for the TO problem, Q-learning with no eligibility traces is more appropriate.

With regard to the action selection policies, we remark that in the case of the Q-learning algorithm the intelligent action selection procedure [3] converges faster than the softmax selection mechanism, as this procedure efficiently guides the exploration of the search space. On the other hand, for the look-ahead mechanism the training process during an episode has a time complexity of $\theta(n^2)$, while for the softmax policy this complexity is $\theta(n)$, where n is the number of samples considered in the ordering process. We will further investigate how an intelligent action selection mechanism, based on softmax instead of ϵ-Greedy, could influence the outcome.

5. Conclusions and Further Work

In the present study we investigated several variations to a reinforcement learning based approach for the biological temporal ordering problem, tackled from a computational perspective. Three Q-learning based algorithms have been experimentally evaluated and compared.

The algorithms were tested on a time series gene expression data set, consisting of samples extracted from yeast cells affected by dithiothrietol exposure, at eight different time points [6]. The tests showed that the traditional Q-learning algorithm performs well with both considered action selection policies (intelligent look-ahead procedure [3] and softmax [12]), retrieving the correct ordering in less than 1 minute, for all tested values of the parameters, except for one case. The two algorithms that combine Q-learning with eligibility traces are also able to obtain the correct solution, but only for certain parameter settings and after a high number of training epochs.

We plan to extend the evaluation of the Q-learning based algorithms for the other data sets we used in [3], to further develop the analysis. We will also investigate possible improvements of these models by adding various local search mechanisms or combining the softmax policy with the intelligent action selection procedure introduced in [3], by testing different values for the decay parameter λ (in the case of $Q(\lambda)$ and naive $Q(\lambda)$), by decreasing the action selection parameters (ϵ and τ) during the training process or by extending the model to a distributed RL approach.

ACKNOWLEDGEMENT

I thank my thesis advisor, Professor Gabriela Czibula for the guidance, ideas and suggestions for this paper. This work was possible with the financial support of the Sectoral Operational Programme for Human Resources Development 2007-2013, co-financed by the European Social Fund, under the project number POSDRU/107/1.5/S/76841 with the title Modern Doctoral Studies: Internationalization and Interdisciplinarity.

References

[1] Chapman D., Kaelbling L. P. *Input generalization in delayed reinforcement learning: an algorithm and performance comparisons*, Proc. of the 12th International Joint Conference on Artificial Intelligence **2**, Morgan Kaufmann Publishers Inc., 1991, pp. 726-731, Sydney, New South Wales, Australia.

[2] Cohen W. W., Schapire R. E., Singer Y. *Learning to order things*, J Artif Intell Res **10**, 1999, pp. 243-270.

[3] Czibula G., Bocicor M.I., Czibula I.G. *Temporal Ordering of Cancer Microarray Data through a Reinforcement Learning based Approach*, Evolutionary Bioinformatics, 2012. Submitted for review.

[4] Czibula I. G., Czibula G, Bocicor M. I., *A Software Framework for Solving Combinatorial Optimization Tasks*, Studia Universitatis "Babes-Bolyai", Informatica, Proc. of KEPT 2011, Special Issue **LVI**(3), Babes-Bolyai University, 2011, pp. 3-8.

[5] Klopf A.H. *Brain function and adaptive systems. A heterostatic theory. Technical Report AFCRL-72-0164*, Air Force Cambridge Research Laboratories, Bedford, MA, 1972.

[6] Gasch A. P., Spellman P. T., Kao C. M., Carmel-Harel O., Eisen M. B., Storz G., Botstein D., Brown, P. O. *Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes*, Molecular Biology of the Cell **11**(12), 2000, pp. 4241-4257.

[7] Lin L.J. *Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching*, Machine Learning **8**, 1992, pp. 293-321.

[8] Peng J. *Efficient Dynamic Programming-Based Learning for Control*, PhD thesis, Northeastern University, Boston, MA, 1993.

[9] Perez-Uribe A. *Introduction to Reinforcement learning*, http://lslwww.epfl.ch/∼anperez/RL/RL.html, 1998.

[10] Singh S. P., Sutton R. S. *Reinforcement Learning with Replacing Eligibility Traces*, Machine Learning **22**, 1996, pp. 123-158.

[11] Susnea I., Vasiliu G., Filipescu A., Radaschin A. *Virtual Pheromones for Real-Time Control of Autonomous Mobile Robots*, Studies in Informatics and Control **18**(3), 2009, pp. 233–240.

[12] Sutton R.S., Barto A. G. *Reinforcement Learning: An Introduction*, MIT Press 1998.

[13] Thrun S. *The Role of Exploration in Learning Control*, Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches, Van Nostrand Reinhold, 1992, Florence, Kentucky.

[14] Watkins, C. J. C. H. *Learning from Delayed Rewards*, PhD thesis, Cambridge University, Cambridge, England, 1989.

Babeş-Bolyai University, Department of Computer Science, 1, M. Kogălniceanu street, 400084 Cluj-Napoca, Romania

E-mail address: `iuliana@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

REDUNDANT SPATIAL INDEX FOR SOLVING RANGE QUERIES

LEON ŢÂMBULEA, ADRIAN SERGIU DĂRĂBANT, AND ANDREEA NAVROSCHI-SZASZ

Abstract. Lately, new applications arise that manage large collections of location and points of interest objects. These are frequently accessed nowadays in mobile and web applications. In order to be able to query and update the position of these objects, several index structures are used. Each of these structures has its own advantages in solving a certain concrete problem. The term POI (Point of Interest) is employed in the context of use of mobile devices. A characteristic of POI collections of objects is their relatively static character (these objects do not usually change their location). In the index structures recommended for these collections a POI object is stored a single time. In this paper we suggest the alteration of an existing index structure by memorizing several times the addresses of some POI objects. The purpose of this multiple storage consists in reducing the response time for range queries.

Key-Words: Grid, R-tree, query POI objects, index redundancy

1. Introduction

A POI (Point of Interest) is a position or a complex 3D structure with some information associated with it (ID, civil address, category, name, description, etc.) [8].

The term of position (location) refers to a physical point on the Earth surface (specified within a system of coordinates). Some categories of POI objects are static (shopping center, speed camera, petrol station), but there are also POI objects valid only for a certain period of time (for example, a cultural event scheduled for a certain time interval).

Lately there are many applications that use spatial data queries, particularly POI objects collections. These applications start from a specified position

Received by the editors: November 28, 2012.

2000 *Mathematics Subject Classification.* 03G10.

1998 *CR Categories and Descriptors.* H.2.2 [**Database Management**]: Physical Design – *Access methods*; H.2.8 [**Database Management**]: Database applications – *Spatial databases and GIS*; H.2.4 [**Database Management**]: Systems – *query processing*.

or a position determined by a GPS device and they look for POI objects located near this position, with possible additional restrictions, returning the position and the associated information for the determined objects. For such applications it is necessary that the responses to these queries are obtained as quickly as possible. Large processing power and storage are generally not enough in order to achieve acceptable response times and high accuracy on this problem. One also needs specific types of indexes adapted to POI data and POI query processing. Among the index structures mostly used for this kind of queries we mention R-tree and the various versions of R-tree [2, 5, 9], the grid structures [1, 6, 7], etc. In each of these structures, a reference to an element from the POI objects collection is memorized a single time (there is no redundancy in the index).

In this paper we analyze the possibility of memorizing several times certain information (addresses of POI objects) in order to reduce the response time for these queries.

The rest of this paper is organized in the following way: section 2 shortly presents the R-tree and grid structures and the way to determine the responses to the queries, section 3 describes the suggested changes for the index associated to a grid, in section 4 we emphasize the results obtained for some experiments, and in section 5 we formulate some conclusions.

2. R-tree and Grid Structures

The POI objects collection is stored into a data base. Each object has an *idPOI*, a *position(x,y)*, and other associated information.

A range query reference is specified by two opposite points in a rectangle D. With this type of query we ask for all POI objects located inside the rectangle D.

To avoid the sequential access to all the objects of the data base, we build an index. For each POI object in the index we memorize at least ($idPOI$ and (x, y)).

2.1. **R-tree.**

The R-tree structure [2] was widely studied and used. An R-tree is a balanced tree with two types of nodes:

- end nodes, where we store a sequence of values $(idPOI, (x, y))$;
- internal nodes, where we store a sequence of values $(idchild, MBR)$. *idchild* is a pointer to an internal or end node, and MBR (minimum bounding rectangle) is the smallest rectangle that includes all the objects that arise in the sub-trees of this node. There is no criterion for the order of elements from an internal or end node.

As parameters of the index we have [4]:

- the dimension of a node of the R-tree, from which we can determine the maximum number of inputs to the node: the number of $(idchild, MBR)$ pairs from an internal node, or the number of $(idPOI, (x, y))$ values from an end node. Let M be the maximum number of entries that will fit in one node.
- the minimum number of entries in one node (which is not the root node) is given by a second parameter $m \leq M/2$. This parameter dictates the minimum occupation of a node.

An internal node is analyzed (the nodes referred by this one are covered) if the associated MBR intersects the rectangle D. For a terminal node all the $(idPOI, (x, y))$ values are examined and the objects for which $(x, y) \in D$ are included in the query answer. Depending on the query, it's possible that several branches of the tree need to be covered.

2.2. **Grid.** The domain where the POI objects are located is divided into a regular network of square cells (resulting into a matrix of cells) (see Figure 1). The objects located inside such a cell are attached to this one and are stored into a list. For a more efficient browsing of the list, the objects can be stored into a linked list of blocks (a block may contain a specified maximum number of $(idPOI, (x, y))$ elements).

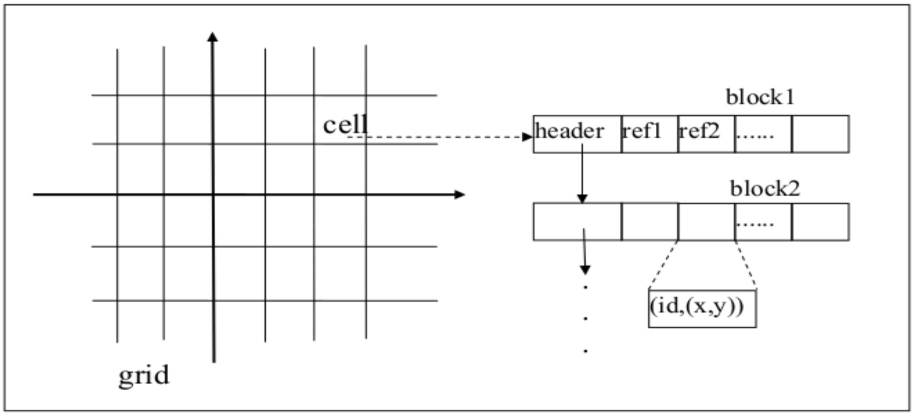


FIGURE 1. Grid index structure

The parameters of this index are [10]:

- The dimension used for dividing the domain of POI objects (the side of a square cell from the grid);
- The dimension of a block.

To determine the answer to a range query the following two steps need to be taken:

(a) We determine the cells that are completely included inside the rectangle D (all the elements from these cells are included in the answer);

(b) We determine the cells that are partially covered by the rectangle D. For the objects of these cells we perform the test $(x, y) \in D$.

The POI objects are not uniformly spread in the plane, therefore some cells will have many blocs (of objects) associated, and the search from step b) can include a lot of data. In [3], the authors propose the use of a several levels grid (the cells with many objects are organized as new grids).

Another possible index consists in storing the list of POI objects that belong to a certain cell into a R-tree. Therefore the index structure is made of a collection of R-trees and a matrix of references to these R-trees.

3. Redundant Index

Let d be the dimension of a cell from the grid built for the collection of POI objects (according to section 2.2). We assume that the rectangle D from the range query is a square and it corresponds to the following query: find all POI objects located within a maximum distance d_0 from a point (x_0, y_0). To increase the efficiency of the search, we first find the POI objects located on the coordinate axes (on each direction), within a maximum distance d_0 from point (x_0, y_0), therefore:

$D(x_0, y_0; d_0) = \{(x, y) \in \Re \times \Re |\ |x - x_0| \leq d_0, |y - y_0| \leq d_0\}$

Let s be the maximum possible value of d_0. If s is large, then:

- the number of cells from the grid queried is large (according to section 2.2);
- the answer to the range query specified by $D(x_0, y_0, d_0)$ contains a large number of recordings, situation that is detrimental for most applications that use such queries.

Next we suggest a possible alteration of the index structure if $s < d/2$, and in the next section we prove that in some cases the response time for range queries respecting the above conditions is reduced.

We build a "main index" for the grid, as suggested in Section 2.2. For the above hypothesis $(s < d/2, d_0 \leq s)$, the answer to a range query is located in one, two or four cells of the grid, as can be seen in Figure 2.

Let q be a range query, $D(x_0, y_0, d_0)$ the associated square and $z(x_0, y_0)$ the cell in the grid where point (x_0, y_0) appears. The answer to the query q is found by browsing:

- the objects associated to the cell $z(x_0, y_0)$. Let $n(q)$ the number of objects in this cell;

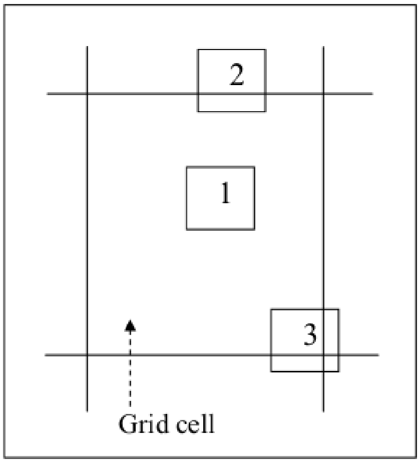


FIGURE 2. Possible overlays of the range query answer over the grid area

- the objects from the neighboring cells that intersect D. Let $n_s(q)$ be the number of objects in these cells. If D is in the first case stated by Figure 2, then $n_s(q) = 0$.

We will denote by $n_1(q)$ the number of objects queried in order to find the answer to q, so $n_1(q) = n(q) + n_s(q)$.

An additional index - "the secondary index" is added to the grid. It contains, for a given cell z, references to objects in the neighboring cells (those that share a side with z) and located within a minimum distance s from the sides of the cell z. The areas containing the additional objects attached to a cell are highlighted in Figure 3.

With this version of index, the answer to a query q specified by $D(x_0, y_0, d_0)$ is found by:

- browsing the $n(q)$ objects from the main index associated to the cell z;
- browsing the objects in the secondary index associated to the cell z, if D is in case 2 or 3 stated by Figure 2. Let $n_{si}(q)$ be the number of objects in the secondary index that need to be queried. If $D \subseteq z$, then $n_{si}(q) = 0$.

We will denote by $n_2(q)$ the number of objects queried in order to find the answer to q, therefore $n_2(q) = n(q) + n_{si}(q)$.

Let C be the collection of POI objects. For a certain d (the dimension of a cell from the grid) and s (the maximum dimension for the d_0 values from the

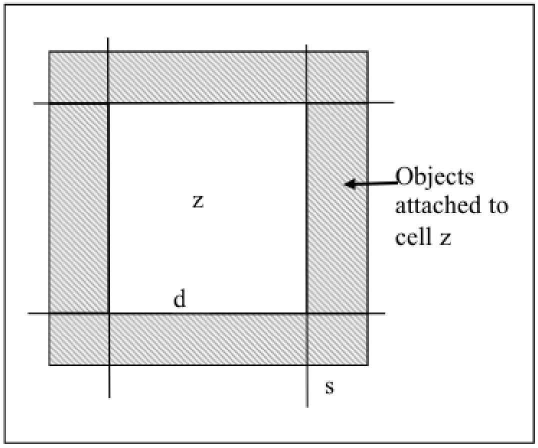


FIGURE 3. Redundant objects attached to a grid cell in the secondary index

range query) we can build the two index structures (the main and secondary one) by a single traversal of collection C.

For an object $ob = (id, (x, y)) \in C$ we determine:

(a) $(i, j) = (\lfloor \frac{x}{d} \rfloor, \lfloor \frac{y}{d} \rfloor)$ and the object is included in the main index of cell (i, j);

(b) $if\ x \in I_1 = [i \cdot d, i \cdot d + s)\ then\ left := i - 1\ else\ left := null$;
$if\ x \in I_2 = [(i+1) \cdot d - s, (i+1) \cdot d)\ then\ right := i + 1\ else\ right := null$;
$if\ y \in J_1 = [j \cdot d, j \cdot d + s)\ then\ bottom := j - 1\ else\ bottom := null$;
$if\ y \in J_2 = [(j+1) \cdot d - s, (j+1) \cdot d)\ then\ top := j + 1\ else\ top := null$;

(c) If at least one of the previous four conditions is satisfied then object *ob* is included in the secondary index of cell (m, n). We denote by $S_i(ob; m, n)$ the operation of inserting *ob* in the secondary index.

```
if (left!= null) and (top!=null) then $S_i$(ob; left, top);
if (top!=null) then $S_i$(obj; i, top);
if (right!=null) and (top!=null) then $S_i$(ob; right, top);
if (right!=null) then $S_i$(ob; right, j);
if (right!=null) and (bottom!=null) then $S_i$(ob; right, bottom);
if (bottom!=null) then $S_i$(ob; i, bottom);
if (left!=null) and (bottom!=null) then $S_i$(ob; left, bottom);
if (left!=null) then $S_i$(ob; left, j);
```

Figure 4 illustrates the above presented cases. It can be seen that:

- each object is included in the main index for a single cell;
- objects from the subareas z_2, z_4, z_6, z_8 are contained in a single cell of the secondary index;
- objects from subareas z_1, z_3, z_5, z_7 are contained in three cells of the secondary index;

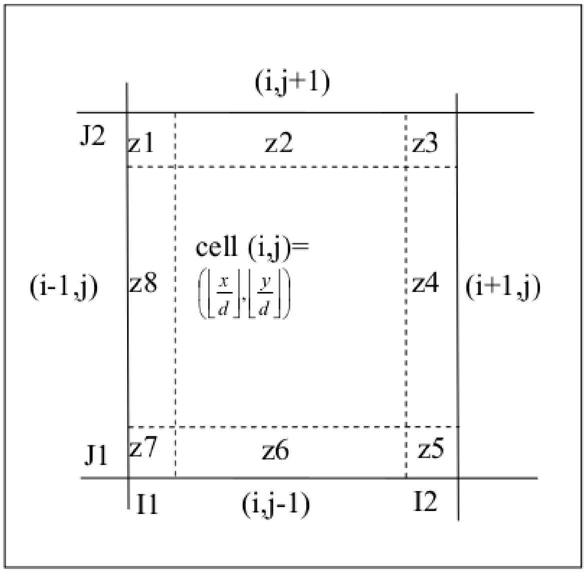

(i,j+1)
J2
z1
z2
z3
(i-1,j)
z8
cell (i,j)=
$\left(\left\lfloor \frac{x}{d} \right\rfloor, \left\lfloor \frac{y}{d} \right\rfloor\right)$
z4
(i+1,j)
J1
z7
z6
z5
I1
(i,j-1)
I2

FIGURE 4. Objects included in the secondary index.

By redundantly storing references to some objects, the index size grows, with the size of the secondary index. The induced redundancy coefficient can be expressed by the following formula:

(1) $$c = \frac{size(secondary\_index)}{size(main\_index)}$$

This coefficient depends on the d and s system parameters and on the object distribution in the generated grid.

For a set of range queries Q we can determine if using the main and a secondary index is more efficient than the main only index version. In order to achieve this we introduce the efficiency coefficient:

$$e = \frac{\sum\limits_{q \in Q} n_2(q)}{\sum\limits_{q \in Q} n_1(q)} \tag{2}$$

For $e < 1$, the total number of accessed objects when applying the new indexing schema for all queries in Q is smaller than in a classical grid structure index.

4. Experimental results

In this section we present the experimental results obtained when varying the values of the parameters introduced in the previous sections.

4.1. **The Object Database Collection.**

In the conducted experiments we used a real POI database collection provided by one of the major actors in this field on the market. The main characteristics of this collection are:

- the number of POI objects is 1,195,398;
- the area covered by the database collection is $latitude \in [-54.8, 78.2], longitude \in [-179.8, 179.4]$.

For experiments we generated 100,000 range queries, each query being expressed by a location and a range $(x_0, y_0; d_0)$. The (x_0, y_0) locations were randomly generated on the area covered by the POI collection. For each given s, the d_0 parameter values were randomly generated in the interval $[s \div 4, s]$.

4.2. **Measurements.**

For each d we compute the number of non-empty cells in the grid. Table 1 shows the number of non-empty cells for a few values of the d parameter.

d(km)	50	100	150	200	300
Cell Count	11,412	5,243	3,268	2,295	1,384

Table 1. Number of non-empty cells vs grid size.

For a given d and s (a subset of the experimental values) and for all queries in Q Table 2 shows:

- *Secondary Index Size* - the size of the secondary index;
- c - the redundancy coefficient;
- $N(q) = \sum\limits_{q \in Q} n(q)$;

- *Additionally inspected cells* - the number of additional grid cells intersected by all queries;
- $N_s(Q) = \sum\limits_{q \in Q} n_s(q)$;
- $N_{si}(Q) = \sum\limits_{q \in Q} n_{si}(q)$;
- the efficiency coefficient $e = \frac{N(Q)+N_{si}(Q)}{N(Q)+N_s(Q)}$;

From the Table 2 it can be seen that the efficiency coefficient has a value less than 1 for many situations. In all these cases the number of analyzed objects for a variant with a secondary index is smaller than the number of analyzed objects for a main grid index without redundancy.

Also, the redundancy coefficient has in these cases reasonable values. This means the overhead incurred for the multiple storage of the same objects would not impact significantly the data structures stored in the main memory. By

d (km)	s	Secondary Index Size	c	N(Q)	Additionally inspected cells	$N_S(Q)$	$N_{SI}(Q)$	e
50	5	470,919	0.42	481,095	24,907	103,381	40,642	0.893
50	10	985,209	0.88	481,095	50,078	242,636	170,561	0.900
50	15	1,600,183	1.43	481,095	74,781	293,400	385,180	1.119
100	5	227,479	0.20	1,853,072	12,468	221,971	52,538	0.918
100	10	456,930	0.41	1,853,072	25,061	513,589	176,288	0.857
100	20	1,017,484	0.91	1,853,072	50,403	880,128	682,995	0.928
100	30	1,670,481	1.50	1,853,072	75,182	1,248,760	1,616,484	1.119
150	5	165,392	0.15	4,206,558	8,299	276,158	45,101	0.948
150	10	336,591	0.30	4,206,558	16,630	676,243	187,649	0.900
150	15	510,677	0.46	4,206,558	24,844	930,088	446,989	0.906
150	20	693,832	0.62	4,206,558	33,276	1,249,643	837,634	0.924
150	30	1,110,691	1.00	4,206,558	49,852	1,721,729	1,745,026	1.004
200	5	111,985	0.10	7,320,215	6,232	610,148	68,648	0.932
200	10	222,985	0.20	7,320,215	12,615	1,153,923	214,695	0.889
200	20	511,359	0.46	7,320,215	25,107	1,953,250	797,491	0.875
200	30	800,983	0.72	7,320,215	37,600	2,768,881	1,697,463	0.894
200	40	1,091,544	0.98	7,320,215	49,984	3,578,632	3,188,074	0.964
200	50	1,389,982	1.25	7,320,215	62,557	3,955,536	4,818,800	1.077
300	5	81,718	0.07	17,126,951	4,187	579,144	51,908	0.970
300	10	158,232	0.14	17,126,951	8,347	1,201,525	186,371	0.945
300	20	331,249	0.30	17,126,951	16,612	2,522,389	768,913	0.911
300	30	533,765	0.48	17,126,951	24,796	3,617,474	1,857,947	0.915

TABLE 2. Number of non-empty cells vs grid size.

analyzing these two factors over a set of queries Q and a typical database, one can decide whether adding a secondary redundant index like the one proposed above can improve or not the query response time.

5. Conclusions

In this paper we proposed an improvement on the indexing of a statical collection of POI objects by adding an additional index. The additional index uses only redundant references that are copies of some of the entries from the main grid index. In some cases the proposed method improves the response time by 10%-15% compared to a typical R-tree or Grid index at the cost of some additional required memory. Update operations are considered in the same context as for classical R-trees and Grid indexes and are usually expensive. This is a well known problem of the R-tree index as well. The conducted experiments show that there are real life situations where applying the proposed method helps reducing query response time at the cost of very little added memory.

References

[1] J. L. Bentley, J. H. Friedman, *Data Structures for Range Searching*, ACM Comput. Surv., 11, 4, 397409, 1979.
[2] A. Guttman, *R-Trees - A Dynamic Index Structure for Spatial Searching*, SIGMOD Rec., 14(2):4757, 1984.
[3] D. V. Kalashnikov, S. Prabhakar, S. E. Hambrusch, *Main Memory Evaluation of Monitoring Queries Over Moving Objects*, Distrib. Parallel Databases, 15, 2, 2004, 117-135.
[4] Ling. Liu, M. Tamer zsu (Eds.), *Encyclopedia of Database Systems*, Springer, 2009.
[5] Y. Manolopoulos, A. Nanopoulos, A.N. Papadopoulos, Y. Theodoridis, *R-Trees: Theory and Applications*, Series in Advanced Information and Knowledge Processing, Springer 2005.
[6] Ming Qi, Guangzhong Sun, Yun Xu, *Query as Region Partition in Managing Moving Objects for Concurrent Continuous Query*, International Journal of Research in Computer Science, 2 (1): pp. 1-6, 2011.
[7] J. Nievergelt, H. Hinterberger, K. C. Sevcik, *The Grid File: An Adaptable, Symmetric Multikey File Structure*, ACM Transactions on Database Systems, 9(1):3871, 1984.
[8] *Points of Interest Core*, W3C Working Draft 12 May 2011, http://www.w3.org/TR/poi-core/.
[9] A.Sabau, *Management of Spatio-Temporal Databases*, PhD Thesis, Cluj-Napoca 2007.
[10] D. Sidlauskas , S. Saltenis , Ch. W. Christiansen , J. M. Johansen , D. Saulys, *Trees or Grids? Indexing Moving Objects in Main Memory*, Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2009, Seattle, Washington.

Babes Bolyai University, Faculty of Mathematics and Computer Science, Cluj Napoca, Romania

E-mail address: leon@cs.ubbcluj.ro, dadi@cs.ubbcluj.ro, deiush@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

ASPECT MINING. PAST, PRESENT, FUTURE

GRIGORETA S. COJOCAR

Abstract. Aspect mining is a research domain that tries to identify crosscutting concerns in already developed software systems. The goal is to refactor the analyzed system to use aspect oriented programming in order to ease the maintainability and evolution of the system. In this paper we briefly describe the aspect mining techniques proposed so far, we analyze them using three new criteria, and we discuss some possible future research directions considering the current state of the art.

1. Introduction

Ever increasing software systems made designing and implementing them a complex task. Software systems are composed of many different concerns, where a concern is a specific requirement or consideration that must be addressed in order to satisfy the overall system. The concerns are divided in core concerns and crosscutting concerns. The core concerns capture the central functionality of a module, while crosscutting concerns capture system-level, peripheral requirements that cross multiple modules. The current paradigms like procedural or object oriented programming provide good solutions for the design and implementation of core concerns, but they cannot deal properly with crosscutting concerns. Different approaches have been proposed for the design and implementation of crosscutting concerns: subject oriented programming [38], composition filters [1], adaptive programming [23], generative programming [10], aspect oriented programming (AOP) [19]. From these approaches, the aspect oriented programming approach has known the greatest success both in industry and academia.

In order to design and implement a crosscutting concern, AOP introduces four new concepts: *join point* (i.e., a well-defined point in the execution of a program), *pointcut* (i.e., groups a set of join points and exposes some of the values in the execution context of those join points), *advice* (i.e., a piece of

Received by the editors: November 24, 2012.

2010 *Mathematics Subject Classification.* 68N19, 68N99.

1998 *CR Categories and Descriptors.* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – *Restructuring, reverse engineering, and reengineering*.

Key words and phrases. aspect mining, analysis, crosscutting concerns.

code that is executed at each join point in a pointcut), and a new modularization unit called *aspect*. The aspect is woven to generate the final system, using a special tool called *weaver*. Some of the benefits that the use of AOP brings to software engineering are: better modularization, higher productivity, software systems that are easier to maintain and to evolve. Nowadays, there are many programming language extensions to support AOP: AspectJ for Java [3], AspectC++ for C++ [2], etc.

For more than a decade researchers have tried to develop techniques and tools to (automatically) identify crosscutting concerns in already developed software systems, without using AOP. This area of research is called *Aspect Mining*. The goal is to identify the crosscutting concerns, and then to refactor them to aspects, in order to obtain a system that can be easily understood, maintained and modified.

In order to identify crosscutting concerns, the techniques try to discover one or both symptoms that appear when designing and implementing crosscutting concerns using the existing paradigms: code *scattering* and code *tangling*. Code scattering means that the code that implements a crosscutting concern is spread across the system, and code tangling means that the code that implements some concern is mixed with code from other (crosscutting) concerns.

Until now, many different approaches have been used for aspect mining, and different techniques have been proposed. In this paper we try to analyze the state of the art of aspect mining from the perspective of the proposed goal and to identify other possible future research directions in this field. The main contributions of this paper are to analyze the existing aspect mining techniques using three new criteria: industry usage, IDE integration and integration with AO refactoring, and to discuss possible new research directions in this field.

The paper is structured as follows. Section 2 presents an overview of the aspect mining techniques proposed so far. Section 3 analyzes the aspect mining techniques using three new different criteria. In Section 4 we discuss some possible future research directions considering the current state of the art of this field.

2. Overview of Aspect Mining Techniques

The first approaches in aspect mining were query-based search techniques. The developer had to introduce a so-called seed (eg., a word, the name of a method or of a field) and the associated tool showed all the places where the seed was found. Very soon, researchers discovered that this approach to aspect mining has some important disadvantages: the tool user had to have an in-depth knowledge of the analyzed system, as he/she had to figure out the seed(s) to be introduced, and the large amount of time needed in order to filter the results displayed. There are many query based aspect mining tools proposed: Aspect Browser [12], The Aspect Mining Tool(AMT) [13], Multi-Visualizer(AMTEX) [44], Feature Exploration and Analysis Tool(FEAT) [33], QJBrowser [31], JQuery [17], Prism [45] and Theme/Doc [4]. Except for the last one, all the other techniques are performing the search in the source code of the mined system. The Theme/Doc tool is searching for the seed in the requirements specifications.

Starting with 2004 researchers have focused on developing aspect mining techniques that do not require an initial seed from the user. These techniques try to identify the crosscutting concerns starting just from some kind of system representation (the source code, the requirements documentation, some execution traces, etc.), and are called automated aspect mining techniques. Different approaches were used: metrics, clustering, clone detection techniques, association rules, formal concept analysis, natural language processing, etc. In the following we briefly describe the automated aspect mining techniques proposed so far.

Marin et al. [26] have proposed an aspect mining technique that looks for methods that are called from many different call sites and whose functionality is needed across different methods, classes, and packages. The authors aim at finding such methods by computing the *fan-in* metric for each method using the static call graph of the system. Their approach relies on the observation that scattered, crosscutting functionality that largely affects the code modularity is likely to generate high fan-in values for key methods implementing this functionality.

Tonella and Ceccato [40] have proposed to use dynamic code analysis, feature location and formal concept analysis [11] for aspect mining, as follows. Execution traces are obtained by running an instrumented version of the program under analysis for a set of scenarios (use cases). The relationship between execution traces and executed computational units (methods) is subjected to concept analysis. The execution traces associated with the use-cases are the objects of the concept analysis context, while the executed methods are the attributes. In the resulting concept lattice, the concepts that satisfy both the scattering and the tangling conditions are considered as aspect candidates.

Breu and Krinke have proposed an aspect mining technique based on execution relations [6]. The proposed approach has two versions: a dynamic one [6] and a static one [20, 21]. They introduce the notion of *execution relation*, that describes the kind of relation that may exist between the executions of two methods. In the dynamic version the execution relations are extracted from program traces, and in the static version the execution relations are extracted from the control flow graph. They identify recurring execution patterns which describe certain behavioral aspects of the software system, and expect these patterns to be potential crosscutting concerns which describe recurring functionality in the program and thus are possible aspects. The authors have focused only on method executions as they wanted to analyze object-oriented systems where logically related functionality is encapsulated in methods.

Sampaio et al. [34] have proposed an approach for mining aspects from requirements related documents. Their approach builds upon the ideas of Theme/ Doc approach [4], but uses corpus-based natural language processing techniques in order to effectively enable the identification of aspects in semi-automated way. The main goal of their approach is to determine potential aspect candidates in requirements documents regardless of how they are structured (e.g., informal descriptions, interviews, structured documents).

Kim and Tourwé have proposed an aspect mining technique that relies on the assumption that naming conventions are the primary means for programmers to associate related but distant program entities [41]. Their technique tries to identify potential aspects and crosscutting concerns by grouping program entities with similar names. They apply formal concept analysis where the objects are all the classes and methods in the analyzed program and the attributes are the identifiers associated with those classes and methods. The authors chose for inspection only the groups that contain at least a given number of objects (a given threshold) and that are crosscutting (i.e., the involved methods and classes belong to at least two different class hierarchies).

Breu and Zimmermann tried to solve the problem of aspect mining taking a historical perspective [7]: they mine the history of a project (version archives) and identify code changes that are likely to be crosscutting concerns. Their analysis is based on the hypothesis that crosscutting concerns evolve within a project over time. A code change is likely to introduce such a concern if the modification gets introduced at various locations within a single code change.

Some authors tried to use clone detection techniques that aim at finding duplicated code, which may have been slightly adapted from the original. They base their research on the observation that typically source code implementing a crosscutting concern involves a great deal of duplications. Since the code belonging to a crosscutting concern cannot be cleanly captured inside a single abstraction, using the current programming paradigms, it cannot be reused. Therefore, developers are forced to write the same code over and over again, and are tempted to just copy paste the code and adapt it slightly to the context.

Shepherd et al [36] proposed the first automatic aspect mining technique based on clone detection. They based their analysis on AspectJ, particularly on the `before` advice. The technique consists in identifying initial refactoring candidates for the `before` advice using a control-based comparison, followed by filtering based on data dependence information. They used two types of clone detection techniques for identifying crosscutting concerns: PDG-based and AST-based clone detection techniques.

Bruntink et al. [9] tried to evaluate the usefulness and accuracy of clone detection techniques in aspect mining. The existing clone detectors usually produce output consisting of pairs of clones, i.e., they report which pairs of code fragments are similar enough to be called clones. The authors then investigate the groups of code fragments that are all clones of each other, called *clone classes*, in order to find aspect candidates.

Bruntink has extented the approach described in [9] considering metrics that grade the obtained clone classes [8]. The considered metrics were defined with the purpose of improving maintainability when aspects are used.

Orlando Mendez has also studied the applicability of clone detection techniques to aspect mining [30]. However, he used only one clone detector and applied it for one case-study.

Many authors have tried to use clustering for crosccutting concerns identification. *Clustering* is a division of data into groups of similar objects [5, 16]. Each group, called *cluster*, consists of objects that are similar between themselves and dissimilar to objects of other groups.

Shepherd and Pollock [37] used clustering to find methods with similar name as an indication of crosscuttingness. They perform agglomerative hierarchical clustering in order to group methods. The objects to be clustered are the names of the methods from the software system under analysis. The authors have developed a tool that helps users navigate and analyze the obtained clusters. The rest of the approach is just manual analysis of the obtained results using the tool.

Moldovan and Şerban have proposed a clustering based aspect mining approach that tries to discover crosscutting concerns by finding attributes of the *code scattering* symptom [28]. The authors use the vector space model based approach with two different vector space models, and different clustering algorithms (hard k-means clustering, fuzzy clustering, hierarchical agglomerative clustering, genetic clustering, etc) in order to group the methods from the software system into clusters.

Şerban and Moldovan also proposed an approach based on graph [35]. This approach is similar to the clustering one, but they use graphs, and in order to obtain a partition of the software system under analysis.

He and Bai [14] have proposed an aspect mining technique based on dynamic analysis and clustering that also uses association rules. They first use clustering to obtain crosscutting concern candidates and then use association rules to determine the position of the source code belonging to a crosscutting concern in order to ease refactoring. Execution traces are generated for an instrumented version of the software system, and for specified scenarios and inputs. Every scenario has a called-method sequence. If there exists a group of codes that has similar action, i.e., similar called-method sequence, and it frequently appears in execution traces, then a crosscutting concern may exist. Similar called-method sequences are considered possible crosscutting concerns code. Clustering analysis is used to find similar called-method sequences. The scenarios are the objects to be clustered, and the methods from the software systems are the attributes.

Maisikeli has proposed a dynamic aspect mining technique that uses a neural network clustering method called Self Organizing Map (SOM) [24]. He used a set of legacy benchmark programs to determine the most relevant software metrics that can be used for aspect mining (i.e., dynamic fan-in/fan-out, information flow, method spread, method cohesion contribution, etc.). The mined software system is executed to compute the value of these metrics. Based on these metrics he constructs a vector space model that is submitted as input to SOM for clustering. The results obtained by SOM are then manually analyzed to identify crosscutting concerns.

Rand Mcfadden has expanded the approach of Moldovan and Serban by using model based clustering [32]. She investigated the performance of different model based clustering algorithms with six vector space models (the two defined by Moldovan and Serban, and four new ones).

Vidal et al. have proposed another aspect mining technique based on dynamic analysis and association rule mining [42]. They execute the system using a set of scenarios in order to obtain execution traces. These execution traces are given as input to an association rule algorithm to find interesting associations among methods. The rules obtained are classified and filtered out in order to remove redundant rules or rules with utility methods.

Huang et al. have proposed an aspect mining technique inspired by the link analysis of information retrieval technology [15]. They try to discover the crosscutting concerns in the concern graphs extracted from the program using a two-state model. They compute the program elements that are in the *scatter* and *centralization* states, and use a ranking technique to select the crosscutting concerns candidates.

3. Analysis of Aspect Mining Techniques

Many different aspect mining techniques have been proposed so far, some of them in the last two years. However, if the techniques proposed in the beginning used very different apprroaches, the last ones (proposed in the last two-three years) are more an improvement of some of the previously proposed techniques. Even so, the results obtained by the new aspect mining techniques did not improve significantly. They obtained better results, but not much better.

In 2008, Mens et al. [27] have conducted an analysis of the problems the proposed aspect mining techniques were encountering. They have identified as main problems: poor precision, poor recall, subjectivity, scalability, lack of empirical validation. They have also identified the causes of these problems. In their opinion there are three main root causes: inappropriateness of the techniques used to mine for aspects, lack of a precise definition of what constitutes an aspect, and inadequate representation of the aspect mining results.

Even though the study conducted by Mens et al. describes most of the problems that exist in the aspect mining research field, there are other criteria that must also be considered when analyzing the aspect mining techniques, like the usage of aspect mining techniques in industry, integration of techniques with IDEs, and link of these techniques with aspect oriented refactoring tools. In the following we analyze the aspect mining techniques using these criteria.

3.1. **Aspect mining techniques used in industry.** Have any of the aspect mining techniques been used for complex projects? In Section 2 were briefly described the aspect mining techniques proposed so far. Most of the techniques used as case-studies JHotDraw version v5.4b1 [18] and Carla Laffra implementation of Dijkstra algorithm [22]. However, these case studies cannot be considered as complex. The former is a small to medium size software system, but the later case study is a small one consisting of only 6 classes. There are just a few case studies used (i.e., Tomcat, Eclipse v3.2M3) that can be considered as more complex. There are also no other reports or surveys describing the use of any of the aspect mining techniques for more complex software systems.

3.2. **IDE Integration.** Even though there are so many aspect mining techniques proposed, most of them cannot be used as there is no associated tool available. The only technique that can be used by others is the Fanin technique that has an Eclipse plugin available. The technique proposed by Vidal et al. [42] also described the use of an Eclipse-based tool, called AspectRT, however it is not publicly available. There are also a few techniques which can be recreated by following and using the same tools as the proponents of the techniques. However, for most of them, there is no tool available which makes it difficult for others to use them for other case studies or software systems.

3.3. **Integration with AO Refactoring.** There are a few reports available [39, 43] about using the results obtained by different aspect mining techniques in order to refactor the mined system to use aspect oriented constructs. For both reports the conclusion was that not all the crosscutting concerns that exist in a software system can be easily redesigned and implemented using AOP. For some crosscutting concerns, even the aspect oriented paradigm is not a good solution.

Some refactorings were proposed in order to ease the migration to an aspect oriented system [29]. However, except for the Vidal et al. technique, none of the techniques take into the consideration the subsequent refactoring step. This may be due to the facts that all the techniques still require a large amount of user involvement in order to analyze the results obtained by them, and that there are still a large number of false positives in the results presented. Some of the authors have considered refactoring the case studies used for aspect mining [25], however the approach used is mainly manually, without tool support.

4. Future of Aspect Mining

Considering the problems discovered by Mens et al. [27], and the additional criteria discussed in Section 3, it is very unlikely that any of the existing aspect mining techniques will be adopted by the industry in the near future. Without a major change in the approach used for aspect mining, the industry practitioners will not consider using an aspect mining technique. In the following we discuss some possible research directions, that might ease the adoption from industry.

4.1. **Top-down approaches.** In the beginning, the researchers have considered using a top-down approach for aspect mining. Using a catalog of known crosscutting concerns, these kind of approaches should focus on identifying the crosscutting concerns from the catalog. This may reduce execution time, the large number of false positives, and the user involvement in analyzing the obtained results.

4.2. **Identify only refactorable crosscutting concerns.** The case studies used for aspect mining and AO refactoring have already shown that only a susbset of the crosscutting concerns (CCCs) that exist in an object-oriented software system may be refactored into aspects. Future aspect mining techniques should focus on identifying only the refactorable crosscutting concerns. This may ease the integration of the next step: refactoring to use AOP. In this case, the techniques should also considered the right level of granularity for refactorable CCCs. The existing aspect mining techniques consider different levels of granularity: statement for clone-detection based techniques, methods for almost all techniques. However, there is still the question of which level is better: statement or method? If we consider only the refactorable CCCs, we should already know how they will be refactored, and it might help in identifying the granularity level used for mining CCCs.

4.3. **Create a catalogue of refactorable CCCs.** In order to identify the refactorable CCCs, we first need to know the crosscutting concerns that are refactorable into aspects. For that we need to create a catalogue. We may start by putting the most known crosscutting concerns that can be designed and implemented using AOP, like security, transaction management, logging and add new CCC as they appear in practice.

4.4. **Tool support.** It is very important that future aspect mining techniques consider developing an associated tool, that can be integrated with existing IDEs. Without such tools, the industry might not adopt/use the technique, even thought it may obtain good results.

5. Conclusions

In this paper we have have briefly described the existing aspect mining techniques, and, then, we have analyzed them using criteria like industry adoption, IDE integration, and subsequent refactoring. We have also discussed some future directions that should be considered for aspect mining.

Many different aspect mining techniques have been proposed so far, however case studies have shown that they do not perform very good: they have low precision, a large number of false positive, they still require a large amount of user involvement, and they cannot be integrated with refactoring tools.

Trying to discover all crosscutting concerns that exist in software systems is not suited for aspect mining and aspect oriented refactoring. Other approaches should be considered for aspect mining in order to be able to get near to one of the objective of aspect mining, that of refactoring the identified crosscutting concerns into aspects. Considering a top down approach, in the future, may be more efficient as it may reduce the number of results presented to the user, it may increase precision, it may decrease the time needed to identify the crosscutting concerns, and it may also make possible integration with refactoring tools.

References

[1] Mehmet Aksit. *On the Design of the Object Oriented Language Sina.* PhD thesis, Department of Computer Science, University of Twente, The Netherlands, 1989.
[2] AspectC++ Homepage. http://www.aspectc.org/.
[3] AspectJ Project. http://eclipse.org/aspectj/.
[4] Elisa Baniassad and Siobhán Clarke. Finding Aspects in Requirements with Theme/Doc. In *Proceedings of Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design*, Lancaster, UK, March 2004.

[5] Pavel Berkhin. Survey of Clustering Data Mining Techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[6] Silvia Breu and Jens Krinke. Aspect Mining Using Event Traces. In *Proceedings of International Conference on Automated Software Engineering (ASE)*, pages 310–315, 2004.

[7] Silvia Breu and Thomas Zimmermann. Mining Aspects from Version History. In Sebastian Uchitel and Steve Easterbrook, editors, *21st IEEE/ACM International Conference on Automated Software Engineering (ASE 2006)*. ACM Press, September 2006.

[8] Magiel Bruntink. Aspect Mining Using Clone Class Metrics. In *Proceedings of the 2004 Workshop on Aspect Reverse Engineering (co-located with WCRE 2004)*, November 2004. Published as CWI technical report SEN-E0502, February 2005.

[9] Magiel Bruntink, Arie van Deursen, Remco van Engelen, and Tom Tourwé. On the use of clone detection for identifying crosscutting concern code. *IEEE Transactions on Software Engineering*, 31(10):804–818, 2005.

[10] Krzysztof Czarnecki and Ulrich Eisenecker. *Generative Programming: Methods, Tools, and Applications.* Addison-Wesley, 2000.

[11] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis.* Springer-Verlag, Berlin, Heidelberg, New York, 1996.

[12] William G. Griswold, Yoshikiyo Kato, and Jimmy J. Yuan. AspectBrowser: Tool Support for Managing Dispersed Aspects. Technical Report CS1999-0640, UCSD, March 2000.

[13] Jan Hannemann and Gregor Kiczales. Overcoming the Prevalent Decomposition of Legacy Code. In *Advanced Separation of Concerns Workshop,at the International Conference on Software Engineering (ICSE)*, May 2001.

[14] Lili He and Hongtao Bai. Aspect Mining using Clustering and Association Rule Method. *International Journal of Computer Science and Network Security*, 6(2):247–251, February 2006.

[15] Jin Huang, Yansheng Lu, and Jing Yang. Aspect mining using link analysis. In *Proceedings of the 2010 Fifth International Conference on Frontier of Computer Science and Technology*, pages 312–317. IEEE Computer Society, 2010.

[16] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, New Jersey, 1998.

[17] Doug Janzen and Kris De Volder. Navigating and Querying Code Without Getting Lost. In *Proceedings of Aspect-Oriented Software Development*, pages 178–187, Boston, USA, 2003. ACM Press.

[18] JHotDraw Project. http://sourceforge.net/projects/jhotdraw.

[19] Gregor Kiczales, John Lamping, Anurag Menhdhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. In *Proceedings European Conference on Object-Oriented Programming*, volume LNCS 1241, pages 220–242. Springer-Verlag, 1997.

[20] Jens Krinke. Mining control flow graphs for crosscutting concerns. In *13th Working Conference on Reverse Engineering: IEEE International Astrenet Aspect Analysis (AAA) Workshop*, pages 334–342, 2006.

[21] Jens Krinke and Silvia Breu. Control-Flow-Graph-Based Aspect Mining. In *Workshop on Aspect Reverse Engineering (WARE)*, 2004.

[22] Carla Laffra. Dijkstra's Shortest Path Algorithm.
http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/
DijkstraApplet.html.

[23] Karl J. Lieberherr. Component Enhancement: An Adaptive Reusability Mechanism for Groups of Collaborating Classes. In J. van Leeuwen, editor, *Information Processing '92, 12th World Computer Congress*, pages 179–185, Madrid, Spain, 1992. Elsevier.

[24] Sayyed Garba Maisikeli. *Aspect mining using self-organizing maps with method level dynamic software metrics as input vectors.* PhD thesis, 2009.

[25] M. Marin. Refactoring JHotDraws Undo concern to Aspectj. In *Proceedings of the First Workshop on Aspect Reverse Engineering (WARE)*, 2004.

[26] Marius Marin, Arie van, Deursen, and Leon Moonen. Identifying Aspects Using Fan-in Analysis. In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004).*, pages 132–141. IEEE Computer Society, 2004.

[27] Kim Mens, Andy Kellens, and Jens Krinke. Pitfalls in Aspect Mining. In *Proceedings of the 2008 15th Working Conference on Reverse Engineering*, WCRE '08, pages 113–122, Washington, DC, USA, 2008. IEEE Computer Society.

[28] Grigoreta Sofia Moldovan and Gabriela Serban. Aspect Mining using a Vector-Space Model Based Clustering Approach. In *Proceedings of Linking Aspect Technology and Evolution (LATE) Workshop*, pages 36–40, Bonn, Germany, March, 20 2006. AOSD'06.

[29] M. P. Monteiro and J. M. Fernandes. Towards a catalog of aspect-oriented refactorings. In *Proceedings of the 4th international conference on Aspect-oriented software development*, pages 111–122, 2005.

[30] Orlando Alejo Mendez Morales. Aspect Mining Using Clone Detection. Master's thesis, Delft University of Technology, The Netherlands, August 2004.

[31] Rajeswari Rajagopolan and Kris De Volder. A Query Based Browser Model. Master's thesis, University of British Columbia, Canada, July 2002.

[32] Renata Rand Mcfadden. *Aspect mining using model-based clustering.* PhD thesis, 2011. AAI3445077.

[33] Martin P. Robillard and Gail C. Murphy. Concern Graphs: Finding and Describing Concerns Using Structural Program Dependencies. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 406–416, 2002.

[34] Américo Sampaio, Neil Loughran, Awais Rashid, and Paul Rayson. Mining Aspects in Requirements. In *Early Aspects 2005: Aspect-Oriented Requirements Engineering and Architecture Design Workshop (held with AOSD 2005)*, Chicago, Illinois, USA, 2005.

[35] Gabriela Serban and Grigoreta Sofia Moldovan. A Graph Algorithm for Identification of Crosscutting Concerns. *Studia Universitatis Babes-Bolyai, Informatica*, LI(2):53–60, 2006.

[36] David Shepherd, Emily Gibson, and Lori Pollock. Design and Evaluation of an Automated Aspect Mining Tool. In *2004 International Conference on Software Engineering and Practice*, pages 601–607. IEEE, June 2004.

[37] David Shepherd and Lori Pollock. Interfaces, Aspects, and Views. In *Proceedings of Linking Aspect Technology and Evolution Workshop(LATE 2005)*, March 2005.

[38] Subject oriented programming. http://www.research.ibm.com/sop/.

[39] Maximilian Störzer, Uli Eibauer, and Stefan Schöffmann. Aspect mining for aspect refactoring: An experience report. In *Towards Evaluation of Aspect Mining*, Nantes, France, July 2006. at ECOOP 2006.

[40] Paolo Tonella and Mariano Ceccato. Aspect Mining through the Formal Concept Analysis of Execution Traces. In *Proceedings of the IEEE Eleventh Working Conference on Reverse Engineering (WCRE 2004)*, pages 112–121, November 2004.

[41] Tom Tourwé and Kim Mens. Mining Aspectual Views using Formal Concept Analysis. In *SCAM '04: Proceedings of the Source Code Analysis and Manipulation, Fourth IEEE*

International Workshop on (SCAM'04), pages 97–106, Washington, DC, USA, 2004. IEEE Computer Society.

[42] Santiago Vidal, Esteban S. Abait, Claudia Marcos, Sandra Casas, and J. Andrés Díaz Pace. Aspect mining meets rule-based refactoring. In *Proceedings of the 1st Workshop on Linking Aspect Technology and Evolution*, PLATE '09, pages 23–27, New York, NY, USA, 2009. ACM.

[43] Isaac Yuen and Martin P. Robillard. Bridging the gap between aspect mining and refactoring. In *Proceedings of the 3rd workshop on Linking aspect technology and evolution*, LATE '07, New York, NY, USA, 2007. ACM.

[44] Charles Zhang, Gilbert Gao, and Arno Jacobsen. Multi Visualizer. http://www.eecg.utoronto.ca/ czhang/amtex/.

[45] Charles Zhang and Hans-Arno Jacobsen. PRISM is Research In aSpect Mining. In *OOPSLA*, Vancouver, British Columbia, Canada, 2004. ACM Press.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: `grigo@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

HOW THE KERNELS CAN INFLUENCE IMAGE CLASSIFICATION PERFORMANCE

LAURA DIOŞAN(1) AND ALEXANDRINA ROGOZAN(2)

Abstract. Support Vector Machines deliver state-of-the-art performance in real-world applications and are now established as one of the standard tools for machine learning and data mining. A key problem of these methods is how to choose an optimal kernel and how to optimise its parameters. Selection of the most appropriate kernel highly depends on the problem at hand and fine tuning its parameters can easily become a tiresome and awkward task. Our purpose is to investigate how the used kernels and their parameters influence the learning performance in the context of a particular classification task: object recognition. The numerical results indicate that the best kernel function depends on the problem to be solved.

1. Introduction

The performance of a classification algorithm is strongly influenced by two ingredients: first, a suitable representation of the objects to be categorized and second a powerful decision maker algorithm on top of this representation. Even if the first aspect has been deeply discussed and analysed in the community of computer vision and the second one inside the machine learning community, the combination of them (how to combine the image representation with a learning algorithm) is still a highly challenging problem.

One of the most important issues in computer vision is how to extract relevant image features. The features that characterise an image can be classified from many points of view. An important criterion is the area from that the feature is extracted (global features and local feature). Another important criterion is the complexity of extraction (low-level features or high level features).

Received by the editors: November 25, 2012.

2000 *Mathematics Subject Classification.* 68T05,91E45.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Concept learning*.

Key words and phrases. Object recognition, Kernel descriptors, Support Vector Machines, Kernel selection.

Furthermore that to extract these features from an image, it is very important to store all these characteristics in an adequate representation such that a learning algorithm can work with them in order to label the corresponding images by a particular class. The most efficient representations are: bag of words [21] and kernels of local features [8].

The second aspect that must be considered when the problem of object recognition has to be solved is the classification algorithm. Since the classification must be performed in an automatically manner, a machine learning algorithm can be utilised. The general problem of machine learning is to search a, usually very large, space of potential hypotheses to determine the one that will best fit the data and any prior knowledge. In supervised image classification, we are given a training set of images and their corresponding labels. The goal is to learn (based on the training set) a classifier to label unseen images.

There are many learning algorithms today and their performances (estimated by different measures, e.g. classification accuracy, solution correctness, solution quality or speed of performance) are related not only to the problem to be solved, but also to their parameters. Therefore, the best results can be achieved only by identifying the optimal values of these parameters. Although this is a very complex task, different optimisation methods have been developed in order to optimise the parameters of Machine Learning algorithms.

In this paper we present a short survey of the most important image features and how they can be involved in a particular representation suitable for an automatically learning process based on kernel methods, since the kernel-based methods have been proved to reach good efficiency in solving such type of problems (with a particular emphasis of Support Vector Machines (SVMs) [23, 24]). Furthermore, we will compare the classification performance by taking into account different image representations and different kernels involved in SVM learning. In fact, we will develop a process of kernel selection at two levels:

- image processing level - at this stage we try to identify which is the most suitable kernel patch descriptors and its kernel in order to transform an image into an efficient and representative vector of features;
- learning level - here we try to identify the kernel involved in the decision process with the highest performance; the kernel is utilised to map the input of the classification algorithm from a non-linear separable space into a linear separable one.

The paper is organized as follows: Section 2 outlines the most important image features and representations, while Section 3 outlines the theory behind SVM classifiers giving a particular emphasis to the kernel functions. This is followed by Section 4 where our study case and the results of the experiments are presented and discussed. Finally, Section 5 concludes the paper.

2. Image representation

A highly challenging problem in computer vision is how to extract relevant image features. The features that characterise an image can be classified from many points of view. An important criterion is the area from that the feature is extracted: if the entire image is used, then some global features are computed, while if one or more image regions (patches) are utilised, then local features are determined. Another important criterion is the complexity of extraction. Image features can be extracted from scratch that means the features are extracted directly from the image (and in this case we discuss about low-level features) or can be computed based on some previously extracted features (in this case high-level features are obtained).

Examples of low-level features or image descriptors include: characteristics extracted from an image (global feature) or from one or more patches of an image (local features) around salient interest points or regular grids, characteristics regarding the edges, corners or blobs from the image/patches. The most popular (due to their success) low-level image descriptors are:

- orientation histograms such as Scale-Invariant Feature Transform (SIFT) [15] - based on a rectangular grid [12];
- Gradient Location and Orientation Histogram (GLOH) [17] - based on a log-polar grid [12];
- Histogram of Oriented Gradients (HOG) [6] - based on a radial grid [12];
- Speeded-Up Robust Features (SURF) and Haar [12];

while one of the best high-level descriptor is the kernel view of orientation histograms [1]. The results presented by L. Bo in [1] indicate that the performance of an image classifier based on the family of kernel descriptors surpasses the performance of a classifier that works only with low-level features (for instance a classifier based on SIFT features or only on HOG). In their work, the authors have indicated that the kernel descriptors are able to convert the pixel attributes into compact patch-level features. For pixel attributes the authors have considered the orientation, the position and the colour of each pixel (from a patch). Furthermore, the authors have introduced three types of match kernels to measure similarities between image patches. They have introduced these new similarity measures since the previous ones (based on HOG) suffer from discretisation (HOG can be considered a special case of linear kernels, but with a restrictive set of values – 1 or 0). The investigated match kernels from [1] are:

- the gradient match kernel (able to capture image variations) based on a kernel of magnitudes, an orientation kernel and a position kernel;
- the colour kernel (able to describe image appearance) based on a colour kernel and a position kernel;
- the local binary pattern kernel (able to capture local shape more effectively) based on a kernel of standard deviations of neighbour pixels, a kernel of binarized pixel value differences in a local window and a position kernel.

Furthermore that to extract these features from an image, it is very important to store all these characteristics in an adequate representation such that a learning algorithm can work with them in order to label the corresponding images by a particular class. These features extracted from an image can be used directly or indirectly by a classification algorithm. In the first case, the image features are simply concatenated into vectors that are utilised as input data by the classifier. The main drawback of such approach is the length of input vectors (sometimes it can be huge and the computational cost of the learning process is very large). Therefore, other efficient representations of image features have been identified and two of them are:

- Bag of words [21] - each local feature is represented with the closest visual word (from a predefined visual vocabulary) and a histogram is computed by counting the occurrence frequencies of words in the entire image (global representation). This histogram is actually used as image descriptor by a learning/recognition algorithm.
- Kernels over local features [8] – a kernel function is required in order to compare two images by using the previously extracted features (local descriptors). The local features are mapped into a low dimensional space by using kernel functions on sets such as:
  - sum match kernel [11] - adds all kernels corresponding to all combinations of local features extracted from two images;
  - neighbourhood match kernel [18] - similar to the previous one, but take into account the spatial location of local features also;
  - pyramid match kernels [9, 13, 14] - the local features are transformed into a multi-resolution histogram;
  - efficient match kernels [2] - set-level features are constructed by averaging the resulting feature vectors.

3. Learning algorithm

The general problem of Machine Learning is to search a, usually very large, space of potential hypotheses to determine the one that will best fit the data and any prior knowledge. In 1995, SVMs marked the beginning of a new era in the paradigm of learning from examples. Rooted to the Statistical Learning Theory and the Structural Risk Minimization principle developed by Vladimir Vapnik at AT&T in 1963 [23, 24], SVMs gained quickly attention from the Machine Learning community due to a number of theoretical and computational merits.

SVMs are a group of supervised learning methods that can be applied to classification or regression. SVMs arose from statistical learning theory; the aim being to solve only the problem of interest without solving a more difficult problem as an intermediate step. SVMs are based on the structural risk minimisation principle, closely related to regularisation theory. This principle incorporates capacity control to prevent over-fitting and thus is a partial solution to the bias-variance trade-off dilemma.

One issue with SVMs is finding an appropriate positive definite kernel (and its parameters) for the given data. A wide choice of kernels already exists. Many data or applications may still benefit from the design of particular kernels, adapted specifically to a given task (i.e. kernels for vectors, kernels for strings, kernels for graphs, Fisher kernels or rational kernels). There are only some hints for working with one or another of these classic kernels, because there is no rigorous methodology to choose *a priori* the appropriate one between them. Moreover, the kernel parameters influence the performance of the SVM algorithm. The selection of the penalty error for an SVM (that controls the trade-off between maximizing the margin and classifying without error) is also critical in order to obtain good performances. Therefore, one has to optimise the kernel function, the kernel parameters and the penalty error of the SVM algorithm in order to guarantee the robustness and the accuracy of an SVM algorithm. Chapelle [4] has proposed to denote the kernel and SVM parameters as hyper-parameters.

3.1. **Generalities of SVM.** Initially, SVM algorithm has been proposed in order to solve binary classification problems [23]. Later, these algorithms have been generalized for multi-classes problems. Consequently, we will explain the theory behind SVM only on binary-labelled data.

Suppose the training data has the following form: $D = (x_i, y_i)_{i=\overline{1,m}}$, where $x_i \in \Re^d$ represents an input vector and each y_i, $y_i \in \{-1, +1\}$, the output label associated to the item x_i. SVM algorithm maps the input vectors to a higher dimensional space where a maximal separating hyper-plane is constructed [23]. Learning the SVM means to minimize the norm of the weight vector (w in Eq. (1)) under the constraint that the training items of different classes belong to opposite sides of the separating hyper-plane. Since $y_i \in \{-1, +1\}$ we can formulate this constraint as:

$$y_i(w^T x_i + b) \geq 1, \ \forall i \in \{1, 2, \ldots, m\}, \tag{1}$$

where the primal decision variables w and b define the separating hyper-plane and v^T represents the transpose of v.

The items that satisfy Eq. (1) with equality are called support vectors since they define the resulting maximum-margin hyper-planes. To account for misclassification, e.g. items that do not satisfy Eq. (1), the soft margin formulation of SVM has introduced some slack variables $\xi_i \in \Re$ [5].

Moreover, the separation surface has to be nonlinear in many classification problems. SVM was extended to handle nonlinear separation surfaces by using a feature function $\phi(x)$. The SVM extension to nonlinear datasets is based on mapping the input variables into a feature space $\mathcal{F}$ of a higher dimension and then performing a linear classification in that higher dimensional space. The important property of this new space is that the data set mapped by ϕ might become linearly separable if an appropriate feature function is used, even when that data set is not linearly separable in the original space.

Hence, to construct a maximal margin classifier one has to solve the convex quadratic programming problem encoded by Eq. (2), which is the primal formulation of it:

$$\begin{array}{ll} \text{minimise}_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^m \xi_i & \\ \text{subject to:} & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \forall i \in \{1, 2, \ldots, m\}. \end{array} \tag{2}$$

The coefficient C (usually called *penalty error* or *regularization parameter*) is a tuning parameter that controls the trade off between maximizing the margin and classifying without error. Larger values of C might lead to linear functions with smaller margin, allowing to classify more examples correctly with strong confidence. A proper choice of this parameter is crucial for SVM to achieve good classification performance.

Instead of solving Eq. (2) directly, it is a common practice to solve its dual problem, which is described by Eq. (3):

$$\begin{array}{ll} \text{maximise}_{a \in \Re^m} \sum_{i=1}^m a_i - \frac{1}{2} \sum_{i,j=1}^m a_i a_j y_i y_j \phi(x_i)^T \phi(x_j) & \\ \text{subject to} \quad \sum_{i=1}^m a_i y_i = 0, & \\ \qquad\qquad 0 \leq a_i \leq C, \forall i \in \{1, 2, \ldots, m\}. & \end{array} \tag{3}$$

In Eq. (3), a_i denotes the Lagrange variable for the i^{th} constraint of Eq. (2).

The optimal separating hyper-plane $f(x) = w \cdot \phi(x) + b$, where w and b are determined by Eq. (2) or Eq. (3) is used to classify the un-labelled input data x_k:

$$y_k = sign\left(\sum_{x_i \in S} a_i \phi(x_i)^T \phi(x_k) + b \right), \tag{4}$$

where S represents the set of support vector items x_i.

We will see in the next section that is more convenient to use a kernel function $K(x, z)$ instead of the dot product $\phi(x)^T\phi(z)$.

3.2. **Kernel formalism.** The original optimal hyper-plane algorithm proposed by Vapnik in 1963 was a linear classifier [23]. However, in 1992, Boser, Guyon and Vapnik [3] have suggested a way to create non-linear classifiers by applying the *kernel trick*. Kernel methods work by mapping the data items into a high-dimensional vector space $\mathcal{F}$, called feature space, where the separating hyper-plane has to be found [3]. This mapping is implicitly defined by specifying an inner product for the feature space via a positive semi-definite kernel function: $K(x, z) = \phi(x)^T\phi(z)$, where $\phi(x)$ and $\phi(z)$ are the transformed data items x and z [20]. Note that all we required is the result of such an inner product. Therefore we do even not need to have an explicit representation of the mapping ϕ, neither to know the nature of the feature space. The only requirement is to be able to evaluate the kernel function on all the pairs of data items, which is much easier than computing the coordinates of those items in the feature space.

The kernels that correspond to a space embedded with a dot product belong to the class of positive definite kernels. This has far-reaching consequences. The positive definite and symmetric kernels verify the Mercer's theorem [16] - a condition that guarantees the convergence of training for discriminant classification algorithms such as SVMs. The kernels of this kind can be evaluated efficiently even though they correspond to dot products in infinite dimensional dot product spaces. In such cases, the substitution of the dot product with the kernel function is called the *kernel trick* [3].

In order to obtain an SVM classifier with kernels one has to solve the following optimization problem:

$$\begin{aligned} &\text{maximise}_{a\in\Re^m} \textstyle\sum_{i=1}^m a_i - \frac{1}{2}\sum_{i,j=1}^m a_i a_j y_i y_j K(x_i, x_j) \\ &\text{subject to} \quad \textstyle\sum_{i=1}^m a_i y_i = 0, \\ &\qquad\qquad\quad 0 \le a_i \le C, \forall i \in \{1, 2, \ldots, m\}. \end{aligned} \tag{5}$$

In this case, Eq. (4) becomes:

$$y_k = sign\left(\sum_{x_i\in S} a_i K(x_i, x_k) + b\right), \tag{6}$$

where S represents the set of support vector items x_i.

There are a wide choice for a positive definite and symmetric kernel K from Eq. (6). The selection of a kernel has to be guided by the problem that must be solved. Choosing a suitable kernel function for SVMs is a very important step for the learning process. There are few if any systematic techniques to assist in this choice. Until now, different kernels for vectors have been proposed [22]; the most utilised of them by an SVM algorithm are listed in Table 1.

TABLE 1. The expression of several classic kernels.

Name	**Expression**	**Type**
Linear	$K_{Lin}\ \ (x,z) = x^T \cdot z$	projective
Polynomial	$K_{Pol}\ \ (x,z) = (x^T \cdot z + coef)^d$	projective
Normalised Polynomial	$K_{NPol}(x,z) = \frac{K_{Pol}(x,z)}{\sqrt{K_{Pol}(x,x)K_{Pol}(z,z)}}$	projective
Laplacian	$K_{Lapl}(x,z) = exp(-\frac{\lvert x-z\rvert}{\sigma})$	radial
Exponentail	$K_{Exp}(x,z) = exp(-\frac{\lvert x-z\rvert}{2\sigma^2})$	radial
Gaussian	$K_{Gauss}(x,z) = exp(-\frac{\lvert x-z\rvert^2}{2\sigma^2})$	radial
Euclidean	$K_{Euclid}(x,z) = \frac{\lvert x-z\rvert^2}{2\sigma^2}$	radial

While one of the first feelings about SVM algorithms is that they can solve a learning task automatically, it actually remains challenging to apply SVMs in a fully automatic manner. Questions regarding the choice of the kernel function and the hyper-parameters values remain largely empirical in real-world applications. While default setting and parameters are generally useful as a starting point, major improvements can result from careful choosing of an optimal kernel. There are many types of kernels for vectors and several criteria could be used for classifying them.

While SVM classifiers intrinsically account for a trade off between model complexity and classification accuracy [24], the generalization performance is still highly dependent on appropriate selection of the penalty error C and kernel parameters. Thus, several methods could be used to optimise the hyper-parameters of an SVM classifier.

Ideally, we would like to choose the value of the kernel parameters that minimise the true risk of the SVM classifier. Unfortunately, since this quantity is not accessible, one has to build estimates or bounds for it.

Cross-validation is a popular technique for estimating the generalization error and there are several interpretations [25]. In k-fold cross-validation, the training data is randomly split into k mutually exclusive subsets (or folds) of approximately equal size. The SVM decision rule is obtained by using $k-1$ subsets on training data and then tested on the subset left out. This procedure is repeated k times and in this manner each subset is used for testing once. Averaging the test error over the k trials gives a better estimate of the expected generalization error.

4. Study case

4.1. **Proposed framework.** Our aim is to investigate how the kernel function influences the performance of learning process. Therefore, we considered the framework proposed by L. Bo [1] for image classification and we test different kernel functions. We already establish that the selection of the kernel function is very important for SVM (see [7]). This time, our investigation about how kernel function affects classification process is developed at two levels:

- at the level of image descriptor and
- at the level of learning process.

In the first case, based on the available code of Kernel descriptors developed by Xiaofeng Ren (*http : //www.cs.washington.edu/ai/Mobile_Robotics /projects/kdes/*), we have tested different kernels when the local features are extracted from an image. Because we work only with gray images, we investigate only the kernels descriptors able to capture image variations (gradient match kernels [1]). As we already presented in Section 2, the Bo's gradient match kernel is composed by three kernels: a kernel of magnitudes, an orientation and a position kernel.

The magnitude kernel is a linear one and its role is to measure the similarity of gradient magnitudes of two pixels. The magnitude kernel type cannot be changes since it must be an equivalent of histogram of gradients in the feature map (a pixel has associated a feature vector obtained by multiplying the magnitude and the orientation of a pixel over all considered orientation bins).

The other two kernels involved in Ren's computation of the gradient match kernel, the orientation kernel (for computing the similarity of gradient orientations) and the position kernel (for measuring how close two pixels are spatially), are actually Gaussian kernels. Therefore, we have changes the implementation and we have involved in the feature extraction process more possible orientation and position kernels (Exponential, Laplacian, Euclidean). The expression of these kernels is given in Table 1.

In the second case, that of learning, the classifier utilised in our experiments is a kernel-based SVM. Again, we have tried to use several kernels with different parameters during the learning process in order to identify the best one. These kernels are the linear kernel, the Polynomial kernel, the Gaussian kernel and the Normalised Polynomial kernel. For the Polynomial kernel several exponents have been tested (2, 3), for parameter $\frac{1}{2\sigma^2}$ of Gaussian kernel the following values have been checked: 0.1, 0.01, 0.001, 0.0001 and for Normalised Polynomial kernel the exponent was 2.

The dual version of the optimisation problem which arises during the training of support vector machines was solved by Sequential minimal optimization (SMO) algorithm [19], since it is able to quickly solved the quadratic programming optimisation problem of SVM. We have chosen this formulation of SVM since the duality theory provides a convenient way to deal with the constraints and, in this form, the optimisation problem can be solved in terms of dot products, that allows using the kernel trick. Furthermore, SMO requires an amount of memory that increases only linear with the training set size, being able to handle very large training sets - as in the image classification case. These aspects are different to L. Bo's framework that is based on primal formulation of SVM and on conjugate gradient optimisation methods (in fact, Newton optimisation).

4.2. **Numerical experiments.** Several numerical experiments about how kernel selection influences the classification process in the case of image recognition task are presented. A benchmark (http://www.cs.unc.edu/ lazebnik) was considered in our study-case. More details about these data can be found in [14].

For all datasets a binary classification problem was actually solved: D1 corresponds to a classification between bedroom and kitchen images, D2 corresponds to a decision coast images *vs.* forest images, while in D3 we have to delimitate industrial scenes to suburban scenes. In all the cases, 50 images are utilised (the decision model is trained on 2/3 of them, while 1/3 of images are used for testing). All the experiments are performed by using a cross-validation technique of 3 folds and they are performed by using the Weka tool [10].

In order to measure the classification performance, the accuracy rate was actually computed. The accuracy rate represents the number of correctly classified items over the total number of items from a given data set.

In Tables 2, 3, 4 are presented the average accuracy rates for each dataset by considering different image descriptor kernels (when the SVM input vectors are actually constructed) and different SVM kernel functions for the first dataset (D1), the second dataset (D2) and the third dataset (D3), respectively.

Several remarks cab ne done based on the results from Tables 2, 3, 4. Regarding the kernel descriptors, the Gaussian kernel is the best one in 2 cases (D2 and D3), but for the first data the Exponential kernel perform better. Furthermore, if we consider all combinations, the Exponential and the Gaussian kernel have won 9 times each in terms of accuracy rate. Taking into account the computation time required for evaluating the kernel expression, we promote to use the exponential kernel (since involve just a simple norm, not a square one – see Table 1).

SVM kernel *vs.* Kernel Descriptor	Exponential	Gaussian	Laplacian	Euclid
Lin	74%	69%	65%	45%
Poly(2)	70%	67%	62%	50%
Poly(3)	59%	67%	55%	53%
Gaussian(0.1)	51%	55%	48%	46%
Gaussian(0.01)	50%	52%	43%	49%
Gaussian(0.001)	54%	54%	40%	42%
Gaussian(0.0001)	49%	49%	38%	53%
NormPoly(2)	49%	51%	45%	54%

TABLE 2. Accuracy rates obtained for dataset D1 by SVM algorithm with different kernel functions on images represented by different kernel descriptors.

SVM kernel *vs.* Kernel Descriptor	Exponential	Gaussian	Laplacian	Euclid
Lin	97%	99%	98%	66%
Poly(2)	94%	96%	96%	67%
Poly(3)	80%	80%	95%	63%
Gaussian(0.1)	83%	83%	85%	59%
Gaussian(0.01)	80%	80%	72%	60%
Gaussian(0.001)	57%	57%	72%	63%
Gaussian(0.0001)	56%	56%	75%	51%
NormPoly(2)	62%	62%	64%	49%

TABLE 3. Accuracy rates obtained for dataset D2 by SVM algorithm with different kernel functions on images represented by different kernel descriptors.

SVM kernel *vs.* Kernel Descriptor	Exponential	Gaussian	Laplacian	Euclid
Lin	79%	88%	80%	59%
Poly(2)	83%	80%	77%	57%
Poly(3)	68%	65%	66%	54%
Gaussian(0.1)	65%	63%	68%	44%
Gaussian(0.01)	60%	63%	56%	52%
Gaussian(0.001)	65%	51%	60%	54%
Gaussian(0.0001)	67%	57%	63%	49%
NormPoly(2)	57%	49%	57%	53%

TABLE 4. Accuracy rates obtained for dataset D3 by SVM algorithm with different kernel functions on images represented by different kernel descriptors.

Regarding the SVM kernels, the best results are obtained for all datasets by using a linear kernel (that means the data can be simple separate by a hyper plane, without requiring hyper spheres or conjunction of feature as in the case of Gaussian and Polynomial kernel, respectively).

5. Conclusions

An important problem was investigated in this paper: how kernel functions can affect the performance of object recognition process. In fact, the kernel functions are involved at two levels: that of extraction of image features and that of learning method. The most promising kernel function involved in the classification algorithm seems to be the simple linear one, while in the case of kernel descriptors (that extract image features) the results indicate that we cannot identify a best kernel. Each problem seems to be solved better with other kernel type. Therefore, we plan to investigate how we can automatically adapt the kernel descriptor (and its parameters) to a given set of images in order to increase the classification performance, but without reducing the generality and the extrapolation power of the method. Furthermore, other match kernels (colour and shape kernels) can be considered.

References

[1] Bo, L., Ren, X., and Fox, D. Kernel descriptors for visual recognition. In *NIPS* (2010), J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds., Curran Associates, Inc, pp. 244–252.

[2] Bo, L., and Sminchisescu, C. Efficient match kernel between sets of features for visual recognition. In *NIPS* (2009), Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds., Curran Associates, Inc, pp. 135–143.

[3] Boser, B. E., Guyon, I., and Vapnik, V. A training algorithm for optimal margin classifiers. In *COLT* (1992), D. Haussler, Ed., ACM Press, pp. 144–152.

[4] Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. Choosing multiple parameters for Support Vector Machines. *Machine Learning 46*, 1/3 (2002), 131–159.

[5] Cortes, C., and Vapnik, V. Support-Vector Networks. *Machine Learning 20*, 3 (1995), 273–297.

[6] Dalal, N., and Triggs, B. Histograms of Oriented Gradients for human detection. In *CVPR* (2005), C. Schmid, S. Soatto, and C. Tomasi, Eds., vol. 2, pp. 886–893.

[7] Diosan, L., Rogozan, A., and Pecuchet, J.-P. Improving classification performance of Support Vector Machine by genetically optimisation of kernel shape and hyper-parameters. *Applied Intelligence 2*, 36 (2012), 280–294.

[8] Eichhorn, J., and Chapelle, O. Object categorization with svm: Kernels for local features. Tech. rep., In Advances in Neural Information Processing Systems, 2004.

[9] Grauman, K., and Darrell, T. J. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV* (2005), pp. II: 1458–1465.

[10] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *SIGKDD Explorations 11*, 1 (2009), 10–18.

[11] Haussler, D. Convolution kernels on discrete structure. Tech. Rep. UCSC-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, July 1999.

[12] Johnson, M. Generalized descriptor compression for storage and matching. In *BMVC* (2010), F. Labrosse, R. Zwiggelaar, Y. Liu, and B. Tiddeman, Eds., British Machine Vision Association, pp. 1–11.

[13] Kumar, A., and Sminchisescu, C. Support kernel machines for object recognition. In *ICCV* (2007), pp. 1–8.

[14] Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR* (2006), pp. II: 2169–2178.

[15] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[16] Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society 83*, 559 (1909), 415–446.

[17] Mikolajczyk, K., and Schmid, C. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell 27*, 10 (2005), 1615–1630.

[18] Parsana, M., Bhattacharya, S., Bhattacharyya, C., and Ramakrishnan, K. R. Kernels on attributed pointsets with applications. In *NIPS* (2007), J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds., Curran Associates, Inc, pp. 1129–1136.

[19] Platt, J. Fast training of Support Vector Machines using Sequential Minimal Optimization. In *Advances in Kernel Methods — Support Vector Learning* (Cambridge, MA, 1999), B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., MIT Press, pp. 185–208.

[20] Schölkopf, B. The kernel trick for distances. In *NIPS* (Cambridge, MA, 2000), T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., MIT Press, pp. 301–307.

[21] Sivic, J., and Zisserman, A. Video google: A text retrieval approach to object matching in videos. In *ICCV* (2003), pp. 1470–1477.

[22] Taylor, J. S., and Cristianini, N. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[23] Vapnik, V. *The Nature of Statistical Learning Theory.* Springer, 1995.

[24] Vapnik, V. *Statistical Learning Theory.* Wiley, 1998.

[25] Wahba, G., Lin, Y., and Zhang, H. GACV for Support Vector Machines. In *Advances in Large Margin Classifiers*, B. Smola and S. SchRolkopf, Eds. MIT Press, Cambridge, MA, 1999.

(1) Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: lauras@cs.ubbcluj.ro

(2) LITIS, EA - 4108, INSA, Rouen, France
E-mail address: arogozan@insa-rouen.fr

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

BUILDING INTELLIGENT KNOWLEDGE MANAGEMENT SYSTEMS

OVIDIU PÂRVU AND SANDA M. DRAGOŞ

Abstract. Knowledge management systems are used in most of the large companies and institutions throughout the world. Currently, most of them are not designed to adapt to the dynamic changes of the surrounding environment. Intelligent plugins can be integrated in these systems in order to increase their reactivity and adaptability. In the present paper, we describe an intelligent plugin for a knowledge management system (KMS) used for managing the PhD theses in the Babes-Bolyai University. The KMS which uses our plugin is called Doctorate Theses Administration Platform (DTAP) [9, 10]. Our plugin uses an intelligent agent in order to help users go through all the required steps of handling a thesis. The behaviour of the application changes dynamically based on the interaction with the users. Results show that both the time needed to manage the theses and number of encountered errors were reduced considerably after the integration of the above mentioned components.

1. Introduction

The term *Knowledge Management System* has many definitions in the literature, but in the present paper we will use only the meaning indicated in [2, 11]: "The KMS represents a class of information systems (ISs) applied to managing organizational knowledge". Simply stated, these systems are used for managing the knowledge of an organization. The collective knowledge of the organization is stored either in a repository or in an archive of documents. At micro level everyone in the organization is contributing to this database, so it can be seen as a dynamic entity which is continuously evolving over

Received by the editors: November 12, 2012.

2010 *Mathematics Subject Classification.* 68T05.

1998 *CR Categories and Descriptors.* J.1 [**Computer Applications**]: Administrative Data Processing – *Education*; I.2.1 [**Computing Methodologies**]: Artificial Intelligence – *Applications and Expert Systems*; H.3.4 [**Information Systems**]: Information Storage and Retrieval – *Knowledge Management Systems and Expert Systems*.

Key words and phrases. knowledge management system, education, reinforcement learning, templating system.

time. A KMS eases the management of the information by using its built in mechanisms.

According to [1, 2] the most important functions of such a system are: *creation*, *storage* and *retrieval*, *transfer* and *application of knowledge*.

The *creation* of knowledge is accomplished by developing new knowledge or replacing existing one from the repository. The KMS includes a repository which allows sharing, creating, amplifying and enlarging the organizational knowledge [2]. Everyone has to collaborate and participate to the contents of the database in order for it to expand. Whether it is a document, a procedure or a new standard, it is important for it to be integrated in the system which every colleague can access.

This brings us to the second function of *storing* (and *retrieving*) data from the repository of the KMS. These are the basic operations which must be carried out in order for the information exchange to take place. Simple file sharing is not enough and specific tools which aid collaboration should be provided. These tools can include database interrogation forms, a logging system which records everything that has been done and maybe also small software modules which have an intelligent tutor integrated. Using these tools, even inexperienced employees can retrieve information from the database in a format which is easy to understand. The ones who share or store information must find the experience both professionally and personally rewarding, such that they remain motivated to use the system [12]. Some companies use rewarding systems to motivate experienced employees to take the time to document their work even if they may not ever need to make use of such knowledge. In other companies, it is part of the job description to write small tutorials and guides, such that everyone has to do it. There are also companies in which every modification is logged, such that history records are generated automatically and can be used as samples for good/bad practices.

A KMS, as its name suggests, is not about exchanging information, but knowledge. It is clear why there is a need for the third function which is the *transfer* of knowledge between employees. In order for the exchange to be done in an optimal way, the system must provide an efficient communication support not only between individuals, but also between different departments within the same company. For instance, employees from one department may post a question on a forum and colleagues from all over the company may view and answer this question. Another useful utility is one that allows employees to check the changes done by other colleagues.

The aim of a KMS is that employees collaborate and *apply* in practice the gained knowledge in order to obtain better results than before.

2. Doctorate Theses Administration Platform

These mechanisms of a KMS can be applied when designing an application for managing the PhD theses in a university. We implemented such an application for the Babeş-Bolyai. It is called the Doctorate Theses Administration Platform and we will refer to it as DTAP in the rest of the paper. Previous to DTAP, the employees of the Institute of Doctoral Studies (IDS) were using obsolete and time consuming methods for managing PhD theses. This led to the implementation of this system. First of all, the information regarding the PhD students was archived on paper. Therefore, searching for a specific thesis or backing up the entire archive implied a lot of work. Secondly, all the documents required by the Romanian Educational system were filled in manually by the employees of the IDS using Microsoft Word templates. This was both time consuming and error prone. Finally, the information which had to be made public was written by hand for every thesis. Considering that the information was displayed taking into account different sorting criteria (e.g. title, author's name etc.), the information had to be duplicated for each one of these sorting criteria. DTAP addresses these issues by using the mechanisms described next.

First of all, it stores all the information regarding the PhD candidates into a database. Therefore, backing up the entire information implies backing up the database, which is not a difficult operation when using a database management system. Moreover, all the data which needs to be made public does not have to be written by hand anymore. Queries could retrieve the needed information from the database considering the sorting criteria chosen by the user. DTAP also includes a tool for generating reports as ".xls" (Microsoft Excel) files. These are very useful to the employees of IDS, because they must create annual reports regarding the performance of the University and its staff (e.g. Supervisor which coordinated most PhD theses in 2012).

Using the data regarding the PhD candidates and a set of templates stored in the database, the employees of the IDS can generate automatically all the documents required by the Romanian Ministry of Education (i.e. Adresă minister, Anunţ susţinere, Decizia rectorului, Listă documente, Proces verbal, Referat preliminar, Referat final). The procedure of generating the documents can be split in two steps. The first step is retrieving the required templates from the database. The second step is to parse the templates, replace variables from the template with their actual values and format certain parts of the template accordingly (e.g. bold, italic). The generated document is stored in a temporary folder until it is downloaded. Immediately after the user downloads it, the document is removed in order to reduce the memory usage. Due to lack of space the algorithm and architecture of the system are not presented here, but more details can be found in [9, 10]. We would like to emphasize the improvement obtained from using this system. The total time needed to generate the documents for a thesis prior to DTAP was 7 * 180 seconds = 1260 seconds, whereas the total time needed to generate the documents using DTAP is 7 * 3 seconds = 21 seconds. Thus, using DTAP the employees of the IDS will need only 1.66% of the time they required before to generate all the documents.

This paper, however, presents the integration of an intelligent plugin in the KMS. The intelligent agent helps the users of DTAP by offering suggestions regarding the steps that need to be followed when managing a thesis. Suggestions aim is to prevent human errors which could delay the submission of the thesis and the PhD students viva respectively.

In section 3 and 4 we will describe other approaches of implementing intelligent systems and how our own approach is different from the existing ones. Moreover, we will give a short introduction for reinforcement learning, the method we used for implementing the intelligent plugin in DTAP.

3. Intelligent Knowledge Management Systems

According to John McCarthy Artificial Intelligence is "*the science and engineering of making intelligent machines*". In order for these machines to fulfill their task, they have to interact with the surrounding environment. Learning from the interaction with the environment underlies all theories of learning and intelligence [13]. Such intelligent systems are used for developing adaptive educational systems. In our case, we have used the reinforcement learning (RL) strategy for implementing an intelligent educational agent.

RL is a class of solutions in which the problem to be solved is defined in terms of rewards and punishments [6]. An autonomous agent will learn based on the received rewards and punishments which actions to choose in order to reach its goal. These rewards and punishments are given by a trainer depending on the problem to solve. For instance, if the agent wants to learn how to play a game, the trainer can punish him for losing, reward him for winning and offer him no payoff for simply playing [8]. The task of the agent is to learn the most optimal strategy for reaching its goal.

Single agents or multi agent implementations using RL have been used for implementing various educational systems [3]. The difference between these two types of systems is the fact that in a single agent system the learning process is influenced only by the interaction with the environment, while in the multi agent system the learning process is influenced by all the other agents present in the system as well. There are multiple examples of educational systems which use RL in order to dynamically adapt their behavior to the changes of the environment.

Among these we would first like to mention the educational system RLATES [7]. In this case the pedagogical policy applied to one student is improved based on the experience of other students with similar characteristics. The experiments show that the system learns through a trial and error technique at the same time the students learn. One of the benefits of using RL is that it eliminates the problem of overtraining encountered at supervised learning.

On the other hand, Cordillera [5] implements a different learning strategy. Cordillera uses a method of RL which improves the efficiency of the system by inducing which policies should be used for the students. The system starts with a set of policies denoted as exploratory corpus. This exploratory corpus was collected while letting the system make random decisions while interacting with real students. The feedback received from the users determines which policy is the most appropriate one.

A different approach is presented in [4] where the end purpose of the system is to discover efficient/inefficient teaching strategies. The efficient teaching tactics can be applied by teachers in real classrooms while the inefficient ones should be avoided. This approach was used in the implementation of NormGain and InvNormGain [4] which learn what are the best and worst teaching strategies respectively through their interaction with the students.

The characteristic which all the above described software products share is the fact that they are directly addressing the needs of teachers and/or students.

4. Navibot

Our own approach is different than the ones presented above. The intelligent agent does not aid the students, but the employees of the IDS. Therefore, it is still an intelligent educational system, but students are no longer the main beneficiary. The current version of our implementation is a single agent system.

We implemented DTAP before designing the intelligent educational agent called Navibot. In order not to mingle with the source code of the original system, we chose to implement a plugin which is independent of DTAP and whose integration in the existing system requires minimal modifications.

The purpose of the plugin is to learn the sequence of actions executed by an employee of the IDS while managing a PhD thesis. Based on the acquired information, Navibot will make suggestions regarding the next action that should be executed considering the current state of the system. For instance, after inserting a thesis, documents a, b and c must be generated. Then, the information about the thesis is updated. In the end, documents d, e, f and g

can be generated as well. Keeping track of all these steps is not a big problem, but after doing this separately for every thesis every day, one may forget about generating a particular document or about updating a certain field of a thesis from time to time. It is in the nature of humans to commit errors, but preventing errors is always better than having to correct them. Hence, the purpose of Navibot is to help the users of DTAP manage the PhD theses without committing errors by pointing them the next step in the process of managing a thesis.

From the point of view of the user, the plugin is simple to use. The available use cases are the following:

- Display the window containing the hint.
- Hide the window containing the hint.
- Follow the action suggested by Navibot.

The plugin learns the sequence of actions dynamically while DTAP is being used. The benefit of this fact is that no supervisor is required to offer training data to the plugin in order for it to learn. Secondly, the plugin will automatically update its information at runtime when the sequence of actions changes.

As stated before, we used a RL algorithm for implementing the plugin. Navibot learns the optimal policy, the correct sequence of actions, from the feedback received for its suggestions.

The first step in designing such a plugin is to establish the set of states. In our case, the number of states used by the plugin is small, because only the relevant features of DTAP were considered. This decision was taken having in mind that this is a web application whose performance is influenced by the connectivity of the user to the Internet. Therefore, it was very important to keep the number of computations done by the plugin at a minimum. Otherwise, it would have affected the performance of DTAP and the plugin would no longer have been useful.

The set of states contains 10 elements, namely *Înregistrare teză (0)*, *Editare teză (1)*, *Vizualizare teze (2)*, *Generare Adresă minister (3)*, *Generare Anunţ susţinere (4)*, *Generare Decizia rectorului (5)*, *Generare listă documente (6)*, *Generare Proces verbal (7)*, *Generare Referat final (8)* and *Generare Referat preliminar (9)* where the string represents the name of the state and the number enclosed in parentheses its identifier. We associated to each state a unique identifier in order to use a simpler notation when referring to them.

The set of actions represents all the possible pages that a user of DTAP can visit. These actions will be displayed as links on the web pages. Users can access any of the above mentioned states. Therefore, the set of actions contains elements of the form "Go to state: s_i", where i represents the identifier of the state. For instance, one of the actions in the set is “Go to state: s_1” which is equivalent with “Go to page: Editare teză”.

Using the set of states and the set of actions we will define the set of transitions. There are no restrictions related to the navigation between the considered pages. Therefore, we can access any page from any other page of DTAP. Thus, the set of transitions contains all the pairs (s_i, s_j) where s_i, s_j belong to the set of states. An example of a transition would be (s_3, s_4) which can be expressed in natural language as follows. Starting from page “Generare Adresă minister” (s_3) go to page “Generare Anunţ susţinere” (s_4).

The learning process of the agent included in the plugin is driven forward by the received rewards. For each given suggestion which was wrong the agent receives a negative feedback. Otherwise, it receives a positive feedback. Therefore, the feedback, known in the literature as reward, can take either one of the following values:

$$r_k = \begin{cases} -0.5, \text{suggestion was wrong} \\ +1.0, \text{suggestion was correct} \end{cases}$$

Simply stated, the agent receives a return with the value “-0.5” for wrong hints and “+1.0” for correct ones. All the received rewards are cumulated in the return value “R” which is computed as follows:

$$R_k = \sum_{i=0}^{k} \gamma^{k-i} r_i$$

However, the updates are done at every step in order to reduce the space complexity of the plugin. Thus, the return value is updated at every step using the recurrence formula:

$$R_k = r_k + \gamma R_{k-1}$$

Experiments have shown that the plugin learns quicker when the value of γ is 0.9.

As the number of states is small our decision was to use the state-action value function. Therefore, the return value for each state-action pair (s, a) is stored. This information is stored in a matrix of dimensions N x N called *actions matrix*, where N represents the number of states. At the beginning of the algorithm every element of the matrix is initialized with the value 1, thus obtaining a matrix of 10 by 10 having all elements = 1.

In order to read the return of choosing the action starting from state 2 and which goes into state 4, the value of the cell with coordinates (3, 5) is read. Generally, in order to check which is the return of choosing the action starting from state i and which goes into state j, one must read the value of the cell with the coordinates *(i+1, j+1)*. One of the benefits of this implementation is the access to the return values in $O(1)$.

All the concepts presented above are implemented in the Navibot plugin. The class diagram of the plugin is presented in figure 1. In order to be able to reuse this structure for all plugins using a RL algorithm, the ReinforcementLearning, State and Action classes are abstract ones.

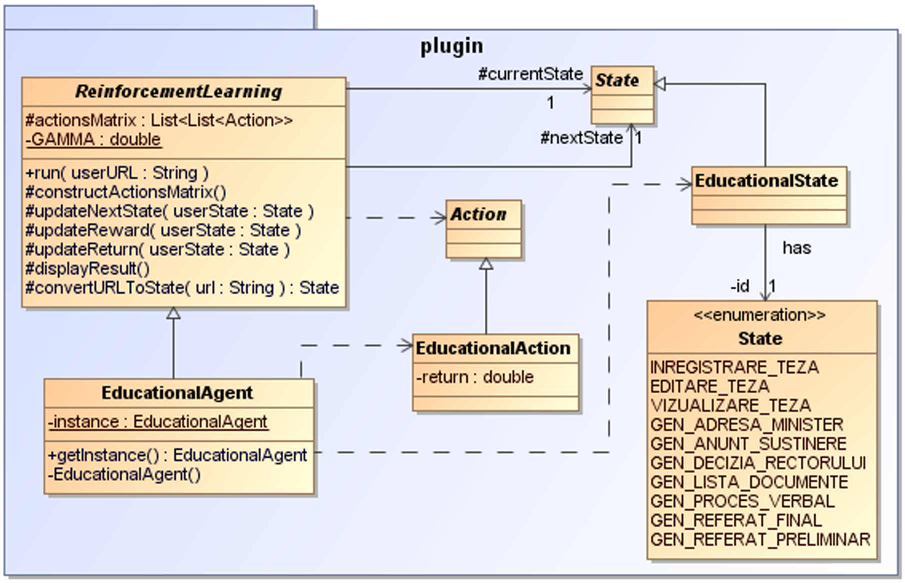


FIGURE 1. Navibot class diagram

The only public method defined in the ReinforcementLearning class is the method "run". This method describes the general behavior of a RL algorithm independent of the representation of the State and Action classes. The classes which are used for actually running the plugin are inheriting the abstract classes and overriding all abstract methods. In our case, these classes are EducationalAgent, EducationalState and EducationalAction.

The general public method "run" from the abstract class ReinforcementLearning is presented in the following using the Pseudocode notation. This method is called every time the user requests a new page. However, in order not to delay the delivery of the requested page to the user, the run method is called in an asynchronous manner using AJAX. As soon as the execution of the method "run" ends, the results are returned and the hint is displayed.

Algorithm 1 Public method *run*

```
 1: get the $current\ state$ of the application
 2: if $actionsMatrix$ is initialized then
 3:     get value of $reward$
 4:     update corresponding $return$
 5:     choose the $next\ state$ for the hint considering the maximum return
 6: else
 7:     construct $actionsMatrix$
 8:     initialize $next\ state$
 9: end if
10: display $next\ state$ as hint
```

An execution of the method "run" will be described in the following. The description is given from an algorithmic point of view which means we will refer to states and not to pages.

Let us assume that the user wants to visit state 1. Therefore, the current state will be set to 1 (line 1).

If the action matrix is initialized, then the value of the reward is computed (line 3). The reward is "-0.5" if the previous suggested state is different from the current state or "+1.0" if they are equal. The return value corresponding to the previous suggestion is updated (line 4) using the value of the reward. Afterwards, the next state has to be determined (line 5). This is done by accessing the row corresponding to state 1 from the actions matrix. According to the explanations given above, this is row 2. Each column from the matrix corresponds to one state and has 1 value in the selected row. The algorithm finds the column containing the maximum value from the row. The state corresponding to this column is chosen as the next state.

On the other hand, if the action matrix is uninitialized, then it is constructed (line 7) and the value of all the cells is set to 1. The next state is initialized with a *NULL* value (line 8).

In the end, the hint is displayed (line 10). If the next state is different from *NULL*, then this state is recommended to the user. Otherwise, the hint will display a message which states that the plugin is currently being initialized.

Next, we will present some results obtained using the Navibot plugin. In figure 2 we describe the number of requests needed to learn sequences of actions of variable length. We mention that the number of requests depends also on which are the considered states.

We considered the number of requests as a unit of measure, because this is the most appropriate quantifier when working with web applications. We also provide an estimate related to the time needed to learn the optimal policy assuming that a user makes an average of 6 requests/minute.

Our estimate is that the maximum time needed for learning a sequence, independent of its length, is less than 20 minutes. However, the sequence of actions changes only in exceptional cases, which means that once learned, the optimal policy will rarely change.

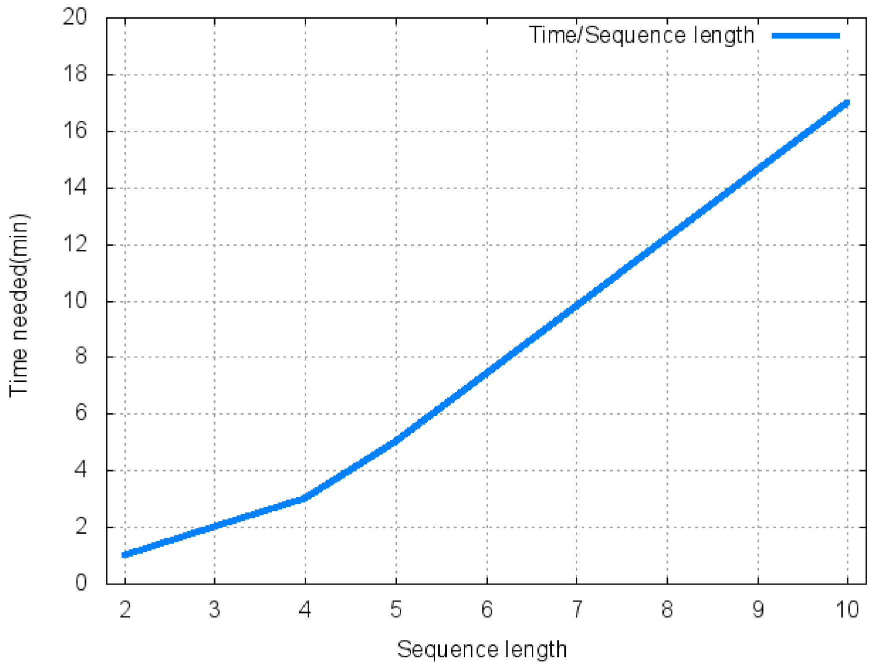


Figure 2. Navibot learning time

The schedule of the employees of the IDS assumes working 8 hours/day. Furthermore, the probability of committing errors at the beginning of the day is almost nonexistent, but it rises as the number of hours spent at work increases. Therefore, towards the end of the day when users need the help of Navibot most, the optimal policy will be learnt and accurate hints will be provided.

5. Conclusions and future work

We would like to emphasize the benefits of implementing the intelligent knowledge management system DTAP.

First of all, DTAP is an educational intelligent software system whose end users are the people working in administration and not the students or teachers. Its behaviour is changing dynamically based on its interaction with the users. The accuracy of the suggestions offered by the system increases throughout the learning process leading to good suggestions in less than 20 minutes. This guarantees that the number of encountered errors will be reduced after a relative short period of time.

The intelligent educational agent Navibot helps the users keep track of their progress when managing a thesis. Moreover, it prevents them from forgetting to make necessary updates or generate all the documents for a thesis.

A future version of Navibot will replace the single-agent system with a multi-agent equivalent. In order to achieve this, the present version of the plugin needs to be modified, such that all interactions of the users with the system are taken into consideration during the learning process. Results of an agent need to be persisted as a shared resource which can be accessed by all the other agents in the system. Therefore, we could no longer use a matrix for storing the rewards, but use a database instead.

We consider that our system should represent only the initial brick of a larger construction. The same principles which were incorporated into DTAP can be used for maintaining Bachelor and Masters theses. Moreover, the design of the application is generic such that only small adjustments are required for adapting it to the requirements of other universities. Therefore, DTAP could be used for managing the PhD theses in all the universities obeying the rules of the Romanian educational legislation.

References

[1] M.S. Abdullah, I. Benest, A. Evans, and C. Kimble. Knowledge Modelling Techniques For Developing Knowledge Management Systems. In *Proceedings of the 3rd European Conference on Knowledge Management*, pages 15–25, Dublin, Ireland, 2002.

[2] M. Alavi and D.E. Leidner. Knowledge Management and Knowledge Management Systems: Conceptual foundations and research issues. *MIS Quarterly*, 25:107–136, 2001.

[3] B.R. Barricelli, M. Padula, and P.L. Scala. Personalized web browsing experience. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, HT '09, pages 345–346, New York, NY, USA, 2009. ACM.

[4] M. Chi, K. VanLehn, and D. Litman. Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. 2010.

[5] M. Chi, K. Vanlehn, D. Litman, and P. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180, April 2011.

[6] A.R. Gavin. *Problem Solving With Reinforcement Learning*. 1995.

[7] A. Iglesias, P. Martnez, R. Aler, and F. Fernandez. Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. *Applied Intelligence*, 31:89–106, 2009. 10.1007/s10489-008-0115-1.

[8] T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 2 edition, 1997.
[9] O. Pârvu. *Intelligent web application for managing the PhD theses in the Babeş-Bolyai University*. Cluj-Napoca, Romania, 2012.
[10] O. Pârvu and S. Dragoş. Interactive and intelligent doctorate theses administration platform. In *Zilele Academice Clujene (ZAC)*, 2012.
[11] M.J. Rosenberg. *E-learning : Strategies for Delivering Knowledge in the Digital Age*. McGraw-Hill Professional, New York, 2001.
[12] D. Stenmark. Information vs. knowledge the role of intranets in knowledge management. In *Proceedings of the XXXV Annual Hawaii International Conference on System Sciences (HICSS-02), 7-10 January*. IEEE, 2002. Exteneded version available from http://w3.informatik.gu.se/ dixi/.
[13] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: `ovidiu.parvu@gmail.com, sanda.dragos@ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

CLUSTERING, TIERED INDEXES AND TERM PROXIMITY WEIGHTING IN TEXT-BASED RETRIEVAL

IOAN BADARINZA AND ADRIAN STERCA

Abstract. In this paper we present a textual retrieval system based on clustering and tiered indexes. Our system can be used for exact phrase matching and also for improved keyword search by employing term proximity weighting in the similarity measure. The document retrieval process is constructed in an efficient way, so that not all the documents in the database need to be compared against the searched query.

1. Introduction

Textual-based web search accounts for a large part of the traffic in the Internet nowadays. The majority of the Internet traffic no longer flows through core routers, but through edge distribution networks like the one of the Google search engine. Although there are several forms of information retrieval (i.e. textual information retrieval, sound-based information retrieval, video information retrieval etc.) the one that evolved the most is text-based information retrieval and this is reflected in the commercially web search engines available today. In this paper we present an information retrieval systems which offers besides keyword search functionality also exact phrase matching. Our system uses a similarity measure which favors documents that contain large portions of consecutive terms from the query, so it can be used in detecting plagiarism in a scientific paper. If the candidate document does not contain groups of consecutive terms from the query, the similarity measure defaults to a classical cosine similarity and the search is a keyword search (not an exact phrase search). The index structure is based on clustering the saved documents and regular term-frequency/inverse-document-frequency indexes.

Received by the editors: November 27, 2012.

2010 *Mathematics Subject Classification.* 68U35, 68M11.

1998 *CR Categories and Descriptors.* H.3.3 [**Information Systems**]: Information Storage and Retrieval – *Information Search and Retrieval*; H.3.1 [**Information Systems**]: Information Storage and Retrieval – *Content Analysis and Indexing*.

Key words and phrases. textual information retrieval, tiered indexes, document clustering, term proximity.

The rest of the paper is structured as follows. In section 2, the general structure of an IR system is outlined and work related to ours is mentioned. Then, section 3 presents the first part of our IR system, the inverted index, followed by section 4 which presents the second part, the retrieval algorithm of our system. In section 5 we present the results of some preliminary evaluations of our system and the paper ends with conclusions in section 6.

2. Information retrieval fundamentals and related work

Every information retrieval system is build from two main parts: the index structure and the retrieval/ranking algorithm. There are several space models for an IR system [1]: the boolean model, the probabilistic model [2], [3], vector space model, linguistic model.

Most IR systems extract keywords from the documents after an initial prefiltering phase (which includes stop words elimination, stemming and lemmantization) and build an inverted index. Most IR systems assign to each keyword t from document d a weight like the following [4]:

$$tf\_idf_{t,d} = tf_{t,d} \times idf_t$$

where $tf_{t,d}$ is the number of occurrences of term t in document d (i.e. term frequency) and idf_t is the inverse term frequency (i.e. number of occurrences of term t in all indexed documents). There are several variants for the term weight formula, but most of them use in some form the term frequency and the inverse document frequency.

The ranked retrieval algorithm compares the query given by the user to all or most documents in the collection and based on some similarity measure between the query and a document it returns the top k relevant documents. A very used similarity measure is the cosine similarity used in the vector space model. For exact phrase matching, a positional index (i.e. an index holding positions in a document for each term) must be used and the similarity measure should include some form of term proximity scoring [5], [6], [7].

3. The index structure of the system

Our system represents documents in the vector space model where each document is viewed as a vector with different document terms and for each term, the system maintains a regular term-frequency/inverse-document-frequency weight [1]. More specifically, for each term t, the index structure holds a postings list with an entry for each document from the collection in which t occurs. Each entry stores the document ID, the term weight in that document and a list of positions were the term appears in that document. The term weight for term t in document d is:

$$wf\_idf_{t,d} = (1 + log(tf_{t,d})) \times idf_t$$

All documents indexed by the system are grouped in clusters/groups based on the similarity between their representative vectors and each group has a leader which is chosen in a random way. When a query is submitted to the IR system, it first checks with the groups' leaders and then it continues the search within the group with the leader most similar to our query. In this way all other non similar documents are excluded from the search and only the most relevant ones are considered which decreases the runtime execution of the query. This cluster pruning heuristic is very useful when new documents are added to the collection. In an IR system, this is done by a crawler. The algorithm used for building the index structure of our system is described in the following lines:

```
The crawler indexing algorithm:
Input: R = {$url_1$, $url_2$, ..., $url_n$} // crawler's repository
       L = {$l_1$, $l_2$, ..., $l_m$} // the existing leaders of the indexed clusters where
                          // $l_i$ is the representation of a document in the
                          // vector space model

For all $r \in R$ do
    d ← getHtml(r);
    d ← filter(d);
    init v;  // v is the representation of document d in vector space model
    for all $t \in d$ do          //for each term t from document d
        $v[t] = wf\_idf_{t,d}$;
    end for;
    init Sim;
    for all $l \in L$ do
        Sim(l) ← Similarity(v,l); // computing similarity between
                                  // document d and cluster leader l
    end for;
    l' ← MAX(Sim);
    insert d in CLUSTER(l');
end for;
```

The cluster based crawler first takes from the repository an url and gets the html source code of that page. The next step is text formatting, deleting the html tags, stop words elimination (e.g. and, or etc.), deleting javascript and css code etc. The following step is index creation based on the term weight $wf\_idf_{t,d}$ for all the terms that appear in the document. In order to find the most appropriate cluster to which this document should be added, the most similar cluster leader from the collection is found and the new document is added to the cluster of this leader. When measuring the similarity between the vectors v and l, the representations of the document d and a cluster leader, the classical cosine similarity metric is used:

$$Similarity(v, l) = \frac{v \circ l}{\|v\| \cdot \|l\|}$$

where the numerator is the dot product of vectors v and l and $\|.\|$ symbolizes the Euclidean norm.

4. The retrieval algorithm of the system

Our information retrieval system uses a combination of clustering and tiered indexes for document searches. When using tiered indexes we set a similarity threshold at a higher value when we search for a document at the first tier and decrease that value at tier two and so on until we find the desired number of documents. This means that the user can search for a whole document and the system will return the most similar indexed documents.

Because of our similarity metric, the retrieval algorithm of our system is a combination between exact phrase retrieval and keyword based retrieval. This means that although the search is essentially a keyword search based on cosine similarity between vectors containing term weights, the vector representation of the query and the vector representation of a candidate document, documents that contain large groups of consecutive terms from the query are favored when returning the results (thus, considered more relevant than documents that do not contain groups of consecutive terms from the query).

The retrieval algorithm returns the top k documents most similar to our query (off course, the query is represented in vector space, in order to be compared to other documents) from the document collection. The algorithm is the following:

```
The document retrieval algorithm:
Input: L = {$l_1$, $l_2$, ..., $l_m$} // the leaders of indexed clusters where $l_i$ is the
                             // vector representation in the vector space model
       q                     // q is the vector representation of the query

Score ← [];
init minimum_threshold;
init similarity_threshold;
```

```
index ← 0;
while (index $\leq$ k) or (similarity_threshold > minimum_threshold) do
        L' ← first_three_similar_leaders(q, L, similarity_threshold);
        L ← L - L';
        for all $l \in L'$ do
                for all $d' \in CLUSTER(l)$ do
                        if (similar(q,d') $\geq$ similarity_threshold)
                                Score[d'] ← similar(q, d');
                                index ← index +1;
                        end if;
                end for;
        end for;
        similarity_threshold ← similarity_threshold - 1;
end while;
for all d' in Score[] do
        Score[d'] ← Score[d'] + title_metadata_url_score(d');
end for;
Sort(Score);
return Score
```

The first step is the search of the most similar leaders from the clusters, which add some speed to the algorithm because the document is compared only to the leaders and not to all documents from the collection. After getting the first three most similar clusters, the document is searched in these leaders' clusters. The function that is used for similarity computation between the vector representations of 2 documents, d_1 and d_2, is a modified cosine similarity function that takes into account matching groups of consecutive words:

$$similar(d_1, d_2) = \frac{d_1 \circ d_2}{\|d_1\| \cdot \|d_2\|} + \left(1 - \frac{1}{N_{d_1,d_2}}\right) \tag{1}$$

where the denominator represents the dot product between vectors d_1 and d_2, $\|.\|$ represents the Euclidean norm of a vector and N_{d_1,d_2} is the length (in terms) of the largest group of consecutive terms that occurs both in d_1 and d_2. Two documents that have a large group of consecutive terms occurring in both will have a value close to 1 for the second part of the $similar(d_1, d_2)$ formula. If documents d_1 and d_2 have no terms in common or if they have terms in common, but no groups of consecutive terms in common, N_{d_1,d_2} is set to 1 and the formula $similar(d_1, d_2)$ defaults to a classical cosine similarity metric. So the two documents, d_1 and d_2, are more similar when the value of the $similar()$ function is higher and less similar when its value is lower. Please note that the formula (1) is not a metric in the mathematical sense since it does not satisfy the triangle inequality property, but it is a semimetric. The values of the semimetric (1) will be between 0.0 and 2.0. The reason that the semimetric contains the N_{d_1,d_2} term is to implement a flexible form of exact phrase matching.

After the first leaders most similar to the query were found, the next step takes place which contains the actual extraction of the k-th most relevant documents that have a similarity value at least as higher as the threshold. The relevant documents that would be returned to the user are searched in the clusters of the selected leaders. The extraction of the first k documents is based on tiered indexes and the following heuristics were used:

- In the first tier, the document will be searched in the first 3 most similar leaders' clusters and the extracted documents must be at least 50% similar with the searched document;
- If the number of returned documents after the first tier is lower than k, than the search goes to tier 2 where the similarity threshold is set to 40%;
- If after tier 2 the number of returned documents is lower than k than it goes to tier 3 where the similarity threshold is set to 20%;
- If the tier 3 search is done and there still aren't k returned documents, than the found documents are returned.

The last step in the algorithm is the rank and score computation for the extracted documents. For score computation, the following factors are taken in consideration: similarity percentage calculated with the formula (1), the words from documents title, key words from meta tags and the words from the url as follows: the score increases with 1 if words from the query are found in the meta data, with 2 if words from the query are found in the document's title and with 2 if words from the query are found in the document's url. Considering the score computation, we can say that this algorithm has support for web pages that were optimized for searched engines.

5. Evaluation

In order to evaluate our text retrieval system we performed initial tests on a rather small document collection consisting of 100 documents, most of them crawled and indexed from the wikipedia.org website. The tests showed that our systems retrieves relevant documents to a large degree of the returned results. We detail in the following lines the results of two tests. In the first test we used a long query of about 70 terms and in the second query we used a smaller query of about 20 terms. Let this query be referred to by Q in both tests. In order to test the efficiency of our modified cosine similarity measure, 7 documents from our 100 documents collection were artificially created:

- Document D_1 contains just the query, Q
- Document D_2 contains the query Q, repeated 3 times
- Document D_3 contains query Q, then some random text, then another occurrence of Q, then other random text
- Document D_4 contains half of Q, followed by some random text, then the other half of Q, followed by another random text
- Document D_5 contains a large portion of random text followed by Q and followed by another random text
- Document D_6 containing some text which resembles Q, but is not the text from Q
- Document D_7 which contains the first third of Q followed by some random text, then the second third of Q, then followed by another random text, then the final third of Q and some random text

For both tests, we set the parameter k of the retrieval algorithm to 10. When the longer query was given to the system, the system retrieved the following documents in the specified order and with the specified similarity score:

Document	Similarity score
D_1	1.99
D_2	1.98
D_3	1.63
D_4	1.44
D_5	1.39
D_7	1.32
Da (irrelevant)	0.0304
Db (irrelevant)	0.0293
Dc (irrelevant)	0.0265
Dd (irrelevant)	0.0248

Precision = 6/10 = 0.6
Recall = 6/6 = 1.0	

The retrieved documents for the short query of about 20 terms are:

Document	Similarity score
D_1	1.89
D_2	1.88
D_3	1.53
D_5	1.46
D_4	1.35
D_7	1.23
D_6 (irrelevant)	0.26
Da (irrelevant)	0.035
Db (irrelevant)	0.028
Dc (irrelevant)	0.026
Precision $= 6/10 = 0.6$ Recall $= 6/6 = 1.0$	

We can see from both tables that the relevant documents were returned, the documents containing large portion of Q have higher similarity score and there is a significant distance between the similarity score for relevant documents and the similarity score for irrelevant documents.

6. Conclusions and future work

In this paper we have presented an information retrieval system based on clusters and tiered indexes that combines exact phrase search with (non-phrase) keyword based search. The system should scale well with a large document collection because it uses clustering in the retrieval process. Initial tests on a rather small sized document collection show that the precision and recall measures of our system have reasonable good values. Of course, in order to assess the full efficiency of such a retrieval system, we need to test it on large collections of documents like the Ad hoc track from the TREC collections [8].

7. Acknowledgments

This work was partially supported by the CNCSIS-UEFISCSU unit of the Romanian Government, through project PN II-RU 444/2010.

References

[1] Manning C.D., Raghavan P., Schutze H, *An introduction to Information Retrieval*, Cambridge University Press, 2009.

[2] Crestani F., Lalmas M., Van Rijsbergen C. J., Campbell I., *Is this document relevant? ... probably: A survey of probabilistic models in information retrieval*, in ACM Computing Surveys, vol 30, no.4, pp.528552, 1998.

[3] Fuhr N., *Probabilistic models in information retrieval*, in The Computer Journal, vol. 35, no.3, pp. 243255, 1992.

[4] Papineni K., *Why inverse document frequency?*, In Proc. North American Chapter of the Association for Computational Linguistics, pp. 18, 2001.
[5] Sadakane K., Imai H., *Text retrieval by using k-word proximity search*, in International Symposium on Database Applications in Non-Traditional Environments, pp.183-188, 1999.
[6] Buttcher S., Clarke C. L. A., Lushman B., *Term proximity scoring for ad-hoc retrieval on very large text collections*, in Proceedings of the 29th annual international ACM SIGIR conference on Research and development in IR, pp. 621622, 2006.
[7] Rasolofo Y., Savoy J., *Term proximity scoring for keyword-based retrieval systems*, in Proceedings of the 25th European Conference on IR Research, pp. 207218, 2003.
[8] *The Text Retrieval Conference*, http://trec.nist.gov .

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: `ionutb@cs.ubbcluj.ro, forest@cs.ubbcluj.ro`