Agents and Data Mining Interaction

International Workshop, ADMI 2011
Taipei, Taiwan, May 2011
Proceedings

Longbing Cao, Ana Bazzan, Andreas L. Symeonidis, Vladimir Gorodetsky, Gerhard Weiss and Philip S. Yu (Eds.)

No Institute Given

Message from the Workshop Chairs

We are pleased to welcome you to attend the 2011 International Workshop on Agents and Data Mining Interaction (ADMI-11), joint with AAMAS 2011. We hope you enjoy the program, and have intensive discussions on Agent Mining research through ADMI-11.

In recent years, Agents and Data Mining Interaction (ADMI, or agent mining) has emerged as a very promising research field. Following the success of ADMI-06 in Hongkong, ADMI-07 in San Jose, AIS-ADM-07 in St Petersburg, ADMI-08 in Sydney, ADMI-09 in Budapest, ADMI-10 in Toronto, the ADMI-11 provides a premier forum for sharing research and engineering results, as well as potential challenges and prospects encountered in the coupling between agents and data mining.

The ADMI-11 workshop encourages and promotes theoretical and applied research and development, which aims at:

- exploit agent-enriched data mining and demonstrate how intelligent agent technology can contribute to critical data mining problems in theory and practice;
- improve data mining-driven agents and show how data mining can strengthen agent intelligence in research and practical applications;
- explore the integration of agents and data mining towards a super-intelligent system;
- discuss existing results, new problems, challenges and impact of integration of agent and data mining technologies as applied to highly distributed heterogeneous, including mobile, systems operating in ubiquitous and P2P environments; and
- identify challenges and directions for future research and development on the synergy between agents and data mining.

The 12 papers accepted by ADMI-11 are from 14 countries. ADMI-11 submissions cover areas from North America, Europe to Asia, indicating the booming of agent mining research globally. The workshop also includes two invited talks by two distinguished researchers.

Following ADMI-09, the papers accepted by ADMI-11 are to be revised and published as an LNAI post-proceedings by Springer. We appreciate Springer, in particular Mr. Alfred Hofmann, for the continuing publication support.

ADMI-11 is sponsored by the Special Interest Group: Agent-Mining Interaction and Integration (AMII-SIG: www.agentmining.org). We appreciate the guideline by the Steering Committee.

More information about ADMI-11 is available from the workshop website:
http://admi11.agentmining.org/.

Finally, we appreciate the contributions made by all authors, program committee members, invited speakers, panelists, and AAMAS 2011 workshop and local organizers.

May 2011

Gerhard Weiss
Philip S Yu
Longbing Cao
Ana Bazzan
Andreas L. Symeonidis
Vladimir Gorodetsky

Organization

General Chair

Gerhard Weiss	University of Maastricht, Netherlands
Philip S Yu	University of Illinois at Chicago, USA

Workshop Co-Chairs

Longbing Cao	University of Technology Sydney, Australia
Ana Bazzan	Universidade Federal do Rio Grande do Sul, Instituto de Informatica, Brasil
Andreas L. Symeonidis	Aristotle University of Thessaloniki, Greece
Vladimir Gorodetsky	Russian Academy of Sciences, Russia

Workshop Organizing Co-Chairs

Dionysis Kehagias	Informatics and Telematics Institute, Centre for Reseach and Technology Hellas, Greece

Program Committee

Ajith Abraham	Norwegian University of Science and Technology, Norway
Eduardo Alonso	University of York, UK
Luis Otavio Alvares	Universidade Federal do Rio Grande do Sul, Brazil
Ioannis Athanasiadis	Dalle Molle Institute for Artificial Intelligence, Switzerland
Sviatoslav Braynov	University of Illinois at Springfield, USA

Valerie Camps	University Paul Sabatier, France
William Cheung	Hong Kong Baptist University, HK
Sung-Bae Cho	Yonsei University, Korea
Frans Coenen	University of Liverpool, UK)
Yves Demazeau	CNRS, France
Giovanna Di Marzo Serugendo	Birkbeck College, UK
Tapio Elomaa	Tampere University of Technology, Finland
Boi Faltings	Artificial Intelligence Laboratory, The Swiss Federal Institute of Technology in Lausanne
Nathan Griffiths	University of Warwick, UK
Mirsad Hadzikadic	University of North Carolina, Charlotte, USA
Ahmed Hambaba	San Jose State University. USA
Heikki Helin	TeliaSonera Finland Oyj, Finland
Henry Hexmoor	University of Arkansas, USA
Ken Kaneiwa	National Institute of Information and Communications Technology, Japan
Hillol Kargupta	University of Maryland, USA
Oleg Karsaev	SPIIRAS, Russia
Dionysis Kehagias Kristian Kersting Yasuhiko Kitamura Matthias Klusch	DFKI, Germany
Matthias Klusch	DFKI, Germany
Daniel Kudenko	University of York, UK
Vipin Kumar	University of Minnesota, USA
Kazuhiro Kuwabara	Ritsumeikan University, Japan
Yaodong Li	Institution of Automation, Chinese Academy of Sciences, China
Jiming Liu	Hong Kong Baptist University, China
Eleni Mangina	University College Dublin, Ireland
Vladimir Marik	Czech Technical University in Prague, Czech Republic
Pericles Mitkas	Aristotle University of Thessaloniki, Greece
Joerg Mueller	Technische University Clausthal, German
Mircea Negoita	WellTech, New Zealand
Wee Keong Ng	Nanyang Technological University, Singapore
Ngoc Thanh Nguyen	Wroclaw University of Technology, Poland
Eugenio Oliveira	University of Porto, Portugal
Michal Pechoucek	Czech Technical University, Czech Republic
Jean-Marc Petit	University Clermont-Ferrand II, France
Martin Purvis	University of Otago, New Zealand
Zbigniew Ras	University of North Carolina, USA
Sandip Sen	University of Tulsa, USA
Simeon Simoff	University of Technology Sydney, Australia

Table of Contents

Obtaining an Optimal MAS Configuration for Agent-Enhanced Mining Using Constraint Optimization

Chayapol Moemeng, Can Wang, Longbing Cao

Quantum Computing and Intelligent Systems,
Faulty of Engineering and Information Technology,
University of Technology, Sydney
P.O. Box 123, Broadway, NSW 2007, Australia
{mchayapol,cawang,lbcao}@it.uts.edu.au

Abstract. We investigate an interaction mechanism between agents and data mining, and focus on agent-enhanced mining. Existing data mining tools use workflow to capture user requirements. The workflow enactment can be improved with a suitable underlying execution layer, which is a Multi-Agent System (MAS). From this perspective, we propose a strategy to obtain an optimal MAS configuration from a given workflow when resource access restrictions and communication cost constraints are concerned, which is essentially a constraint optimization problem. In this paper, we show how workflow is modeled in the way that can be optimized, and how the optimized model is used to obtain an optimal MAS configuration. Finally, we demonstrate that our strategy can improve the load balancing and reduce the communication cost during the workflow enactment.

Keywords: Constraint Optimization, Workflow Management System, Agent and Data Mining Interaction

1 Introduction

A challenge in agent and data mining interaction (ADMI) is the interaction itself: whether agent enhances the data mining process, or data mining is used to improve agent intelligence [1]. Given the current pace of research progression from these two fields, introducing an entirely new mechanism would face the issues in terms of acceptance and adoption by the ADMI community.

At first, let us look at the successful existing data analysis and mining tools (data mining tools, for short). Take SAS Analytics[1], RapidMiner[2], KNIME[3], as examples, they manage complicated data analysis and mining components

[1] SAS website: http://www.sas.com/
[2] Rapid-i website: http://rapid-i.com/
[3] KNIME website: http://www.knime.org

through workflow-style tools. Workflow not only provides a high level of visualization which increases the usability of the system, but also eases system developments in terms of re-usability from the computational resources, such as data analysis and mining components. As a result, users can define arbitrary workflows for their requirements with the support of these application tools. As for today, Workflow Management System (WfMS) has become a de facto standard for data mining tools [2]. Contemporarily, major efforts on improving WfMS performance by using different system architectures and engineering strategies [3, 4] have been made.

Besides, in terms of data mining techniques, there exist two major constraints which significantly impact the performance of the data mining process: *large data sets* and *resource access restriction* [5]. In fact, transmitting a large amount of data over the network could degrade such performance due to the enforced resource access restriction, and may lead the system into inconsistency state and various costs [3] which involve message passing for workflow management activities. Consequently, handling the aforementioned constraints together with the performance improvement at the same time is a challenging task.

However, from the standpoint of our interests in ADMI, adjustable anatomy of Agent-based Workflow Management System (A-WfMS) has been investigated to support dynamic constraints analysis recently [6]. Within A-WfMS, essentially, the workflow functions as a process descriptor and a Multi-Agent System (MAS) handles the workflow engine by coordinating the workflow enactment [7–10].

Our recent work [2] has shown the successful use of workflow system as an interaction platform between agent and data mining. Further, in this paper, we focus on how to obtain the optimal MAS configuration for the agent-enhanced data mining when given an optimized workflow model. In fact, this problem is a class of Constraints Optimization Problem (COP) in which the hard constraint (resource access restriction) must be satisfied and the soft constraint (data transfer minimization) can be optimized.

The main contributions of this paper are in threefold: (i) we explore a type of A-WfMS data mining process explicitly, (ii) we present a strategy to interact between agents and data mining in terms of agent-enhanced mining based on constraint optimization framework, and (iii) we provide the ADMI community with a mechanism that can be quickly and inexpensively accepted and adopted.

The rest of the paper is structured as follows. Section 2 reviews the related work for this paper. Problem statement, including preliminary assumptions, case description, and relevant definitions, is explained in Section 3 in which the POCL plan [11] is used to present our task description. Section 4 specifies the constraint optimization and the MAS configuration schemes individually. Afterwards, we evaluate the strategy and analyse the results in Section 5. Finally, we end the paper in Section 6.

2 Related Work

Integration of agents and WfMS have been introduced in the late 1990s. Pioneer work [7, 8] argued that agents are suitable for workflows since the nature of the requirements could always evolve over time. Therefore, automatic process improvement is desirable, which can then intelligently adapt to the changing environmental conditions. Recent A-WfMSs, JBees [9] and i-Analyst [10], use collaborating software agents as the basis of the systems provides more flexibility than existing WfMSs. JBees uses Coloured Petri nets as process formalism However, JBees' allocation of process agent (workflow engine) does not concern the cost of communication and resource access restrictions. On the other hand, i-Analyst is specifically designed for data mining process using agent technology for workflow execution layer. It provides a range of capabilities, including workflow construction, workflow management, algorithm development, resource management, workflow enactment, and reporting. The mechanism for MAS configuration of i-Analyst is described in this paper.

In later years, attentions have been moved to the implementation techniques. The increasing growth of the Internet has played a major role in the development of WfMS. Recently and remarkably, web Services is a promising method to provide the computational resources for workflow, and such workflow benefits from the support for distributed process models [12]. Although web services technology is becoming a main stream in workflow integration, Huhns [13] argued that web services alone may not completely fulfil distributed WfMS requirements due to the fact that web services know only about themselves, rather than any meta-level awareness; web services are not designed to utilize or understand ontologies; besides, web services are not capable of autonomous action, intentional communication, or deliberatively cooperative behavior. Accordingly, Buhler and Buhler [6] showed a critical survey of workflow, web services, and agent technologies to make a point that web services and MAS will become imperative parts of the future WfMS development. To our extend, we aim at an agent-enhanced data mining system in which computational resources (operations) are not required to be web services. The location requirement of the web service does not support our approach, because rather than transmitting large data over the network to the designated operation at a service provider, we prefer the operation to be transmitted to the data site instead.

Some works attempting to improve the performance of A-WfMS have become incrementally popular. For instance, Yoo et al. [4] proposed an architecture that involves agents in workflow engine layer, therefore, agents can help distribute workloads of a naming/location server and a workflow engine effectively. Bauer and Dadam [3] proposed a variable server assignment for distributed WfMS which allows dynamic server assignment without expensive run-time analysis. Their approaches use the advanced techniques in system engineering to help minimize the communication load in the distributed system, while our approach takes great advantage of the pre-construct configuration which has its own unique benefit: that is once the agent organization has been acquired and tasks have been allocated to the corresponding agents, they may perform tasks in advance,

such as loading independent data and libraries, without having to wait for dependent task completions. In our understanding, improving the performance means minimizing the related costs in A-WfMS.

3 Problem Statement

This section depicts the building blocks of this paper. We propose, firstly, the preconditions of the involved data mining tool. Secondly, the case to be used in the rest of the paper is specified. And finally, the relevant definitions are formalized. In this work, the concerned system is a data mining tool with an embedded WfMS. The WfMS has the information of the size of data sources. The workflow execution layer is deployed on a group of computers (hosts) on a network. Each host has an agent platform to perform as a workflow engine. Lastly, the topology of the network does not change significantly. We have implemented such system in our previous work [10].

3.1 Case Description

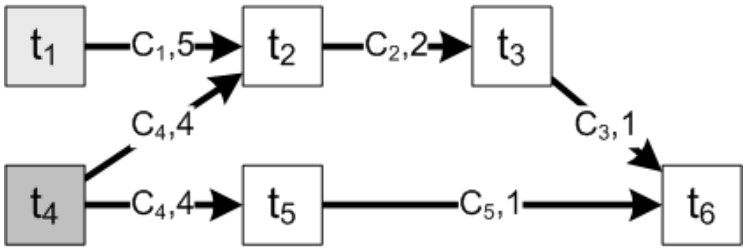


Fig. 1: A Workflow

For the rest of the paper, we set up a workflow and describe the MAS configuration to be obtained from the above one.

To begin with, Fig. 1 is a workflow that consists of six tasks. All the tasks are connected with each other by one or more directed arcs. Each arc is inscribed with a cause c_i and a cost in a numeric value. A cause c is both an effect of task t_i and a precondition of task t_j [11] which can be written as $t_i \xrightarrow{c} t_j$.

Highlighted tasks t_1 and t_4 are *location-specific tasks*, which must be executed at the specific location only; while other tasks are *non-location-specific*.

In addition, MAS configuration is a document that describes the organization of agents in a platform. The description of the configuration is varied by the types of agent platform. Let us use a generic agent platform, WADE, for our common understanding. WADE is an extension of JADE (Java Agent Development Framework) [4]. Listing 1.1 shows a sample of MAS configuration used in

[4] WADE website, http://jade.tilab.com/wade/index.html

```
<platform name="Configuration-1">
 <hosts>
  <host name="host1">
   <containers>
    <container name="Execution-Node-1">
     <agents>
      <agent name="performer1" type="WE Agent"/>
      <agent name="performer2" type="WE Agent"/>
     </agents>
    </container>
   </containers>
  </host>
 </hosts>
</platform>
```

Listing 1.1: Sample MAS Configuration

WADE. Specifically, an agent *platform* is composed of multiple hosts. Each *host* may contain multiple agent containers, in which each *container* may have multiple agents. *Agents*, in particular, are type specific. In this example, agents are of *Workflow Engine (WE) Agent* type. Note that every element (host, container, and agent) must have a specific name.

3.2 Definitions

We desire to obtain the MAS configuration that is optimal for a workflow. In order to get such document, we establish the following definitions. In this work, our definition of a multi-agent plan extends that of the Partial-Order Causal-Link (POCL) plan. The POCL plan definition has been well-established in the planning community [11]. Our definition is also inspired by the framework proposed by Cox and Durfee [14].

Definition 1. *A **Localized POCL (L-POCL) plan** is a tuple* $P = \langle T, \succ_T, \succ_C, L_T, \mathscr{C} \rangle$ *where*

- T *is a set of the tasks,*
- $\succ_T$ *is the temporal orders on* T*, where* $e \in \succ_T$ *is a tuple* $\langle t_i, t_j \rangle$ *with* $t_i, t_j \in T$*,*
- $\succ_C$ *is the causal partial orders on* T*, where* $e \in \succ_C$ *is a tuple* $\langle t_i, t_j, c \rangle$ *with* $t_i, t_j \in T$ *and* c *is a cause,*
- L_T *is a set of the locations and the corresponding tasks on* T*, where* $e \in L_T$ *is a tuple* $\langle l_i, t_j \rangle$ *with location* $l_i \in L$ *and task* $t_j \in T$*,*
- $\mathscr{C}$ *is the cost function mapping* $\mathscr{C} : T \times T \rightarrow \{0, \Re\}$*, this function* $\mathscr{C}(t_i, t_j) = 0$ *if* $l_{t_i} = l_{t_j}$*, otherwise* $\mathscr{C}(t_i, t_j) = r$ *where* $t_i, t_j \in T$*, and* $r \in \Re$ *is a real value related with the size of data to be transmitted.*

Location and Cost Function. We have added location set L_T and cost function $\mathscr{C}$ to the original POCL plan. The L-POCL plan itself does not fully represent the workflow, since it does not maintain the information about process and execution conditions. However, it is still adequate to capture necessary information for constraint optimization. Here, location can be in any form that uniquely represents a physical place, such as IP address, host name, etc. A non-location-specific task has $\emptyset$ as its location. In relation to L_T, we introduce two more related sets $L_{|T|}$ and T_L^*.

- $L_{|T|}$: describes the number of tasks at the same location, i.e., $e \in L_{|T|}$ is a tuple $\langle l_i, n\rangle$, where l_i is the location, $t_i \in T$, and n is the number of tasks at location l_i.
- T_L^*: is a set of tasks $t \in T$ whose locations cannot be changed.

In terms of COP, the *hard constraint* must be satisfied strictly; in this scenario, that is the location of each task $t \in T_L^*$ cannot be changed. However, the *soft constraint* is preferred to be optimized. In this case, the concerned global function is the communication cost $\mathscr{C}$ between each task, for the reason that the data transmission between locations incurs communication costs. But when the tasks are at the same location, communication cost is marked to zero.

Consider the workflow as shown in Fig. 1, the L-POCL plan concepts involved are exemplified as follows:

$$
\begin{aligned}
T &= \{t_1, t_2, t_3, t_4, t_5, t_6\} \\
\succ_T &= \{\langle t_1, t_2\rangle, \langle t_1, t_3\rangle, \langle t_1, t_6\rangle, \langle t_2, t_3\rangle, \langle t_2, t_6\rangle, \langle t_3, t_6\rangle, \\
&\qquad \langle t_4, t_2\rangle, \langle t_4, t_3\rangle, \langle t_4, t_5\rangle, \langle t_4, t_6\rangle, \langle t_5, t_6\rangle\} \\
\succ_C &= \{\langle t_1, t_2, c_1\rangle, \langle t_2, t_3, c_2\rangle, \langle t_3, t_6, c_3\rangle, \langle t_4, t_2, c_4\rangle, \langle t_4, t_5, c_4\rangle, \langle t_5, t_6, c_5\rangle\} \\
T_L^* &= \{t_1, t_4\} \\
L_T &= \{(l_1, t_1), (l_2, t_4), (\emptyset, t_2), (\emptyset, t_3), (\emptyset, t_5), (\emptyset, t_6)\} \\
L_{|T|} &= \{(l_1, 1), (l_2, 1), (\emptyset, 4)\} \\
\mathscr{C}(i,j) &= \begin{cases} 5 & \text{if } i = t_1 \text{ and } j = t_2, \\ 2 & \text{if } i = t_2 \text{ and } j = t_3, \\ 1 & \text{if } i = t_3 \text{ and } j = t_4, \\ 4 & \text{if } i = t_4 \text{ and } j = t_2, \\ 4 & \text{if } i = t_4 \text{ and } j = t_5, \\ 1 & \text{if } i = t_5 \text{ and } j = t_6 \end{cases}
\end{aligned}
$$

Note that L_T and $\mathscr{C}$ are also illustrated in Fig 2a Initial L_T table.

Multi-Agent L-POCL. We have shown that a workflow can be modeled by a L-POCL plan. However, the MAS configuration also requires information about agents and their task assignments. The task assignment is to be mapped with the corresponding location. Therefore, a multi-agent version of L-POCL is defined as follows:

Definition 2. *A **Multi-Agent L-POCL plan (ML-POCL)** is a tuple* $P = \langle A, T, \succ_T, \succ_C, L_T, \mathscr{C}, X \rangle$ *where*

- $\langle T, \succ_T, \succ_C, L_T, \mathscr{C} \rangle$ *is the embedded POCL plan,*
- A *is a set of the agents,*
- *Execution assignment* X *is a set of the tuples with the form* $\langle t, a \rangle$*, representing that the agent* $a \in A$ *is assigned to execute task* $t \in T$*.*

L_T and X are necessary components to construct a MAS configuration. Consider Listing 1.1, **hosts** are all locations found in L_T; **agents** are specified in X. L_T and X are mutually linked by tasks. Note that the naming scheme of an agent has no restriction as long as the name is unique.

4 Optimal MAS Configuration

This section describes how to obtain an optimal MAS configuration in terms of constraint optimization. A constraint optimization algorithm, in this case, is to minimize the soft constraint $\mathscr{C}$. The optimization reflects the change of L_T to an optimal one L_T^*. Then L_T^* is used to guide the construction of the optimal MAS configuration.

4.1 Constraint Optimization

The optimization algorithm migrates workflow tasks based on location. It uses the global cost function $\mathscr{C}$ as a guide by reducing the costly routes by minimizing the number of locations in the workflow.

Algorithm 1: Optimize $\mathscr{C}$

```
input  : L_T
output: L*_T
L*_T = L_T;
while there exists non-location-specific task in L*_T do
    foreach non-location-specific task t ∈ L*_T do
        let T_x be a set of location-specific tasks adjacent to t;
        if T_x = ∅ then continue;
        let t_x ∈ T_x be a task that maximizes C(t, t_x) or C(t_x, t).;
        update (∅, t) ∈ L*_T to (l_x, t) where l_x is the location of t_x.;
        if t_x maximizes C(t, t_x) then
            C(t, t_x) ← 0;
        else
            C(t_x, t) ← 0;
return L*_T;
```

Given a workflow as shown in Fig. 1, each arc is marked with the estimated size of data (presume in mega-bytes) to be transmitted between relevant tasks. Tasks t_1 and t_4 are at l_1 and l_2 respectively, and the locations of them cannot

$\mathscr{C}$	l_1,t_1	$\emptyset,t_2$	$\emptyset,t_3$	l_2,t_4	$\emptyset,t_5$	$\emptyset,t_6$
l_1,t_1	-	5	-	-	-	-
$\emptyset,t_2$	-	-	2	-	-	-
$\emptyset,t_3$	-	-	-	-	-	1
l_2,t_4	-	4	-	-	4	-
$\emptyset,t_5$	-	-	-	-	-	1
$\emptyset,t_6$	-	-	-	-	-	-

Fig 2a Initial L_T table

$\mathscr{C}$	l_1,t_1	l_1,t_2	$\emptyset,t_3$	l_2,t_4	$\emptyset,t_5$	$\emptyset,t_6$
l_1,t_1	-	0	-	-	-	-
l_1,t_2	-	-	2	-	-	-
$\emptyset,t_3$	-	-	-	-	-	1
l_2,t_4	-	4	-	-	4	-
$\emptyset,t_5$	-	-	-	-	-	1
$\emptyset,t_6$	-	-	-	-	-	-

Fig 2b Migrate t_2 to t_1

$\mathscr{C}$	l_1,t_1	l_1,t_2	l_1,t_3	l_2,t_4	l_2,t_5	$\emptyset,t_6$
l_1,t_1	-	0	-	-	-	-
l_1,t_2	-	-	0	-	-	-
l_1,t_3	-	-	-	-	-	1
l_2,t_4	-	4	-	-	0	-
l_2,t_5	-	-	-	-	-	1
$\emptyset,t_6$	-	-	-	-	-	-

Fig 2c Migrate t_5 to l_2 and t_3 to l_1

$\mathscr{C}$	l_1,t_1	l_1,t_2	l_1,t_3	l_2,t_4	l_2,t_5	l_2,t_6
l_1,t_1	-	0	-	-	-	-
l_1,t_2	-	-	0	-	-	-
l_1,t_3	-	-	-	-	-	1
l_2,t_4	-	4	-	-	0	-
l_2,t_5	-	-	-	-	-	0
l_2,t_6	-	-	-	-	-	-

Fig 2d Optimized L_T^*

Fig. 2: Cost Function $\mathscr{C}$ Optimization

be changed since $t_1, t_2 \in T_L^*$. The corresponding location set L_T and cost function $\mathscr{C}$ can be presented in the tabular form as shown in Fig. 2a. Each header cell contains the combinations of each task $t_j (j = 1, \cdots, 6)$ and its location $l_i (i = 1, 2)$, note that $\emptyset$ means unspecified location. Moreover, '$-$' denotes the situation of unavailable connection. As $optimize()$ is applied to the workflow, for instance, we get $\mathscr{C}(t_1, t_2) = 5$, where t_1 is at l_1 and t_2 is non-location-specific. The algorithm gradually evolves to the point L_T^* that every task is assigned with a optimal location. The algorithm performs the migration in two fashions: cost reduction and load balancing.

Cost reduction optimization: The algorithm searches for every non-location-specific task that is adjacent to at least one location-specific task. Firstly, $(\emptyset, t_2)$ is found with two adjacent location-specific tasks t_1 and t_4. As $\mathscr{C}(t_1, t_2) > \mathscr{C}(t_4, t_2)$, t_2 is migrated to the same location of task t_1, i.e., l_1. Tasks t_1 and t_2 are now at the same location l_1, then $\mathscr{C}(t_1, t_2)$ is updated to zero. Subsequently, the algorithm searches again and finds another non-location-specific task t_3, and then migrates it to the location of task t_2, i.e., l_1. The same principle goes on for non-location-specific task t_5, it is migrated to the location of t_4, i.e., l_2.

Load balancing optimization: As the cost reduction optimization goes on, the last non-location-specific task t_6 has two choices to migrate, t_3's location l_1 and t_5's location l_2, and both the values of $\mathscr{C}(t_3, t_6)$ and $\mathscr{C}(t_5, t_6)$ equal to 1. Now, let us consider the number of tasks at each location, indicated as $L_{|T|} = \{(l_1, 3), (l_2, 2), (\emptyset, 1)\}$. A simple load balancing advises to migrate task t_6 to location l_2. However, it also depends on other factors such as computing power which we do not cover in this paper, we will consider it for our future work. The final L_T^* is shown in Fig. 2d.

4.2 Obtaining MAS Configuration

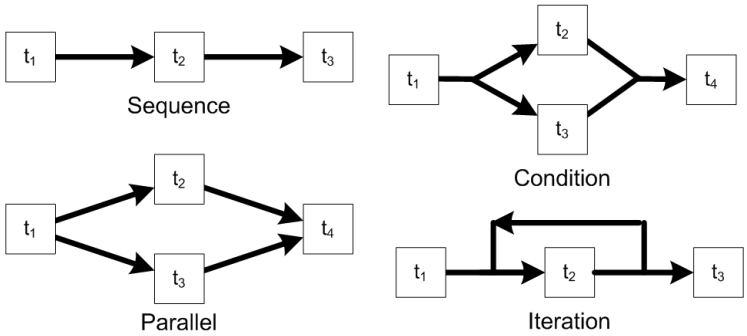


Fig. 3: Workflow Constructs

This section discusses the way to obtain the optimal MAS configuration. The basic configuration rule is to assign one task to one agent, thus $|A| = |T|$ regardless of locations. But agents are related to the structure of L_T^* which has been attained in the previous steps. The execution assignment X of ML-POCL plan is also needed to obtain the desired configuration. Instead of starting out with one agent for one task, one agent is assigned to one location, thus initially $|A| = |L|$ regardless of tasks. Tasks that can be executed without having to wait for other tasks should be allocated with an agent. At each location $l \in L_T^*$, determine the agent allocation according to the workflow constructs (illustrated in Fig. 3):

- **Sequence**: If task t_i has a successive task t_j defined in $\succ_T$, an agent a is assigned to t_i and t_j, s.t. $(t_i, a), (t_j, a)$.
- **Parallel**: If task t_i and t_j are not linked with temporal ordering as defined in $\succ_T$, two agents a_1 and a_2 are assigned to tasks t_i and t_j respectively, s.t. $(t_i, a_1), (t_j, a_2)$. For example in Fig. 1, if tasks t_2 and t_5 were at the same location, they are parallel.
- **Condition**: If task t_i is to choose either t_j or t_k exclusively, which means only one successive task will be executed, therefore an agent a is assigned to all t_i, t_j, t_k, s.t. $(t_i, a), (t_j, a), (t_k, a)$.
- **Iteration**: Iterative task is a sequential form of condition and loop. Let task t_i be the entrance to the iteration, T_i is the set of tasks to be repeated in a loop, t_j is the decision-making task, and t_k is the successive task of t_j, then an agent a is assigned to all tasks $t \in \{t_i, t_j, t_k\} \cup T_i$.

With respect to iteration, we cannot precisely calculate the number of iterations which will directly affect the total costs. But this is a factor that will influence every scheme of configuration proportionally.

5 Evaluation

In this section we demonstrate that the overall communication cost can be significantly reduced by taking advantage of the optimal MAS configuration strategy. For this purpose, we have simulated the execution of the randomly generated workflow in a distributed system composed of 5 hosts (locations). We randomly generate 20 workflows consisting of 10-20 tasks with random structures. In a workflow, 20-30% of all tasks are location specific. And each causal link is assigned with a random cost value between 1-5 (assumed mega-bytes). This simulation generates three MAS configurations for each workflow:

1. **Centralized configuration (C)**: All tasks run at one host.
2. **Equally distributed configuration (E)**: Equally assign a host with a number of tasks.
3. **Optimal configuration (O)**: The method we have proposed.

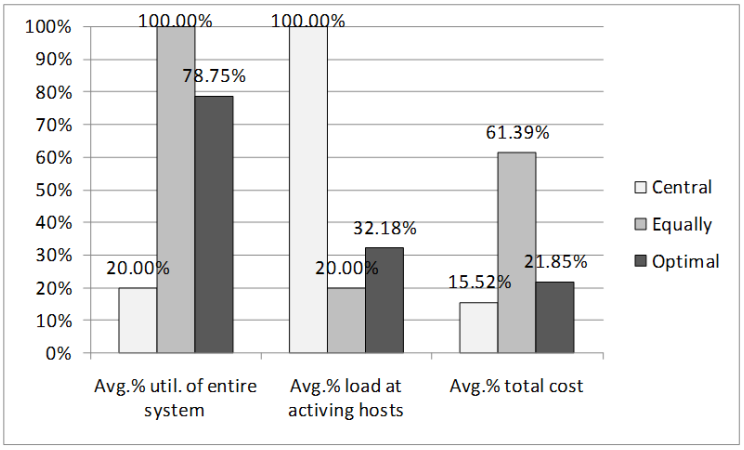


Fig. 4: Results of the Simulation

Based on the 20 workflows and 5 hosts, we calculate three evaluation values (the results are shown in Fig. 4):

1. **The average percentage of utilization of the entire system**: indicates the percentage of hosts being used to run the tasks. **C** uses only one host; **E** distributes all tasks to all hosts, so all hosts are utilized; and **O** uses only the hosts that have location-specific tasks, s.t., the number of active hosts is equal to $|L_T|$.
2. **The average percentage of load at the active hosts**: indicates the percentage of the number of tasks assigned to the active hosts. **C** runs all tasks at the only active host, s.t. 100%; **E** distributes all tasks to all hosts,

s.t. the number of tasks per host is $|T|/|\texttt{host}|$; and **O** uses only the hosts that have location-specific tasks, therefore, it is slightly higher than that of **E**, $|T|/|L_T|$.

3. **The average percentage of total cost**: indicates the amount of communication cost compared to the maximum cost in each workflow of the three configurations. In this simulation, we add one extra unit of cost for each causal link although all communications happened at the same host in order to demonstrate the activity that happens during the workflow enactment, otherwise **C** will always have zero cost. **C** communicates at a single host internally, then the cost is minimum; **E** communicates through out the system, so the cost is the highest among all; and **O** communicates only between selected hosts, therefore, the cost is between **E** and **C**.

The results of the simulation show that the proposed optimal MAS configuration both reduces the communication cost significantly and maintains a high level of system utilization. Its load balancing is better than that of centralized configuration, and obviously it cannot be better than that of equally distributed configuration as it does not use all of the hosts. With the proposed approach, we can achieve an optimal configuration that satisfies the hard constraints (location specific tasks) and optimizes the soft constraints (communication cost).

6 Conclusion and Future Work

We have presented a strategy to obtain an optimal MAS configuration for agent-enhanced data mining. The method utilizes the existing component in modern data mining tools, i.e., the workflow. The workflow is modeled with our proposed ML-POCL plan. The plan is then optimized and used to obtain an optimal MAS configuration. The result shows that the attained MAS configuration optimally reduces the communication cost and maintains a high level of system utilization.

However, there are a few issues yet to be improved. The constraint optimization process assumes all costs ($\mathscr{C}$) between tasks to be pre-defined. But in a more complex situation, the configuration should deal with cost function $\mathscr{C}$ dynamically since the environment may change and affect it. Another issue is that most WfMSs use data pipe-lining to pass data from one task to another, while global resource sharing scheme allows data production from one activity to be published to the naming directory service for later reference and re-use. This will help boost re-usability of the resource.

References

1. Cao, L., Gorodetsky, V., Mitkas, P.: Agent mining: The synergy of agents and data mining. Intelligent Systems, IEEE **24**(3) (2009) 64–72
2. Moemeng, C., Zhu, X., Cao, L. In: Integrating Workflow into Agent-Based Distributed Data Mining Systems. Volume 5980 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg (2010) 4–15

3. Bauer, T., Dadam, P.: Efficient Distributed Workflow Management Based on Variable Server Assignments. In: Advanced Information Systems Engineering, 12th International Conference CAiSE 2000. Volume 1789. (2000) 94–109
4. Yoo, J.J., Suh, Y.H., Lee, D.I., Jung, S.W., Jang, C.S., Kim, J.B.: Casting Mobile Agents to Workflow Systems: On Performance and Scalability Issues. In Mayr, H.C., Lazansky, J., Quirchmayr, G., Vogel, P., eds.: Database and Expert Systems Applications. Volume 2113 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg (August 2001) 254–263
5. Klusch, M., Lodi, S., Gianluca, M.: The role of agents in distributed data mining: issues and benefits. IEEE Comput. Soc (2003)
6. Buhler, P.A., Vidal, J.M.: Towards Adaptive Workflow Enactment Using Multiagent Systems. Information Technology and Management **6**(1) (January 2005) 61–87
7. Judge, D.W., Odgers, B.R., Shepherdson, J.W., Cui, Z.: Agent-enhanced Workflow. BT Technology Journal **16**(3) (July 1998) 79–85
8. Odgers, B.R., Shepherdson, J.W., Thompson, S.G.: Distributed Workflow Coordination by Proactive Software Agents. In: In Intelligent Workflow and Process Management. The New Frontier for AI in Business IJCAI-99 Workshop. (1999)
9. Ehrler, L., Fleurke, M., Purvis, M., Savarimuthu, B.: Agent-based workflow management systems (WfMSs). Information Systems and E-Business Management **4**(1) (January 2006) 5–23
10. Moemeng, C., Zhu, X., Cao, L., Jiahang, C.: i-Analyst: An Agent-Based Distributed Data Mining Platform. IEEE (December 2010)
11. Weld, D.S.: An Introduction to Least Commitment Planning. AI Magazine **15**(4) (1994) 27–61
12. Savarimuthu, B.T., Purvis, M., Purvis, M., Cranefield, S.: Agent-based integration of Web Services with Workflow Management Systems. In: AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM (2005) 1345–1346
13. Huhns, M.: Agents as Web services. Internet Computing, IEEE **6**(4) (2002) 93–95
14. Cox, J., Durfee, E.: An efficient algorithm for multiagent plan coordination. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM (2005) 828–835

Exploiting Domain Knowledge in Making Delegation Decisions

Chukwuemeka David Emele[1], Timothy J. Norman[1],
Murat Şensoy[1], and Simon Parsons[2]

[1] University of Aberdeen, Aberdeen, AB24 3UE, UK
[2] Brooklyn College, City University of New York, 11210 NY, USA
{c.emele,t.j.norman,m.sensoy}@abdn.ac.uk
parsons@sci.brooklyn.cuny.edu

Abstract. In multi-agent systems, agents often depend on others to act on their behalf. However, delegation decisions are complicated in norm-governed environments, where agents' activities are regulated by policies. Especially when such policies are not public, learning these policies become critical to estimate the outcome of delegation decisions. In this paper, we propose the use of domain knowledge in aiding the learning of policies. Our approach combines ontological reasoning, machine learning and argumentation in a novel way for identifying, learning, and modeling policies. Using our approach, software agents can autonomously reason about the policies that others are operating with, and make informed decisions about to whom to delegate a task. In a set of experiments, we demonstrate the utility of this novel combination of techniques through empirical evaluation. Our evaluation shows that more accurate models of others' policies can be developed more rapidly using various forms of domain knowledge.

1 Introduction

In many settings, agents (whether human or artificial) engage in problem solving activities, which often require them to share resources, act on each others' behalf, communicate and coordinate individual acts, and so on. Such problem-solving activities may fail to achieve desired goals if the plan is not properly resourced and tasks delegated to appropriately competent agents. Irrespective of the field of endeavour, the overall success of problem-solving activities depends on a number of factors; one of which is the selection of appropriate candidates to delegate tasks to (or share resources with). However, successful delegation decisions depend on various factors. In norm-governed environments, one of the factors for making successful delegation decisions is the accuracy of the prediction about the policy restrictions that others operate with.

In multi-agent environments, agents may operate under policies, and some policies may prohibit an agent from providing a resource to another under certain circumstances. Such policies might regulate what resources may be released to a partner from some other organisation, under what conditions they may be used, and what information regarding their use is necessary to make a decision. In addition, policies may govern actions that can be performed either to pursue individual goals or on behalf of another.

In Emele *et al.* [1], we show that intelligent agents can determine what policies others are operating within by mining the data gathered from past encounters with that agent (or similar agents) as they collaborate to solve problems. This prior research uses a novel combination of argumentation-derived evidence (ADE) and machine learning in building stable models of others' policies. Here we explore the question that given agents may have access to some background (or ontological) domain knowledge, how can we exploit such knowledge to improve models of others' policies? To do this, we propose the use of ontological reasoning, argumentation and machine learning to aid in making effective predictions about who to approach if some other collaborator is to be delegated to perform a task on the behalf of another.

The rest of this paper is organised as follows. Section 2 discusses delegation in norm-gorverned environments. Section 3 presents our approach for learning policies. Section 4 reports the results of our evaluations. Section 5 summarises our findings, discusses related work and outlines future directions.

2 Delegating in Norm-governed environments

Task delegation, in general, is concerned with identifying a suitable candidate (or candidates) to transfer authority to act on one's behalf [4]. In other words, we are concerned with finding candidate agents that will achieve a given goal on our behalf. Implicitly, this model is based on experience gathered from past encounters. In a norm-governed multi-agent system where each agent is regulated by a set of rules, referred to as *policies* (or norms), the policies could determine what actions an agent is (or is not) allowed to perform. Delegation, in this case, is not just a matter of finding agents that possess the appropriate expertise (or resource); if an agent has the required expertise but is operating under a policy that prohibits the performance of that action, it may not take on the delegated task. Nevertheless, delegation in such settings entails finding agents who possess the required expertise (or resources), and whose policies permit it to perform the required action (or provide the required resource). If an agent is allowed to perform an action (according to its policy) then we assume it will be willing to perform it when requested, provided it has the necessary resources and/or expertise, and that doing so does not yield a negative utility. In our framework, an agent that has been assigned a task is solely responsible for that task. However, an agent can delegate an aspect of a task to another. For example, agent x is responsible for performing task T_x but could delegate the provision of some resource R_r required to fulfill T_x to another agent y_1. Provided agent y_1 has resource R_r, and does not have any policy that forbids the provision of R_r then we assume y_1 will make R_r available to x.

From the above example, we see that agent x needs to find effective ways of delegating the provision of resource R_r. In order to delegate a task successfully, we need to find out the agent whose policy constraints will most likely, according to a chosen metric, permit it to execute the delegated task. In our framework, whenever there is a task to be delegated, policy predictions are generated alongside the confidence of those predictions from the policy models that have been learned over time. Confidence values of favourable policy predictions are easily compared to determine which candidate to

delegate the task to. In our case, confidence values range from 0 to 1, with 0 being *no confidence* in the prediction, and 1 being *complete confidence*.

The delegating agent explores the candidate space to identify suitable candidates to whom it can delegate a task. In these terms, our delegation problem is concerned with finding potential candidates whose policies permit it to perform the delegated task, and thereafter, selecting the most promising candidate from the pool of eligible candidates. Borrowing ideas from economics, we assume that some payment will be made to an agent for performing a delegated task (e.g. payment for the provision of a service).

Example 1 *Consider a situation where an agent x is collaborating with a number of agents, y_1, y_2, y_3, and y_4, to solve an emergency response problem. Let us assume that agent x does not have a helicopter in its resource pool, and that each of agents y_1, y_2, y_3, and y_4 can provide helicopters, jeeps, vans, bikes, fire extinguishers, and unmanned aerial vehicles (UAVs).*

Agent x in Example 1 has to decide which of the potential providers, y_1, y_2, y_3, and y_4 to approach to provide the *helicopter*. Let us assume that the four providers advertise similar services. Agent x, at this point, attempts to predict the policy of the providers with respect to task delegation (or resource provision). This prediction is based on policy models built from past experience with these providers (or similar agents). Assuming the predictions are as follows: (i) y_1 *will accept to provide the helicopter with 0.6 confidence*; (ii) y_2 *will accept with 0.9 confidence*; (iii) y_3 *will accept with 0.7 confidence*; and (iv) y_4 *will decline with 0.8 confidence*. If the decision to choose a provider is based on policy predictions alone, then y_2 is the best candidate.

3 Learning agent policies

The framework we propose here enables agents to negotiate and argue about task delegation, and use evidence derived from argumentation to build more accurate and stable models of others' policies. The architecture of our framework, sketched in Figure 1, enables agents to learn the policies and resource availabilities of others through evidence derived from argumentation, and improve those models by exploiting domain knowledge. The dialogue manager handles all communication with other agents. The learning mechanism uses machine learning techniques to reason over the dialogue and attempts to build models of other agents' policies and resource availabilities based on arguments exchanged during encounters. The arguments include the features that an agent requires in order to make a decision about accepting a task delegation or not. The agent attempts to predict the policies of others by reasoning over policy models (built from past experience). Such reasoning is further improved by exploiting background domain knowledge and concept hierarchies in an ontology (see Section 3.4).

3.1 Policies

In this framework, agents have policies that govern how resources are deployed to others. In our model, policies are conditional entities (or rules) and so are relevant to an agent under specific circumstances only. These circumstances are characterised by a set of features, e.g., vehicle type, weather conditions, etc.

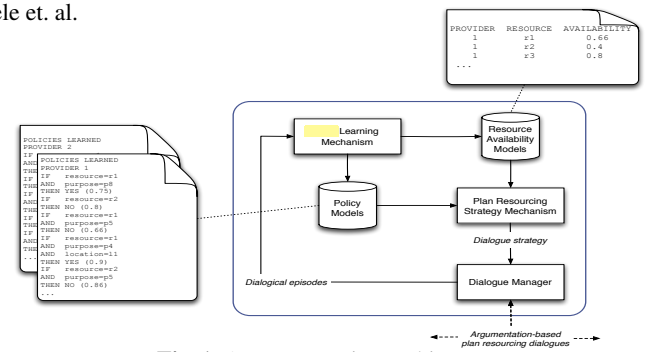


Fig. 1. Agent reasoning architecture.

Definition 1 (Features). *Let $\mathcal{F}$ be the set of all features such that $f_1, f_2, \ldots \in \mathcal{F}$. We define a feature as a characteristic of the prevailing circumstance under which an agent is operating (or carrying out an activity).*

Our concept of policy maps a set of features into an appropriate policy decision. In our framework, an agent can make one of two policy decisions at a time, namely (1) *grant*, which means that the policy allows the agent to provide the resource when requested; and (2) *deny*, which means that the policy prohibits the agent from providing the resource.

Definition 2 (Policies). *A policy is defined as a function $\Pi : \boldsymbol{\mathcal{F}} \rightarrow \{grant, deny\}$, which maps feature vectors of agents, $\boldsymbol{\mathcal{F}}$, to appropriate policy decisions.*

In order to illustrate the way policies may be captured in this model, we present an example. Let us assume that f_1 is resource, f_2 is purpose, f_3 is weather report (with respect to a location), f_4 is the affiliation of the agent, and f_5 is the day the resource is required, then $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}_3$ in Figure 2 will be interpreted as follows: $\mathbb{P}_1$**:** You are **permitted** to release a *helicopter* (h), to an agent if the $helicopter$ is required for the purpose of transporting relief materials (trm); $\mathbb{P}_2$**:** You are **prohibited** from releasing an *aerial vehicle* (av) to an agent in bad weather conditions - e.g. volcanic clouds (vc); $\mathbb{P}_3$**:** You are **permitted** to release a *jeep* (j) to an agent.

Policy Id	f_1	f_2	f_3	f_4	f_5	Decision
$\mathbb{P}_1$	h	trm				grant
$\mathbb{P}_2$	av		vc			deny
$\mathbb{P}_3$	j					grant
$\mathbb{P}_4$	c		vc	xx		grant
...	...	...	...	...	...	...
$\mathbb{P}_n$	q	yy	w	xx	z	deny

Fig. 2: An agent's policy profile.

In the foregoing example, if *helicopter* is intended to be deployed in an area with volcanic clouds then the provider is forbidden from providing the resource but might offer a ground vehicle (e.g. *jeep*) to the seeker if there is no policy prohibiting this and the resource is available. Furthermore, whenever a seeker's request is refused, the seeker may challenge the decision, and seek justifications for the refusal. This additional evidence is beneficial, and could be used to improve the model, hence, the quality of decisions made in future episodes.

3.2 Argumentation-based Negotiation

Figure 3 illustrates the protocol employed in this framework, which guides dialogical moves. Our approach in this regard is similar to the dialogue for resource negotiation

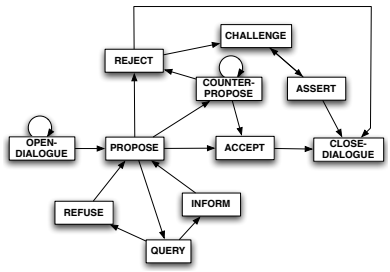


Fig. 3. The negotiation protocol.

proposed by McBurney & Parsons [3]. To illustrate the sorts of interaction between agents, consider the example dialogue in Figure 4. Let x and y be seeker and provider agents respectively. Suppose we have an argumentation framework that allows agents to ask for and receive explanations (as in Figure 4, *lines* 11 *and* 12), offer alternatives (counter-propose in Figure 3), or ask and receive more information about the attributes of requests (*lines* 4 *to* 9), then x can gather additional information regarding the policy rules guiding y concerning provision of resources.

#	**Dialogue Sequence**	**Locution Type**
1	x: Start dialogue.	OPEN-DIALOGUE
2	y: Start dialogue.	OPEN-DIALOGUE
3	x: Can I have a *helicopter* for $0.1M reward?	PROPOSE
4	y: What do you need it for?	QUERY
5	x: To transport relief materials.	INFORM
6	y: To where?	QUERY
7	x: A refugee camp near Indonesia.	INFORM
8	y: Which date?	QUERY
9	x: On Friday 16/4/2010.	INFORM
10	y: No, I can't provide you with a *helicopter*.	REJECT
11	x: Why?	CHALLENGE
12	y: I am not permitted to release a *helicopter* in volcanic eruption.	ASSERT
13	x: There is no volcanic eruption near Indonesia.	CHALLENGE
14	y: I agree, but the ash cloud is spreading, and weather report advises that it is not safe to fly on that day.	ASSERT
15	x: Ok, thanks.	CLOSE-DIALOGUE

Fig. 4. Dialogue example.

Negotiation for resources takes place in a turn-taking fashion. The dialogue starts, and then agent x sends a request (propose in Figure 3) to agent y, e.g. line 3, Figure 4. The provider, y, may respond by conceding to the request (accept), rejecting it, offering an alternative resource (counter-propose), or asking for more information (query) such as in line 4 in Figure 4. If the provider agrees to provide the resource then the negotiation ends. If, however, the provider rejects the proposal (line 10, Figure 4) then the seeker may challenge that decision (line 11), and so on. If the provider suggests an alternative then the seeker evaluates it to see whether it is acceptable or not. Furthermore, if the provider agent needs more information from the seeker in order to make a decision, the provider agent would ask questions that will reveal the features it requires to make a decision (query, inform/refuse). The negotiation ends when agreement is reached or all possibilities explored have been rejected.

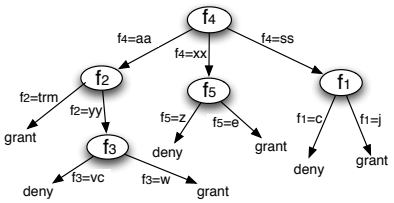


Fig. 5. Example decision tree.

3.3 Learning from past experience through dialogue

When an agent has a collection of experiences with other agents described by feature vectors (see Section 3.1), we can make use of existing machine learning techniques for learning associations between sets of discrete attributes (e.g. $f_1, f_2, \ldots f_n \in \mathcal{F}$) and policy decisions (i.e., *grant* and *deny*). In previous research we investigated three classes of machine learning algorithms: (i) instance-based learning (using k-nearest neighbours); (ii) rule-based learning (using sequential covering); and (iii) decision tree learning (using C4.5). Figure 5 shows an example decision tree representing a model of the policies of some other agent learned from interactions with that agent. Nodes of the decision tree capture features of an agent's policy, edges denote feature values, while the leaves are policy decisions.

The machine learning algorithms were chosen to explore the utility of different classes of learning techniques. Instance-based learning is useful in this context because it can adapt to and exploit evidence from dialogical episodes incrementally as they accrue. In contrast, decision trees, and rule learning are not incremental; the tree or the set of rules must be reassessed periodically as new evidence is acquired.[3] Learning mechanisms such as sequential covering, decision trees do have a number of advantages over instance-based approaches; in particular, the rules (or trees) learned are more amenable to scrutiny by a human decision maker.

The training examples used in each learning mechanism are derived from plan resourcing episodes (or interactions), which involves resourcing a task t using provider y and may result in ($\boldsymbol{F}_y, grant$) or ($\boldsymbol{F}_y, deny$). In this way, an agent may build a model of the relationship between observable features of agents and the policies they are operating under. Subsequently, when faced with resourcing a new task, the policy model can be used to obtain a prediction of whether or not a particular provider has a policy that permits the provision of the resource. In this paper, we take this aspect of the research further by investigating *semantic-enriched decision trees* ($STree$), which extend C4.5 decision trees using ontological reasoning to explore how much domain knowledge can improve learning.

[3] For these algorithms we define a *learning interval*, ϕ, which determines the number of plan resourcing episodes (or interactions) an agent must engage in before building (or re-building) its policy model.

3.4 Learning from domain knowledge

In this paper, we argue that domain knowledge can be used to improve the performance of machine learning approaches. Specifically, in this section, we will describe how we can exploit domain knowledge to improve C4.5 decision trees.

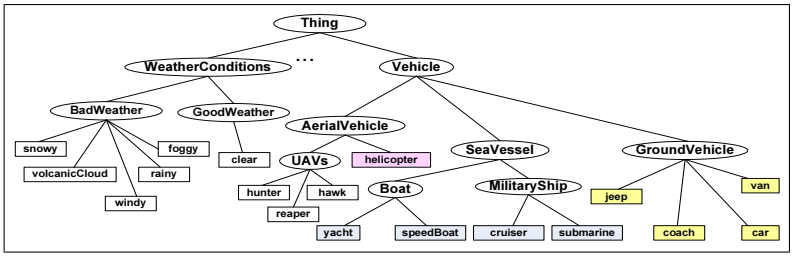


Fig. 6. A simple ontology for vehicles and weather conditions. Ellipsis and rectangles represent concepts and their instances respectively.

Domain Knowledge Domain knowledge consists of such background information that an expert (in a field) would deploy in reasoning about a specific situation. Semantic Web technologies allow software agents to use ontologies to capture domain knowledge, and employ ontological reasoning to reason about it [2]. Figure 6 shows a part of a simple ontology about vehicles and weather conditions. The hierarchical relationships between terms in an ontology can be used to make generalisation over the values of features while learning policies as demonstrated in Example 2. Policies are often specified using numerical features (e.g., vehicle price) and nominal features (e.g., vehicle type). Each nominal feature may have a large set of possible values. Without domain knowledge, the agent may require a large training set containing examples with these nominal values. However, domain knowledge allows agents to reason about to the terms unseen in the training set and learn more general policies with fewer number of training examples.

Example 2 *Suppose agent x in Example 1 has learned from previous interactions with agent y_1 that there is a policy that forbids y_1 from providing a helicopter when the weather is rainy, foggy, snowy, or windy. In addition, suppose agent x has learned from previous experience that agent y_1 is permitted to provide a jeep in these conditions. This information has little value for x if it needs a helicopter when the weather is not rainy, foggy, snowy, or windy but volcanic clouds are reported. On the other hand, with the help of the ontology in Figure 6, agent x can generalise over the already experienced weather conditions and expect that "agent y_1 is prohibited from providing helicopters in bad weather conditions". Such a generalisation allows x to reason about y_1's behavior for the cases that are not experienced yet. That is, with the help of the domain knowledge, agent x can deduce (even without having training examples involving volcanic clouds directly) that agent y_1 may be prohibited from providing a helicopter if there is an evidence of volcanic clouds in the region.*

C4.5 Decision Tree Algorithm In this section, we shortly describe the induction of C4.5 decision trees. Then, in the following section, we describe how domain knowledge can be exploited during tree induction.

The well-known C4.5 decision tree algorithm [6] uses a method known as divide and conquer to construct a suitable tree from a training set S of cases. If all the cases in S belong to the same class C_i, the decision tree is a leaf labeled with C_i. Otherwise, let B be some test with outcomes $\{b_1, b_2, \ldots, b_n\}$ that produces a partition of S, and denote by S_i the set of cases in S that has outcome b_i of B. The decision tree rooted at B is shown in Figure 7, where T_i is the result of growing a sub-tree for the cases in S_i. The root node B is based on an attribute that best classifies S. This attribute is determined using information theory. That is, the attribute having the highest *information gain* is selected.

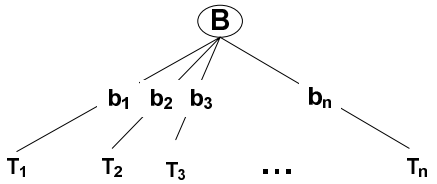


Fig. 7: Tree rooted at the test B and its breaches based on its outcomes.

Information gain of an attribute is computed based on *information content*. Assume that testing an attribute A in the root of the tree will partition S into disjoint subsets $\{S_1, S_2, \ldots, S_t\}$. Let $RF(C_i, S)$ denote the relative frequency of cases in S that belong to class C_i. The information content of S is then computed using Equation 1. The *information gain* for A is computed using Equation 2.

$$I(S) = -\sum_{i=1}^{n} RF(C_i, S) \times log(RF(C_i, S)) \tag{1}$$

$$G(S, A) = I(S) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} \times I(S_i) \tag{2}$$

Once the attribute representing the root node is selected based on its information gain, each value of the attribute leads to a branch of the node. These branches divide the training set used to create the node into disjoint sets $\{S_1, S_2, \ldots, S_t\}$. Then, we recursively create new nodes of the tree using these subsets. If S_i contains training examples only from the class C_i, we create a leaf node labeled with the class C_i; otherwise, we recursively build a child node by selecting another attribute based on S_i. This recursive process stops either when the tree perfectly classifies all training examples, or until no unused attribute remains.

Figure 8 lists 10 training examples, where $Type$, Age, and $Price$ are the only features. C4.5 decision tree algorithm makes induction only over numerical attribute values. However, it could not make induction or generalisation over the nominal attribute values (i.e., terms). For instance, a decision node based on the *price* test in Figure 9 can be used to classify a new case with price $\$250,000$, even though there is no case in the training examples with this price value. However, a new case with an unseen type, for instance a *submarine*, cannot be classified using the decision node based on the attribute $Type$.

#	Type	Age	Price	Class
1	Van	10	10,000	$grant$
2	Van	5	20,000	$grant$
3	Car	8	5,000	$grant$
4	Car	15	1,000	$grant$
5	Coach	2	200,000	$grant$
6	Yacht	20	300,000	$deny$
7	Yacht	2	500,000	$deny$
8	Speedboat	4	8,000	$deny$
9	Speedboat	15	2,000	$deny$
10	Cruiser	10	100,000	$deny$

Fig. 8: Training examples.

Semantic-enriched decision trees Here, we propose semantic-enriched decision trees ($STree$) built upon the subsumptions relationships between terms in the ontology. These relationships can be derived automatically using an off-the-shelf ontology reasoner [2]. The main idea of $STree$ is to replace the values of nominal attributes with

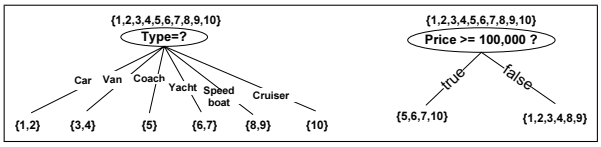


Fig. 9. Decision nodes created using the tests on $Type$ (on left) and $Price$ (on right).

more general terms iteratively during tree induction, unless this replacement results in any decrease in the classification performance.

Algorithm 1 Generalising values of nominal attribute A in training set S.

1: **Input :** S, A
2: **Output:** T
3: $g = G(S, A)$
4: $baned = \{Thing\}$
5: $terms = getAttributeValues(S, A)$
6: **while** $true$ **do**
7: **if** $\exists t$ **such that** $t \in T \wedge t \notin baned$ **then**
8: $t' = generalise(t)$
9: $T' = replaceWithMoreSpecificTerms(T, t')$
10: $s = replaceAttributeValues(S, A, T')$
11: **if** $G(s, A) = g$ **then**
12: $S = s$ **and** $T = T'$
13: **else**
14: $baned = baned \cup \{t\}$
15: **end if**
16: **else**
17: **break**
18: **end if**
19: **end while**

Algorithm 1 summarises how the values of A are generalised for S. First, we compute the original gain $G(S, A)$ (line 3). Second, we create a set called $banned$, which contains the terms that cannot be generalised further (line 4). Initially, this set contains only the top concept $Thing$. Third, we create the set T that contains A's values in S (line 5). While there is a generalisable term $t \in T$ (lines 6-18), we compute its generalisation t' using ontological reasoning (line 8) and create the set T' by replacing more specific terms in T with t' (line 9). If this term is an instance of a concept, then the generalisation of the term is the concept, e.g., $Boat$ is generalisation of $Yacht$. If the term is a concept, its generalisation is its parent concept, e.g., $SeaVessel$ is generalisation of $Boat$. For instance, let S be the data in Figure 8, then T would contain $Yacht$, $Speedboat$, $Cruiser$, Van, Car, $Coach$, and $Cruiser$. If Car is selected as t, t' would be $GroundVehicle$. In this case, T' would contain $Yacht$, $Speedboat$, $Cruiser$, and $GroundVehicle$. Next, we check if the generalisation leads to any decrease in the information gain. This is done by creating a temporary training set s from S by replacing A's values in S with the more general terms in T' (line 10) and then comparing $G(s, A)$ with the original gain g (line 11). If there is no decrease in the information gain, S and T are replaced with s and T' respectively; otherwise t is added to $banned$. We iterate through until we cannot find any term in T to generalise without any decrease in the information gain.

The output of the algorithm would be $\{SeaVessel, GroundVehicle\}$ for the examples in Figure 8, because any further generalisation results in a decrease in information gain. Hence, a decision node based on $Type$ attribute would be as shown in Figure 10 (left hand side). A new test case (11, $Submarine$, $40years$, $\$800,000$) would be

classified as $deny$ using this decision node, because a submarine is a sea vessel and all known sea vessels are labeled as $deny$. If the actual classification of the case is $grant$ instead of $deny$, the decision node would be updated as seen in Figure 10 (right hand side), because generalisation of $Submarine$ or $Cruiser$ now results in a decrease in the information gain.

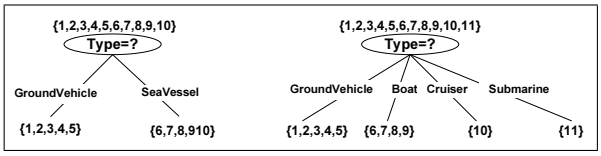


Fig. 10. Decision nodes using the generalisation of cases in Figure 8 (left hand side) and after the addition of a new case (11, $Submarine$, 40, 800, 000, $grant$) (right hand side).

4 Evaluation

In evaluating our approach, we employed a simulated agent society where a set of seeker agents interact with a set of provider agents with regard to resourcing their plans over a number of runs. Each provider is assigned a set of resources. Providers also operate under a set of policy constraints that determine under what circumstances they are permitted to provide resources to seekers. In the evaluation reported in this section, we demonstrate that it is possible to use domain knowledge to improve models of others' policies, hence increase their predictive accuracy, and performance. To do this, we consider two experimental conditions (i.e. closed and open). There are five features that are used to capture agents' policies, namely *resource type*, *affiliation*, *purpose*, *location*, and *day*. In the open scenario, each feature can have up to 20 different values, whereas only 5 different values are allowed in the closed scenario. In each scenario, six agent configurations (RS, SM, $C4.5$, kNN, SC, and $STree$) are investigated. In configuration RS, random selection is used. In SM, simple memorisation of outcomes is used. In $C4.5$, C4.5 decision tree classifier is used. In kNN, k-nearest neighbour algorithm is used. In SC, sequential covering rule learning algorithm is used. Lastly, in $STree$, agents use semantic-enriched decision trees to learn policies of others.

Seeker agents were initialised with random models of the policies of providers. 100 runs were conducted in 10 rounds for each case, and tasks were randomly created during each run from the possible configurations. In the control condition (random selection, RS), the seeker randomly selects a provider to approach. In the SM configuration, the seeker simply memorises outcomes from past interactions. Since there is no generalisation in SM, the *confidence* (or prediction accuracy) is 1.0 if there is an exact match in memory, else the probability is 0.5.

Figure 11 gives a graphical illustration of the performance of six algorithms we considered in predicting agents' policies in the closed scenario. The results show that $STree$, SC, kNN, $C4.5$ and SM consistently outperform RS. Furthermore, STree, SC and kNN consistently outperform C4.5 and SM. It is interesting to see that, with relatively small training set, SM performed better than C4.5. This is, we believe, because the model built by C4.5 overfit the data. The decision tree was pruned after each set of 100 tasks and after 300 tasks the accuracy of the C4.5 model rose to about 83% to

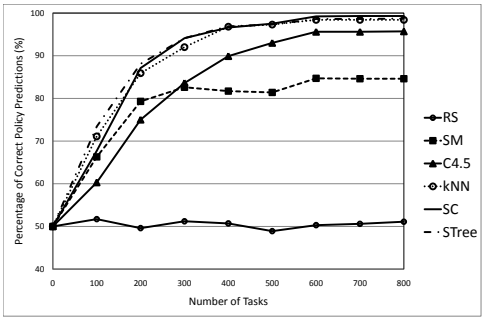
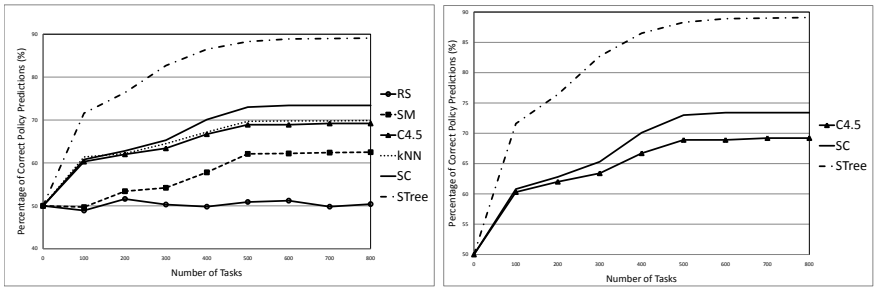


Fig. 11. The effectiveness of exploiting domain knowledge in learning policies (closed).

tie with SM and from then C4.5 performed better than SM. Similarly, STree performed much better than SC with relatively small training set. We believe, this is because STree takes advantage of domain knowledge and so can make informed inference (or guess) with respect to feature values that do not exist in the training set. After 400 tasks the accuracy of SC reached 96% to tie with STree. We believe, at this point, almost all the test instances have been encountered and so have been learned (and now exist in the training set for future episodes).



(i) (ii)

Fig. 12. The effectiveness of exploiting domain knowledge in learning policies (open).

Figure 12 illustrates the effectiveness of four learning techniques (C4.5, kNN, SC, and STree) and SM in learning policies in the open scenario. The result shows that the technique that exploits domain knowledge (STree) significantly outperforms the other techniques that did not. The decision trees (i.e. STree and C4.5) were pruned after each set of 100 tasks and after 300 tasks the accuracy of the STree model had exceeded 82% while that of C4.5) was just over 63%.

Tests of statistical significance were applied to the results of our evaluation, and they were found to be statistically significant by a t-test with $p < 0.05$. Furthermore, in both the open and closed scenarios, the learning technique that incorporates domain knowledge (STree) consistently yielded higher prediction accuracy (when training set is relatively small) than those without domain knowledge. These results show that exploiting domain knowledge in decision making can help increase the performance of agents. These results confirm our hypotheses, which state that exploiting appropriate

domain knowledge in learning policies mean that more accurate and stable models of others' policies can be derived more rapidly than without exploiting such knowledge.

5 Discussion

We have proposed an agent decision-making mechanism where models of other agents are refined through argumentation-derived evidence from past dialogues, and these models are used to guide future task delegation. Our evaluations show that accurate models of others' policies could be learned by exploiting domain knowledge. We believe that this research contributes both to the understanding of argumentation strategy for dialogue among autonomous agents, and to applications of these techniques in agent support for human decision-making.

Sycara *et al.* [5] report on how software agents can effectively support human teams in complex collaborative planning activities. One area of support that was identified as important in this context is guidance in making policy-compliant decisions. This prior research focuses on giving guidance to humans regarding their own policies. Our work complements the approach of Sycara *et al.* by allowing agents to support humans in developing models of others' policies and using these in decision making. Our approach extends decision trees with ontological reasoning. Zhang and Honavar have also extended C4.5 decision trees with Attribute-value taxonomies [7]. Their approach is similar to $STree$, but it does not allow ontological reasoning during tree induction. Unlike their approach, our approach can directly incorporate existing domain ontologies and exploits these ontologies during policy learning.

In future, we plan to extend other matching learning methods with domain knowledge and explore how much this extension improves policy learning and enhances agents' support for human decision-making.

References

1. Emele, C.D., Norman, T.J., Parsons, S.: Argumentation strategies for plan resourcing. In: Proceedings of AAMAS 2011. p. to appear. Taipei, Taiwan (2011)
2. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
3. McBurney, P., Parsons, S.: Games that agents play: A formal framework for dialogues between autonomous agents. Journal of Logic, Language and Information 12(2), 315 – 334 (2002)
4. Norman, T.J., Reed, C.: A logic of delegation. Artificial Intelligence 174(1), 51–71 (2010)
5. Sycara, K., Norman, T.J., Giampapa, J.A., Kollingbaum, M.J., Burnett, C., Masato, D., McCallum, M., Strub, M.H.: Agent support for policy-driven collaborative mission planning. The Computer Journal 53(1), 528–540 (2009)
6. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco, 2nd edn. (2005)
7. Zhang, J., Honavar, V.: Learning decision tree classifiers from attribute value taxonomies and partially specified data. In: Proceedings of the International Conference on Machine Learning (2003)

Toward a methodology for agent-based data mining and visualization

Elizabeth Sklar[1,2], Chipp Jansen[1,3], Jonathan Chan[1] and Michael Byrd[2]

[1] Brooklyn College, The City University of New York, USA
[2] The Graduate Center, The City University of New York, USA
[3] Hunter College, The City University of New York, USA
sklar@sci.brooklyn.cuny.edu,chipp@chipp.org,jonmchan@gmail.com,mbyrd1@gmail.com

Abstract. We explore the notion of agent-based data mining and visualization as a means for exploring large, multi-dimensional data sets. In Reynolds' classic flocking algorithm (1987), individuals move in a 2-dimensional space and emulate the behavior of a flock of birds (or "boids", as Reynolds refers to them). Each individual in the simulated flock exhibits specific behaviors that dictate how it moves and how it interacts with other boids in its "neighborhood". We are interested in using this approach as a way of visualizing large multi-dimensional data sets. In particular, we are focused on data sets in which records contain time-tagged information about people (e.g., a student in an educational data set or a patient in a medical records data set). We present a system in which individuals in the data set are represented as agents, or "data boids". The flocking exhibited by our boids is driven not by observation and emulation of creatures in nature, but rather by features inherent in the data set. The visualization quickly shows separation of data boids into clusters, where members are attracted to each other by common feature values.

1 Introduction

We are motivated to explore the notion of agent-based data mining visualization, taking inspiration from the *Artificial Life* and *Information Visualization* communities. Advances in computer graphics, processor speed and networking bandwidth over last decade have made possible the application of dynamic techniques for information visualization that were previously limited to high-end graphics laboratory settings. In addition, the rapid rise in popularity of certain types of data visualization environments, particularly those from the *Geographic Information Systems (GIS)* community, have made commonplace otherwise obscure techniques for examining vast multi-dimensional data sets. *Google Earth* [2, 8, 6] is one example, which has brought to the masses the notion of zoomable interfaces and allowed everyday computer users to explore 3-dimensional geographic data sets facilitated by, what are now considered to be, standardized

controls. Our work takes advantage of these trends by hypothesizing that when any multi-dimensional data set is projected onto 2 or 3 dimensions, it can be explored as if it were a geographic landscape. Such "data landscapes" could be studied dynamically in interesting ways if points in the landscape are represented as software agents that move in response to various stimuli.

The work presented here develops this idea, with the long term goal of providing a participatory agent-based data mining and visualization system in which the user and the system collaboratively explore a data landscape. Classical statistics and many machine learning methods are often fine (and the right choice) when the user has some notion of what is in their data set and how they want to analyze their data. Looking for a bell curve in a data set of student grades is an example where a standard statistical method, like computing mean and standard deviation, is an appropriate choice. Using K-means [9] when clustering data into a known number of groups is an example where a standard machine learning method is appropriate. But when the number of clusters and even the selection of features on which to cluster the data are unknown *a priori*, other techniques must be investigated. Our hypothesis is that a *participatory* system in this type of situation can take advantage of superior human skills for quick visual understanding and intuition, facilitating a user to easily identify and nudge an evolutionary data mining process into a desired direction.

This paper is organized as follows. Section 2 describes the related work in this area, which was inspired by seminal work on "flocking boids" [16]. Sections 3 details our approach. Section 4 presents results from applying our approach to a sample data set. Finally, we conclude in Section 5 with summary remarks.

2 Related Work

In 1987, Cliff Reynolds [16] produced the classic work on flocking in artificial systems. Reynolds' model focuses on graphical aspects, and the aim was to produce a realistic (from a graphical standpoint) simulation of a group of identical agents (which Reynolds calls "boids"). Each agent is given the same behavioral rules, which includes instructions for how to react to others in an agent's "neighborhood". These interaction instructions consist of three independent rules. First, there is a *separation* rule, which implements collision avoidance, preventing the agents from bumping into each other. Second, there is an *alignment* rule, which causes agents to move at the same velocity and heading as those around them. Third, there is a *cohesion* rule, which encourages agents to gravitate towards a common center. When put together, the composite set of agents exhibits emergent group behavior that looks like *flocking*—in the same way that schools of fish or flocks of birds or herds of land animals move together. This work has proven to be highly influential and has inspired most of the subsequent research in the area of artificial flocking.

Proctor and Winter [15] developed the notion of *information flocking*, about 10 years after Reynolds' work. Their aim was to visualize patterns of behavior of users visiting web sites. They simulated artificial "fish" and associated a fish

with each user, mining users' clickstreams to provide input to their simulation. A user clicking on a URL was interpreted as interest in the topic(s) displayed on the page. A matrix of user interests was updated, and the fish responded by grouping together—showing users who shared similar interests.

Moere [11] provides a comprehensive overview of decentralized data visualization work conducted in the two decades since Reynolds' original work. He divides the work in the field into three categories: *information particle* animation; *information flocking*; and *cellular ant* methods. The first category involves simulating a group of *information particles*, or *"infoticles"* (i.e., agents), in three-dimensional space. Agents react to *forces* in the system, moving in response to them. The forces can emanate from static points in the agents' artificial landscape, acting as fixed point attractors (or repellers), as well as from dynamic points—other agents. The forces influence the velocity and direction of movement of each agent. With an infoticle, data values are assigned to each agent. The forces acting on an agent are calculated according to the similarity (or dissimilarity) of the agents' data values in relation to those of other nearby agents or fixed-point attractors in the landscape.

The second category, *information flocking*, describes the method introduced by Proctor and Winter [15], discussed above. Picarougne *et al.* [12] extend the work of Proctor and Winter by introducing the notion of an "ideal" distance between agents that is proportional to the difference in feature values between the agents. They compare to the K-means clustering method and state that their results are similar. Another example that extends Proctor and Winter's work suggests using the visualization in tandem with other algorithms for data classification [1]. Moere [10] offers a nice extension to the Proctor and Winter work by applying the method to time-varying datasets and using financial stock market data as an example. The data values (e.g., prices or "quotes" of a given stock) are represented by agents, and the agents' values change over time. Moere introduces two additional behavior rules: *data similarity*, where agents stay close to others with similar data values; and *data dissimilarity*, where agents move away from others with different data values. The method highlights changes in "data behavior"—as data values change over time, agents that have similarly changing data values end up clustering together.

The third category combines ideas from *artificial ant systems* [4], *ant foraging* [3] and *cellular automata* [19]. The general idea is that artificial agents ("ants") move around a two-dimensional grid world and collect "food"—i.e., data. The aim is to bring like pieces of data together, emulating the way that ants gather food particles and deposit them in shared nests. The agents' movement is constrained by cellular-automata-like rules in which they respond only to neighboring grid cells. While an interesting alternative, many comment that this approach takes more time to reach a solution than other methods. The reason is that the agents' movements are constrained (by the cellular automata rules), and so even if an agent "knows" where it is going, it still has to find a path to get there. The advantage, however, is that forcing the agent to explore an indirect path will lead it to encounter other agents and new data (food) sources, which

may eventually result in a more robust solution (even if it takes longer to find). Handl and Meyer [7] review ant-based clustering algorithms, comparing results of different implementations and discussing approaches from an optimization perspective.

Cui *et al.* [20] employ a flocking-based algorithm for document clustering analysis. Each document object is represented as a "boid" and projects the document's TFIDF[4] vector onto a two-dimensional space. The authors compare their flocking-based clustering algorithm with K-means clustering and Ant clustering. They found that the flocking algorithm ran faster than the Ant algorithm, while K-means ran the fastest. However, the K-means method requires an *a priori* estimate of the correct number of groups, whereas the flocking and Ant-based methods do not. This makes the flocking algorithm the better choice for a dynamically changing data set, like the one used by the authors.

Picarougne *et al.* [13] describe a biologically inspired clustering algorithm, called *FClust*, which applies similar, but not identical, techniques to those of Proctor and Winter. The authors also model agents as representations of data points moving in a two-dimensional landscape. However, instead of applying three forces to the agents' motion, the authors apply two forces: one that attracts agents with similar data values and one that repels agents with dissimilar data values. They determine experimentally the threshold for similarity, showing visually that different threshold values produce different clustering results, as one would expect. They apply their method to several data sets and compare results to K-means clustering methods. They also describe a component of their system referred to as "interactive clustering" in which a user can select and label data elements on a visual display and the system will perform clustering on the user-selected classes. The authors define an "entropy" measure (inspired by Shannon [17]) for evaluating the stopping condition for the system.

The work we present in the remaining sections has been inspired and informed by these and other examples, all of which were found primarily in the *Artificial Life* and *Information Visualization* literature. Our approach is most similar to that of Moere [10] and Picarougne *et al.* [13]. However, we take an *agent-based* perspective. Each agent in our system can only compare its data values to others in its geographic neighborhood, much as a robot in a multi-robot team can only respond to teammates that are within range of its visual or range sensors or communication network. Where Moere and Picarougne *et al.* stress two behavior rules (essentially data similarity and dissimilarity), we stick with Reynolds' original three behavior rules (separation, cohesion and alignment); and we introduce an additional *meta-level* flocking step in which groups that are similar emit an attractive force and tend to move towards each other. The details are described in the next section.

[4] Term Frequency, Inverse Document Frequency—a common metric used in Natural Language Processing (NLP)

3 Approach

Our approach is built on the *information flocking* paradigm described in the previous section, with modifications highlighted below. We associate a "data boid", or agent, with each record in an n-dimensional data set (i.e., a data set with n features in each record). In our visual environment, the agent moves around a two-dimensional geographic landscape, and its position is influenced by the relative values of its data features compared with those of its neighbors. Iterating over a number time steps, or "frames", we animate the agents by first applying the three standard flocking rules, then *grouping* agents based on their feature value and geographic proximity, next executing a *meta-flocking* step that pulls similar groups together, and finally applying a *braking* factor as clusters of like agents converge.

The standard flocking procedure involves computing vectors representing three forces acting on each agent, followed by combining the three forces using a weighting scheme. The three force vectors are computed as follows:

- **Separation.** A steering vector is computed that points away from agents in the neighborhood with dissimilar feature values:

$$sep = \frac{1}{n} \sum_{i=1}^{n} d_i (P - P_i) \tag{1}$$

  where n is the number of agents in the neighborhood within a specified feature distance (Δ), d_i is the geographic distance to neighbor i, P is the (x, y) position of the agent performing the computation, and P_i is the (x, y) position of neighbor i.

- **Cohesion.** A steering vector is computed that points toward the center of the group of agents in the neighborhood with similar feature values:

$$coh = \frac{1}{n} \sum_{i=1}^{n} P_i \tag{2}$$

  where n is the number of agents in the neighborhood within a specified feature distance (Δ) and P_i is the (x, y) position of neighbor i.

- **Alignment.** A velocity vector is computed to match the average speed and heading of agents in the neighborhood with similar feature values:

$$ali = \frac{1}{n} \sum_{i=1}^{n} V_i \tag{3}$$

  where n is the number of neighboring agents within a specified feature distance (Δ) and V_i is the velocity of neighbor i.

The vectors are weighted and applied together to update the agent's position:

$$velocity = \psi \cdot sep + \alpha \cdot ali + \gamma \cdot coh \tag{4}$$

The weighting scheme amongst the three vectors is important, and there is a delicate balance between them. For the results presented here, we used default weights of $\psi = 1.5$, $\alpha = 1.0$ and $\gamma = 1.0$.

The standard flocking algorithm only considers geographic distance (d) between agents, whereas we use geographic distance to determine an agent's neighborhood and distance in feature space (Δ) to determine whether agents should be attracted to or repel their neighbors. The distance in feature space is computed as follows. Multi-featured data sets frequently contain a mixture of *categorical* and *quantitative* types of features. We compute the distance between individual features, based on their type and normalized to $[0 \ldots 1]$ (where 0 is most similar and 1 is most dissimilar), and then calculate the average over all features. The distance between two categorical feature values is:

$$\delta(a_i, b_i) = (a_i == b_i \ ? \ 1 : 0) \tag{5}$$

where a_i and b_i represent the values of the i-th feature for agents a and b, respectively. The distance between two quantitative feature values is:

$$\delta(a_i, b_i) = \frac{|a_i - b_i|}{max_i} \tag{6}$$

where max_i is the range of possible values for feature i. The overall feature distance between two agents is thus:

$$\Delta(a, b) = \frac{\sum_{i=1}^{n} \delta(a_i, b_i)}{n} \tag{7}$$

After the standard flocking rules are applied to all the agents, a *grouping* step takes place, followed by a *meta-flocking* step. Grouping is an organizational step in which the agents are partitioned into virtual groups based on their locations in geographic space. This is done using a recursive algorithm in which each agent looks at its neighbors (other agents within a fixed physical distance), and those neighbors look at their neighbors, and so on, adding to the group all neighbors (and neighbors' neighbors, etc.). Agents' positions do not change during the grouping step.

The meta-flocking step pulls together groups that have similar feature values. For each group, a set of feature values is calculated that represents the center of the group in feature space. For quantitative feature values, the center is the mean over all group members. For categorical feature values, the center is the mode (i.e., most popular) over all group members. A pairwise comparison is made between all groups' feature centers. A cohesion steering vector for the group is computed that points toward the geographic center of mass of other groups with similar feature values:

$$coh_g = \frac{1}{n} \sum_{i=1}^{n} M_i \tag{8}$$

where n is the number of groups within a specified feature distance (Δ) and M_i is the (x, y) position of the center of mass of group i. Note that this is different from the standard (agent-level) cohesion rule for two reasons. First, it only takes into account the feature distance between groups and does not use the geographic distance. Second, it compares with every group in the system, not just groups that are within close graphic distance proximity.

Finally a *braking* factor is applied to all members of groups whose feature values within the group are similar. As group membership settles, the system converges and the agents belonging to settled groups move more slowly and eventually halt.

As the clusters of agents evolve, the system computes two metrics to assess how the process is performing:

- *pctGood* computes the percentage of groups that fall within a pre-specified "goodness" threshold. The aim is for this value to converge to 1.0. The *goodness* of a group is a measure of the variance in group members' feature values. A goodness value close (or equal) to 0 indicates that the group members are closely matched; a higher goodness value indicates a more diverse group. Goodness is calculated as:

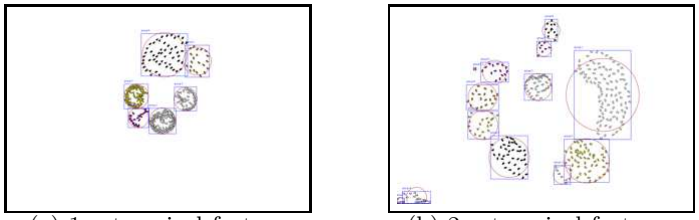
$$goodness = \sum_{i=1}^{n} \left(\sum_{f \in Q} \sigma(f) + \sum_{f \in C} (1 - \frac{match(f)}{n}) \right) \tag{9}$$

  where n is the number of agents in the group, Q is the set of each agents' quantitative features, C is the set of each agents' categorical features, $\sigma(f)$ is the standard deviation of (quantitative) feature f across all agents in the group, and $match(f)$ is the number of agents in the group with (categorical) feature f matching the mode for that feature in the group.

- *pctOverlap* computes the percentage of groups that have overlapping feature values. Since the system determines group membership dynamically, based on which agents have come within small graphical distances of each other while they float around in space, there is no guarantee that multiple groups with the same feature values will not appear. The overall aim is to find the minimal set of groups that have different feature values, since it is not desirable to have different groups that overlap in feature space. The aim is for this value to converge to 0.0.

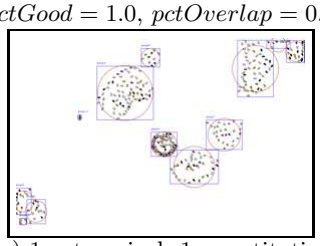
4 Experiments and Results

We implemented our approach in a prototype system, developed using Processing [14]. The system can run in an interactive mode, where the user selects features on which to cluster, and the system responds in real-time. The user can change feature selection during clustering, and the system will adjust. To assess the efficacy of our method for clustering, we ran a series of experiments in

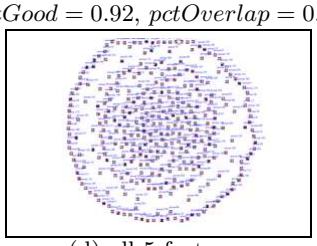


(a) 1 categorical feature
293 frames
$pctGood = 1.0$, $pctOverlap = 0.0$

(b) 2 categorical features
1000 frames
$pctGood = 0.92$, $pctOverlap = 0.08$



(c) 1 categorical, 1 quantitative
1000 frames
$pctGood = 0.83$, $pctOverlap = 0.08$

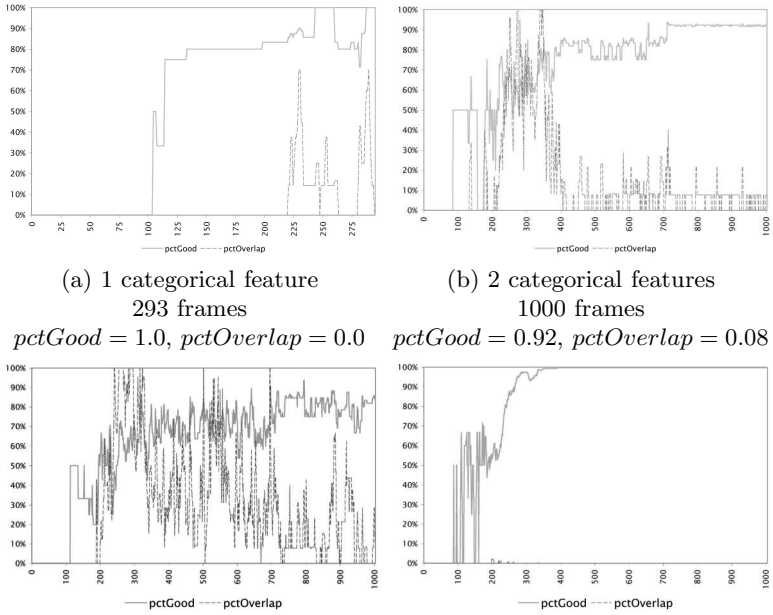


(d) all 5 features
1000 frames
$pctGood = 0.99$, $pctOverlap = 0.0$

Fig. 1. Screen shots at end of 4 different runs, each clustering on different sets of features, as indicated above.

a batch mode, using a 5-dimensional sample set of university enrollment data, gathered over a 4-and-a-half-year period. The data was partitioned into subsets, each representing one term; on average, there are 375 data records per term. Each record represents an instance of a student taking one class in a specific term. The fields in each record are: a unique identifier for the student (categorical feature), a course number identifying which class was taken (categorical feature), the student's gender (categorical feature), a numeric code indicating the student's ethnic origin (categorical feature), and the grade earned by the student in the class (quantitative feature). A subset of data for one term contains all the students who took classes in that term. A student who took more than one class in a given term is represented by multiple data records.

Figure 1 contains screen shots from 4 representative runs of the system, captured at the end of each run. The end of a run is either the point at which $pctGood = 1.0$ and $pctOverlap = 0.0$ or a maximum number of frames have elapsed. We used 1000 as the maximum number of frames for the runs presented here. The figure captions show the number of frames elapsed for each run, and the final values of *pctGood* and *pctOverlap*. Figure 2 shows the improvement in *pctGood* and *pctOverlap* rates through the course of each corresponding run. When clustering on one categorical feature value, the system quickly converges to the correct solution (in 293 frames); whereas in all the other cases, the system timed out after 100 runs. Even so, the percentage of "good" groups was high, ranging between 83% and 99%.



(a) 1 categorical feature
293 frames
$pctGood = 1.0$, $pctOverlap = 0.0$

(b) 2 categorical features
1000 frames
$pctGood = 0.92$, $pctOverlap = 0.08$

(c) 1 categorical, 1 quantitative
1000 frames
$pctGood = 0.83$, $pctOverlap = 0.08$

(d) all 5 features
1000 frames
$pctGood = 0.99$, $pctOverlap = 0.0$

Fig. 2. Improvement in evolution metrics: *pctGood* and *pctOverlap*

At the end of each run, we computed two scores to determine how well the clustering process worked: *within cluster* score and *between cluster* score. The "within cluster" score measures the disparity of the feature values in the cluster. The goal is to minimize the "within" cluster score and to maximize the "between" cluster score. The "within" cluster score is determined by calculating the feature distance (equation 7) from the representative center of the group in feature space (see the description of the meta-flocking step in Section 3) to each member of the group. The "between" cluster score is determined by calculating the average over all pairwise feature distances between representative (feature) centers of each group. Using these two metrics, we compared our results to two standard clustering algorithms from the data mining literature: K-means [9] and *Cobweb* [5]. We ran these algorithms using WEKA [18] on our same data set.

Figures 3 and 4 compare the results of our agent-based data visualization method to K-means and Cobweb. Each horizontal bar represents an average over multiple runs[5]. The shaded bands within each bar represent the percentage of runs in which either all three methods had the same (best) result (labeled

[5] The number of runs varied between 4 and 31, depending on the number of features being clustered.

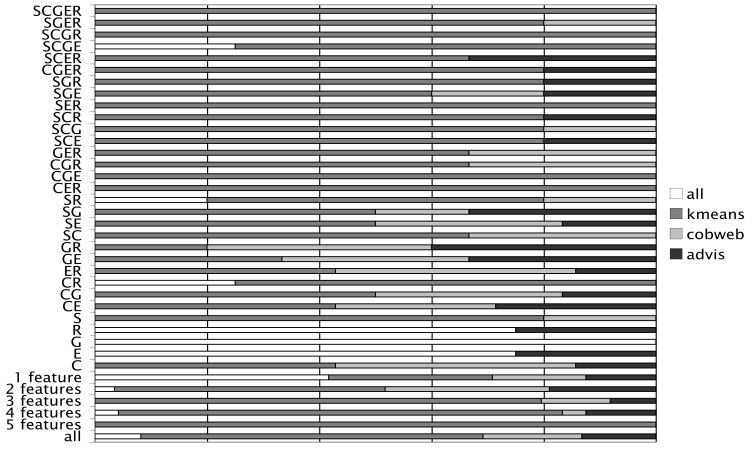


Fig. 3. "Within Cluster" score comparison

"all"), versus the percentage of runs in which each method had the same result (labeled "kmeans" or "cobweb" or "advis"). The bottom bar in each plot shows the average results over all combinations of features. Reading up from the bottom, the next 5 bars show the average results over combinations of 5, 4, 3, 2 or 1 feature. Then, the next 31 bars represent average results over specific combinations of features. In general, *K*-means produces the best results for "within" cluster score. For "between" cluster score, the results are split between Cobweb and our agent-based data flocking method. The table below quantifies how far off each method was when it was not categorized as the "best":

	kmeans	cobweb	advis
within cluster	0.87%	38.23%	47.39%
between cluster	1.58%	2.18%	1.12%

The values in the table are expressed as percentages and are computed as follows. For example, in the kmeans column, in the cases where *K*-means did not produce the best "within" cluster score, the difference is calculated between the score computed by *K*-means and the score computed by whichever method was best for that case. Then that difference is expressed as a percentage of the best cluster score—to give a sense of the proportion of the error. In the cases when our method (labeled "advis") does not produce the best "between" cluster score, it has the smallest error rate as compared with the cases where the other two methods are not the best.

5 Summary

We have described preliminary work in the development of an agent-based data mining and visualization method for clustering multi-dimensional data sets. Our

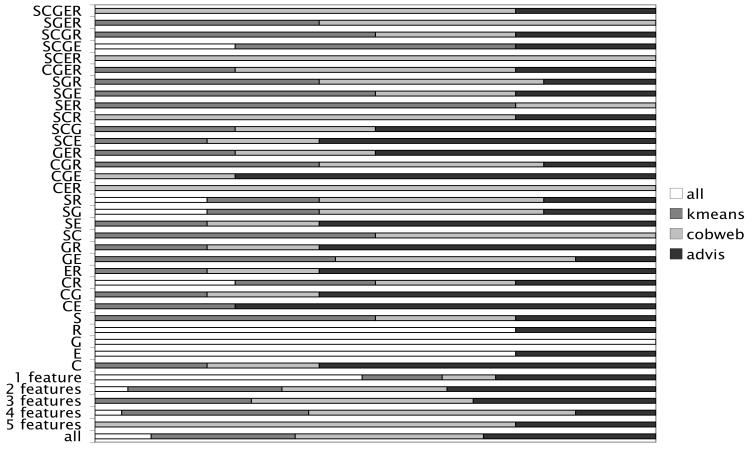


Fig. 4. "Between Cluster" score comparison

technique extends early work in the area of information flocking and introduces several strategies that help the system converge on a clustering task. The first strategy involves "grouping" and "meta-level" flocking steps in which groups of data boids are identified and nudged toward each other, based on the similarity between feature values amongst the groups. The second strategy is a "braking" adjustment that causes the data boids to slow down as the system converges on good clusters. Experiments were conducted on a sample data set, showing promising results for "between cluster" scoring as compared with standard techniques. The advantage that our method has over standard techniques is the ability to adapt during clustering to changes in clustering criteria.

We plan to apply evolutionary machine learning techniques to evaluate various parameter settings, including weights (equation 4), "goodness" threshold and geographic neighborhood distance, in order to improve clustering results. Eventually, the aim is for the system to dynamically explore various combinations of features while clustering, learning to converge on the set of features that offer the best cluster scores. In addition, we also will apply our method to real-time data streams, where the feature values in the data set change while the clustering process is occurring. Because our method is highly dynamic, it should be able to respond in near real-time (depending on the relative speed at which new data comes in compared to the time it takes clusters to form).

6 Acknowledgments

This work was supported by a grant from the National Science Foundation (#CNS-0722177).

References

1. Aupetit, S., Monmarché, N., Slimane, M., Guinot, C., Venturini, G.: Clustering and dynamic data visualization with artificial flying insect. In: Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation (GECCO), Lecture Notes In Computer Science. pp. 140–141. Springer-Verlag (2003)
2. Butler, D.: Virtual globes: The web-wide world. Nature 439, 776–778 (2006)
3. Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chrétian, L.: The dynamics of collective sorting: Robot-like ants and ant-like robots. In: From Animals to Animats: 1st International Conference on Simulation of Adaptative Behaviour. pp. 356–363 (1990)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics-Part B 26(1), 1–13 (1996)
5. Fisher, D.H.: Knowledge Acquisition Via Incremental Conceptual Clustering. Machine Learning 2, 139–172 (1987)
6. Google: Google earth. `http://earth.google.com` (2005)
7. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. Swarm Intelligence 1(2), 95–113 (2007)
8. Lisle, R.J.: Google earth: a new geological resource. Geology Today (2006)
9. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. pp. 281–297 (1967)
10. Moere, A.V.: Time-varying data visualization using information flocking boids. In: Proceedings of IEEE Symposium on Information Visualization. pp. 10–12 (2004)
11. Moere, A.V.: A model for self-organizing data visualization using decentralized multiagent systems. In: Prokopenko, M. (ed.) Advances in Applied Self-organizing Systems, Advanced Information and Knowledge Processing, vol. Part III, pp. 291–324. Springer (2008)
12. Picarougne, F., Azzag, H., Venturini, G., Guinot, C.: On data clustering with a flock of artificial agents. In: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI). pp. 777–778 (2004)
13. Picarougne, F., Azzag, H., Venturini, G., Guinot, C.: A new approach of data clustering using a flock of agents. Evolutionary Computation 15(3), 345–367 (2007)
14. Processing: `http://www.processing.org/` (2010)
15. Proctor, G., Winter, C.: Information flocking: Data visualisation in virtual worlds using emergent behaviours. In: Proceedings of Virtual Worlds. pp. 168–176. Springer-Verlag (1998)
16. Reynolds, C.W.: Flocks, Herds and Schools: A Distributed Behavioral Model. In: International Conference on Computer Graphics and Interactive Systems. pp. 25–34 (1987)
17. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal 27, 379–423 (1948)
18. WEKA: `http://www.cs.waikato.ac.nz/ml/weka/` (2010)
19. Wolfram, S.: Cellular automata as models of complexity. Nature 311, 419–424 (1984)
20. Xiaohui Cui, J.G., Potok, T.E.: A flocking based algorithm for document clustering analysis. Journal of Systems Architecture 52(8-9), 505–515 (2006)

Opinion Formation in the Social Web: Agent-based Simulations of Opinion Convergence and Divergence in Sub-Communities

Pawel Sobkowicz[1], Michael Kaschesky[1] and Guillaume Bouchard[2]

[1] Bern University of Applied Sciences, E-Government, Morgartenstrasse 2a, Bern, Switzerland
{ pawelsobko@gmail.com, michael.kaschesky@gmail.com }

[2] Xerox Research Center Europe, 6 chemin de Maupertuis, Meylan, France
{ guillaume.bouchard@xrce.xerox.com }

Abstract. Recent research on opinion formation in the social web – particularly blogs, comments, and reviews – investigates opinion dynamics but reaches opposite conclusions whether consensus formation occurs or not. To address this issue, a model of consensus formation is described that takes into account not only factors leading to convergence of opinions, but also those that strengthen their divergence. Nonlinear interplay between these tendencies might lead to interesting results, and decoupling the technical basis of the interactions (e.g. network dynamics) from the human perspective of opinions and sympathies (e.g. social dynamics) is at the core of such an approach.

Keywords: Opinion Research, Agent-based Simulations, Measurement, Economics, Reliability, Experimentation, Human Factors.

1 Introduction

Study of dynamics of opinion spreading is of important practical value in many diverse domains, from commercial marketing through judging effectiveness of government activities to counterterrorism. The field of study has been recently subject of intensive growth, drawing upon experiences from many disciplines. In addition to traditional sociological studies, tools borrowed from statistical physics, Bayesian statistical methods and agent-based computer modeling have become important. Moreover, the increased role played by the Internet based communication channels allowed the researchers to access massive amounts of data documenting people's activities, expressed sentiments and interactions. This changes the perspective of the research field, enabling the use of datasets obtained from the Internet to improve theoretical models to the point of usability in predicting social behavior.

Recent research on opinion formation in the social web – particularly blogs, comments, and reviews – investigates opinion dynamics but reaches opposite conclusions whether consensus formation occurs or not. Many standard opinion dynamic models postulate a form of averaging of opinions towards a mean value [6, 11], others assume that as a result of the interaction between two agents one of them changes his or her opinion by adopting the other opinion [28]. Unfortunately in large part these studies concentrate on mathematical formalisms or Monte Carlo simulations, and not on descriptions of real-life phenomena. However, situations where consensus formation does not occur have also been observed [26]. Indeed, prolonged interactions may even increase the rift between participants. This effect should be studied in more detail, as it possibly suggests modifications of the models of consensus formation in other situations. The need of bringing simulations and models closer to reality has been realized and voiced quite a few times [9, 24].

The persistence of differences of opinions exhibited in online political discussions [26] stands in contrast to observations of Wu and Huberman [29], who measured a strong tendency towards moderate views in the course of time for book ratings posted on Amazon.com. However, there are significant differences between book ratings and expression of political views. In the first case the comments are generally favorable and the voiced opinions are not influenced by personal feuds with other commentators. Moreover, the spirit of book review is a positive one, with the official aim of providing useful information for other users. This helpfulness of each of the reviews is measured and displayed, which promotes pro-sociality and 'good' behavior. In the case of political disputes it is often the reception in one's own community that counts, the show of force and verbal bashing of the opponents. The goal of being admired by supporters and hated by opponents promotes very different behavior than in the cooperative activities. For this reason, there is little to be gained by a commentator when placing moderate, well-reasoned posts – neither the popularity nor status is increased. These kinds of behavioral considerations must be taken into account by social modeling computer and particularly by models of opinion formation (for recent review see [4]).

Both history and literature are full of examples of undying feuds, where acts of aggression follow each other, from Shakespearean Verona families to modern political or ethnic strife. Observations of the Internet discussions should therefore be augmented by sociological data on flesh-and-blood conflicts and arguments, and the dynamics of opinion shifts. But even before such studies are accomplished the basic assumptions of the sociophysical modeling of consensus formation should be expanded. This is a very interesting task, because ostensibly we are faced with two incompatible sets of observations:

Using 'facts' and 'evidence' to support their viewpoint, participants in political discussions tend to reinforce their opinions, strengthening their resolve with each exchange. In this case, interactions do not lead to opinion averaging or switching.

Most participants do have well defined opinions. These must have formed in some way. Specific opinions on concrete events or people cannot be explained by genetic or cultural coding – they must be reached individually in each case.

We suggest the existence of two mechanisms: fast consensus formation within one's own group (including adoption of common, stereotyped views and beliefs); and persistence of differences between groups. An interesting experimental confirmation

of such phenomenon has been published by Knobloch-Westerwick and Meng [14], whose findings *"demonstrate that media users generally choose messages that converge with pre-existing views. If they take a look at 'the other side,' they probably do not anticipate being swayed in their views. [. . .] The observed selective intake may indeed play a large role for increased polarization in the electorate and reduced mutual acceptance of political views."*

In the remaining paper, we address the question of how individual characteristics of individual opinion adoption can be incorporated into opinion formation modeling. We describe a model of consensus formation that takes into account not only factors leading to convergence of opinions, but also those that strengthen their divergence. Nonlinear interplay between these tendencies might lead to interesting results, and decoupling the technical basis of the interactions (e.g. network dynamics) from the human perspective of opinions and sympathies (e.g. social dynamics) is at the core of such an approach.

2 Research Background

Recent years have brought significant interest in interdisciplinary studies, combining tools and methods known from physics with social analyses. Computational models of opinion formation often combine results derived from statistical physics with agent based simulations. Within a simplified framework focusing on selected aspects (such as interpersonal communication network, susceptibility to influences, contrariness etc.) it is possible to derive general trends of behavior of large societal groups. However, one of the major problems with opinion modeling by computer simulation is the lack of real-life data. Recent works reiterate the need for a real data by emphasizing real-life evidence over conceptual models and theory and prediction, explanation and guiding data collection in opinion modeling observation. Our task here is a challenge and innovation, using real data for large-scale social observations, analyzing its past evolution, and simulating potential future developments.

2.1 Research Problem

Opinion modeling based on sociophysics typically focuses on global properties of the modeled system. However, to be able to provide practical insights and real-life tools for social monitoring and policy formulation, models must include factors absent from simplified viewpoints. An important way of extending such an approach is via agent-based simulations, which allow integration of agent descriptions (characteristics) that are more detailed on the micro-level of individual behavior and therefore enable the combination and cross-fertilization of observations across several levels of social organization, from the individual micro-levels to the aggregated macro-levels of society.

Due to the rise of online media and user-generated content, opinion modeling has become very popular in recent years. The recent review by Castellano et al. summarizes many interesting results obtained via theoretical analyses and computer

modeling [4]. There are several models of opinion formation (Sznajd-Weron [28], Deffuant [6], voter model, Krause-Hegelsman [11]), and a significant number of research papers based on these approaches. Other approaches have been proposed by Nowak and Latané [18, 19] and Kacperski and Hołyst [12, 13]. Many of these models suffer from three important deficiencies:

- Most of them focus on achieving consensus and fail to account for divergences, such as persistence of minority groups, reluctance to accept policies, influence of individuals and subgroups etc.
- There is a fundamental problem in relating 'computer time' (usually measured in simulation steps) with real time for social phenomena, which is important for predicting dynamic real situations.
- Lastly, many current studies are abstracted from social reality, only a few attempts to explain concrete events or actions observed in the real world [24].

In addition, several works on propagation models for online information diffusion are based on simple infection models, which address one or more of the following issues:

- Link structure between blogs to track flow of information. Classification-based technique to model opinion propagation (called "infection") [1];
- Macroscopic topic model and microscopic propagation model [10];
- Tracing chain-letter data, propagation trees, small world network [17].

2.2 Existing Approaches

Cascading models are better suited to track and simulate large-scale information cascades, which is at the core of Simoo research. In cascading threshold models, an agent (person) will adopt an opinion if a specified fraction of its neighbors have adopted that opinion. The network is initialized by "seeding" a small number of agents with new information. If it spreads throughout a significant portion of the network, it is referred to as an information cascade, resembling the spread of ideas and fads in real-world complex systems. As said earlier, the foundations were laid out by Thomas C. Schelling [21]. Some recent approaches include:

- Modelling opinion formation as cascades based on efficient inference and learning algorithms [20];
- Cascades for modelling the propagation of recommendations for commercial products based on evolution of the recommendation network over time [13];
- Propagation of rumours and ideas based on a cascading model [16].

Research on the diffusion of information over time often assumes a prior model for information diffusion that is valid on every individual in the network. The goal of the inference technique is to recover the most probable graph that explains how information propagates in blogs, news feeds, emails, etc. There is a direct analogy between information diffusion and disease propagation. The most common approach is to model the population as a network of individuals fitting into one of three categories: Susceptible to the disease (S), currently Infected (I) and Recovered (R). In the case of information diffusion, the following analogy is made:

- S-state is the state of the individual before hearing the news (or discovering the new product),

- I-state corresponds to the state when user is keen to relay information/speak about the product, and
- R-state means that the user does not speak about the news or the product anymore.

When studying such models, researchers found interesting differences between classical disease diffusion and information propagation [10]:

- network structure on the internet is different because it typically has a power-law structure,
- transmission model for information must account for decay in the infection rate
- parameters of the diffusion model (such as information adoption rate) are dependent on individual characteristics, which are less important for disease contagion.

Estimations and probability intervals for parameters in complex systems, such as communication networks, are best calculated with probabilistic inference techniques, sometimes called Bayesian inference although the modeling is not always Bayesian (i.e. assuming a distribution over the parameters). The literature on social or network learning can be distinguished according to two criteria: whether learning is adaptive or myopic, and whether individuals learn from communication of exact signals or from the payoff of others, or simply from observing others' actions. Typically, probabilistic models focus on learning from past actions and communications, while most, but not all, myopic learning models focus on learning only from communications [3, 22]. In addition, research in engineering studies related problems, especially motivated by aggregation of information collected by decentralized sensors. These existing models and methods are used for modeling special aspects of social communicated networks.

2.3 Agent-based Modeling of Opinions

Among advances necessary to overcome existing shortcomings is providing better correspondence between parameters describing computer agents used in simulations and psychological and sociological characteristics of human participants, relevant to specific situations. Such mapping will account for, among others:

- correlations between opinions held on various subjects (multidimensional opinion description);
- dependence of individual opinion changes on pre-existing emotional states and on emotional formulation of the messages and policies;
- variances of attention and of communication capacities of individuals;
- tendencies to preserve one's own opinions and selective attention to views differing from them.

Therefore, the objectives of the proposed research approach are as follows:

- opinion diffusion model, based on advanced agent-based modelling that could be used in variety of policy situations, especially in the context of analysis of data obtainable from online media;

- analyze which factors describing individual and social situation are relevant for opinion change modeling, by comparing simulations with observations and analyses;
- baseline computational algorithms for simulations of opinion formation and diffusion;
- statistical learning algorithms for effective opinion simulations with predictive capabilities.

There is a rich literature on agent-based modeling of opinions, but the main interest is often in finding conditions which lead to a society reaching unanimous opinion, either via internal interactions or due to external influences. For this reason the field is often called 'consensus formation'. On the other hand, the real world examples of societies where full consensus is present are quite rare. In fact, where important issues are at stake, we observe extremely stable situations of opinion splits, ranging from relative parity (for example in political setup of many countries) to configurations where small minority persists despite strong social pressures (for example extremist views).

Our goal is to study opinion formation taking into account some processes that are usually omitted, which, in our opinion play a crucial role in real social situations and increase the stability of split opinion configurations. We study changes of opinion in conjunction with other processes, such as segregation. Current 'standard' approaches to opinion formation modeling focus on changes of individual opinions due to encounters between agents. It is surprising that the tendency of voluntary social separation due to attitude or opinion similarities and differences, well known and recorded even in folklore ('birds of a feather group together') has been neglected in opinion simulations so far. Instead of the choice of withstanding the arguments of an opponent or surrendering to them, most of us have the choice of severing the relationships with them. This results in relatively closed sub-societies, with little interaction between them, especially when the issues under study are of crucial value. A classical example of social modeling study invoking the separation tendencies is the famous Schelling model of social segregation.

It should be noted that the increased reliance of society members and organizations on Internet channels of communication provide unprecedented levels of flexibility of association. People are free to join groups of interest and express their views. The perceived anonymity and lack of face-to-face contact encourages expressions of extreme opinions. This leads to situation in which social networks, in addition to these built on similarity of opinions and interests, may also be built on differences and even hate [26]. Thus, social segregation, as seen through the Web media is more complex than traditional modals allow.

Figure 1 below shows an example of changes in simulated opinion distribution and social network topology leading to opinion diffusion and network rearrangement. While the predominant opinion reaches an increasing number of previously undecided participants, minorities may remain untouched, separated from the main part of society. The left of Figure 1 shows the initial social state with a large majority of undecided participants without a specified opinion on the issue (grey), and smaller groups of proponents and opponents concerning the issue (respectively light grey and black). The social links enabling information flow and opinion influences exist

between all groups. The right of Figure 1 shows the final state of simulated opinion diffusion with most of the participants having adopted the proponents' position. However, small minorities persist at the outskirts of society, mainly because they have severed and cut most of the information links with the majority [24]. Their existence and vitality can be monitored for example, via internet media activity, especially when filtering for high negative emotions. This enables policymakers to adjust implementation to better integrate dissenters and communication to better target expected benefits and consequences, by monitoring the effects of activities of individual persons and leaders [25].

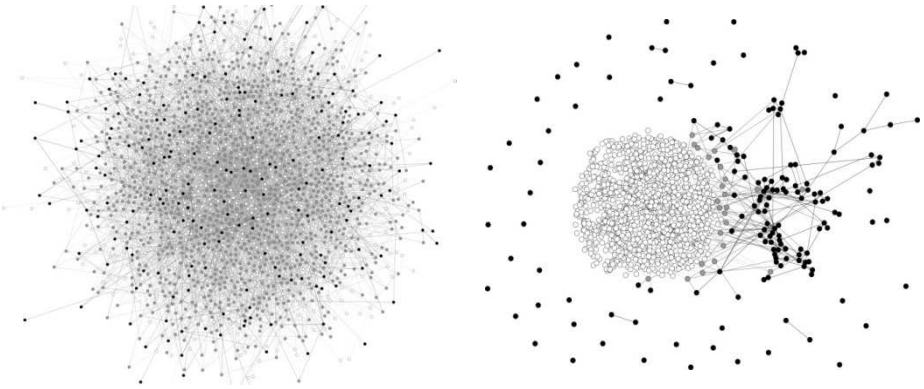


Figure 1: Initial state with large majority undecided (left) and final state of simulated opinion diffusion with majority adopting a mainstream opinion (right)

3 Simulation Model with Predictive Capabilities

The simulations presented here allow computer agents to cut the social links with those they disagree with and form a new links with agents sharing the same opinion. This changes the network structure. To take into account the fact that in real societies some links cannot be broken (e.g. family or work relationships) we have simulated situations where certain percentage of the links remain *static*, while the rest are *free*, allowing changes in topology of the social network.

The first novel aspect of the simulations is direct inclusion of agents with no preferred opinion (neutral agents). This allows significant change from models where only committed agents are present, changing both the social network and opinion change dynamics. Appropriate real life examples of application of the simulation include political preferences and highly controversial opinions on topics such as abortion or evolution. Within the model, the strength of influence between agents decreases with their social separation, reflecting the fact that our opinions are swayed less by remote acquaintances or strangers than by the closest associates.

Secondly, the opinion of a given agent may be changed in reaction to perceived cumulative social opinion of others, corresponding to a properly averaged 'peer

pressure' rather than the individual encounters. Many of the classical models have stressed the importance of such individual contacts in opinion changes of agents, but in our belief the constant background of perceived opinions, resulting from numerous meetings (not necessarily related to the issue in question) and information on opinions held by other society members is more relevant. In a way, this can be described as each agent continuously measuring and responding to the 'discomfort' due to difference between its opinion and properly averaged opinions of other agents. In addition to this intrinsic pressure it is possible to simulate, via simple parameterized factor the external influence, aimed to describe propagandist efforts.

The main features of the model can be summarized as follows [see 25]:

- A set of N interacting agents forms a society. Agents are connected to form a scale-free Albert-Barabási network. Such networks are typical for many social environments, exhibiting fat-tailed, power-law distribution of social connectivity. One of the most important features is presence of highly connected hubs, which significantly influence information and opinion flow within society. Such networks were found in many empirical studies e.g. [2,8,7,5].
- Each agent i has, at a given time, his 'opinion' $\sigma(i)$ (+1 , −1 or 0). The addition of the class of neutrals forms the crucial enhancement of the model.
- The network is assumed to have a dynamic character. Depending on simulation parameters, certain ratio of links is assumed to be *static*, corresponding to family or workplace connections, which are difficult to break even if there is a difference in opinions of the connected agents. The rest of the links are *free*, and may be cut off when the agents have conflicting (+1/−1) opinions. To preserve the general network properties the process of destruction and creation of new links is performed in a way that keeps the number of connections constant. The creation of replacement links may happen within strict similarity scenario: new links form only between agents with the same opinion (+1/+1 and −1/−1) or in promiscuous scenario, when links may form between agents that are not in conflict.
- Influence of agents on other agents is governed by their respective opinions as well as on their social distance. We measure this distance simply via the number of links separating the agents in the network. This is described in detail in later part of the paper.
- Opinion of an agent is changed due to combined influence of all other agents. To describe the natural human tendency of resisting such external pressures, we have added in our simulations additional factor s_o, describing the influence of an agent on itself, always opposing the direction of external pressures.

As the starting distribution of all the above mentioned parameters is random, it has little connection with real societies, with their correlated structures and patterns of opinions. To keep the model closer to reality, we start with a relatively small number of time steps (<5000, corresponding to 2.5 time steps per agent), during which the agents adjust the network links but are not allowed to change their opinions. This pre-separates (if there are free links) the network in accordance with initial random distribution of opinions. The reason for such initial phase is to avoid spurious results that arise from fully random distributions. For simulations where all the links are assumed static, this phase simply does not change anything. Later simulations involve

much larger number of steps (>200 steps per agent). In each of them first a randomly chosen agent is allowed to change opinion and then the same agent may adjust its network neighborhood by cutting a link with an opponent and adding one with a friendly agent. These two key processes of the simulation, modeling opinion adoption of agents and modeling network ties on opinion adoption, are described below.

3.1 Modeling Opinion Adoption of Agents

For a randomly chosen 'initializing' agent *i* we calculate the network separation for all other agents and separate them into the four spheres *S1* to *S4*. These are composed of, respectively, the direct neighbors, the second neighbors, third neighbors and the rest of the society. The strength of the influence of an agent *j* on *i* depends on the sphere that *j* belongs to. For each agent j we calculate also a modification factor which reflects our human tendency to give more value to opinions of those who agree with us and to give less credibility to contrary opinions. Thus $f(i,j)=2$ if $\sigma(i)=\sigma(j)$ and $f(i,j)=1$ otherwise. In Figure 1 below, *S1* corresponds to nearest neighbors, *S2* to next nearest neighbors etc. *S4* holds the large majority of the "rest of society". Influences of agents within each sphere are averaged and normalized, and the relative importance of the spheres is described by geometrically decreasing series with ratio *d*. In this work we have used $d=0.3$, which leads to 70.5 percent of the weight of the influence originating from the immediate sphere of neighbors *S1*.

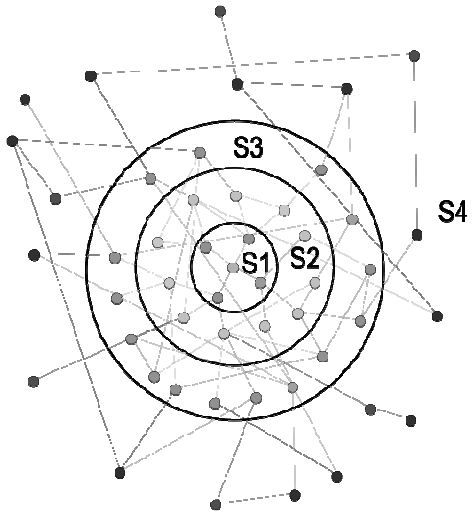


Figure 2: Schematic division of social network into four spheres, following the distance from the chosen agent *i*.

For each of the spheres S_K we calculate the averaged, normalized influence over agent *i* as follows:

$$I_K = \frac{\sum_{j \in S_K} f(i,j)\sigma(j)s(j)}{\sum_{j \in S_K} f(i,j)s(j)} \quad (1)$$

where $s(j)$ measures individual strength of agent *j*. The denominator acts to normalize I_K, so that if all agents *j* within S_K are of the same opinion, then their effect has the maximum absolute value $|I_K|=1$. Allowing selected agents to have $s(j)$ much larger than the rest provides grounds for modeling leaders.

The overall influence of the society on agent *i* is then calculated as a weighted sum of I_K. To reflect the observation that close acquaintances have much more influence than strangers we have assigned geometrically decreasing weights, for the spheres of increasing distance. The total influence of the society on agent *i* is as follows:

$$T(i) = \sum_{K=1}^{4} I_K \, d^{K-1} \frac{1-d}{1-d^4} \quad (2)$$

where $d<1$ and the last factor serves to provide normalization in addition to social pressure. Choice of *d* determines how important are close acquaintances in influencing agent's opinion, for example choice of $d=0.3$ leads to 70.5 percent of influence coming from the nearest neighbors, and only about 2 percent from sphere S_4, containing the large number of agents forming the "rest of society". It is possible to include in *T(i)* and additional element describing external influence, such as media coverage, government propaganda etc. applied independently of agent-to-agent contacts. This is done by adding a constant term h, the same for all agents. The change of agent's opinion is determined by comparing the external influences of the society and propaganda with self influence factor s_o, which determines the general susceptibility of agents in simulation to such influences. It is one of the main simulation parameters, upon which the stability of social mix depends to a large degree.

3.2 Modeling Network Ties on Opinion Adoption

The second step in simulations is the modification of network connections. To maintain the number of links constant we have alternated cutting and adding links. The first of these processes is very simple. Keeping the previously chosen agent *i*, we pick another, 'target' agent *t* randomly from the set of nearest neighbors that disagree with *i*. The link between *i* and *t* is then cut off. Obviously, if all *i's* neighbors agree with it, then we move to next simulation round.

Creating new links is more complex. The target agent *t* is not chosen totally at random among the spheres but rather with probability decreasing on social distance between *i* and *t*, so that probability of forming a new link with agents from further spheres decreases geometrically. Again, we chose this form for conceptual simplicity,

with the aim of getting a simple equivalent to the observations that we meet more often with our close social neighbors. The newly formed link between *i* and *t* is treated as a free one. The resulting network preserves, with only minor deviations, the general degree distribution characteristic for *AB* networks.

When there are no neutral agents in the simulations the process leads very quickly to separation of subnetworks of +1 and −1 opinion holders if the free links are dominant. As Figure 3 below shows, there are only a few links between agents supporting different opinions, dividing the society into separate parts, with weak chances of achieving consensus of opinions, as each agent is surrounded socially by proponents of his own view. While the average number of links per agent remains unchanged and only minor changes in degree distribution are observed, other relevant network properties change radically, for example the shortest path distribution and clustering coefficient. This is especially important when all the links are free, because +1 and −1 opinion holders segregate to separate communities. If we calculate the network properties of these communities separately, then in the case of large majority we observe that there is similarity to the original configuration in such characteristics as clustering coefficient. On the other hand, the minority has much larger fraction of disconnected agents, so that there is significant difference in related network characteristics.

As Figure 4 shows, even if a third of links between the separate communities are held static, there are more links between supporters of opposite views but social division remains. When including neutral agents into the simulations, significant changes in the network dynamics are observed. As Figure 5 shows, neutral agents, marked in gray serve as a link between the opposing opinion holders, shortening the average communication paths, even though there are no direct links between the two communities. In short, presence of neutrals acts as social glue between the committed groups, even when all links are free.

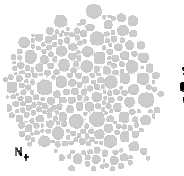


Figure 3: Social network resulting from simulation where no neutral agents are present and all links are free.

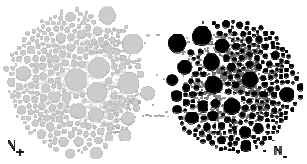


Figure 4: Social network resulting from simulation where no neutral agents are present and 1/3 of the links are static.

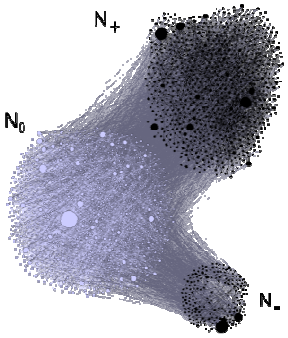


Figure 5: Social network resulting from simulation where neutral agents are present.

4 Conclusion

Changes in the social network influence opinion dynamics. In comparison to situations where only committed +1 and -1 agents are present [25], the range of situations where a mix of opinions persists within a society is extended. Depending on the s_o values and the ratio of free links within the network, we observed interesting regimes of opinion dynamics. For large enough s_o, the initial social configuration remains stable, with only a few conversions. At the other extreme, for small s_o, neutrals are quickly converted to majority opinion, but the time gained allows the opposing minority to separate itself from the rest of society. Interesting behavior may be observed for intermediate values.

In a first phase, described by the exponential function $a - b\ exp\left(-\frac{x}{t}\right)$, some neutral agents are converted by the majority while the number of minority agents remains almost constant. At some stage a second process may appear, described by logistic function centered at t_o. During the second phase almost all neutrals adopt the majority opinion. The tempo of the change is relatively constant between various runs of simulations, but the onset of this process may varies over very wide range of values, so that some simulations remain in a three state society (such as the one depicted in Figure 5) even after many hundreds of thousands of time steps.

The capacity to provide nontrivial behavior by the simulation model is an indication that despite model simplicity, there are enough features to suggest its usability for practical applications of understanding social behavior. Of course there are still many issues related to precise mapping of simulation environment to real life situations, but it has to be remembered that the ultimate goal of computer simulations is not in models *per se*, but in enabling to ask the right questions, improve observations of social phenomena and finally to provide predictions for evolution of specific social cases [13]. For these reasons the current model still needs improvements based on such mapping of simulation parameters to real life.

References

1. E. Adar & L.A. Adamic. Tracking information epidemics in blogspace. IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), 2005.
2. R. Albert and A. L. Barabási. Statistical Mechanics of Complex Networks. *Review of Modern Physics*, 74:67–97, 2002.
3. S. Bikhchandani, D. Hirshleifer and I. Welch. Learning from the Behavior of Others: Conformity, Fads, and Informational Cascades. *Journal of Economic Perspectives*, 12(3), 151-170, 1998.
4. C. Castellano, S. Fortunato & V. Loreto. Statistical physics of social dynamics. *Rev.Mod.Phys.* 81, 591-646, 2009.
5. A. Chmiel, K. Kowalska, and J.A. Hołyst. Scaling of human behavior during portal browsing. *Physical Review E*, 80(6):66122, 2009.

6. G. Deffuant, D. Neau, F. Amblard, and G. Weisbuch. Mixing beliefs among interacting agents. *Advances in Complex Systems*. 3, 87–98, 2000.
7. F. Ding and Y. Liu. Modeling opinion interactions in a BBS community. *The European Physical Journal B.* 78(2), 245–252, 2010.
8. S.N. Dorogovtsev and J.F.F. Mendes. Evolution of networks. *Advances in Physics*, 51:1079–1087, 2002.
9. J.M. Epstein. Why model? *Journal of Artificial Societies and Social Simulation*. 11(4), 12, 2008.
10. D. Gruhl, R. Guha, D. Liben-Nowell, A. Tomkins. Information diffusion through blogspace. Thirteenth International World Wide Web Conference, ACM Press, New York, NY, May 2004.
11. R. Hegselmann and U. Krause. Opinion dynamics and bounded confidence models, analysis, and simulation. *Journal of Artifical Societies and Social Simulation (JASSS).* 5(3), 2002.
12. K. Kacperski and J.A. Hołyst. Phase transitions as a persistent feature of groups with leaders in models of opinion formation. *Physica A*. 287, 631–643, 2000.
13. Kacperski, K. and Hołyst, J.A.. Opinion formation model with strong leader and external impact: a mean field approach. *Physica A*. 269, 511–526, 1999.
14. S. Knobloch-Westerwick, J. Meng, *Communication Research*. 36, 426, 2009.
15. J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. ACM Trans. Web, May 2007.
16. J. Leskovec, J, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Patterns of cascading behavior in large blog graphs. Society of Industrial and Applied Mathematics-Data Mining. Minneapolis, April 2007.
17. D. Liben-Nowell and J. Kleinberg. Tracing information flow on a global scale using internet chain-letter data. Proceedings of National Academy of Sciences of the USA, 105(12), 2008.
18. A. Nowak and M. Lewenstein. Modeling Social Change with Cellular Automata. In R. Hegselmann, U. Mueller, and K.G. Troitzsch (eds), Modelling and Simulation in the Social Sciences From A Philosophy of Science Point of View, 249–285. Kluver, Dordrecht, 1996.
19. A. Nowak, J. Szamrej, and B. Latané. From Private Attitude to Public Opinion: A Dynamic Theory of Social Impact. *Psychological Review*. 97(3), 362–376, 1990.
20. M. G. Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2010.
21. T. C. Schelling. Hockey helmets, concealed weapons, and daylight saving: a study of binary choices with externalities. *The Journal of Conflict Resolution*. 17(3), September 1973.
22. L. Smith and P. Sorensen. Pathological Outcomes of Observational Learning. *Econometrica*. 68(2), 371-398, 2000.
23. P. Sobkowicz. Effect of leader's strategy on opinion formation in networked societies with local interactions. *International Journal of Modern Physics C (IJMPC)*. 21(6): 839-852, 2010.
24. P. Sobkowicz. Modeling opinion formation with physics tools: call for closer link with reality. *Journal of Artificial Societies and Social Simulation*. 12(1), 11, 2009.
25. P. Sobkowicz. Studies of opinion stability for small dynamic networks with opportunistic agents. *International Journal of Modern Physics C (IJMPC)*. 20(10), 1645–1662, 2009.
26. P. Sobkowicz and A. Sobkowicz. Dynamics of hate based Internet user networks. *The European Physical Journal B.* 73(4), 633–643, 2010.
27. T.F. Smith and M.S. Waterman. Identification of Common Molecular Subsequences. *J. Mol. Biol.* 147, 195-197, 1981.
28. K. Sznajd-Weron and J. Sznajd. Opinion Evolution in Closed Community. *Int. J. Mod. Phys. C*. 11, 1157–1166, 2000.
29. F. Wu and B.A. Huberman, in Proceedings of the Workshop on Internet and Network Economics (2008).

Enhancing Agent Intelligence through Evolving Reservoir Networks for Power Load and Settlement Price Predictions in Power Stock Markets

Kyriakos C. Chatzidimitriou, Antonios C. Chrysopoulos, Andreas L. Symeonidis, and Pericles A. Mitkas

Electrical and Computer Engineering Department
Aristotle University of Thessaloniki
GR 54124, Thessaloniki, Greece
{kyrcha,achryso}@issel.ee.auth.gr
{asymeon,mitkas}@eng.auth.gr
http://issel.ee.auth.gr

Abstract. *Time Series Analysis* has bloomed during the last half of the century as a result of the need for reliable methods to estimate and predict the pattern, or behaviour, of events or systems. In recent years, *Time Series Prediction* and clustering have been employed in hyperactive and evolving environments, where temporal data play an important role. *Power Stock Markets* are such highly dynamic and competitive auction environments, additionally perplexed by constrained power laws in the various stages, from production to transmission and consumption. As with all real-time auctioning environments, the limited time available for decision making provides an ideal testbed for time series prediction. Within the context of this paper we employ *Recurrent Neural Networks* in the form of *Echo State Networks*, in order to generate power load and settlement price prediction models, in typical *Day-ahead Power Markets*. The Echo State networks are encapsulated inside Cassandra platform, a general-purpose *Multi-Agent System* that exploits *Data Mining* techniques, and are autonomously adjusted to the problem at hand using *Neuroevolution* techniques, in order to increase the autonomy of the platform and minimize user involvement. The system has been tested in a real-world scenario, that of the Greek Energy Stock Market.

Keywords: Data Mining, Power Stock Markets, Reservoir Computing, Multi-Agent System, Neuroevolution

1 Introduction

Agent Technology (AT) has been successfully employed in various aspects of real-world trading and electronic markets [8]. In highly dynamic, uncertain and multi-player markets, such as (Power) Stock Markets, decisions have to be made continuously within limited time, while data are generated at extreme rates.

Thus, agents are considered a suitable candidate for building efficient trading mechanisms, where their intelligence is based on algorithms that are able to adapt to the data at hand, using no or limited user input.

To this end and based on authors' prior work, we argue that an agent developed can employ Data Mining (DM) techniques, in order to extract useful nuggets of knowledge that could give him/her a predictive advantage over other competitors. In the case of stock markets, one could apply time series analysis on data in order to either identify the nature of the phenomenon represented by the sequence of observations or forecast (predict future values of the time series variable). In each of these cases it is necessary that the pattern of observed time series data is identified and more or less formally described. Once the pattern is established, we can then interpret and integrate it with other data (i.e., use it in our theory of the investigated phenomenon, e.g., seasonal commodity prices). Regardless of the depth of our understanding and the validity of our interpretation (theory) of the phenomenon, we can extrapolate the identified pattern to predict future events.

To this end we have fused AT and DM and developed a Multi-Agent System (MAS) capable of efficiently handling the deluge of available data and of practicing various DM methodologies, in order to reach what seems to be an efficient prediction of the prices of goods of interest. The testbed for our platform was the Greek Power Stock Market, which is a dynamic, partially observable environment, giving room for applying a wide variety of strategic approaches. The fact that each decision made affects instantly the future moves or decision of the platform and the Stock Market itself, makes this undertaking really challenging.

In order to capture the temporal and non-linear dynamics of the Power Stock Market signals, we employed *Echo State Networks* (ESNs), a neural network function approximator. In order to promote the autonomy of the platform, networks are adapted using neuroevolution. Short-term predictions are made for load and price time-series and are tested against standard regression techniques previously employed in the Cassandra system. Results with respect to load forecasting were excellent, while for price forecasting, which is a much more complex time-series, promising.

2 Technologies Overview

In this section we provide a brief overview of the constituents of the proposed learning method, as well as of the application domain.

2.1 Echo State Networks and NEAR

The idea behind *reservoir computing* (RC) and in particular Echo State Networks (ESNs) [9] is that a random *recurrent neural network* (RNN), created under certain algebraic constraints, could be driven by an input signal to create a rich set of dynamics in its reservoir of neurons, forming non-linear response signals. These signals, along with the input signals, could be combined to form

the so-called *read-out function*, a linear combination of features, $y = \mathbf{w}^T \cdot \phi(\mathbf{x})$, which constitutes the prediction of the desired output signal, given that the weights, w, are trained accordingly.

A basic form of an ESN is depicted in Figure 1. The reservoir consists of a layer of K input units, connected to N reservoir units through an $N \times K$ weighted connection matrix W^{in}. The connection matrix of the reservoir, W, is an $N \times N$ matrix. Optionally, an $N \times L$ backprojection matrix W^{back} could be employed, where L is the number of output units, connecting the outputs back to the reservoir neurons. The weights from input units (linear features) and reservoir units (non-linear features) to the output are collected into an $L \times (K+N)$ matrix, W^{out}. For this, the reservoir units use $f(x) = tanh(x)$ as an activation function, while the output units use either $g(x) = tanh(x)$ or the identity function, $g(x) = x$.

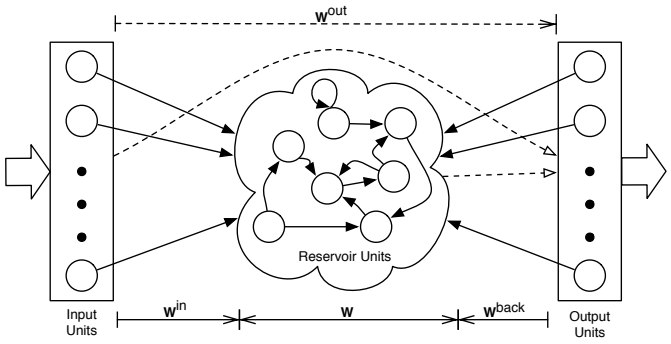


Fig. 1. A basic form of an ESN. Solid arrows represent fixed weights and dashed arrows adaptable weights.

One may refer to [9, 10] for best practices for generating ESNs, in the sense of procedures for generating the random connection matrices W^{in}, W and W^{back}. These could be briefly summarized in the following: (i) W should be sparse, (ii) the mean value of weights should be around zero, (iii) N should be large enough to introduce more features for better prediction performance, (iv) the spectral radius, ρ, of W should be less than 1 to practically (and not theoretically) ensure that the network will be able to function as an ESN. Finally, a weak uniform white noise term can be added to the features for stability reasons.

In current work, we consider discrete time models and ESNs without backprojection connections. As a first step, we scale and shift the input signal, $\mathbf{u} \in \mathbb{R}^K$, depending on whether we want the network to work in the linear on the non-linear part of the sigmoid function. The reservoir feature vector, $\mathbf{x} \in \mathbb{R}^N$, is given by Equation 1:

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{W}^{in}\mathbf{u}(t+1) + \mathbf{W}\mathbf{x}(t) + \mathbf{v}(t+1)) \quad (1)$$

where $\mathbf{f}$ is the element-wise application of the reservoir activation function and $\mathbf{v}$ is a uniform white noise vector. The output, $\mathbf{y} \in \mathbb{R}^L$, is then given by Equation 2:

$$\mathbf{y}(t+1) = \mathbf{g}(\mathbf{W}^{out}[\mathbf{u}(t+1)|\mathbf{x}(t+1)]) \tag{2}$$

with $\mathbf{g}$, the element-wise application of the output activation function.

NeuroEvolution of Augmented Topologies (NEAT) [14] is a topology and weight evolution of artificial neural networks algorithm, constructed on four principles that have established it as a reference algorithm in the area of NeuroEvolution. First of all, the network, i.e. the phenotype, is encoded as a linear genome (genotype), making it memory efficient with respect to algorithms that work with full weight connection matrices. Second, using the concept of *historical markings*, newly created connections are annotated with innovation numbers. During crossover, NEAT aligns parent genomes by matching the innovation numbers and performs crossover on these matching genes (connections). The third principle is to protect innovation through *speciation*, by clustering organisms into species in order for them to have time to optimize by competing only in their own niche. Last but not least, NEAT initiates with minimal networks, i.e. networks with no hidden units, in order to: (a) initially start with a minimal search space and, (b) justify every complexification made in terms of fitness. NEAT complexifies networks through the application of structural mutations, by adding nodes and connections, and further adapts the networks through weight mutation by perturbing or restarting weight values. The above successful ideas could be used in other NE settings in the form of a meta-search evolutionary procedure. Within the context of our approach, we adopt this model in order to achieve an efficient search in the space of ESNs.

NeuroEvolution of Augmented Reservoirs (NEAR) utilizes NEAT as a meta-search algorithm and adapts its four principles to the ESN model of neural networks. The structure of the evolutionary search algorithm is the same like in NEAT, with adaptations made mainly with respect to gene representation, crossover with historical markings, clustering, thus including some additional evolutionary operators related to ESNs. A major differentiation from NEAT is that both evolution and learning are used in order to adapt networks to the problem at hand. A complete description of NEAR can be found in [5].

2.2 Power Market Auctions

The deregulation of the Power Market has given room for the development of Open Markets, where participants are able to choose between different energy products in different periods of time and may negotiate on their "product portfolio". These portfolios can be negotiated under three different market regimes:

- *The Long-term Market*, where participants come to direct agreements in form of long-term contracts.
- *The Day-ahead Market*, where buyers place their bids in 24 hourly auctions, in order to establish a contract for the next day.

– *The Real-time Market*, where buyers place their bids in order to establish a contract for the next hour.

In Power Market Auctions, two are the most important entities:

1. *The Market participants* (or Players)
2. *The Independent System Administrator* (ISA)

A *Player* is defined as any financial entity that accesses the Power Market [16]. In general, this entity may represent a group of Production Units and/or a group of Consumers. Players participating in the Power Market as *Producers* submit their power supply offers in pre-specified time intervals. Each offer contains the amount of supplying Power, as well as the minimum price one is willing to accept. On the other hand, Players participating in the Power Market as *Consumers* submit their power demands within the same time intervals, along with the maximum price they are willing to pay for it.

The *ISA* is the administrator of the Power System, and also the Administrator of the Power Stock Market (more entities may exist, but they are merged into one for simplicity reasons). Among others, ISA is responsible for the *Settlement Price* of the Power Market, taking into consideration the limitations of the power system. ISA collects bids for each auction and has to calculate two curves: the *aggregate Supply Curve* (ascending) and the *aggregate Demand Curve* (descending). In the simplified case where no transmission limitations exist, Settlement of the Market is achieved at the intersection of the two curves (Figure 2). The intersection point specifies the Settlement Price of the Market (SPM), as well as the load to be provided/offered.

Though demand is mostly covered by Long-term and Day-ahead market transactions, one should stress the following. The fact that power cannot be stored for long periods of time, urged for dictates development of a mechanism that can efficiently balance the supply and demand of power and can be easily and constantly controlled. Such a balance between production and consumption is ensured through the utilization of a Real-Time Market model. This model bears the highest risk for participants, since the malfunction of an Electric node or the shutting down of a Power line can bring exaggerated rising or falling of loads and prices. Nevertheless, higher risks also imply profit maximization potential for players willing to participate in this Market.

3 Related Work

Various approaches have been employed for analyzing the behavior of Power Markets, some of which employ AT and DM primitives. These are discussed next.

In the early 2000s, during the bloom of MAS utilization, the Electric Power Research Institute (EPRI) developed SEPIA (Simulator for Electrical Power Industry Agents), a multi-agent platform capable of running a plethora of computing experiments for many different market scenarios [2, 1]. The Argonne National

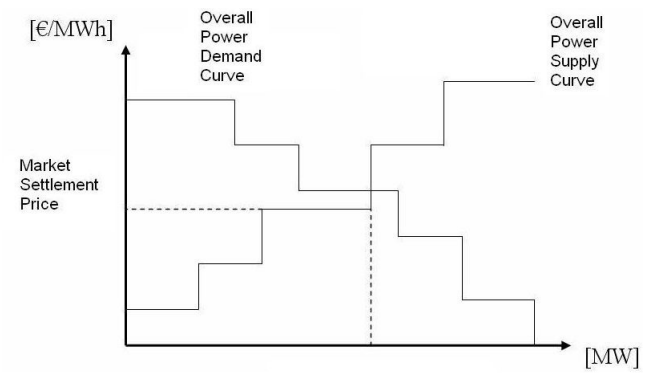


Fig. 2. The Aggregate Power Supply and Demand Curves

Laboratory, on the other hand, developed EMCAS (Electricity Market Complex Adaptive System) [6], an efficient implementation for handling the Electric Energy Market. Through EMCAS, one may study the complex interactions between the physical entities of the market, in order to analyze the participants and their strategies. Players' learning is based on genetic algorithms, while EMCAS supports stock market transactions, as well as bipartite contracts. Finally, Petrov and Sheble [12] introduced Genetic Programming in their simulation and tried to model the bipartite Auctions for the Electric Energy Market, by the use of agents. One of the players incorporates knowledge represented as a Decision-Making Tree, which is developed by Genetic Programming. The rest of the agent-players incorporate ruled-defined behaviors.

Special attention should be drawn to work by Melzian [11], who developed EMSIM (Energy Market SIMulator), in an effort to derive a deeper understanding of the bidding behaviour at the EEX (European Energy Exchange), the impact of changes in market design and individual strategies of the participants. Additionally, work by Bremer et al. [4] should also be denoted. They built an agent-based environment for modeling and simulating adaptive household consumers responding to dynamic electricity pricing. Focus was given on the analysis of household load shifting potential under different tariffs and negotiation strategies.

Although efficient, related work is of narrow-scope and static. The solutions implemented are focused on specific problems and most of them utilize a preselected DM technique, with no ability to adapt and/or expand. This is the main reason why *Cassandra* is built as a general-purpose prediction tool. Different prediction types and different DM algorithms are supported, while Cassandra can manipulate various data types and various datasets. New prediction models can be easily generated to replace previous ones. In the case of time series, the time window used for prediction can be modified (day, week, month, year), while combination of prediction models with differing time windows can be performed, in order to improve prediction performance.

We demonstrate the power of Cassandra by focusing on the newly added algorithms, ESNs and NEAR. With respect to power load forecasting, ESNs have been studied in [13, 3], where data were drawn from the "World-wide competition within the EUNITE network"[1]. Neither of the approaches optimizes all topology, reservoir properties and weights at the same time, in order to adapt the function approximator with minimum human intervention. Additionally, we deal with settlement price forecasting as well, which is a much more demanding problem.

4 The Cassandra MAS

Cassandra is an Agent-Based Framework that employs DM For Prediction and Rapid Problem Solving. Through Cassandra the user may select the appropriate dataset(s), preprocess it, and select a Data Mining API to build his/her models (WEKA and R are fully supported, other APIs are partially supported also). Results are displayed and stored, while action can be taken (in an autonomous or semi-autonomous manner), when deemed appropriate.

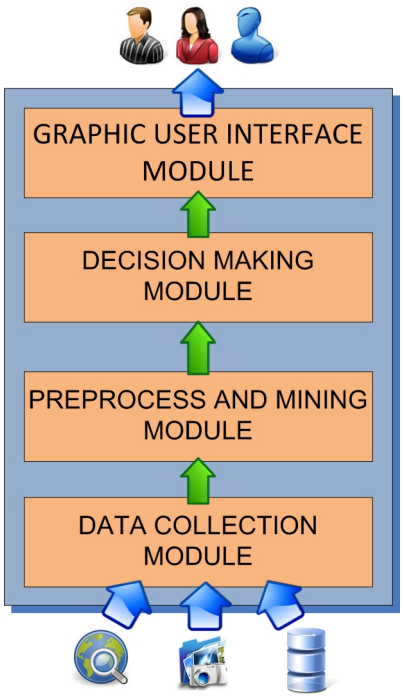


Fig. 3. Cassandra 4-Layer Architecture

[1] http://neuron.tuke.sk/competition/index.php

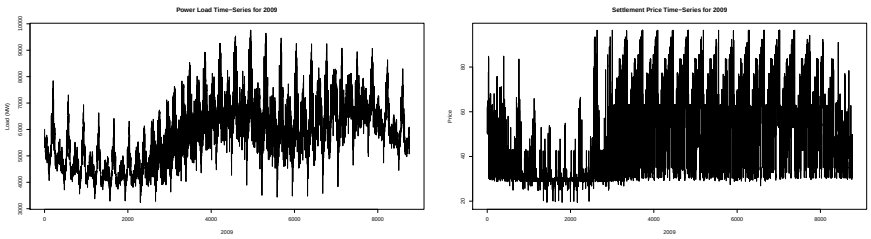
4.1 Architecture

Cassandra follows the IRF Architecture Model (Intelligent Recommendation Framework) [15], which defines a 4-layer functional model for the agent system. IRF is usually employed in enterprises for the optimization of the administration mechanism, since it can automate and integrate all the data producing or data demanding facets of a company. Taking a closer look of the Power Market through the IRF prism, one may identify several tasks that have to be tackled:

- Collection of the historical data from previous auctions and processing of the data.
- Application of the suitable DM algorithms in order to build the necessary forecasting models for the values in question.
- Integration of generated models in the Business Intelligence of the System and evaluation of the results.
- Continuous monitoring of the PSM in order to validate the efficiency of the platform.

As expected, Cassandra employs a modular architecture (Figure 3), where each module is responsible for one of the aforementioned tasks. The platform also provides a wrapper around all modules and ensures communication with the system users. The modules comprising Cassandra are:

1. Data Collection Module (DCM): Collecting historical data, either from files provided by the user, or directly from an API or the web.
2. Data Processing and Mining Module (DPMM): Processing datasets, preparing training sets and applying DM algorithms.
3. Decision Making Module (DMM): Aggregating all information in order to make the optimal decision in any given occasion.
4. Graphic User Interface Module (GUIM): Interacting with the System Users. It must be user-friendly, and easily comprehensive.



(a) Load forecasting (b) Price forecasting

Fig. 4. Load and price timeseries.

5 Experiments

Figure 4 displays the two time series (power load and settlement prices) for year 2009. One may easily observe how irregularities are more intense in Figure 4(b) than in Figure 4(a), indicating a much more difficult task, especially in the absence of knowledge of the causes of these fluctuations.

Collected data span from the 1st of March 2003 to the 30th of September 2010. We have devised two tasks: a) to make a 1-day forecast and b) to make a 1-week forecast, using only the trained data as known input and connecting the outputs of the networks to the inputs for predictions beyond the training data. For the first task, the goal was to predict data power load and prices for September 30th, 2010, while for the second task, the week between 24-30 September, 2010. One year's values were used as a washout sequence in order to initialize the networks and the next year's data were used as the training data. In the NEAR case, fitness was calculated for the week prior to the prediction day and week, respectively. The fitness function was defined as $1/MAPE$, where MAPE is the Mean Absolute Percentage Error. A population of 50 networks evolved over 50 generations. The same initial neurons were used as in the ESN standard case.

Table 1. MAPE and MAXIMAL error values for different algorithms with respect to power load forecasting.

Method	**MAPE (%) 1-day**	**MAXIMAL (MW) 1-day**
ESN	2.26	**279**
NEAR	**2.08**	322
Linear Regression	2.55	389
M5'	2.45	376
Pace Regression	2.55	388
Method	**MAPE (%) 7-days**	**MAXIMAL (MW) 7-days**
ESN	4.87	**1188**
NEAR	**4.28**	1238
Linear Regression	8.34	1336
M5'	4.81	1507
Pace Regression	8.33	1337

Table 1 provides the Mean Absolute Percentage Error (MAPE) and the maximal error (MAXIMAL), which are given by equations:

$$MAPE = 100 \cdot \frac{\sum_{i=1}^{N} \sum_{j=1}^{H} \frac{|L_{Rij} - L_{Pij}|}{L_{Rij}}}{N \cdot H}$$

and

$$MAXIMAL = max(|L_{Rij} - L_{Pij}|)$$

Table 2. ESN parameters.

ESN Parameter	**Value (Load)**	**Value (Price)**
Spectral Radius	0.85	0.5
Density	25%	25%
Reservoir Activation Function	*tanh*	*tanh*
Output Activation Function	*tanh*	*identity*
Reservoir Size	200	30
Noise level	10^{-5}	10^{-7}

where L_{Rij} is the real value at day i and at hour j and L_{Pij} the predicted value. ESNs, and especially the NEAR methodology, exhibit optimal behavior for both tasks. Due to their random initialization procedure, ESNs sometimes tend to become unstable and provide bigger errors. NEAR, on the other hand, bias the search towards ESNs that are able to generalize better.

The M5', Pace and Linear Regression implementations are the ones provided by the WEKA API [7]. For plain ESN, the parameters used can be found in Table 2.

Table 3, displays the MAPE and MAXIMAL errors for settlement price forecasting using the ESN and NEAR algorithms.

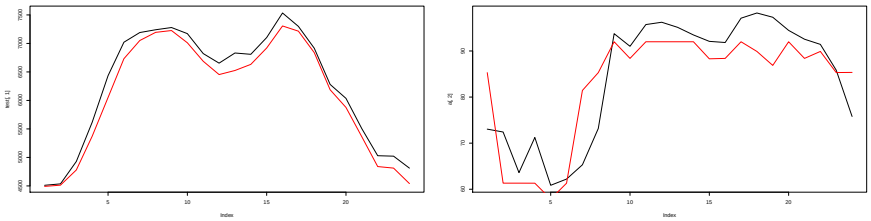
Table 3. MAPE and MAXIMAL error values for ESN and NEAR algorithms with respect to settlement price forecasting.

Method	**MAPE (%) 1-day**	**MAXIMAL (MW) 1-day**
ESN	11.96	21.68
NEAR	**10.64**	**19.54**
Method	**MAPE (%) 7-days**	**MAXIMAL (MW) 7-days**
ESN	18.08	**41.49**
NEAR	**15.99**	46.31

In Figure 5, we present the forecasts made for both power load and market price on the 30th of September 2010 using the NEAR methodology. The actual and predicted values in Figure 5(a) are indicative of the prediction model ESNs are capable of capturing. In the second case, that of Figure 5(b), one may identify that the prediction model is good in general, nevertheless cannot capture unexpected events. In order to do so, more information on the physical constraints of the power system are needed.

6 Conclusions and Future Work

In this paper, we have explored the use of the ESN and NEAR methodologies as autonomous adaptation methods of reservoir computing topologies, and applied



(a) Load forecasting (b) Price forecasting

Fig. 5. Load and price forecasting for the 30th of September 2010.

them in order to enhance the prediction ability of our MAS system with respect to short term power load and market price values in the Greek power stock market. Prediction of load and prices are very important to both the power system administrators, in terms of the economy and system security, as well as the players involved, giving them strategic advantage over their competitors. Our methodology provides very small error for day-ahead power load forecasts, while for price forecasting it significantly outperforms standard data mining algorithms. We believe that with the incorporation of exogenous factors that interfere with market prices, the prediction capability of the system will further improve.

References

1. Amin, M.: Electricity Pricing in Transition, chap. Restructuring the Electric Enterprise: Simulating the Evolution of the Electric Power Industry with Intelligent Agents, pp. 27–50. Kluwer Academic Publishers (2002)
2. Amin, M., Ballard, D.: Defining new markets for intelligent agents. IEEE IT Professional 2(4), 29–35 (Jul/Aug 2000)
3. Babinec, Š., Pospíchal, J.: Optimization of echo state neural networks for electric load forecasting. Neural Network World 2(7), 133–152 (2007)
4. Bremer, J., Andressen, S., Rapp, B., Sonnenschein, M., Stadler, M.: A modelling tool for interaction and correlation in demand-side market behavior. In: First European Workshop on Energy Market Modeling using Agent-Based Computational Economics. pp. 77–91. Karlsruhe (March 2008)
5. Chatzidimitriou, K.C., Mitkas, P.A.: A NEAT way for evolving echo state networks. In: European Conference on Artificial Intelligence. IOS Press (August 2010)
6. Conzelmann, G., Boyd, G., Koritarov, V., Veselka, T.: Multi-agent power market simulation using emcas. In: IEEE 2005 Power Engineering Society General Meeting. vol. 3, pp. 2829–2834 (June 2005)
7. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computation. Natural Computing Series, Springer-Verlag (2003)
8. He, M., Jennings, N.R., Leung, H.: On agent-mediated electronic commerce. IEEE Transactions on Knowledge and Data Engineering 15(4), 985–1003 (Jul/Aug 2003)

9. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the ''echo state network'' approach. Tech. Rep. GMD Report 159, German National Research Center for Information Technology (2002), http://www.faculty.iu-bremen.de/hjaeger/pubs/ESNTutorialRev.pdf
10. Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review 3, 127–149 (2009)
11. Melzian, R.: Bidding and pricing in electricity markets - agent-based modelling using emsim. In: First European Workshop on Energy Market Modeling using Agent-Based Computational Economics. pp. 49–61. Karlsruhe (March 2008)
12. Petrov, V., Shebl, G.: Power auctions bid generation with adaptive agents using genetic programming. In: 2000 North American Power Symposium. Waterloo-Ontario, Canada (OCtober 2000)
13. Showkati, H., Hejazi, A., Elyasi, S.: Short term load forecasting using echo state networks. In: The 2010 International Joint Conference on Neural Networks (IJCNN). pp. 1–5. Barcelona (July 2010)
14. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
15. Symeonidis, A.L., Mitkas, P.: Agent Intelligence through Data Mining. Springer, USA (2005)
16. Tellidou, A., Bakirtzis, A.: Multi-agent reinforcement learning for strategic bidding in power markets. In: 3rd IEEE International Conference on Intelligent Systems. pp. 408–413. London, UK (September 2006)

Agent Based Middleware for Maintaining User Privacy in IPTV Recommender Services

Ahmed Mohamed, Dimtri Botich

Abstract. Recommender systems are currently used by IPTV providers to help end users finding suitable content according their personalized interests, increase content sales, gain competitive advantage over competing companies and improve the overall performance of the current system by building up an overlay to increase content availability, prioritization and distribution based on users' interests. However current implementations are mostly centralized recommender systems (CRS) where the information about the users' profiles is stored in single server. That's why users usually consider CRS as malicious service because in order to obtain accurate recommendations the user might have to reveal information that is considered private such as watching history, previous buying behavior and items ratings. This type of design poses a severe privacy hazard, since the users' profiles are fully under the control of CRS and they have to trust it to keep their profiles private. In this paper, we present our efforts to build a private centralized recommender service (PCRS) using collaborative filtering techniques by introducing an agent based middleware we call AMPR for ensuring user profile privacy in recommendation process; AMPR preserve the privacy of its users when using the system and allow sharing data among different user in the network. We also introduce two obfuscations algorithms embedded in AMPR that protect users' profile privacy and preserve the aggregates in the dataset to maximize the usability of information in order to get accurate recommendations. Using these algorithms gives the user complete control on the privacy of his personal profile. We also provide an IPTV network scenario and experimentation results

Keywords: privacy; clustering; IPTV; recommender system; Multi-Agent.

1 Introduction

Internet protocol television (IPTV) is one of the largest services in ICT; it broadcasts multimedia content (e.g. movies, news programs and documentaries) in digital format via broadband internet networks using packet switched network infrastructure. Differently from conventional television, IPTV allows an interactive navigation of the available items[1]. Recently IPTV providers employ automated recommender systems by collecting information about users' preferences for different items to create users' profile. The preferences of a user in the past can help the recommender system to predict other items that might be interested for him in the future.

Collaborative filtering (CF) technique is utilized for recommendation purposes as one of the main categories of recommender systems. CF is based on the assumption that people with similar tastes prefer the same items. In order to generate a recommendation, CF cluster users with the highest similarity in their interests, then dynamic recommendations are then served to them as a function of aggregate cluster

interests. Thus, the more the users reveal information about their preferences, the more accurate recommendations provided to them. However at the same time the more information is revealed to the recommender service about users' profile, the lower users' privacy levels can be guaranteed. This trade-off acts also as a requirement when recommended system process design is being conducted using CF technique. Privacy aware users refrain from providing accurate information because of their fears of personal safety and the lack of laws that govern the use and distribution of these data. Most service providers would try their best to keep the user's data private but occasionally, especially when they are facing bankruptcy, they might sell it to third parties in exchange of financial benefits. In the other side many services providers might violate users' privacy for their own commercial benefits. Based on a survey results in [2, 3] the users might leave a service provider because of privacy concerns. Privacy is the main concern for a significant number of users, these concerns range from user discomfort with the fact that a system gathers information about his preferences and purchase history at remote servers other than their own devices. This information breaches the privacy of the users on two levels.

1. The user's real identity is available to the central server; the server can associate the user's profile which contains his private information to his real identity. This is an obvious privacy breach, considering that a user does not want the link between his real identity and his profile to be revealed, yet he wants to use the service.
2. If the user is not known to the server, the server can try to deanonymize the user's identity by correlating the information contained in the user's profile and some information obtained from other databases [4] .

In this paper we proposed an agent based middleware for private recommendation (AMPR) that bear in mind privacy issues related to the utilization of collaborative filtering technique in recommender systems and allow sharing data among different users in the network. We also present two obfuscations algorithms that protect user' privacy and preserve the aggregates in the dataset to maximize the usability of information in order to get accurate recommendations. Using these algorithms, gives the user complete control on his personal profile, so he can make sure that the data does not leave his personal Set Top Box (STB) until it is properly desensitized. In the rest of this paper we will generically refer to news programs, movies and video on demand contents as Items. In section II describes some related work. Section III we introduce a combined IPTV network scenario landing our private centralized recommender service (PCRS) using collaborative filtering techniques. Section IV introduces the proposed obfuscation algorithms used in our framework. Section V describes some experiments and results based on obfuscation algorithms for IPTV network control. Section VI includes conclusions and future work. Finally some relevant references are included.

2 Related Work

Existing Recommender systems are mostly based on collaborative filtering; others focus on content based filtering based on EPG data. The majority of the literature addresses the problem of privacy on collaborative filtering technique, Due to it is a potential source of leakage of private information shared by the users as shown in [5]. [6] Propose a theoretical framework to preserve privacy of customers and the commercial interests of merchants. Their system is a hybrid recommender that uses

secure two party protocols and public key infrastructure to achieve the desired goals. In [7, 8] the author proposed a privacy preserving approach based on peer to peer techniques, he suggests forming users' communities, where the community will have a aggregate user profile representing the group as whole and not individual users. Personal information will be encrypted and the communication will be between individual users and not servers. Thus, the recommendations will be generated at client side. [9, 10] suggest another method for privacy preserving on centralized recommender systems by adding uncertainty to the data by using a randomized perturbation technique while attempting to make sure that necessary statistical aggregates such as mean don't get disturbed much. Hence, the server has no knowledge about true values of individual rating profiles for each user. They demonstrate that this method does not lower considerably the obtained accuracy of the results. Recent research work [11, 12] pointed out that these techniques don't provide levels of privacy as were previously thought. [12] Pointed out that arbitrary randomization is not safe because it is easy to breach the privacy protection it offers. They proposed a random matrix based spectral filtering techniques to recover the original data from perturbed data. Their experiments revealed that in many cases random perturbation techniques preserve very little privacy. Similar limitations were detailed in [11]. Storing user's profiles on their own side and running the recommender system in distributed manner without lying on any server is another approach proposed in [13], where authors proposed transmitting only similarity measures over the network and keep users profiles secret on their side to preserve privacy. Although this method eliminates the main source of threat against user's privacy, but they need high cooperation among users to generate useful recommendations.

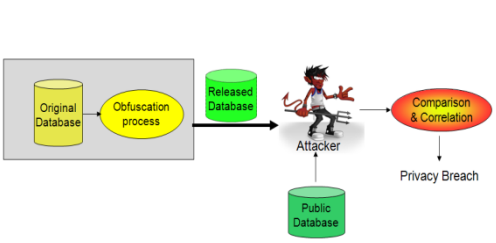


Fig. 1. Attack model for existing Techniques

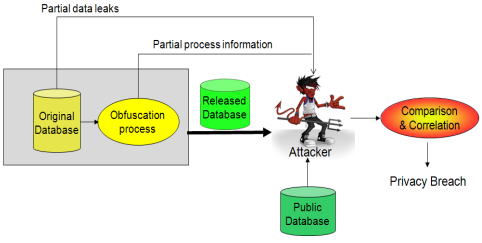


Fig. 2. Attack model for this work

In this paper, we propose agent based middleware (AMPR) to protect the privacy of user's profile form the attack model presented in [14]. The attack model for data obfuscation is different from the attack model for encryption-based techniques, but no common standard has been implemented for data obfuscation. Existing techniques has primarily considered a model where the attacker correlates obfuscated data with data from other publicly-accessible databases in order to reveal the sensitive information (figure 1). In this work, we consider a model where the attacker colludes with some users in the network to obtain some partial information about the process used to obfuscate the data and/or some of the original data items themselves (figure 2). The attacker can then use this partial information to attempt to reverse engineer the entire data set.

3 Combined IPTV network scenario

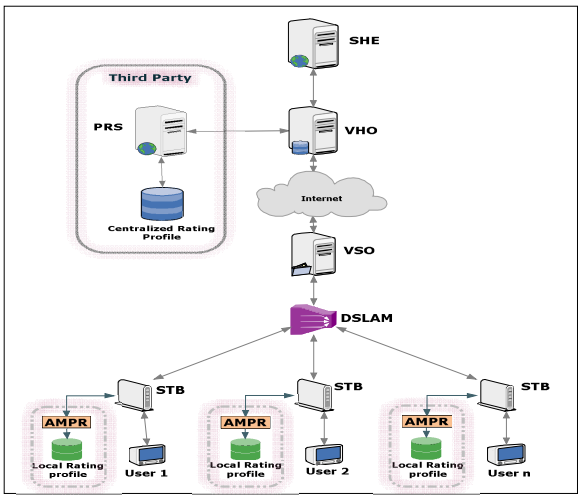


Fig. 3. Illustration of proposed combined IPTV Network

We consider the scenario where PCRS is implemented as a third-party service and users provide their profiles to that external service in order to get recommendations. IPTV uses this business model to reduce the required computational power, expenses or expertise to maintaining an internal recommender system. There are two requirements should be satisfied when using this business model:

- IPTV providers care about the privacy of their catalogue which is considered an asset for their business. In the meantime they are willing to offer real users' ratings for different masked items to offer better recommendations for their users and increase their revenues.
- In the other side, privacy aware users worry about the privacy of their profiles, so sending their real ratings harm their privacy.

AMPR employ a set of algorithms by which the users can changes their profiles to the privacy level they desire, and then submit it to the PCRS. We assume that PCRS follow the semi-honest model, which is realistic assumption because the service provider needs to accomplish some business goals and increase his revenues. Intuitively, the system privacy is high if the PCRS is not able to construct the real ratings for users based on the information available to it. Figure (3) shows our combined solution of hierarchical IPTV architecture with our proposed PCRS service and AMPR. The network consists of super head end (SHE) where all items are collected and distributed, Video Hub office (VHO) that receives content from SHE and distributes it to a number of video serving offices (VSO). VSO stores the content and distribute it to user's Set top box (STB). The Set top box is an electronic appliance that connects to both the network and the home television, with the advancement of data storage technology each STB is equipped with a mass storage device like Cisco STB. We use the user's Set top box (STB) storage to host AMPR at client side. In the other side, PCRS is the entity operating the recommender system that is a third-party recommender service provider makes recommendations by consolidating the information received from multiple sources. The user's profile is obfuscated using AMBR then sent to PCRS for making recommendations, which are then sent back to the corresponding user. We alleviate the user's identity problems by using anonymous

pseudonyms identities for users. Figure 4 outlines the complete framework of our proposed system.

3.1 PCRS Components

PCRS maintains a masked catalogue of items hashed using VHO key (or a group key), obfuscated user profiles (users' pseudonym and their obfuscated ratings). The PCRS communicates with the user through a manager unit. The PCRS also maintains peer cache which is an updated database for information about peers that participate in previous recommendations formulation. The peer cache is updated from peer list database at client side. The user is the client who wishes to receive recommendation list from the PCRS about items that might be interesting for him to pay for view. The clustering manager is the entity responsible for building recommendations model based obfuscated centralized rating profiles.

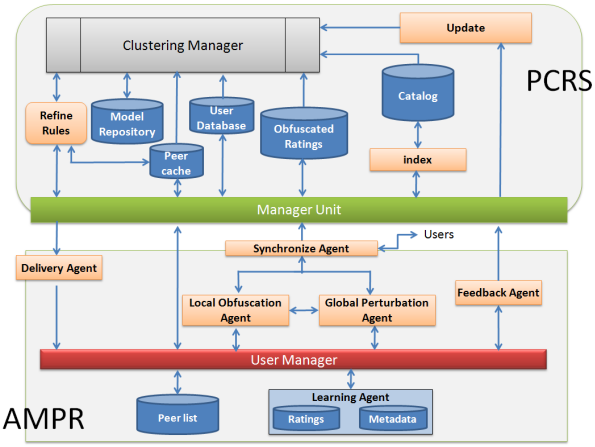


Fig. 4. PCRS framework

3.2 AMPR Components.

The AMPR in the user side consists of different co-operative agents. Learning agent captures user's interests about items to build rating table and meta-data table explicitly and implicitly, more details about learning unit will be published in future work. The local obfuscation agent implements CBT obfuscation algorithm to achieve user privacy while sharing the data with other users or the system. The global perturbation agent executes G-algorithm on the locally obfuscated collected profiles. This two stage obfuscation processes based on a new set of algorithms to achieve user privacy, these algorithms act as wrappers that obfuscate item's ratings before they are fed to the PCRS. Since the databases are dynamic in nature, the local obfuscation agent desensitizes the updated data periodically, then synchronize agent send it to other users and PCRS. So the recommendations are made on the most recent ratings. More details about the recommendation process described in the next sub-section.

3.3 The Recommendation process

The recommendation process based on the two stage obfuscation processes in our framework can be summarized as following:

1. The target user broadcast a message to other users in the IPTV network to indicate his intention to start recommendations process. He also used local obfuscation agent to sanitize his local profile.
2. Individual users that decide to respond to that request use the local obfuscation agent to obfuscate their local rating profiles based on *CBT* algorithm (mentioned below). Then, they submit their locally obfuscated profiles to the requester.
3. If the size of group formation less than specific value, the target user contacts the PCRS directly to gets recommendation from centralized rating profiles stored in it. Otherwise, the target user incites his global perturbation agent to start *G* algorithm (mentioned below) on the collected local rating profiles.
4. In order to hide items that the group are interested in from PCRS, the target user masks the list of items rated by responding users using anonymous indexes which are linked to the actual items indexes in the Video Hub office (VHO) through a secret map Ω know only by target user see table 1. One important issue to standardize this secret map is to use hashing functions using group generated key or key generated by the VHO to hash or mask his catalogue from PCRS.

Anonymous index or Hash Value	Item index	Item Name	User1 rating	User2 rating	...	UserN rating
A_1	I_1	...	...	...		...
A_2	I_2	...	...	...		...
A_v	I_v	...	...	...		...

Table 1. Secret Map Ω Used by Target User

   Due to that the PCRS will not be able to deal directly with items names but their hash values or anonymous index. Additionally the users' ratings are obfuscated from both PCRS and VHO.
5. The target user sends the globally perturbed ratings profiles together with pseudonyms of users participate in global perturbation process to PCRS.
6. The PCRS insert pseudonyms into user database and their ratings into obfuscated rating database. PCRS updates its model using received ratings, then it produces a list $A_i = \{A_1, A_2, ..., A_y\}$ of anonymous indexes that users in the same cluster have chosen in the past.
7. The target user then submits the globally perturbed users' rating to the other users that participate with him in the process, so they able to update their rating data. Then he unmasks the list A_i using his secret map Ω to get finally list A_i , then he selects the appropriate items for him from it.

4 Proposed obfuscation Algorithms

In the next sections we provide enhanced algorithms that used by agents to obfuscate the user' profiles in a way that obscures user' ratings in the untrusted PCRS with minimum loss of accuracy. In our framework, each user has two datasets representing his profile local rating profile where each user perturbs his profile before merging it

with similar users' profiles that desire to collaborate with him as part of the recommendation process and centralized rating profile is the output of the two obfuscation process where the user get recommendation directly from the PCRS based on previously collected ratings. We perform experiments on real datasets to illustrate the applicability of our algorithms and the privacy and accuracy levels achieved using them.

4.1 Local Obfuscation Using CBT Algorithm

We propose a new obfuscation algorithm called clustering based transformation (CBT) that have been designed especially for the sparse data problem we have here. Based on the idea of block cipher in [15]. We present a new technique to build a Transformation lookup table (TLUT) using clustering techniques then approximate each point in the data set to the nearest representative value in the TLUT (core-point for cluster it belong to) with the help of similarity measures. The output of our obfuscation algorithm should satisfy two requirements:

1. Reconstructing the original data from the obfuscated data should be difficult, in order to preserve privacy.
2. Preserve the similarity between data to achieve accurate results.

We use local learning analysis (*LLA*) clustering method proposed in [16] to create the TLUT. It is important to attain an optimized TLUT because the quality of the TLUT obviously affects the performance of the transformation. LLA builds an initial TLUT and repeats the iteration till two conditions satisfied:

1. The distance function $d(x, c_i)$ between a point x and its corresponding value c_i is minimized.
2. The distortion function between each dataset and its nearest value becomes smaller than a given threshold.

Our algorithm consists of following steps:

1. The users' ratings stored in ratings component (RC) as dataset D of c rows, where each row is sequence of fields $X = x_1\ x_2\ x_3 \ldots\ldots x_m$.
2. Users' ratings dataset D is portioned into $D_1\ D_2\ D_3 \ldots\ldots D_n$ datasets of length L , if total number of attributes in original is not perfectly divisible by L then extra attributes is added with zero value which does not affect the result and later it is removed at step 5.
3. Generate TLUT using LLA algorithm, LLA takes Gaussian Influence function as the similarity measure. Influence function between two data point x_i and x_j is given as

$$f_{Gauss}^{x_i}(x_j) = e^{-\frac{d(x_i, x_j)^2}{2\sigma^2}},$$

The field function for a candidate core-point given by:

$$f_{Gauss}^{D}(x_j) = \sum_{s=1}^{k} e^{-\frac{d(x_j, x_{is})^2}{2\sigma^2}}$$

Clustering is performed on each dataset D_i, resulting to k clusters $C_{i1}, C_{i2}, C_{i3}, ..., C_{ik}$ and each cluster is represented by its core-points, i.e. core-point of j^{th} cluster of i^{th} dataset is $(C_{ij}) = \{c_1, c_2, c_3, ...c_L\}$.Every single row portion falls in exactly one cluster. And The TLUT = (core-point(C_{i1}) , core-point(C_{i2}) , core-point(C_{i3}) ..,core-point (C_{ik}))

4. Each dataset D_i is transformed into new dataset D_i' using generated TLUT, each portion $Y_i = x_{(i-1)L+1}\ x_{(i-1)L+2}\ x_{(i-1)L+3} \ldots\ldots x_{iL}$ replaced by the nearest cluster core-point Z_i = core-point (C_{ij}) in which it falls.

$$Y_i \xrightarrow{transofrmed} Z_i$$

The transformation function is: $T(Y_i) = \{core - point(C_j \leftrightarrow d(Y_i, core - point(C_j) < d(Y_i, core - point(C_Z) \ \square\ Z\}$

5. Now all the n transformed portions of each point is joined in the same sequence as portioned in step 2 to form a new k dimension transformed row data which replaces the X in the original dataset. In this way perturbed dataset D_i' is formed from original dataset D
6. Compute the privacy level by calculating the difference between the original dataset and transformed dataset using Euclidean distance:

$$\Pr ivacy - Level = \frac{1}{mn}\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|x_{ij} - y_{ij}\right|^2}$$

4.2 Global Perturbation using *G* Algorithm

After executing the local obfuscation process, the global perturbation phase starts. The idea is cluster multidimensional data using fast density clustering algorithm, then perturb each dimension in each cluster in such a way to preserve its range. In order to allow obfuscation agent to execute *G* algorithm, we introduce enhanced mean shift (EMS) algorithm which is tailored algorithm for the global perturbation phase that has advantage over previous algorithm proposed in [17], that it requires low computational complexity in clustering large data sets. we employ Gaussian KD-tree [18] clustering to reduce the feature space of the locally obfuscated data. The *G* algorithm consists of two steps:

Step1: Build different density based clusters.

1. We build the tree In a top down manner starting from a root cell similar to [18, 19]. Each inner node of the tree S represents a *d*-dimensional cube cell which stores the dimension S_d along which it cuts, the cut value S_{cut} on that dimension, the bounds of the node in that dimension S_{min} and S_{max}, and pointers to its children S_{left} and S_{right}. All points in the leaf nodes of the k-d tree are then considered as a sample and the kd-tree stores m samples defined as $y_j, j = 1,...,m$ that construct the reduced feature space of the original obfuscated data set.

2. Assign each record x_i to its nearest y_j based on *kd*-search, then compute a new sample, we called it $y_j^*, j = 1,...,m$.
3. Generated y_j^* is a feature vector of *d*-dimensions, that is considered as a more accurate sample of the original obfuscated data set that will be used in the mean shift clustering.
4. The mean shift clustering iteratively performs these two steps:
   - Computation of mean shit vector based on the reduced feature space as following:

$$m(x_j) = \frac{\sum_{y_i^* \in N(y_j^*)} y_i^* g\left(\left\| \frac{x_j - y_i^*}{h} \right\|^2\right)}{\sum_{y_i^* \in N(y_j^*)} g\left(\left\| \frac{x_j - y_i^*}{h} \right\|^2\right)} - x_j, \; j = 1, 2..m$$

     Where $g(x) = -k'(x)$ defined when the derivate of function $k(x)$ exists, and $k(x), 0 \le x \le 1$ is called kernel function satisfying: $k(x) = c_{k,d} k\left(\|x\|^2\right) > 0$, $\|x\| \le 1$ and $\int_{-\infty}^{\infty} k(x)dx = 1$
   - Update the current position x_{j+1} as following:

$$m(x_{j+1}) = \frac{\sum_{y_i^* \in N(y_j^*)} y_i^* g\left(\left\| \frac{x_j - y_i^*}{h} \right\|^2\right)}{\sum_{y_i^* \in N(y_j^*)} g\left(\left\| \frac{x_j - y_i^*}{h} \right\|^2\right)}, \; j = 1, 2..m$$

     Until reaching the stationary point which is the candidate cluster center. x_j will coverage to the mode in reduced feature space, finally we got approximate modes of original data defined as $z_x, x = 1,....,k$ (proof in appendix A).
5. Finally, the points which are in the mode are associated with the same cluster, and then we interpolate the computed modes in samples to the original obfuscated data by searching for the nearest mode z_x for each point x_i .

Step 2: Generating random points in each dimension range

For each cluster C , perform the following procedure.

1. Calculate the interquartile range for each dimension A_i .
2. For each element $e_{ij} \in A$, generate a uniform distributed random number r_{ij} in that range and replace e_{ij} with r_{ij} .

5 Experiments

The proposed algorithms is implemented in C++, we used message passing interface (MPI) for a distributed memory implementation of *G* algorithm to mimic a distributed reliable network of peers. We evaluate our proposed algorithms in two aspects: privacy achieved and accuracy of results. The experiments presented here were conducted using the Movielens dataset provided by Grouplens [20], it contains

users' ratings on movies in a discrete scale between 1 and 5. We divide the data set into a training set and testing set. The training set is obfuscated then used as a database for PCRS. Each rating record in the testing set is divided into rated items t_u and unrated items r_u. The set $t_{u,i}$ is presented to the PCRS for making predication $p_{u,i}$ for the unrated items $r_{u,i}$. To evaluate the accuracy of generated predications, we used the mean average error (MAE) metric proposed in [21].

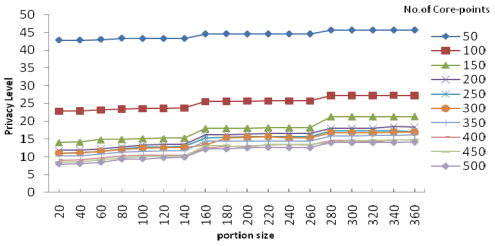


Fig. 6. Privacy level for different partion

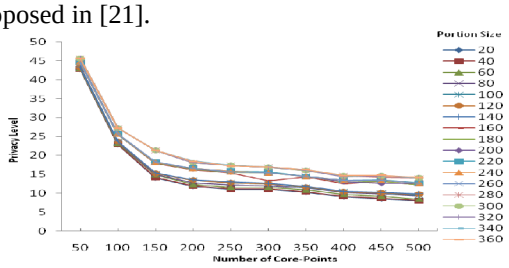


Fig. 5. Privacy level for different no.of core points

The first experiment performed on CBT algorithm, we need to measure the impact of varying portion size and number of core-points on privacy level of transformed dataset. In order to compute it, we keep portion size L constant with different number of core-points and then we vary portion size with constant number of core-points. Based on the results shown in figures (5) and (6); we can deduce that privacy level increases with increase in portion size. In the other side, privacy level reduced with increasing number of core-points as large number of rows used in TLUT; this can be achieved by tuning different parameters in LLA algorithm. Note that low privacy level reduces information loss when PCRS group different users but the transformed data is very similar to original data so attacker can acquire more sensitive information.

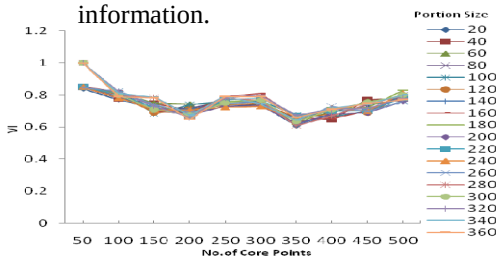


Fig. 8. VI for different no.of core points

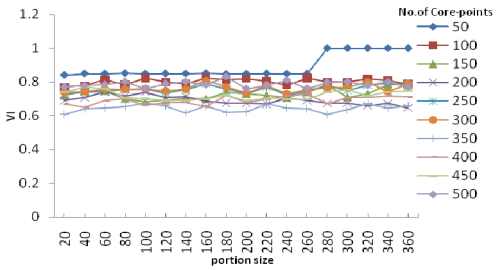


Fig. 7. VI for different partion size

To measure the distortion of the results, we use variation of information (VI) metric. Figure (7) shows VI for different number of core-points. it has been found that for low number of core-points the value of VI is high but slowly it decreases with increase in number of core-points.at certain point it gain rises to local maxima then it gain decreases with some number of core-points. Finally the graph rises again with increasing number of core-points. We analyzing the graph to understand its behavior, and come up with these results:

- High VI at low number of core-points because there is a high chance of points to move from one core-point to another due to low number of core-points. As show in figure (5) privacy level is high at low number of core-points so there are more information loss.

– High VI at high number of core-points because less number of points with each core-point so there is little chance of a point to move from one core-point to another due to high number of core-points that might be higher than optimal number of natural core-points in the dataset.

The second experiment performed on *G* algorithm, we need to measure the impact of sample size on accuracy level and the execution time for the global perturbation process. In order to compute both, we should have a specific threshold value that reflects minimum number of responding users to a target user to start the global perturbation process. We set it to 500 users, otherwise use the PCRS obfuscated ratings database. Figure (9) shows different execution time for step 1 of *G* algorithm based on various sample size. The relation between execution time and number of clusters in step 2 is shown in figure (10). We continue our experiments of *G* algorithm to measure the effect of sample size in on accuracy level, figures (11) illustrate that. We can note that the increase in sample size leads to high accuracy in the predications

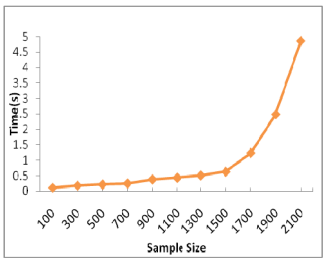


Fig. 9. Execution time for step 1

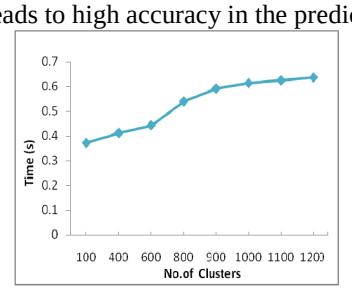


Fig. 10. Execution time for step 2

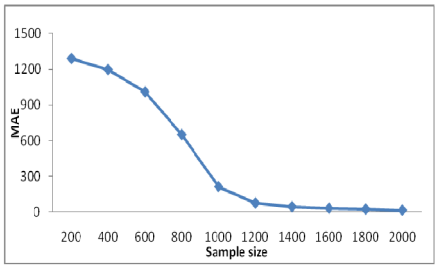


Fig. 11. Relation between sample size and MAE

6 CONCLUSION AND FUTURE WOK

In this paper, we present our ongoing work on building a framework for private centralized recommender system with agent based middleware. We give a brief overview over the system components and recommendations process. Also we present novel algorithms that give the user complete control over his profile privacy using two stage obfuscation processes. We test the performance of the proposed algorithms on real dataset and report the overall accuracy of the recommendations based on different privacy levels. The experiential results shows that preserving users' data privacy for in collaborative filtering recommendation system is possible and the mean average error can be reduced with proper tuning for algorithms parameters and large number of users. We need to perform extensive experiments in other real data set

from UCI repository and compare the performance with other techniques, also we need to consider different data partitioning techniques, identify potential threats and add some protocols to ensure the privacy of the data against these threats.

References

[1] S. Hand and D. Varan, "Interactive Narratives: Exploring the Links between Empathy, Interactivity and Structure," ed, 2008, pp. 11-19.
[2] L. F. Cranor, "'I didn't buy it for myself' privacy and ecommerce personalization," presented at the Proceedings of the 2003 ACM workshop on Privacy in the electronic society, Washington, DC, 2003.
[3] C. Dialogue, "Cyber Dialogue Survey Data Reveals Lost Revenue for Retailers Due to Widespread Consumer Privacy Concerns," in *Cyber Dialogue*, ed, 2001.
[4] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," presented at the Proceedings of the 2008 IEEE Symposium on Security and Privacy, 2008.
[5] F. McSherry and I. Mironov, "Differentially private recommender systems: building privacy into the net," presented at the Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Paris, France, 2009.
[6] A. Esma, "Experimental Demonstration of a Hybrid Privacy-Preserving Recommender System," 2008, pp. 161-170.
[7] J. Canny, "Collaborative filtering with privacy via factor analysis," presented at the Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, Tampere, Finland, 2002.
[8] J. Canny, "Collaborative Filtering with Privacy," presented at the Proceedings of the 2002 IEEE Symposium on Security and Privacy, 2002.
[9] H. Polat and W. Du, "Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques," presented at the Proceedings of the Third IEEE International Conference on Data Mining, 2003.
[10] H. Polat and W. Du, "SVD-based collaborative filtering with privacy," presented at the Proceedings of the 2005 ACM symposium on Applied computing, Santa Fe, New Mexico, 2005.
[11] Z. Huang, *et al.*, "Deriving private information from randomized data," presented at the Proceedings of the 2005 ACM SIGMOD international conference on Management of data, Baltimore, Maryland, 2005.
[12] H. Kargupta, *et al.*, "On the Privacy Preserving Properties of Random Data Perturbation Techniques," presented at the Proceedings of the Third IEEE International Conference on Data Mining, 2003.
[13] B. N. Miller, *et al.*, "PocketLens: Toward a personal recommender system," *ACM Trans. Inf. Syst.*, vol. 22, pp. 437-476, 2004.
[14] R. Parameswaran and D. M. Blough, "Privacy preserving data obfuscation for inherently clustered data," *Int. J. Inf. Comput. Secur.*, vol. 2, pp. 4-26, 2008.
[15] M. Blaze and B. Schneier, "The MacGuffin block cipher algorithm," ed, 1995, pp. 97-110.
[16] A. M. Elmisery and F. Huaiguo, "Privacy Preserving Distributed Learning Clustering Of HealthCare Data Using Cryptography Protocols," in *34th IEEE Annual International Computer Software and Applications*, Seoul, South Korea, 2010.
[17] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *Information Theory, IEEE Transactions on*, vol. 21, pp. 32-40, 2003.
[18] K. Xu, *et al.*, "Efficient affinity-based edit propagation using K-D tree," presented at the ACM SIGGRAPH Asia 2009 papers, Yokohama, Japan, 2009.

[19] A. Adams, *et al.*, "Gaussian KD-trees for fast high-dimensional filtering," *ACM Trans. Graph.*, vol. 28, pp. 1-12, 2009.
[20] S. Lam and J. Herlocker. MovieLens Data Sets [Online]. Available: http://www.grouplens.org/node/73
[21] J. L. Herlocker, *et al.*, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, pp. 5-53, 2004.

Appendix A: Proof of Convergent

Theorem 1: if the kernel function g has a convex and monotonically decreasing profile, the sequence $\{f(j)_{j=1,2....}\}$ will still convergent.

Using the kernel function $x_{j+1} = \frac{\sum_{y_i^* \in N(y_j^*)} y_i^* g\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)}{\sum_{y_i^* \in N(y_j^*)} g\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)}, j = 1, 2..m$

To prove convergence, we have to prove that $f(x_{j+1}) > f(x_j)$

$$f(x_{j+1}) - f(x_j) = \sum_{i=1}^{n} g\left(\left\|\frac{x_{j+1} - y_i^*}{h}\right\|^2\right) - \sum_{i=1}^{n} g\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)$$

But since the kernel is a convex function we have,

$g(x_{j+1}) - g(x_j) \geq g'(x_j)(x_{j+1} - x_j)$ Using it,

$$f(x_{j+1}) - f(x_j) \geq \sum_{i=1}^{n} g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)\left(\left\|\frac{x_{j+1} - y_i^*}{h}\right\|^2 - \left\|\frac{x_j - y_i^*}{h}\right\|^2\right)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)\left(x_{j+1}^2 - 2y_i^* x_{j+1} + y_i^{*2} - (x_j^2 - 2y_i^* x_j + y_i^{*2})\right)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)\left(x_{j+1}^2 - x_j^2 - 2(x_{j+1} - x_j)^T y_i^*\right)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)\left(x_{j+1}^2 - x_j^2 - 2(x_{j+1} - x_j)^T x_{j+1}\right)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)\left(x_{j+1}^2 - x_j^2 - 2(x_{j+1}^2 - x_{j+1}x_j)\right)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)\left(x_{j+1}^2 - x_j^2 - 2x_{j+1}^2 + 2x_{j+1}x_j\right)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)\left(-x_{j+1}^2 - x_j^2 + 2x_{j+1}x_j\right)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)(-1)\left(x_{j+1}{}^2 + x_j{}^2 - 2x_{j+1}x_j\right)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} -g'\left(\left\|\frac{x_j - y_i^*}{h}\right\|^2\right)\left(\left\|x_{j+1} - x_j\right\|^2\right)$$

$= \geq 0$

There we prove that the sequence $\left\{f(j)_{j=1,2....}\right\}$ is still convergent.

Pricing Analysis in Online Auctions using Clustering and Regression Tree Approach

Submitted for Blind Review

Abstract. Auctions can be characterized by distinct nature of their feature space. This feature space may include opening price, closing price, average bid rate, bid history, seller and buyer reputation, number of bids and many more. In this paper, a price forecasting agent (PFA) is proposed using data mining techniques to forecast the end-price of an online auction for autonomous agent based system. In the proposed model, the input auction space is partitioned into groups of similar auctions by k-means clustering algorithm. The recurrent problem of finding the value of k in k-means algorithm is solved by employing elbow method using one way analysis of variance (ANOVA). Based on the transformed data after clustering, bid selector nominates the cluster for the current auction whose price is to be forecasted. Regression trees are employed to predict the end-price and designing the optimal bidding strategies for the current auction. Our results show the improvements in the end price prediction using clustering and regression tree approach.

Keywords: Online auctions, Price forecasting, Bidding strategies, Data mining, Clustering, Regression trees

1 Introduction

Predicting the end price of an auction has become an increasingly important area of research because buyers and sellers can be offered a great benefit by using the above predicted prices[1, 2, 5]. The online auctions are exchange mechanisms which produce a large amount of transaction data. This data can be exploited to forecast the final prices of the auction items, if utilized properly. Researchers' efforts can be noticed in the area of forecasting end-price of an auction using machine learning techniques, functional data analysis, time series analysis [1-8]

Software agent technology is one of the most popular mechanisms used in on-line auctions for buying and selling the goods. Software agent is a software component that can execute autonomously, communicates with other agents or the user and monitors the state of its execution environment effectively [9-11]. Agents can use different auction mechanisms (e.g. English, Dutch, Vickery etc.) for procurement of goods or reaching agreement between agents. Agents make decisions on behalf of consumer and endeavor to guarantee the delivery of item according to the buyer's preferences. These are better negotiators than human being in terms of monitoring, remembering and emotional influence. In these auctions buyers are faced with difficult task of deciding amount to bid in order to get the desired item matching their preferences. This bid

amount can be forecasted effectively by analyzing the data produced as an auction progresses (historical data). This forecasted bid can be exploited by the bidding agents to improve their behaviors. Also the analysis of plethora of data produced in the online auction environment can be done by using DM techniques [1, 4, 7, 8, 12].

Predicting the end price depends on many factors, such as item type, type of auction, quantity available, opening price, number of bidders, average bid amount and many more. Price dynamics of the online auctions can be different even when dealing with auctions for similar items. Functional data analytical tools have been used to characterize different type of auctions that exhibit different price dynamics in [8]. In this paper, a price forecasting agent (PFA) is proposed using data mining techniques to forecast the end-price of an online auction for autonomous agent based system. A clustering based approach is used to characterize different type of auctions. In the proposed model, the input auctions are clustered into groups of similar auctions based on their characteristics using k-means algorithm. To decide the value of *k* in *k-means* algorithm is a recurrent problem in clustering and is a distinct issue from the process of actually solving the clustering problem. The optimal choice of *k* is often ambiguous, increasing the value of *k* always reduce the error and increases the computation speed. In this paper, we are exploring Elbow approach using one way analysis of variance (ANOVA) to estimate the value of *k*. Based on the transformed data after clustering and the characteristics of the current auction whose price is to be forecasted, bid selector nominates the cluster. Regression trees are employed to the corresponding cluster for forecasting the end-price and to design the bidding strategies for the current auction.

The rest of the paper is organized as follows. In section 2 we discuss the related work to the topic. Section 3 illustrates the design of PFA- Price Forecasting Agent describing the data mining mechanism developed for forecasting the end-price and optimizing the bidding strategies for online auctions. Section 4 depicts the experimental results. Section 5 discusses the conclusions of the paper and presents directions for the future work.

2 Related Work

In the recent literature, different approaches have been presented for end price prediction in the online auctions environment. A data mining based multi-agent system has been designed in favor of a multiple on-line auctions environment for selecting the auction, in which the traded item will be sold at the lowest price [4]. The K-means clustering technique has been used to classify auctions into discrete clusters. Clustering operates dynamically on multiple auctions as bid price changes in running auctions. The results of the dynamic clustering are fed into the agents, and by employing probability-based decision making processes, agents deduce the auction that is most likely to close at the lowest price. Experimental results have demonstrated the robustness of the designed system for multiple on-line auctions with little or no available information.

Forecasting [3] has been proposed as a time series problem and has been solved using moving averages and exponential smoothing models. Authors in this paper also emphasized that there is no one best forecasting technique for a particular set of data. Authors used sequence mining to improve the decision mechanism of an agent for predicting bidding strategy of a set of trading agents [7]. Prediction are mostly based on the continuously growing high dimensional bid's history, so authors identified that sequence mining technique can be employed to classify the high dimensional frequent patterns in the bid's history. Also the sliding window concept has been used for feeding the continuous classifier. Classification and meta-classification are applied to predict the final bid.

Predicting end price of an online auction has been stated as a machine learning problem and has been solved using regression trees, multi-class classification and multiple binary classification [2]. Among these machine learning techniques, posing the price prediction as a series of binary classification has been proved to be the best suited method for this task. In the literature, along with the machine learning techniques, traditional statistical methods have also been used to forecast the final prices of the auction item, but the experimental results demonstrated that the machine-learning algorithms such as BP networks outperform traditional statistical models [5]. Seeking the effect of clustering of data was also the concern of the authors [5]. Clustering has increased the accuracy of the results in case of BP networks but decreased the accuracy in logistic regression.

A support system for predicting the end-price of an online auction based on the item description using text mining and boosting algorithm has been proposed [6]. Emphasis is given by the authors on capturing the relevant information for incorporating into the price forecasting models for ongoing online auction [1]. A novel functional K-nearest neighbor (fKNN) forecaster based on functional and non-functional distance has been proposed and has been proved to be effective particularly for heterogeneous auction populations.

LS-SVM (Least Square Support Vector Machine) algorithm has been introduced by Zhou and Wang for forecasting in online electronic commerce [12]. Authors first improved the SVM to solve the sparsity and time lag problems existing in the traditional method and then they established the LS-SVM online forecast model based on the time factor elimination. Experimental results demonstrated almost same values for the forecasted and the actual price.

3 PFA-Price Forecasting Agent

A clustering based method is used to forecast the end-price of an online auction for autonomous agent based system. In the proposed methodology the input auctions are partitioned into groups of similar auctions depending on their different characteristics. This partitioning has been done by using *k-means* clustering algorithm. The value of *k* in *k-means* algorithm is determined by employing elbow method using one way analysis of variance (ANOVA). Based on the transformed data after clustering and the characteristics of the current auction, bid selector nominates

the cluster for price forecasting. End-price is predicted and bidding strategies are designed by using regression trees for the nominated cluster.

The price forecasting and bidding agent is represented in Figure. 1. Formally our approach consists of four steps. First, data is extracted from the bid server as per the requirements to form the agents' knowledge base for online auctions. Let A be the set of the attributes collected for each auction then $A=\{ a_1, a_2 \ldots. a_j\}$ where j is the total number of attributes. Then based on the auctions' characteristics, similar auctions are clustered together. Secondly, k-estimator agent determines the best number of partitions for the overall auction data and then the set of similar auctions are clustered together in k groups. Let C be the set of clusters then $C = \{c_1, c_2 \ldots. c_k\}$ where k is the total number of clusters. Thirdly, based on the transformed data after clustering and the characteristics of the current auction, bid selector nominates the cluster for price forecasting. Finally, regression tree is employed to forecast the end price and to design the bidding strategies for the selected cluster.

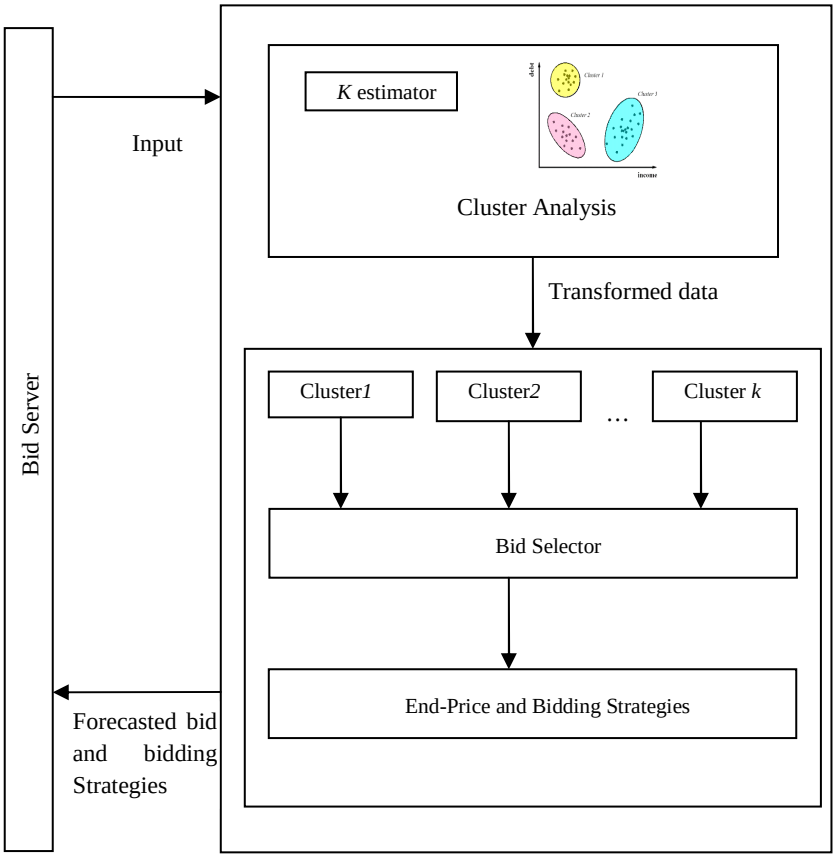


Fig. 1. PFA-Price Forecasting Agent

3.1 *K-means* Cluster Analysis

The main idea of *k-means* clustering is to define *k* centroids, one for each cluster. Initially *k* data are randomly chosen to represent the centroids. Each data point is assigned to the group that has the closest centroid. After assignment of each data point, positions of the *k* centroids are re-calculated as the average value of every cluster. These steps repeat until the centroids no longer move or minimizing the objective function *J*.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \quad (1)$$

Where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j. It indicates the distance of the *n* data points from their respective cluster centers.

K-means clustering technique is used here to categorize different types of auctions based on some predefined attributes from the vast feature space of online auctions. The feature space may include average bid amount, average bid rate, no. of bids, item type, seller reputation, opening bid, closing bid, quantity available, type of auction, duration of the auction, buyer reputation and many more. In this paper, to classify different types of auctions, we focus on only a set of attributes; opening bid, closing price, number of bids, average bid amount and average bid rate. Now $A=\{OpenB_i, CloseP_i, NUM_i, AvgB_i, AvgBR_i\}$

Where *A* is the set of attributes for an auction.

$OpenB_i$ is the starting price of i^{th} auction

$CloseP_i$ is the end price of i^{th} auction

NUM_i is the total number of bids placed in i^{th} auction

$AvgB_i$ is the average bid amount of i^{th} auction and can be calculated as $Avg(B_1, B_2,\ldots..B_l)$ where B_1 is the 1st bid amount, B_2 is the second bid amount and B_l is the last bid amount for i^{th} auction.

$AvgBR_i$ is the average bid rate of i^{th} auction and can be calculated as

$$\frac{1}{n}\sum_{i=1}^{n}\frac{B_{i+1}-B_i}{t_{i+1}-t_i}.$$

where B_{i+1} is the amount of $(i+1)^{th}$ bid, B_i is the amount of i^{th} bid, t_{i+1} is the time at which $(i+1)^{th}$ bid is placed and t_i is the time at which i^{th} bid is placed.

3.2 *K*-estimator

To decide the value of *k* in *k-means* algorithm is a recurrent problem in clustering and is a distinct issue from the process of actually solving the clustering problem. The optimal choice of *k* is often ambiguous, increasing the value of *k* always reduce the

error and increases the computation speed. The most favorable method to find *k* adopts a strategy which balances between maximum compression of the data using a single cluster, and maximum accuracy by assigning each data point to its own cluster. There are several approaches to decide the value of *k*: rule of thumb, the elbow method, information criterion approach, an information theoretic approach, choosing *k* using the Silhouette and cross-validation. In this paper, we are exploring Elbow approach using one way analysis of variance (ANOVA) to estimate the value of *k*. Number of clusters are chosen so that adding another cluster doesn't give much better modeling of the data. A graph has been plotted for percent of variance against the number of clusters. At some point the marginal gain will drop, giving an angle in the graph (Figure. 2). The number of clusters is chosen at this point.

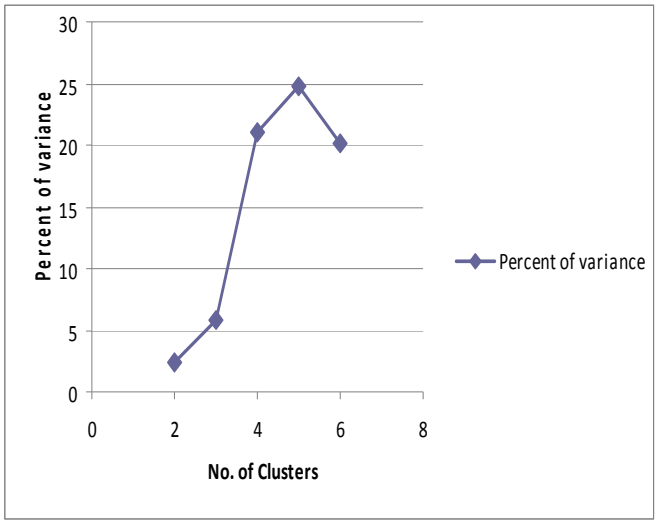


Fig. 2. Choosing the value of *k*

Below is the mathematical model explained for the value of *k*.

Percent of variance=

$$\frac{\sum_{n=1}^{k}\sum_{i=1}^{N_n}(X_{in}-\bar{X})^2-\sum_{n=1}^{k}\sum_{i=1}^{N_n}(X_{in}-\overline{X_n})^2}{\sum_{n=1}^{k}\sum_{i=1}^{N_n}(X_{ij}-\bar{X})^2} \quad (2)$$

Where *k* is the total no. of clusters

N_n is total no. of elements in n^{th} cluster

$\overline{X_n}$ is the mean of distances of auctions from the cluster center in n^{th} cluster

X_{in} is distance of i^{th} auction in n^{th} cluster from its cluster center

3.3 Bid Selector

In order to decide that the current auction belongs to which cluster, the bid selector is activated. Based on the transformed data after clustering and the characteristics of the current auction, bid selector nominates the cluster for the current auction whose price is to be forecasted.

3.4 Extracting Bidding Strategies

Regression Tree is an analytic procedure for predicting the values of a continuous response variable from continuous predictors. We can derive simple if-then rules from these predictions. In this paper, we are employing regression tree to predict the end price and to design the optimal bidding strategies for the targeted cluster. This targeted cluster is the cluster which bid selector nominates for the current auction whose end price is to be forecasted. Regression trees are built in two phases. The first phase is growing phase and the second phase is pruning phase. In the growing phase, the tree is grown until no error reduction on the training possible or a pre-defined threshold has been reached. The resulting model usually over-fits the data. This is overcome by the pruning the tree. The best tree is chosen when both the output variable variance and the cost-complexity factor are minimized. The trade-off between these two criterions should be considered. There are two ways to accomplish this. One is test-sample cross-validation and the other is v-fold cross-validation. In this paper, test-sample cross-validation method is used to determine the best pruned tree for designing the optimal bidding strategies of the online auction.

4 Experimentation

In the proposed approach end-price is predicted and the bidding strategies are designed for an online auction by exploiting regressions trees on each cluster which are generated by applying *k-means* algorithm on the input auctions dataset. The outcome of the proposed model with clustering is compared with the classic (without clustering) model for price prediction. The improvement in the error measure for each cluster for a set of attributes gives support in favor of the proposed model using clustering. Optimal bidding strategies are designed by employing regression trees on each cluster and the model is evaluated by test-sample cross validation approach. Our dataset includes the complete bidding records for 149 auctions for new Palm M515 PDAs. All are 7-day auctions that were transacted on eBay between March and June, 2003 (a sample is available online at;
http://www.rhsmith.umd.edu/digits/statistics/pdfs_docs/Palm_7day_149auctions.xls).

The value of *k* for *k-means* algorithm is estimated by Elbow approach using one way analysis of variance (ANOVA). Percent of variance is calculated after performing clustering for subsequent values of *k* using *k-means* algorithm for estimating the point where marginal gain drops. In our experiment, this point occurs after five numbers of clusters as shown in Table 1 and Figure 2. So we divide the input space in five clusters considering a set of attributes i.e. opening bid, closing price, no. of bids, average bid

amount and average bid rate for a particular auction. These five clusters contain 20.27%, 34.46%, 18.92, 20.95 and 5.41% of auctions' data.

The prediction performance is evaluated using the root mean square error (RMSE) measure. This is used as a measure of the deviation between the predicted and the actual values. The smaller the value of RMSE, the closer is the predicted values to the actual values. Typically the RMSE is defined as

$$RMSE=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i-\bar{y_i})^2} \quad (3)$$

Where $\bar{y_i}$ is the forecasted bid for the i^{th} auction based on the estimated model, y_i is the actual bid amount for the i^{th} auction and m is the total no. of auctions in the cluster.

Table 1. Percent of variance after subsequent clustering

No.of Clusters	Percent of variance
2	2.46
3	5.8
4	21.06
5	24.75
6	20.22

The improvement in root mean square error after clustering for each of the five clusters are 16.43%, 29.47%, 20.76%, 31.17% and 64.91% respectively. This indicated that the proposed price forecasting agent can improve the accuracy of the forecasted price for online auctions.

Optimal bidding strategies are designed by employing regression trees on each cluster. To find the best tree, we have used the test-sample cross validation approach. We randomly divide the data into 80% training and 20% validation set and applied the tree-growing algorithm to the training data and grows the largest tree which over-fits the data. Then the tree is pruned by calculating the cost complexity factor at each step during the growing of the tree and deciding the number of decision nodes for the pruned tree. The tree is pruned to minimize the sum of (1) the output variable variance in the validation data, taken a terminal node at a time, and (2) the product of the cost complexity factor and the number of terminal nodes. In our experiments, we applied regression trees on the second cluster only and designed the bidding rules for the same. Regression tree for cluster 2 is shown in Figure 3. The non-terminal nodes present test/decisions on one or more attributes and the terminal nodes reflect the decision outcomes. Table 2 gives the rule set that sum up this regression tree.

5 Conclusions

In this paper we presented a clustering and regression trees based approach to forecast the end-price and to find the optimal bidding strategies of an online auction for autonomous agent based system. In the proposed model, the input auctions are partitioned into groups of similar auctions by *k-means* clustering algorithm.

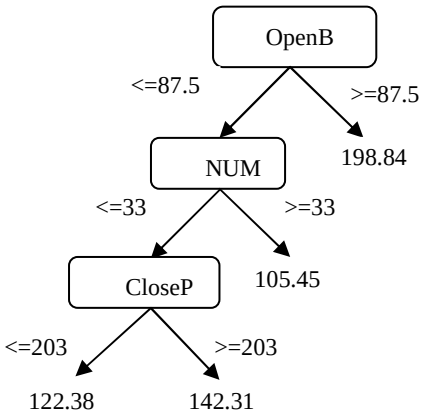


Fig. 3. Regression tree for cluster 2

Table 2. Rule set for cluster 2

1 If OpenB <=87.5	**And NUM** <=33	**And CloseP** <=203	**Then AvgB:** 122.38
2 If OpenB <=87.5	**And NUM** >=33		**Then AvgB:** 105.45
3 If OpenB >=87.5			**Then AvgB:** 198.84
4 If OpenB <=87.5	**And NUM** >=33	**And CloseP** >=203	**Then AvgB:** 142.31

The recurrent problem of finding the value of *k* in *k-means* algorithm is solved by employing elbow method using one way analysis of variance (ANOVA). Then *k* numbers of regression models are employed to estimate the forecasted price of the online auction. Based on the transformed data after clustering, bid selector nominates the cluster for the current auction whose price is to be forecasted. Regression trees are employed to the corresponding cluster for forecasting the end-price and to design the bidding strategies for the current auction. The outcome of the proposed model with clustering is compared with the classic model for price prediction. The improvement in the error measure for each cluster for a set of attributes gives support in favor of the proposed model using clustering. Optimal bidding strategies are designed by employing regression trees on each cluster and evaluating the model by test-sample cross validation approach. Further work will be focused in two directions; first, to improve the prediction model by exploiting decision trees and classification methods and secondly, study the importance of each attribute on the end price prediction for online auctions.

References

1. Zhang, S., Jank, W., Shmueli, G.: Real-time forecasting of online auctions via functional K-nearest neighbors. In: International Journal of Forecasting (2010)
2. Ghani, R., Simmons, H.: Predicting the end-price of online auctions. In: Proceedings of the International Workshop on Data Mining and Adaptive Modeling Methods for Economics and Management, held in conjunction with the 15th European Conference on Machine Learning (2004)
3. Guo, W., Chen, D., Shih, T.: Automatic forecasting agent for e-commerce applications. In: 20th international conference on Advanced Information Networking and Applications (AINA'06), pp.1--4 (2006)
4. Kehagias, D.D., Mitkas, P.A.: Efficient E-Commerce Agent Design Based on Clustering eBay Data. In: IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology Workshops, pp. 495--498 (2007)
5. Xuefeng, L., Lu, L., Lihua, W., Zhao, Z.: Predicting the final prices of online auction items. In: Expert Systems with Applications, Vol. 31, pp. 542--550 (2006)
6. Heijst, D., Potharst, R., Wezel, M.: A support system for predicting ebay end prices. Econometric Institute Report (2006)
7. Nikolaidou, V., Mitkas, P.: A Sequence Mining Method to Predict the Bidding Strategy of Trading Agents. Agents and Data Mining Interaction, pp. 139--151 (2009)
8. Jank, W., Shmueli, G.: Profiling price dynamics in online auctions using curve clustering. Technical report, Smith School of Business, University of Maryland, pp.1--28 (2005)
9. Greenwald, A., Stone, P.: Autonomous bidding agents in the trading agent competition. In: IEEE Internet Computing, pp. 52--60 (2001)
10. Byde, A., Preist, C., Jennings, N.: Decision procedures for multiple auctions. In: ACM First International Joint Conf. on Autonomous Agents and Multi-Agent Systems, pp. 613--620 (2002)
11. Anthony, P., Jennings, N.: Evolving bidding strategies for multiple auctions. In: Proc. of 15th European Conf. Artificial Intelligence, pp. 178--182 (2002)
12. Min, Z., Qiwan, W.: The On-line Electronic Commerce Forecast Based on Least Square Support Vector Machine. In: ICIC '09. Second International Conference on Information and Computing Science, Vol. 2, pp. 75--78 (2009)

Change Point Analysis for Intelligent Agents in City Traffic

Maksims Fiosins*, Jelena Fiosina, and Jörg P. Müller

Clausthal University of Technology, Clausthal-Zellerfeld, Germany,
(Maksims.Fiosins, Jelena.Fiosina)@gmail.com,
Joerg.Mueller@tu-clausthal.de

Abstract. Change point (CP) detection is an important problem in data mining (DM) applications. We consider this problem solving in multi-agent systems (MAS) domains. Change point testing allows agents to recognize changes in the environment, to detect more accurately current state information and provide more appropriate information for decision-making. Standard statistical procedures for change point detection, based on maximum likelihood estimators, are complex and require construction of parametrical models of data. In methods of computational statistics, such as bootstraping or resampling, complex statistical inference is replaced by a large computation volumes. However, these methods require accurate analysis of their precision. In this paper, we apply and analyze a bootstrap-based CUSUM test for change point detection, as well as propose a pairwise resampling test. We derive some useful properties of the tests and demonstrate their application in the decentralized decision-making of vehicle agents in city traffic.

Keywords: Multiagent decision-making, data mining, change point detection, resampling, variance, bootstrapping CUSUM test, traffic control

1 Introduction

Change point (CP) analysis is an important problem in data mining (DM), the purpose of which is to determine if and when a change in a data set has occurred. In multiagent systems (MAS) research, methods of CP analysis are applied, but not widely. One of the most popular agent-related areas of CP analysis application is web mining. Here, agents deal with automatic knowledge discovery from web documents and services, including social networks. CP detection in values of different parameters, such as number of messages, frequency of certain actions, or number of active users in some blogs are relevant for a wide number of applications, such as marketing, production, security. For example, Lu et. al. [10] applied the CUSUM algorithm in combination with shared beliefs for agent-oriented detection of network attacks, McCulloh [9] for detection of changes in social networks.

* Maksims Fiosins is supported by the Lower Saxony Technical University (NTH) project "Planning and Decision Making for Autonomous Actors in Traffic"

A traditional statistical approach to the problem of CP detection is maximum likelihood estimation (MLE) [5],[8]. In this approach, an appropriate data model is constructed, which includes a model of CP. Then a likelihood function for model parameters (including CP) is designed and an estimator of for CP is obtained as a result of the likelihood function minimization. Such an approach requires assumptions about the data model and underlying distributions as well as complex analytical or numerical manipulations with a likelihood function.

As an alternative to the classical approach, methods of computational statistics, like bootstrap or resampling, are widely used [7]. In such methods, complex analytical procedures are replaced by intensive computation, which is effective with modern computers. However, these methods only provide approximate solutions, and the analysis of their accuracy and convergence rate is very important for their correct application.

One of the most popular methods of computational statistics, used for CP analysis, is a cumulative sum bootstrapping test (CUSUM bootstrapping test) [4]. This method does not require assumptions about data models and corresponding distributions; the idea of the test is to construct so-called CUSUM plots: one on the initial sample and one on each of a large number on permuted (bootstrapped) samples. The difference between the initial plot and bootstrapped plots aids to spread regarding the CP existence. This test relies on a visual assessment of whether there is a change in the slope of the CUSUM plot (see Figure 3).

In previous publications, we applied the resampling method for analysis and DM in different kinds of systems, including information, reliability, transportation logistics, software systems, including MAS [1],[3].

The purpose of this paper is twofold. First, we explain how non-parametrical CP detection methods may be integrated into agent decision support by illustrating it on detecentralized traffic routing scenario. Second, we demonstrate how to analyze a precision of the considered tests, taking expectation and variance of resulting estimators as an efficiency criteria.

We discuss two CP tests based on methods of computational statistics: a bootstrap-based CUSUM test, and a novel test, called pairwise resampling test. We analyze the efficiency of both tests as well as show how these tests are applied in MAS systems, focusing on the traffic applications. Case studies are presented to demonstrate how CP analysis is incorporated into agents decision-making processes to verify the potential effect of the proposed approach.

The paper is organized as follows. In Section 2, we describe how CP analysis is incorporated in the decision module of agents. In Section 3, we formulate the CP problem and explain a standard CUSUM bootstrapping approach. In Section 4, we present proposed CP detection algorithms. In Section 5, we provide the most important aspects of the tests efficiency analysis. Section 6 demonstrates a case study, where the proposed methods are used in decentralized traffic scenario. Section 6 contains final remarks, conclusions and outlook.

2 CP Based Decision Making for Agents in Traffic

Appropriate DM tools are very important for a class of MAS where the individual agents use DM technologies to construct and improve a basis for local decision-making and use communication, coordination and cooperation mechanisms in order to improve local decision-making models and to provide a basis for joint decision-making [11]. In order to make appropriate decisions, agents analyze incoming data flows, construct relevant models and estimate their parameters.

The environment and behavior of other agents are subject to changes. Suppose, some other agent decides to change its plans and start acting in a new manner, some part of the environment may become unavailable for agents, new agents may appear etc. So, the old behavior of the agent becomes inappropriate.

Let us consider a simple example from a traffic domain. Let a vehicle agent plans its route through a street network of a city. The agent is equipped with a receiver device, which allows obtaining an information about times needed to travel through the streets. Based on this information, the vehicle agent makes strategic decisions regarding its route. The vehicle does not use only messages from TMC: rather it builds some model based on historical information (such a model is considered by Fiosins et. al. [6]). In the simplest case, the vehicle agent just calculates an average of some set of historical observations.

Now suppose that some change occurs (traffic accident, overloading of some street etc.). The purpose of the agent is to detect this change and make appropriate re-routing decisions.

Let us describe an architecture of an intelligent agent from this point of view. It receives observations (percepts) from the environment as well as communication from other agents (Fig. 1). Communication subsystem is responsible for receiving input data, which then is pre-processed by an initial data processing module. The information, necessary for decision-making, is obtained from initial data by constructing corresponding data models (regression, time series, rule-based etc.). A data models estimation/learning module is responsible for the estimation of the model parameters or iterative estimation ("learning"), which provides an information base for an agent. Mentioned blocks represent a DM module of the agent. The information then is transformed to an internal state of the agent, which is agent's model of the real world. It represents the agent knowledge about the environment/other agents. Based on the internal state, the agent performs its decision-making. The efficiency (utility) functions measure an accordance of the internal state as a model of the external (environment state) with goals of the agent. Based on it, the agent produces (updates) its plan to reach its goal; this process may include the internal state change. As well, the efficiency function itself (or their parameters) can change under learning process.

The decision-making process includes strategic decisions of the agents [6], which define plans of general resource distribution as well as tactical decisions, including operative decisions regarding resource usage. For example, strategic decisions of a vehicle agent may include the route choice, but tactical decisions may include include speed/lane selection. As well, the agent plans its social behavior, i.e. its interactions with other agents. The result of this process is a

construction of a plan (policy). The plan is given to an action module for the actual action selection and execution.

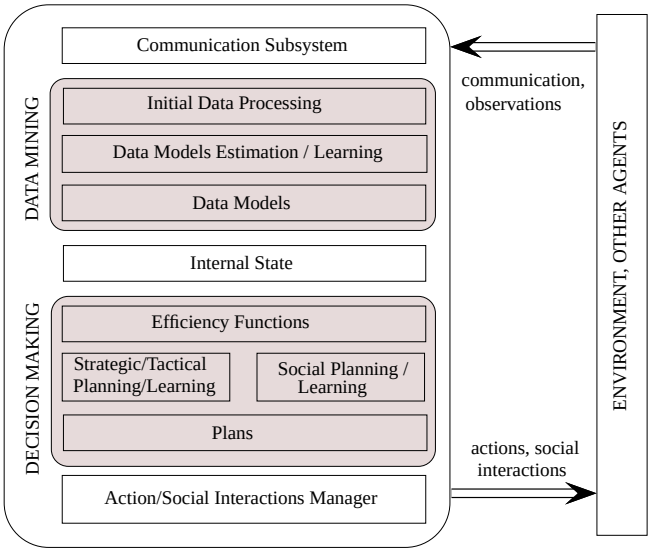


Fig. 1. An agent architecture including DM and decision making modules

Consider an example of DM module of an autonomous vehicle agent.

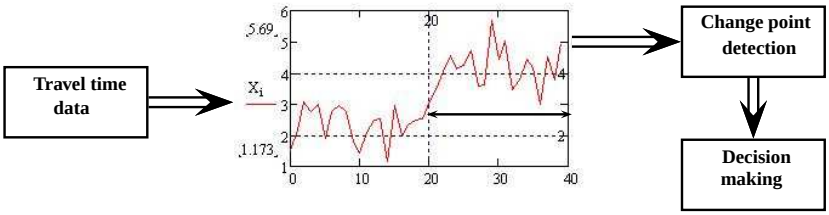


Fig. 2. DM module for strategic (routing) information processing

The vehicle receives travel time data, which are pre-processed by the initial data processing module. Then data is tested on the existence of CP. If it is detected, only data after the last CP are used in future analysis, where the current state (travel time) trough a given street is estimated. In the simplest case an average of travel times after the last CP is used.

In the next Section we present a CP detection problem as well as describe a standard bootstrap-based CUSUM test for CP detection.

3 CP Problem and CUSUM Test

Let us formulate a CP detection problem. Let $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ be a random sample. Let us divide this sample as $\mathbf{X} = \{\mathbf{X}^B, \mathbf{X}^A\}$, where $\mathbf{X}^B = \{x_1^B, x_2^B, \ldots, x_k^B\}$, $\mathbf{X}^A = \{x_1^A, x_2^A, \ldots, x_{n-k}^A\}$. We say that there is a CP at position k in this sample, if elements $\mathbf{X}^B$ are distributed according to a distribution function (cdf) $F_B(x)$, but elements $\mathbf{X}^A$ according to cdf $F_A(x) \neq F_B(x)$. The aim of a CP detection test is to estimate the value of k (clear that in the case of $k = n$ there is no CP). We are interested in a case when $F_B(x)$ and $F_A(x)$ differ by a mean value.

Note that a CP is not always visually detectable. Figure 3 represents two samples: left has exponential distribution, right has normal. Both have a CP at $k = 10$: for the exponential distribution its parameter λ changes from 1/10 to 1/20; for the normal distribution its parameter μ changes from 5 to 7. One should have an experience to see these CP visually.

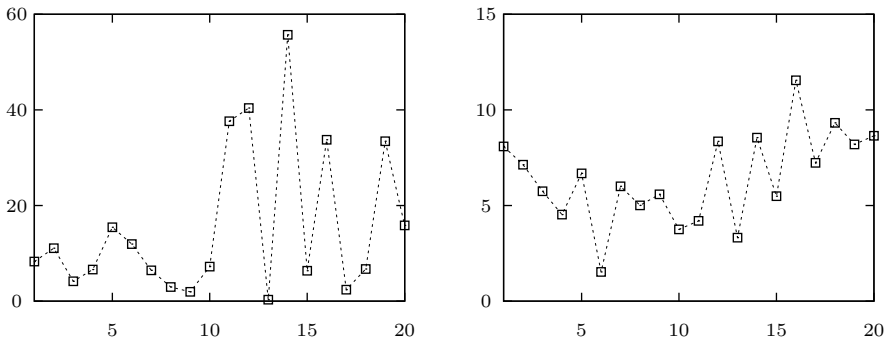


Fig. 3. A sample of 20 observations from exponential (left) and normal (right) distributions with CP at $k = 10$

A popular non-parametric approach for CP analysis is bootstrapping CUSUM test. CUSUM presents the cumulative sum of the differences between individual data values and the mean. If there is no shift in the mean of the data, the chart will be relatively flat with no pronounced changes in slope. Also, the range (the difference between the highest and lowest data points) will be small. A data set with a shift in the mean will have a slope change at the data point where the change occurred, and the range will be relatively large.

The cumulative sum S_i at each point i is calculated by the sample $\mathbf{X}$ by adding the difference between a current value and sample mean to the previous sum as

$$S_i = \sum_{j=1}^{i} (x_j - \bar{X}), \tag{1}$$

where $\bar{X}$ is the mean of the sample $\mathbf{X}$, $i = 1, 2 \ldots n$.

A CUSUM chart starting at zero will always end with zero as well: $S_n = 0$. If a CUSUM chart slopes down, it indicates that most of the data are below the mean. A change in the direction of a CUSUM indicates a shift in the average. At the CP, the slope changes direction and increases, indicating that most of the data points are now greater than the average.

In order to make a CP detection procedure more formal, a measure of the initial CUSUM line divergence from "normal" lines for given data is calculated. It is calculated using a technique known as bootstrapping, whereby N random permutations $\mathbf{X}^{*j}$, $j = 1, 2, \ldots, N$ of $\mathbf{X}$ are generated and corresponding CUSUMS S_i^{*j} are calculated by formula (1).

For a fixed point k the percentage of times where the cumulative sum for the original data exceeds the cumulative sum for the randomized bootstrap data is calculated as $p_k^* = \#\{j : S_k^{*j} \leq S_k\}/N$.

For values of p_k^* near 0 or near 1 we can say that the CP in k occurs. The idea behind this is that values S_k^{*j} approximate the distribution of CUSUMs constructed till k under assumption that data is mixed (values may be taken both before the CP and after the CP).

An example of a CUSUM test is presented on Figure 4.

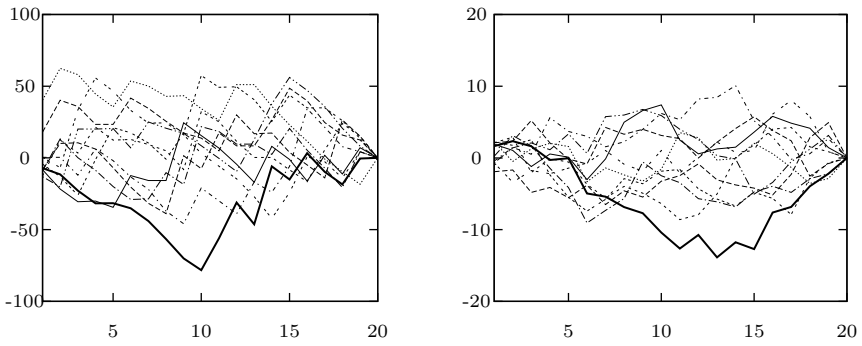


Fig. 4. CUSUM test examples with CP at $k = 10$

Here we can see a minimum of original CUSUM line at point $k = 10$; for other bootstrapped lines there is no minimum.

However, is this test reliable? Will the original CUSUM line be always outside of bootstrapped CUSUM line range? If not, then what is an expected value of the percentage of bootstrapped CUSUMS, which the original CUSUM exceeds? How this percentage differs from its expected value? All these questions should be answered during the method analysis phase; without the accurate analysis the method cannot be correctly applied.

4 Resampling Based CP Tests

4.1 CUSUM Based Test

Let us describe an algorithm for a CUSUM-based test for a CP at the point k. On r-th resampling step, we extract, without replacement, k elements from a sample $\mathbf{X}$ of size n, forming the resample $\mathbf{X}_k^{*r} = \{x_1^{*r}, x_2^{*r}, \ldots, x_k^{*r}\}$. Then we construct r-th resampling CUSUM

$$S^{*r} = \sum_{i=1}^{k}(x_i^{*r} - \bar{X}) = \sum_{i=1}^{k} x_i^{*r} - k\bar{X}, \tag{2}$$

where $\bar{X}$ is an average over the initial sample $\mathbf{X}$.

Each CUSUM is compared with a pre-defined value x, obtaining an indicator value $\zeta^{*r} = 1_{S^{*r} \le x}$.

We make N such realizations, obtaining a sequence of indicators $\zeta^{*1}, \zeta^{*2}, \ldots, \zeta^{*N}$. These values in fact approximate a cdf of CUSUMS. We estimate it as

$$F^*(x) = P\{S^{*r} \le x\} = \frac{1}{N}\sum_{r=1}^{N} \zeta^{*r}. \tag{3}$$

As a value of x we take a value of a CUSUM S, calculated by initial data. So the probability of interest is $F^*(S)$.

Low or high value of this probability allows to spread about CP existence. In principle, we can find maximal (close to 1) or minimal (close to 0) value of this probability on all k and consider this point as a CP.

A corresponding calculation of the probability $F^*(S)$ is presented in Algorithm 1.

Algorithm 1 Function CUSUM_TEST

```
1: function CUSUM_TEST(X, k, N)
2:     S = Σ_{i=1}^{k} (x_i − X̄)
3:     for r = 1 ... N do
4:         X_k^{*r} ← resample(X, k)
5:         S^{*r} = Σ_{i=1}^{k} (x_i^{*r} − X̄)
6:         if S^{*r} < S then ζ^{*r} = 1
7:         else ζ^{*r} = 0
8:     end for
9:     return 1/N Σ_{r=1}^{N} ζ^{*r}
10: end function
```

However, as the same elements can be used for calculation of ζ^{*r} on different realizations r, it leads to a complex structure of dependency between ζ^{*r}. So we should be accurate in results interpretation here. In Section 5 we provide the most important aspects of this test analysis.

4.2 Pairwise Resampling CP Test

We propose an alternative resampling-based CP test; we call it pairwise resampling test. It is based on calculation of the probability $P\{Y \leq Z\}$ that one random variable (r.v.) Y is less than another r.v. Z [2]. Suppose that the sample $\mathbf{X}^B$ contains realizations of some r.v. Y, but the sample $\mathbf{X}^A$ realizations of some r.v. Z. Our characteristic of interest is the probability $\Theta = P\{Y \leq Z\}$.

On r-th resampling step we extract one value y^{*r} and z^{*r} from the samples $\mathbf{X}^B$ and $\mathbf{X}^A$ correspondingly and calculate an indicator value $\zeta^{*r} = 1_{y^{*r} \leq z^{*r}}$.

We make N such realizations, obtaining a sequence of indicators $\zeta^{*1}, \zeta^{*2}, \ldots, \zeta^{*N}$. The resampling estimator of Θ is

$$\Theta^* = \frac{1}{N} \sum_{r=1}^{N} \zeta^{*r}. \tag{4}$$

In order to check if there is a CP, we construct a confidence interval for Θ. We produce v such estimators, denote them $\Theta_1^*, \Theta_2^*, \ldots, \Theta_v^*$. Let us order them, producing an ordered sequence $\Theta_{(1)}^* \leq \Theta_{(2)}^* \leq \ldots \leq \Theta_{(v)}^*$.

Let us select a confidence probability γ for this interval (γ is usually selected 0.95 or 0.99). We accept $[\Theta^*_{(\lfloor \frac{1-\gamma}{2} v \rfloor)}; \Theta^*_{(\lfloor \frac{\gamma}{2} v \rfloor)}]$ as a γ confidence interval for Θ.

Note that in the case of CP absence the probability Θ will be equal to 0.5, and the estimators Θ^* will be close, but different from 0.5. However, is this difference significant? In order to answer we check if value 0.5 traps into the constructed confidence interval (Algorithm 2).

Algorithm 2 Function PAIRWISE_CONFIDENCE

```
1: function PAIRWISE_CONFIDENCE(X, k, N, v, γ)
2:     X^B = subsample(X, 1, k), X^A = subsample(X, k + 1, n)
3:     for j = 1 ... v do
4:         for r = 1 ... N do
5:             y^{*r} ← resample(X^B, 1), z^{*r} ← resample(X^A, 1)
6:             if y^{*r} < z^{*r} then ζ^{*r} = 1
7:             else ζ^{*r} = 0
8:         end for
9:         Θ*_j = Σ_{r=1}^{N} ζ^{*r}
10:    end for
11:    sortΘ*
12:    return [Θ*_{(⌊(1−γ)/2 v⌋)}; Θ*_{(⌊γ/2 v⌋)}]
13: end function
```

There is again a complex dependence structure between ζ^{*r}, and so between Θ^*, because the same elements may be used in comparisons on different realizations. So true coverage probability of constructed interval will differ from γ. The goal of the algorithm analysis is to calculate the true coverage probability of this interval; then we can correctly apply the method.

5 Analysis of the CP tests accuracy

In this Section, we shortly highlight the most important aspects of the methods efficiency analysis. The complete analysis can be found in our articles [1],[2],[3].

5.1 CUSUM Based Test

We are going to calculate an expectation and variance of the estimator (3). This means that we calculate theoretically an average of the estimator and spread of the percentage of cases, when the CUSUM constructed on the original data exceeds CUSUMS constructed on the bootstrapped data.

Let y_r be a number of elements, extracted from $\mathbf{X}^B$; then from $\mathbf{X}^A$ we extract $k - y_r$ elements. Then the expectation of (3) can be expressed as

$$E[F^*(x)] = P\{S^{*r} \leq x\} = \sum_{y_r=2}^{k-1} \int_{-\infty}^{\infty} F_B^{(y_r)}(x-u) dF_A^{(k-y_r)}(u) \cdot p_{y_r}(p), \qquad (5)$$

where $F^{(k)}(x)$ is a convolution of the cdf $F(x)$ with itself.

Variance of (3) can be expressed as

$$Var[F^*(x)] = \frac{1}{N} Var\left[1_{\{S^{*r} \leq x\}}\right] + \frac{(N-1)}{N} Cov\left[1_{\{S^{*r} \leq x\}}, 1_{\{S^{*p} \leq x\}}\right], \qquad (6)$$

for $r \neq p$.

Only the covariance term depends on the resampling procedure, which can be expressed using the mixed moment $\mu_{11} = \left[1_{\{S^{*r} \leq x\}} \cdot 1_{\{S^{*p} \leq x\}}\right]$.

In order to calculate μ_{11}, we use the notation of $\boldsymbol{\alpha}$-pair [1],[3]. Let $\boldsymbol{\alpha} = (\alpha_B, \alpha_A)$, where α_B and α_A are the number of common elements extracted from $\mathbf{X}^B$ and $\mathbf{X}^A$ correspondingly on two different resampling realizations.

Than μ_{11} can be expressed by fixing all possible values of $\boldsymbol{\alpha}$:

$$\mu_{11} = \sum_{\boldsymbol{\alpha}} \mu_{11}(\boldsymbol{\alpha}) P(\boldsymbol{\alpha}). \qquad (7)$$

For the case of exponential and normal distributions we can obtain explicit formulas for the previous expressions.

5.2 Pairwise Resampling CP Test

In order to analyze properties of (4), we introduce a protocol notation [2]. Let us order a sample $\mathbf{X}^B$, obtaining an ordered sequence $\{x^B_{(1)}, x^B_{(2)}, \ldots, x^B_{(k)}\}$. Let $c_i = \#\{x^A_j \in \mathbf{X}^A : x^B_{(i-1)} \leq x^A_j \leq x^B_{(i)}\}$, $x^B_{(0)} = -\infty$, $x^B_{(k+1)} = \infty$. We call $k+1$-dimensional vector $C = \{c_1, c_2, \ldots, c_{k+1}\}$ as a protocol.

For a fixed protocol C the conditional probability of the event $\{Y \leq Z\}$ is

$$q_C = P\{Y \leq Z | C\} = \frac{1}{k(n-k)} \sum_{i=1}^{k} \sum_{j=i}^{k} c_j. \qquad (8)$$

The probability that one resampling estimator Θ_j^* will be less than Θ is given by the binomial distribution with a probability of success (8):

$$\rho_C = P\{\Theta_j^* \leq \Theta | C\} = \sum_{\zeta=0}^{\Theta r-1} \binom{r}{\zeta} q_C^{\zeta} (1 - q_C)^{r-\zeta}. \tag{9}$$

Finally the unconditional probability of coverage is calculated as $\sum_C P_C \cdot R_C$.

6 Case Study

We consider a vehicle routing problem in a street network, where vehicles receive data about travel times and are applying the shortest path algorithm looking for a fastest path to their destination. As travel times are subject to change, that's why CP analysis is performed. If CP is detected, only the data part after the last CP is taken into account.

We suppose, that travel times trough the streets are normally distributed and are subject to changes in the mean. An example of input data is shown in Figure 5 (left). The vehicle analyses CPs in such data for all streets and selects an appropriate fastest route; the route selection process is presented in Figure 5 (right).

Now consider the behavior of CP estimators. In Figure 6 (left) the CUSUM test presented in the case of CP absence. In this case, we can see a big variance of the probability $F^*(x)$ of interest (standard deviation = 0.29). This means, that there exist a big risk of considering some point as a CP, if it is not one. Figure 6 (right) demonstrates the CUSUM test in the case of CP existence. Here we see very good CP detection with practically zero variance at the CP.

Now let us consider the pairwise test. In Figure 7 (left) we see this test in the case of CP absence. Here we see smaller variance of the probability Θ^* of interest (standard deviation = 0.14 on the most of the interval). This means, that a risk of considering some point as a CP, if it is not one, is lower than for the CUSUM test. Figure 7 (right) demonstrates this test in the case of CP. Here detection is not so bad as well, however the variance of the estimator is bigger, so there is a risk to miss this CP.

We can conclude that the CUSUM test detects CP very well; however, it may consider as CP some point, which is not one. In opposite, the pairwise test is more reliable in the case of a CP absence; however, it can miss some CPs.

So for streets where CPs are rare, it is better to use the pairwise test; the CUSUM test is better for streets with often occurred CPs in travel times.

7 Conclusions

CP detection is very important task of DM for MAS, because it allows agents to estimate more accurate the environment state and prepare more relevant information for decentralized planning and decision-making. As classical statistical

methods for CP estimation are relatively complex, it is better to apply methods of computational statistics for this problem.

In this paper, we considered an application of CP detection as a part of DM module of an intelligent agent. We considered two resampling-based CP detection tests: CUSUM-based bootstrapping test and pairwise resampling test. We described algorithms of their application as well as highlighted the most important aspects of their efficiency analysis, taking expectation and variance of the estimators as the efficiency criteria.

We demonstrated an application of CP detection for vehicle agents in city traffic. This allows vehicles to detect CPs in street travel times and select more appropriate path in a street network.

The first test demonstrated good detection of CPs, however has a big variance in the case of CPs absence. The second test has smaller variance in this case, however worse detects existing CPs.

First experiments show that the demonstrated approach allows reducing the travel time of vehicles. In the future we will work on an application of computational statistics methods for different DM procedures in MAS. Another important direction is an application of our approach for different domains.

References

1. Afanasyeva, H.: Resampling-approach to a task of comparison of two renewal processes. In: Proc. of the 12th International Conference on Analytical and Stochastic Modelling Techniques and Applications. pp. 94–100. Riga (2005)
2. Andronov, A.: On resampling approach to a construction of approximate confidence intervals for system reliability. In: Proceedings of 3rd International Conference on Mathematical Methods in Reliability. pp. 34–42. Trondheim, Norway (2002)
3. Andronov, A., Fioshina, H., Fioshin, M.: Statistical estimation for a failure model with damage accumulation in a case of small samples. Journal of Statistical Planning and Inference 139(5), 1685 – 1692 (2009)
4. Antoch, J., Hušková, M., Veraverbeke, N.: Change-point problem and bootstrap. Journal of Nonparametric Statistics 5, 123–144 (1995)
5. Ferger, D.: Analysis of change-point estimators under the null-hypothesis. Bernoulli 7(3), 487–506 (2001)
6. Fiosins, M., Fiosina, J., Müller, J., Görmer, J.: Agent-based integrated decision making for autonomous vehicles in urban traffic. In: Proceedings of 9th International Conference on Practical Applications of Agents and MAS (2011)
7. Gentle, J.E.: Elements of Computational Statistics. Springer (2002)
8. Hinkley, D.V.: Inference about the change-point from cumulative sum tests. Biometrika 58(3), 509–523 (1971)
9. McCulloh, I., Lospinoso, J., Carley, K.: Social network probability mechanics. In: Proceedings of the World Scientific Engineering Academy and Society 12th International Conference on Applied Mathematics. p. 319325. Cairo, Egypt (2007)
10. Peng, T., Leckie, C., Ramamohanarao, K.: Detecting reflector attacks by sharing beliefs. In: In Proceedings of the IEEE GLOBECOM. pp. 1358–1362 (2003)
11. Symeonidis, A., Mitkas, P.: Agent Intelligence Through Data Mining (Multiagent Systems, Artificial Societies, and Simulated Organizations). Springer (2005)

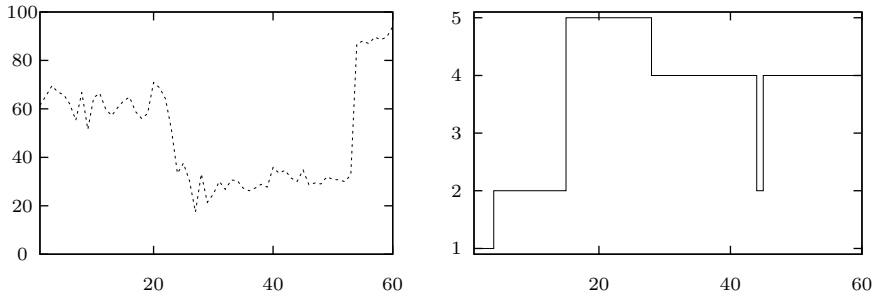


Fig. 5. Travel times on one street with 60 observations (left) and resulting route selection from 5 routes (right)

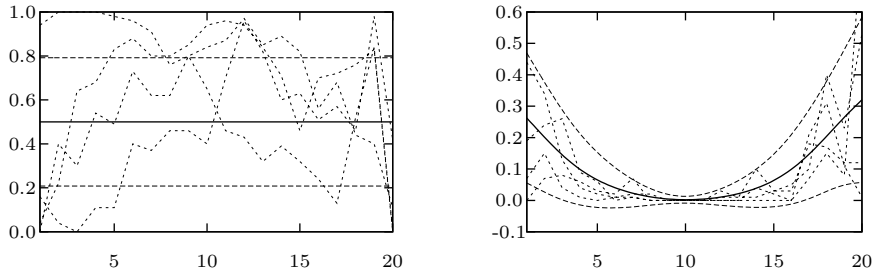


Fig. 6. Results of the CUSUM test without CP (left) and with CP at $k = 10$ (right) for a fragment of 20 observations. Straight line shows the expected value $E[F^*(x)]$ of the estimator $F^*(x)$, dashed lines show the difference between the expected value and standard deviation $E[F^*(x)] - Var[F^*(x)]^{1/2}$ of the estimator $F^*(x)$, dotted lines show several realizations of $F^*(x)$

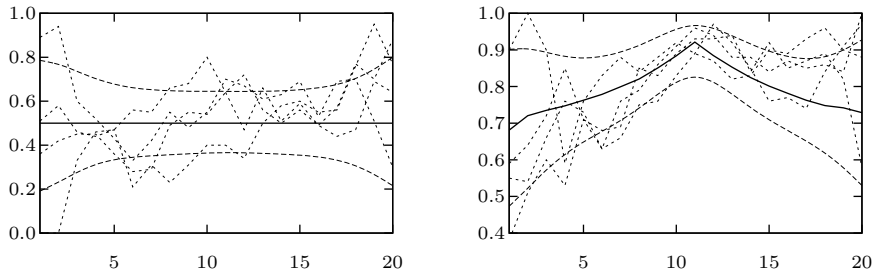


Fig. 7. Results of the pairwise test without CP (left) and with CP at $k = 10$ (right) for a fragment of 20 observations. Straight line shows the expected value $E[\Theta^*]$ of the estimator Θ^*, dashed lines show the difference between the expected value and standard deviation $E[\Theta^*] - Var[\Theta^*]^{1/2}$ of the estimator Θ^*, dotted lines show several realizations of Θ^*

A Data-driven Approach for Resource Gathering in Real-time Strategy Games

Dion Christensen, Henrik Ossipoff Hansen, Jorge Pablo Cordero Hernandez, Lasse Juul-Jensen, Kasper Kastaniegaard, and Yifeng Zeng

Department of Computer Science, Aalborg University, Denmark
{kogle,ossipoff,jorgecordero,ljuul,kkkas,yfzeng}@cs.aau.dk

Abstract. In real-time strategy games, resource gathering is a crucial part of constructing an army and becoming victorious. In this paper we present an algorithm for resource gathering that is developed based on accumulated game data. The algorithm builds upon a queue system for resource collecting agents and optimises resource gathering by utilising travel times of agents in the game world. We implement the algorithm in the testbed of StarCraft: Brood War and compare it with the built-in method for resource gathering in this game. Experimental results show a gain in the amount of resources gathered compared to the built-in method. In addition, the results demonstrate better predictability when our approach is used to gather resources in games.

1 Introduction

In real-time strategy (RTS) games, players engage each other in real-time battles for map domination. This is done by collecting resources, building an army and controlling the units to attack the opponent [2]. A typical RTS game requires players to initially focus on resource gathering, in order to provide an economic foundation for their armies.

Resource management in an RTS game is the act of collecting minerals and converting them into new units. It is an important part of winning a match of an RTS game, since resource management directly implies the ability to create more units. Resource gathering is the concrete act of having agents moving to a resource, spending time gathering the resource and returning to the resource deposit. Intuitively, resource gathering must play an important role in RTS games since battles cannot be won without a steady income of resources. We did an analysis of 200 StarCraft replays gathered from iCCup[1] that suggests a correlation between spending resources and winning the game. Fig. 1 shows the amount of minerals spent by the winners along with their raze score–a score indicating the number of buildings a player has destroyed. The figure shows a tendency to an increased amount of minerals for higher raze scores. This suggests that it is important to obtain a high amount of resources in order to win a match.

[1] http://www.iccup.com/

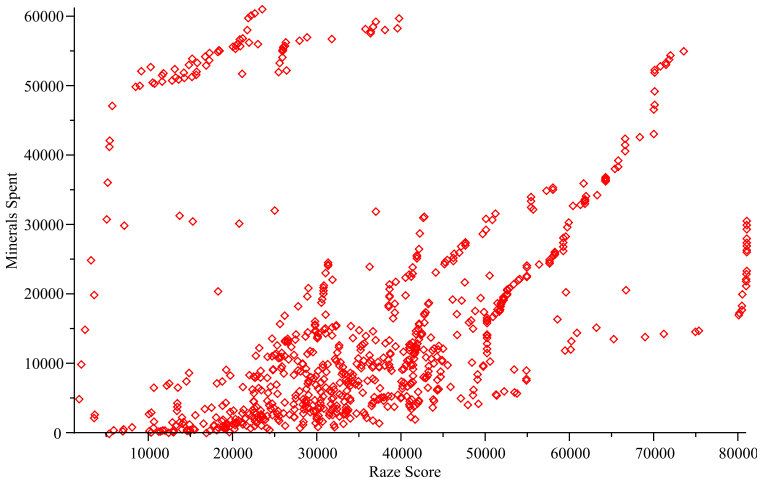


Fig. 1. The amount of minerals gathered compared to the raze score. Data is gathered from the winning players every 120 frames from a total of 200 replays.

Unfortunately, topics on resource management (e.g. resource gathering) have not been well explored in the literatures. Previous work by *Chen et al.* [1] neglects the importance of the time required for agents to travel in the game world. *Wintermute et al.* [5] concern about strategies on preventing agents from colliding when they are gathering resources. We take a step further to find out the difficulty on designing a realistic approach on resource gathering. Essentially, it is due to a large amount of uncertainty in RTS game [3]. As we recall, the development of a resource gathering algorithm requires the computation of travel time between a resource and a resource deposit. However the computation is inaccessible since agents do not enjoy constant travel styles including speed, acceleration and so on.

In this paper we take a data-driven approach to design the resource gathering algorithm in an RTS game particularly in StarCraft: Brood War. We exploit the accumulated gameplay data to compute travel times for agents in the algorithm development. This is made possible due to an increasing amount of available replay data. An example of this is iCCup that supports data for StarCraft, Warcraft III and DotA. The use of replay data has already made an impact on the strategy prediction in RTS games [4]. In addition, in order to track the movement of agents in gathering resource, we implement a queue system to distribute agents across resources in a game world. We evaluate the algorithm performance and compare it with the resource gathering algorithm built in StarCraft: Brood War.

The rest of the paper is organized as follows. Section 2 introduces the algorithm for resource gathering in RTS games. Section 3 shows the algorithm performance in the experiments. Section 4 summarizes our contribution and discusses related research issues.

2 Efficient Resource Gathering

In RTS games, a player usually does not take a direct control on the agents that have been assigned to gather resources. When an agent is ordered to gather resources, the agent will gather the resources, return to the nearest resource deposit, and then continue to gather resources until given a different order.

Depending on the RTS game, different numbers of agents are allowed to gather a resource. If a resource becomes completely occupied when an agent is returning to continue gathering, the agent may go to a different resource, if one exist, or wait until the resource is available. Choosing a different resource may lead to the appearance of an agent regretting their previous goal and choosing a new goal. This will in some cases result in the agent leaving a resource that is becoming available shortly for a resource that will be occupied shortly, thus wasting time on traveling. Waiting for availability may cause an agent to waste time waiting instead of moving to a new resource. Any time spent not moving to the correct resource, or waiting for this resource, causes a loss compared to the optimal behaviour.

The erratic movement may cause the resource income rate to spike or drop– when an agent chooses a new path–making it difficult to predict the amount of resources available at a later point in time. To avoid this, direct control can be applied to the agents, where the future availability of the resource is considered before moving.

2.1 Problem Definition

Given a set of agents $A = \{a_i | i = 1, \cdots, n\}$ and a set of sites $M = \{m_j | j = 1, \cdots, l\}$ located in a two-dimensional euclidean space (each site m_j has an attached amount of resources $r_j \in \mathbb{Z}^+$ and $\forall_{j \neq k} r_j = r_k$), choose a subset $S \subseteq G$ of gathering tasks $G = \{A \times M\}$ such that the total amount of resources $R = l * r_j$ is gathered in a minimal time T.

An amount of resources is gathered when an agent collected a resource from a site and delivered it to a depot D. Consider that for each site m_j there exists exactly one queue Q_j containing the agents that will gather this resource next. A site queue is defined as an totally ordered set $Q_j = \{a_h | h = 0, \cdots, m\} \subseteq A$, where each element is an agent in the queue. The first element $a_0 \in Q_j$ is the agent that may use the resource first. Each agent is assigned to at most one resource queue at a time. When an agent has finished gathering, they are removed from the queue.

We model T as the total time that is required to execute every gathering task $s_{i,j} = (a_i, m_j) \in S$ in time $t_{i,j}$. Specifically, a gathering task $s_{i,j}$ is completed in a round-trip time $t_{i,j}$ after agent a_i travels from D to a site m_j, potentially waits in line on a queue, collects resources and goes back to D. Equation 1 shows the aforementioned round-trip time calculation that we aim to minimize by applying efficient resource gathering.

$$t_{i,j} = tt^i_{j,D} + max\left[0, rt_j - tt^i_{j,D}\right] + C + tt^i_{D,j}, \tag{1}$$

where $tt^i_{j,D}$ is the time required for the agent a_i to travel from depot D to the resource m_j. rt_j is the remaining time for site m_j to become available after all agents in queue Q_j complete their work. Thus, the time agent a_i would wait in line is the remaining time rt_j minus the time that has already been spent on traveling. The constant collecting time C is also added along with the travel time to return to the resource deposit $tt^i_{D,j}$.

By using site queues, it is possible to determine the best site to send agents in order to minimize the time required for the agent to return to D with an amount of resources. Equation 2 states the remaining time rt_j as follows:

$$rt_j = (m+1) * C, \tag{2}$$

in other words, rt_j is the time agent $a_i \notin Q_j$ waits for all $m+1$ agents $a_h \in Q_j$ to finish collecting resources.

2.2 Algorithm Implementation

The equations presented in the previous section have lead to the development of an algorithm based on the same principles. The algorithm is a practical utilization of the equations.

Minimizing the round-trip time for an agent will cause the agent to always pick the resource that will allow for the fastest delivery of an amount of resources in a greedy fashion. The main mining algorithm, seen in Algorithm 1, uses a sub-algorithm *Work(Q,a)*.

Algorithm 1 Resource gathering algorithm

Declarations
$Q_j, \ldots, Q_l$: resource queues
m_j: resource site

```
1:  time ← ∞
2:  for all a_i ∉ ⋃_{j=0}^{l} Q_j do
3:      for h = 0 ... m do
4:          Q_w ← Q_h ∪ {a_i}
5:          if Work(Q_w, a_i) + tt^i_{w,D} < time then
6:              time ← Work(Q_w, a_i) + tt^i_{w,D}
7:              best ← h
8:          end if
9:      end for
10:     Q_best ← Q_best ∪ {a_i}
11: end for
```

The algorithm is run every time when new agents need to be assigned to a resource queue. In line 2 the algorithm iterates through every agent that has not yet been assigned to a queue, followed by a linear run through every resource,

lines 3-10 to find the resource that requires the least amount of time for the agent to return with a deposit, and adds the agent to the queue.

Algorithm 2 takes as parameters a queue Q_j and an agent a and returns the total time spent on traveling to the resource site, waiting in line and gathering from the resource. The algorithm works by recursively calculating the work load of the agents ahead in the queue (lines 5-6) to calculate the time spent waiting in line. Since the time spent on traveling is included in the return value in line 13, the travel time will have to be subtracted from the waiting time in lines 8-12.

Algorithm 2 *Work(Q,a)*

Parameters
Q_j: resource queue
a: agent
Declarations
$p \leftarrow h$, where $a = a_h \in Q_j, 0 \leq h \leq m$: position of a in Q

1: **if** $p = 0$ **then**
2: **return** $tt^i_{j,D} + C$
3: **end if**
4: $workLoad \leftarrow 0$
5: **for** $h = (p-1) \dots 0$ **do**
6: $workLoad \leftarrow workLoad + Work(Q_j, a_h)$
7: **end for**
8: **if** $workLoad > tt^i_{j,D}$ **then**
9: $workLoad \leftarrow workLoad - tt^i_{j,D}$
10: **else**
11: $workLoad \leftarrow 0$
12: **end if**
13: **return** $tt^i_{j,D} + workLoad + C$

Consider the scenario on Fig. 2 in which three agents, A, B and C must gather resources from m_0 and m_1 by transporting resources from these positions to the drop-off location R. By applying Algorithm 1 to this scenario the actions of the agents can be observed in Table 1. Each queue is initially empty and each agent is initially not assigned to a resource.

The first three lines in the trace displays each agent being assigned to their first resource. The algorithm takes into account the distance to be travelled for each agent as well as the remaining time for agents that have already been assigned to a resource. When an agent returns to R, Algorithm 1 is run again, to assign the agent or agents to their next resource. Over the course of a session, each agent may be assigned to different resources, depending on their availability.

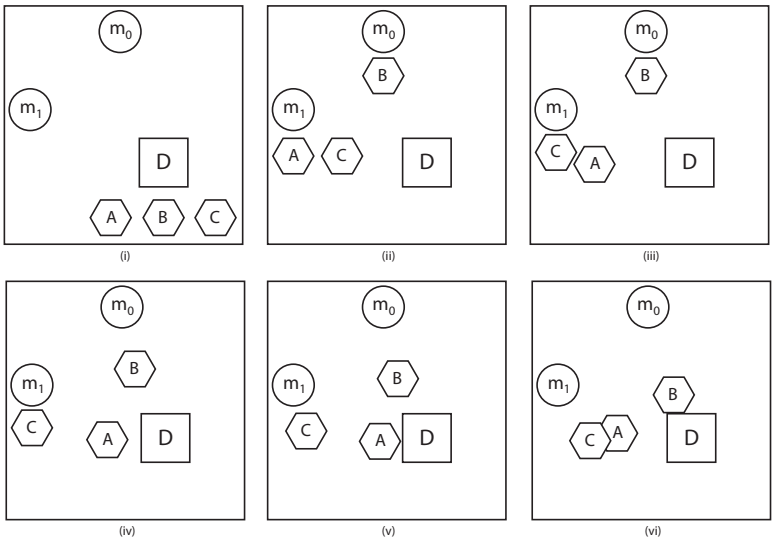


Fig. 2. (i) None of the agents have moved. A is assigned to m_1, B to m_0, C to m_1. (ii) A is done using m_1, B is gathering m_0, C starts gathering m_1. (iii) B is done using m_0, A is moving to D, C is still gathering m_1. (iv) C is done using m_1, A is still moving to D, B is moving to D. (v) A delivers to D and is assigned to m_1, B is still moving to D, C is moving to D. (vi) B delivers to D and is assigned to m_0, A is moving to m_1, C is still moving to D.

Time	**Agent**	**Action**	**Queues**	**Resources**
0	A	Move to m_1 (110)	$Q_{m_0} = \emptyset$ $Q_{m_1} = \{A_0[work \leftarrow 190]\}$	0
0	B	Move to m_0 (126)	$Q_{m_0} = \{B_0[work \leftarrow 206]\}$ $Q_{m_1} = \{A_0[work \leftarrow 190]\}$	0
0	C	Move to m_1 (126)	$Q_{m_0} = \{B_0[work \leftarrow 206]\}$ $Q_{m_1} = \{A_0[work \leftarrow 190], C_1[work \leftarrow 80]\}$	0
190	A	Move from m_1 (90)	$Q_{m_0} = \{B_0[work \leftarrow 16]\}$ $Q_{m_1} = \{C_0[work \leftarrow 80]\}$	0
206	B	Move from m_0 (108)	$Q_{m_0} = \emptyset$ $Q_{m_1} = \{C_0[work \leftarrow 64]\}$	0
270	C	Move from m_1 (90)	$Q_{m_0} = \emptyset$ $Q_{m_1} = \emptyset$	0
280	A	Deliver & move to m_1 (90)	$Q_{m_0} = \emptyset$ $Q_{m_1} = \{A_0[work \leftarrow 170]\}$	8
314	B	Deliver & move to m_0 (108)	$Q_{m_0} = \{B_0[work \leftarrow 188]\}$ $Q_{m_1} = \{A_0[work \leftarrow 136]\}$	16

Table 1. A sample trace for the scenario on Fig. 2. From the left is the current time, the affected agent, the action of the agent, the current state of both resource queues and the amount of gathered resources.

2.3 Algorithm Analysis

In StarCraft: Brood War, agents do not move at a constant speed. An agent, given a movement order while standing still, will first accelerate towards a maximum speed and decelerate before reaching its destination. Since the testbed of the algorithm, StarCraft: Brood War, is a closed source, access to the algorithm used for pathfinding in the game is restricted. To compensate for this, travel times are measured and stored.

Every time an agent moves between the resource deposit and a resource, the time of the travel is recorded. Information on the travel is saved in a lookup table, including the starting position, the destination and the time. As the function will only be used to decide the travel time between a resource deposit and the stationary resource, the amount of starting position/destination pairs are limited, making this approach possible.

Whenever another agent is moving in a path equal to a recorded path, the original value is used and potentially revised. Given enough data, all possible paths an agent may use when mining resources, is known for a specific map.

The information has been gathered by using the resource gathering algorithm where the travel function is defined as a function that returns a previously calculated value. If no data matching the source and destination exists, the function returns a low value to allow the agent to take the path and thereby gather previously unknown data.

The complexity of Algorithm 1 is $O(n^3 + TIME(Travel))$ bounded by the number of agents n, where $TIME(Travel)$ is the time required to run the travel-function. As we use a lookup table to compute travel time thereby making the complexity as $O(n^3)$ assuming a lookup can be made in constant time.

Algorithm 1 is general in the sense that it can be applied for RTS games having the following rules.

- Several resources are present
- A resource contains several units of the resource
- Agents move between a resource and a drop-off location
- Agents do not collide
- Only one agent may use the resource at a time

The travel time between positions must either be known or it must be possible to estimate this value. A general algorithm for calculating travel time has not been presented, as the value is very dependant on the environment. If an estimate for the value is used, the algorithm does not guarantee to use the lowest round-trip time for each agent.

3 Experiments

We evaluated our algorithm against the built-in method used for resource gathering in StarCraft: Brood War. StarCraft: Brood War is a science-fiction RTS game developed by Blizzard Entertainment in 1998. This particular game was

Fig. 3. In-game screenshot from a match in StarCraft: Brood War featuring a range of agents and buildings.

chosen as the testbed for various reasons: it is the best selling RTS game of our time[2], incorporates all the general features of an RTS and is accessible through open source APIs.

3.1 StarCraft Game

The StarCraft game features three distinct races each of which has individual tech trees. The race has a unit type that is capable of gathering resources and constructing buildings. Two types of resources exist–minerals and gas–which are gathered by an agent traveling to the resource, spending time gathering the resource and then returning to a resource deposit to deliver the gathered resources. Fig. 3 shows an example of a match in StarCraft. The middle building is the resource deposit of the *Terran* race. The blue crystals contain the mineral resource while the top left building is an extractor for the gas resource.

All communication with StarCraft is done using the *Brood War Application Programming Interface*[3] (BWAPI), an open source C++ framework allowing two-way communication with StarCraft. BWAPI allows communication through objects and function calls, as opposed to input via human input devices. In effect, this allows the user to, among other things, observe events in the game and react upon these.

[2] According to VGChartz, `http://www.vgchartz.com/`
[3] `http://code.google.com/p/bwapi`

Currently, StarCraft has implemented an automatic resource gathering system that is responsible for resource management in the game. The method is initialized by selecting an agent and invoking the *gather* command on a specific resource. The agent will then move to and gather the resource, return to the resource depot with a deposit, move to the resource again, and so forth. If a resource is occupied on arrival, the agent will move to the closest resource available and start gathering that instead.

3.2 Experimental Setting

We implemented our algorithm and compared on the performance with the built-in resource gathering method in StarCraft. We used both methods to train 18 agents on the map *Astral Balance*. Though the difference of starting positions for agents is usually considered negligible, in order to ensure fairness, all tests were performed in the upper right corner. We put the starting positions next to a mineral cluster consisting of eight mineral fields, each holding 1500 minerals, as well as a single gas geyser. We evaluate the performance of both algorithms on two aspects. One is an average resource collected by agents over game frame and the other is a standard deviation of gathered resources as time passes.

3.3 Experimental Results

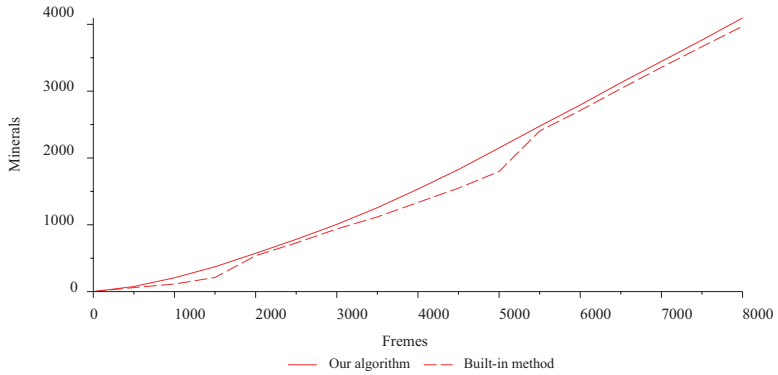


Fig. 4. Comparison between two methods, showing the amount of minerals gathered over time.

Fig. 4 shows a comparison between two methods on the amount of collected minerals in the course of 8000 frames. The data is based on 10 runs and averaged according to minerals for both algorithms. Clearly, our algorithm increases the

income of the player compared to the built-in method. Furthermore, the smoothness of the curve for our algorithm as opposed to the built-in method indicates a higher predictability. The predictability is further elaborated in Fig. 5, picturing the standard deviation in the course of 10 runs. Fig. 5 shows that the standard deviation of the minerals collected by both methods grows as time passes. There is a clear tendency that the deviation of the built-in method grows faster than that of the proposed algorithm.

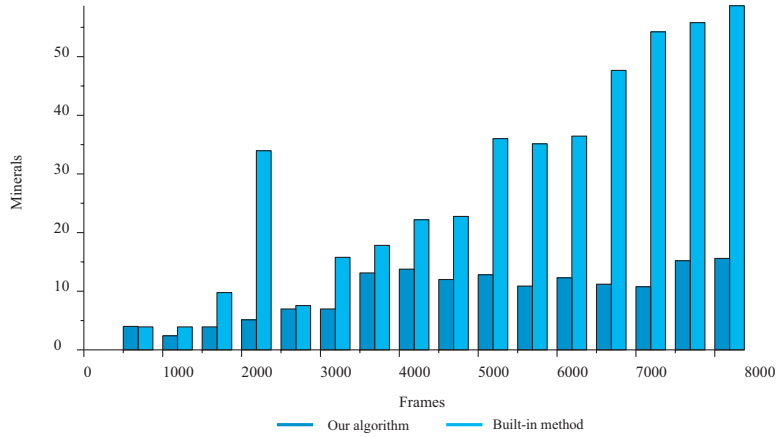


Fig. 5. Comparison on the standard deviation of the both methods.

4 Discussion

Exploiting game data to compute travel times facilitates the design of a resource gathering algorithm in spite of an unknown pathfinding method. Experiments show that the algorithm provides an increase in the amount of resources gathered, compared to the built-in approach. Furthermore the experiments show that our algorithm is more predictable. The predictability can be used to predict the amount of resources collected within some time frames.

Currently our algorithm is more *selfish* in the sense that each agent makes decisions that allow for the fastest mineral gain for itself, but not necessarily the best for all of agents in a team. It might be beneficial to run the algorithm over a group of agents by having them act as a team and optimizing the group income instead of the individual income over time.

The increased predictability of income using our algorithm makes it eligible for use in other aspects of an RTS game. An example is to develop a scheduler for construction of, for example, military units. By using our algorithm it is possible to predict the time when the scheduled production line is finished, even if the resources for the units should first be gathered. The algorithm enables the

possibility of acquiring an accurate estimate on the potential gain from utilizing resources from other areas in the game world. This is interesting for RTS games in which new resource deposits may be created by a player.

Notice that the algorithm depends on the implementation for computing travel times, which need to consider the agent collision. In some RTS games, the travel time is insignificant due to the distance from a resource to a resource deposit, the number of agents or other game specifics. In RTS games similar to StarCraft we expect to observe similar performance of our proposed algorithm.

References

1. H. Chan, A. Fern, S. Ray, N. Wilson, and C. Ventura. Online Planning for Resource Production in Real-Time Strategy Games. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 65–72, September 2007.
2. S. Rabin, editor. *Introduction to Game Development*, chapter 1.1 – A Brief History of Video Games, pages 3–36. Charles River Media Inc., 2005.
3. M. Sharma, M. Holmes, J. Santamaria, A. Irani, C. Isbell, and A. Ram. Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL. In *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.
4. B. Weber and M. Mateas. A Data Mining Approach to Strategy Prediction. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, 2009.
5. S. Wintermute, X. Joseph, and J. E. Laird. SORTS: A Human-Level Approach to Real-Time Strategy AI. In *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference*. The AAAI Press, 2007.

A Multi-Agent Based Approach To Clustering: Harnessing The Power of Agents

Santhana Chaimontree, Katie Atkinson and Frans Coenen

Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK.
{chaimontree,atkinson,coenen}@liverpool.ac.uk

Abstract. A framework for multi-agent based clustering is described whereby individual agents represent individual clusters. A particular feature of the framework is that, after an initial cluster configuration has been generated, the agents are able to negotiate with a view to improving on this initial clustering. The framework can be used in the context of a number of clustering paradigms, two are investigated: K-means and KNN. The reported evaluation demonstrates that negotiation can serve to improve on an initial cluster configuration.
keywords: Multi-Agent Data Mining, Clustering

1 Introduction

Data Mining and Multi-Agent Systems (MAS) are well established technologies which are finding increasing application. One of the current challenges of data mining is how to cope with the ever increasing size of the data sets that we wish to mine. A highlevel answer is to adopt and apply greater computational power. This can be achieved in a number of manners. One approach is to make use of distributed [6, 8, 10, 21, 22, 25] or parallel [13, 26, 27] processing techniques so that several processors can be applied to the problem. This of course assumes that appropriate "farms" of processors are available. However, a more specific disadvantage is that of centralised control and lack of generality. Distributed and parallel data mining techniques typically assume a "master" process that directs the data mining task, therefore control is centralised at the master process and consequently these systems lack robustness. Further, distributed and parallel approaches tend to be directed at specific data mining applications and are difficult to generalise (because of the centralised control inherent to these systems). MAS offer an alternative to handling large quantities of data by harnessing the power of numbers of processors, with the added advantages that control is not centralised and consequently such systems can be much more robust and versatile.

A MAS is essentially a collection of software entities (agents) that are intended to cooperate in some manner so as to undertake some processing task. An important aspect of this cooperation is that the agents behave in an autonomous manner; they *negotiate* with one another to complete a given task rather than being directed to do so by some master process. The idea of adopting MAS technology for data mining is therefore an attractive one. Multi-agent

Data Mining (MADM) or Agent Assisted Data Mining (AADM) also allows for distributed data to be mined effectively without the need to first move the data into a data warehouse. This offers advantages where it is not easy or not possible for data to be first collated. MADM also supports the creation of frameworks that can be allowed to grow in an almost anarchic manner, agents can be easily incorporated into such frameworks as long as they comply with whatever protocols have been specified. The nature of these protocols remains a research issue. A number of Agent Communication Languages (ACLs) have been proposed but often these are difficult to fit to particular applications; it is therefore frequently necessary to extend or partially replace the set of performatives that are specified as part of these ACLs in order to taylor them to the specific scenario.

This paper describes an MADM framework, a set of agent communication performatives and supporting protocol, directed at unsupervised learning (clustering). The framework comprises four general categories of agent: user agents, data agents, clustering agents, validation agents (there are also house keeping agents, but these can be considered to be orthogonal to the task of data mining). Some of these agents are persistent while others are spawned as required, and have a lifetime equivalent to the duration of a given clustering task. To support the desired MADM communicates a dedicated set of performatives have been derived. A particular feature of the framework is that it supports negotiation between agents. This negotiation process allows for the enhancement of clusters once an initial cluster configuration has been established.

2 Previous Work

This section presents a review of some related work to that described in this paper. The section commences with some general background to MADM and then continues with a brief review of some parallel work on MAS clustering, highlighting the distinction between this work and the proposed MAS clustering approach.

There are two main paradigms for the interaction and integration between agent and data mining [4]: (i) data mining-driven agents which is the use of data mining to support the abilities of agents such as adaptation, coordination, learning, reasoning, etc. and (ii) agent-driven data mining, commonly known as Multi-Agent Data Mining (MADM), is the use of a collection of agents to perform data mining tasks. Surveys of agent-based distributed data mining can be found in [12, 17, 20].

PADMA [14] and PAPYRUS [2] are two of the earliest (late 1990s) reported multi-agent clustering systems. These systems aimed to achieve the integration of knowledge discovered from different sites with a minimum amount of network communication and a maximum amount of local computation. PADMA uses a facilitator (or coordinator) agent to direct interaction between the mining agents. As such, it is based on a centralised architecture. PADMA agents are used to access local data and perform analysis. The *local clusters* are collected centrally to generate the *global clusters*. In addition, PADMA can be used to

generate hierarchical clusters in the context of document categorisation. On the other hand, PAPYRUS differs from PADMA in that it is a distributed clustering system. PAPYRUS adopted a Peer-to-Peer model where both data and results can be moved between agents according to given MAS strategies. A more recent multi-agent clustering system is KDEC [16]. KDEC is a distributed density-based clustering algorithm also founded on the Peer-to-Peer model. In KDEC density estimation samples are transmitted, instead of actual data values so as to preserve data privacy and minimize communication between sites. A multi-agent clustering system directed at documents has been proposed in [24]; the objective here is to improve the accuracy and the relevancy of information retrieval processes. Kiselev et al. [15] proposed a clustering agent based system dealing with data streams in distributed and dynamic environments whereby input data sets and decision criteria can be changed at runtime (clustering results are available at anytime and are continuously revised to achieve the global clustering).

The above systems use agents to facilitate data privacy and support distributed data clustering (mining). There is little reported work on an agent based clustering systems that support intra-agent negotiation in order to refine (enhance) cluster configurations. In [1] Agogino et al. proposed an agent-based cluster ensemble approach to generate a best cluster configuration. Reinforcement learning, to maximise a utility function with respect to the original clustering results, is used to achieve the desired best clustering. However, the approach proposed in the paper operates in a different manner by using negotiation among agents to improved the quality of clustering result. It is argued that this approach harnesses the true power of agents.

3 The MADM Framework

The proposed MADM framework, as noted above, comprises four categories of agent:

1. User agents.
2. Data agents.
3. Clustering agents.
4. Validation agents.

User agents are the interface between end users and the MADM environment. The agents are responsible for obtaining the input from the user, spawning clustering agents in order to perform the clustering task and presenting the derived clustering result. To the above list of specific MADM agents we can also add a number of housekeeping agents that are utilised within the MADM framework.

Data agents are the "owners" of data sources. There is a one-to-one relationship between data agents and data sources. Data agents can be thought of as the conduit whereby clustering agents can access data.

Clustering agents are the "owners" of clusters. Groups of clustering agents can be though of as representing a clustering algorithm. With respect to this paper the K-means and K-Nearest Neighbour (KNN) clustering algorithms have

been adopted; however, our collections of clustering agents could have been configured to perform some alternative form of clustering (for example hierarchical clustering). A number of clustering agents will be spawned, as required, by a user agent in order to perform some clustering task. Thus, each clustering agent represents a cluster and is responsible for selecting a record from a data set and determining whether that record would belong to its cluster or not. The number of clustering agents, therefore depends on the number of clusters (K). In the case of the K-means algorithm the number of clusters is predefined; thus, by extension, the number of clustering agents that will be spawned will also be predefined. In the case of the KNN approach only one initial clustering agent will be spawned; then, as the KNN algorithm progresses further clustering agents may be created. Details concerning the operation of K-means and KNN with respect to the proposed MADM framework will be presented in Section 5. Clustering agents collectively have two principal functions: (i) initial generation of a "start" cluster configuration, and (ii) cluster refinement.

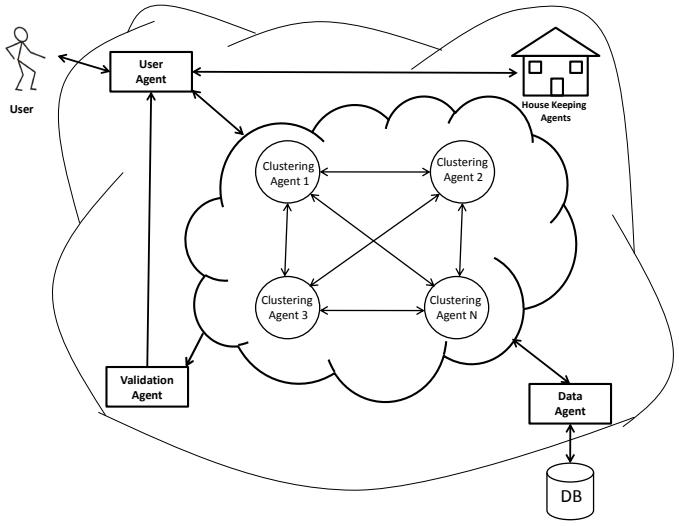


Fig. 1. Proposed MADM Framework

Validation agents are a special type of agent that performs validation operations on clustering results. Each validation agent is the "owner" of a technique for measuring the "goodness" of a given cluster configuration. In the current system validation agents consider either cluster cohesion or cluster separation or both.

A possible configuration for the proposed MADM framework, incorporating the above, is presented in Figure 1. The Figure includes a User Agent, a collection of Clustering Agents, a Data Agent, a Validation Agent and some housekeeping agents. The directed arcs indicate communication between agents. Note that communication can be bidirectional or unidirectional and that the Data Agent directly communicates with each Clustering Agent. Intra-communication

between Clustering Agents takes follows a protocol that permits negotiation about cluster exchange to take place.

The framework has been realised using the Java Agent Development Environment (JADE) [3]. The house keeping agents provided by JADE, include the AMS (Agent Management System) agent and the DF (Directory Facilitator) agent. The AMS agent is responsible for managing and controlling the lifecycle of other agents in the platform, whereas the DF agent provides a "yellow pages" service to allow agents to register their capabilities.

4 Agent Communication within the Framework

The agents within our framework need to be able to communicate to carry out their tasks. JADE provides a communication mechanism that makes use of the FIPA ACL performatives [9]. However, as has been discussed previously in [19], the FIPA ACL has limited applicability to dialogues not involving purchase negotiations. Given that we wish to permit the agents in our system to engage in dialogues about the exchange of items between cluster configurations, we need a more expressive language to support this communication. As such, we have defined and implemented a set of performatives to enable our agents to engage in negotiations about the suitability of moving items/merging between clusters. At the highest level, the performatives are categorised as follows: holding a dialogue; performing the clustering task, negotiating about the movement of items between clusters; informing others about the clustering results.

The performatives are defined axiomatically in terms of the pre-conditions that must hold for an agent to be able to use the performatives and the post-conditions that apply following the use of the performatives. The agents use the performatives as part of a protocol that governs the exchange of information between the agents. For reasons of space we do not include here the details of the semantics of the performatives, but instead describe the communication protocol that the agents follow when using the communication language. We indicate in italics the performatives used at each stage.

A dialogue opens (*mining request*) which triggers a mining request to the data, cluster and validation agents to join. Once the recipient agents have entered the dialogue (*join dialogue*), the clustering task is performed (*inform data*). On completion of the initial clustering, the agents evaluate the cohesion and seperation of their clusters and as a result may broadcast to other agents that items be moved between clusters (*propose item move*). The recipients of the proposal can then reply by accepting (*accept item move*) or rejecting the proposed move (*reject item move*). We also permit retraction of a proposed item move (*retract item move*) if it is subsequently found to yield an unsuitable configuration. When the items have been moved between clusters and the agents are happy to accept the results, the clustering agents inform the validation agent of the new configurations (*inform cluster*). The overall cluster configuration is then sent to the user agent (*inform cluster configuration*), after which moves are made for agents to exit the dialogue (*leave dialogue*) and subsequently end it (*close dialogue*).

The way in which the reconfiguration of clusters happens is dependant upon the clustering algorithm used, as will be described in the next section.

5 Operation

The operation of the proposed MADM clustering mechanism is described in this section. As noted in the foregoing, Clustering Agents are spawned by a User Agent according to the nature of the end user's initial "clustering request". Fundamentally there are two strategies for spawning Clustering Agents: the K-means strategy and the KNN strategy. In the K-means strategy the user pre-specifies the number of clusters, K, that are required; in which case K Clustering Agents are spawned. In the case of the KNN strategy the MADM process decides how many clusters are required, thus initially only one Clustering Agent is spawned (more may be generated later as required). The operation of the proposed MADM clustering mechanism comprises two phases: a bidding phase and a refinement phase. During the bidding phase Clustering Agents compete for records by "bidding" for them in an "auction" setting where the Data Agent acts as the auctioneer. For each record the Clustering Agent that poses the best bid wins the record and includes it in its cluster (see Sub-sections 5.1 and 5.2). During the refinement phase each Clustering Agents tries to pass unwanted records (records that no longer fit within its cluster definition) to other agents. This can also be conceptualised in terms of an auction; each Clustering Agent acts as a local auctioneer and tries to sell its unwanted records by inviting other clustering agents to bid for them. The operation of the refinement phase is entirely independent of the spawning strategy adopted. However the operation of the bidding phase differs according to the nature of the spawning strategy. The rest of this section is organised as follows. In Sub-sections 5.1 and 5.2 the operation of the biding phase with respect to the K-means and KNN spawning strategies are described. Sub-section 5.3 then describes the operation of the refinement phase.

Table 1. Bidding phase founded on K-means Spawning Strategy

Phase I: Bidding using K-means
Input: Dataset ($D = \{d_1, d_2, \cdots, d_n\}$), the desired number of clusters (K) Output: An initial clustering configuration
1. User Agent spawns K Clustering Agents ($D = \{c_1, c_2, \cdots, c_K\}$) 2. Each Clustering Agent sends a data request to the indicated Data Agent 3. Data Agent sends first K records ($\{d_1, d_2, \cdots, d_K\}$) to the K Clustering Agents; d_1 to c_1, d_2 to c_2, and so on. 4. Each Clustering Agent calculates its cluster centroid 5. $\forall d_i \in D$ ($i = K + 1$ to n) 6. $\forall c_j \in K$ ($j = 1$ to K) 7. $bidDistance = d_i -$ *centroid* c_j 8. Allocate d_i to c_j so as to minimise $bidDistance$

Table 2. Bidding phase founded on KNN Spawning Strategy

Phase I: Bidding using KNN
Input: Dataset ($D = \{d_1, d_2, \cdots, d_n\}$), threshold t Output: An initial clustering configuration
1. User Agent spawns a single Clustering Agent (c_1) 2. $K = 1$ 3. $\forall d_i \in D$ ($i = 2$ to n) 4. $\forall c_j \in K$ ($j = 1$ to K) 5. $bidDistance = nearest\ neighbour\ to\ d_i$ 6. IF $\exists c_j \in C$ such that $bidDistance < t$, allocate d_i to c_j so as to minimise $bidDistance$ 7. ELSE $K = K + 1$, spawn Clustering Agent c_K, allocate d_i to c_K

5.1 Biding Phase founded on the K-means Spawning Strategy

The operation of the bidding process with respect to the K-means strategy is presented in Table 1. K clustering agents are spawned to represent the clusters (line 1). Each Clustering Agent is then allocated a single record, by the identified Data Agent, and this record is used to represent the centroid of each cluster (lines 2 to 4). The clustering agents then bid for the remaining records in D (lines 5 to 8). In the current implementation the *bidDistance* equates to the "distance" of d_i to the nearest neighbour of d_i in the cluster. At the end of the process the K clustering agents will collectively hold an initial cluster configuration. Note that, in common with standard K-means clustering, the "goodness" of this initial configuration is very much dependent on the nature of the first K records selected to define the initial clusters.

5.2 Biding Phase founded on the KNN Spawning Strategy

The biding phase founded on the KNN spawning strategy, as noted above, commences with a single Clustering Agent (C_i). The operation of this bidding process is presented in Table 2. Note that the process requires a *nearest neighbour threshold*, t, as a parameter. The threshold is used to determine the "nearest neighbour". If the *bidDistance* is less than the threshold, this record in question is allocated to the "closest" cluster (line 6). If there is no "closest" cluster a new Clustering Agent is spawned (line 7). Note that the chosen value of t can significantly affect the number of Clustering Agents that are spawned. The authors proposed a method to identify the most appropriate value for t in [5].

5.3 Refinement (Negotiation) Phase

The refinement process is presented in Table 3. The refinement process is driven using individual cluster cohesion values and an overall cluster configuration

seperation value. We wish to generate a configuration which minimises the cohesion values and maximises the separation value. The refinement phase commences (line 1) with each Clustering Agent determining its own cluster cohesion value and the set of Clustering Agents collectively determining a separation value. Cluster cohesion (the compactness of a cluster) can be determined, by each clustering agent, simply by considering the distance between the members of its cluster. In order to determine the degree of separation (the distinctiveness of each cluster with respect to each other cluster) agents must communicate with one another. For this latter purpose each Cluster Agent defines its cluster in terms of its centroid and these values are then transmitted to all other clustering agents so that an overall separation value can be determined. With respect to the framework described in this paper the Within Group Average Distance (WGAD) and the Between Group Average Distance (BGAD) metrics were adopted to determine cluster cohesion and separation respectively (alternative methods could equally well have been adopted). These metrics are described in Section 6. The cohesion and separation values are sufficient if the cohesion values are below a specified cohesion threshold and the separation value is above a prespecified separation threshold. If this is not the case the cluster associated with each Clustering Agent (c_i) will be be split into two sub-clusters: a *major sub-cluster* and a *minor sub-cluster*. The splitting is achieved by applying a standard K-means algorithm (with K set to 2) to the records held by c_i. The cluster with the smallest number of records is then designated to be the minor sub-cluster and these records are then put up for auction. The auction proceeds in a similar manner to that described for the bidding phase founded on the K-means strategy (see Sub-section 5.1) with the distinction that the WGAD value is used as the *bidDistance*. This process repeats until satisfactory cohesion and separation values are reached. Note (line 8) that on any given iteration, if a record cannot be allocated to a class (because its WGAD value is too high) it is allocated to an *outlier cluster*.

6 Cluster Configuration Metrics

In order for agents to bid for examples some appropriate metric must be adopted. The process for deciding when and how to move a record also requires recourse to some metric, as does deciding when to split and merge clusters. Essentially there are three ways of measuring the "goodness" of a proposed cluster configuration. We can measure intra-cluster cohesion, we can measure inter-cluster separation or we can adopt both metrics. This section presents a review of some of the metrics that may be used to measure the goodness of a cluster configuration and the metrics used in the context of the work described in this paper.

In context of the work described the authors have adopted the Within Group Average Distance (WGAD) [23] and the Between Group Average Distance (BGAD) metrics:

$$WGAD = \frac{\sum_{i=1}^{i=|C|} dist(x_i, c)}{|C|} \quad . \tag{1}$$

Table 3. Algorithm for refinement phase

Algorithm Phase II: Refinement

Input: a cluster
Output: an improved clustering result

1. For all clusters calculate cluster cohesion and separation values
2. DO WHILE there exists cluster cohesion value > cohesion threshold
   or cluster separation value < separation threshold
3. $\forall c_i \in C$ $(i = 1$ to$K)$
3. Split a cluster c_i into two sub-clusters, c_{major} and c_{minor} using K-means
4. $\forall d \in c_{minor}$
5. $\forall c_j \in C$ $(j = 1$ to K and $j \neq i)$
6. $bidDistance = WGAD_j$ (see Section 6)
7. IF $\exists c_j \in C$ such that $bidDistance < cohesion\ threshold$, allocate d to c_j so as to minimise $bidDistance$
8. ELSE Allocate d to "outlier cluster"
9. IF no "successful" bids end loop

$$BGAD = \sum_{i=1}^{i=K} dist(c_i, c) \ . \tag{2}$$

where $|C|$ is the number of objects in a cluster (i.e. the size of a cluster), c is the cluster centroid, and $dist(x_i, c)$ is the distance between object x_i and the cluster centroid. K is the number of clusters.

The lower the WGAD value the greater the cohesiveness of the cluster, whereas the higher the BGAD value the greater the separation of the clusters from one another. We wish to minimise the WGAD and maximise the BGAD to achieve a best cluster configuration. The target WGAD value, the cohesion threshold, is the average WGAD across the identified set of clusters multiplied by a factor p $(0 < p < 1.0)$. The target BGAD value, the separation threshold, is the BGAD value for the initial cluster configuration multiplied by a factor q $(1.0 < q < 2.0)$. Our experiments indicate that values of $p = 0.8$ and $q = 1.2$ tend to produce good results.

7 Evaluation

To evaluate our approach we experimented with a selection of data sets taken from the UCI machine learning repository [11]. We compared the operation of our MADM approach with the well known K-means [18] and KNN [7] clustering algorithms. In each case we recorded the accuracy of the clustering operation, with respect to the known (*ground truth*) clustering. For the K-means algorithm the number of desired clusters must be pre-specified in order to spawn an appropriate number of clustering agents. For this purpose we have used the number

Table 4. Comparison of the result accuracy provided by K-means task distribution before and after cluster configuration improvement.

No.	Data Set	Num Classes	Accuracy Phase I	Accuracy Phase II	Cohesion threshold	Seperation threshold
1	**Iris**	3	0.89	0.97	1.41	2.03
2	**Zoo**	7	0.76	0.78	1.94	1.95
3	**Wine**	3	0.68	0.70	204.95	296.73
4	**Heart**	2	0.55	0.59	33.87	106.28
5	**Ecoli**	8	0.76	0.80	0.42	0.54
6	Blood Transfusion	2	0.76	0.76	794.16	4582.29
7	**Pima Indians**	2	0.65	0.66	65.08	290.60
8	**Breast cancer**	2	0.79	0.85	283.16	1729.93

of classes given in the UCI repository. The t parameter used for KNN was selected, for evaluation purposes, according to the nature of the input so as to be compatible with the specified number of clusters. The results using K-means and KNN are reported in Tables 4 and 5 respectively. The columns in the tables describe: the number of identified clusters (classes), the accuracy after the initial Phase I clustering (i.e. the accuracy that would be achieved using K-means or KNN without any further negotiation), the accuracy after refinement (Phase II), and the calculated cohesion and separation thresholds. Accuracy values are calculated as follow:

$$Accuracy = \frac{\sum_{i=1}^{i=K} C_i}{m} \tag{3}$$

Where K is the number of clusters, m is the number of records and C_i is the size (in terms of the number of records) of the majority class for cluster i. Note that the ground truth is only used here to evaluate the outcomes.

Table 5. Comparison of the result accuracy provided by KNN task distribution before and after cluster configuration improvement.

No.	Data Set	Num Classes	t	Accuracy Phase I	Accuracy Phase II	Cohesion threshold	Seperation threshold
1	**Iris**	4	0.99	0.84	0.87	1.90	1.93
2	Zoo	7	2.24	0.82	0.73	2.42	2.55
3	Wine	3	164.31	0.65	0.65	281.49	282.98
4	**Heart**	2	115.29	0.55	0.60	21.96	62.51
5	**Ecoli**	7	0.42	0.64	0.67	0.38	0.26
6	Blood Transfusion	2	3500.83	0.76	0.76	1222.54	3090.16
7	Pima Indians	2	149.08	0.65	0.65	133.77	152.08
8	**Breast cancer**	2	826.75	0.75	0.79	205.51	847.98

From the tables it can be seen that in the majority of cases (shown in bold font) agent negotiation serves to enhance the initial clustering, with the K-means approach tending to outperform the KNN approach. Interestingly, in the

case of the Zoo data set when using the KNN approach, negotiation had an adverse effect; the research team conjecture that this is something to do with the large number of classes (and hence the large amount of "splitting") featured in this data set. In the remaining cases the situation remained unchanged, either because a best configuration had been identified immediately, or because the WGAD and BAGD values could not be reduced). It should also be noted that the number of class values given in Table 4 (column three) are the ground truth values; the KNN approach does not always produce the correct number of classes and hence this is why the accuracy values are not always as good as in the case of the K-means approach.

8 Conclusion

A MADM framework to achieve multi-agent based clustering has been described. A particular feature of the framework is that it enables agents to negotiate so as to improve on an initial clustering. The framework can be used in a number of ways. Two approaches were considered: K-means and KNN. Evaluation using a number of datasets taken from the UCI repository indicates that in most cases (and especially when using the K-means approach) a better clustering configuration can be obtained as a result of the negotiation process.

References

1. Agogino, A., Tumer, K.: Efficient agent-based cluster ensembles. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. pp. 1079–1086. AAMAS '06, ACM, New York, NY, USA (2006)
2. Bailey, S., Grossman, R., Sivakumar, H., Turinsky, A.: Papyrus: A system for data mining over local and wide area clusters and super-clusters. IEEE Supercomputing (1999)
3. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: JADE: a java agent development framework. In: Bordini, R.H. (ed.) Multi-agent programming : languages, platforms, and applications, p. 295. New York : Springer (2005)
4. Cao, L., Gorodetsky, V., Mitkas, P.A.: Guest editors' introduction: Agents and data mining. IEEE Intelligent Systems 24(3), 14–15 (2009)
5. Chaimontree, S., Atkinson, K., Coenen, F.: Best clustering configuration metrics: Towards multiagent based clustering. In: Proc 6th Int. Conf. Advanced Data Mining and Applications (ADMA'10). pp. 48–59. Springer LNAI (2010)
6. Coenen, F.P., Leng, P., Ahmed, S.: T-trees, vertical partitioning and distributed association rule mining. pp. 513–516 (2003)
7. Dasarathy, B.V.: Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, Las Alamitos, California (1991)
8. Dasilva, J., Giannella, C., Bhargava, R., Kargupta, H., Klusch, M.: Distributed data mining and agents. Engineering Applications of Artificial Intelligence 18(7), 791–807 (2005)
9. FIPA: Communicative Act Library Specification. Tech. Rep. XC00037H, Foundation for Intelligent Physical Agents (2001), available from the website *http://www.fipa.org*.

10. Forman, G., Zhang, B.: Distributed data clustering can be efficient and exact. ACM SIGKDD Explorations Newsletter 2, 34–38 (2000)
11. Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml
12. Giannella, C., Bhargava, R., Kargupta, H.: Multi-agent systems and distributed data mining. vol. 3191, pp. 1–15 (2004)
13. Kargupta, H., Chan, P. (eds.): Advances in Distributed and Parallel Knowledge Discovery. MIT Press, Cambridge, MA, USA (2000)
14. Kargupta, H., Hamzaoglu, I., Stafford, B.: Scalable, distributed data mining using an agent based architecture. In: Proceedings the Third International Conference on the Knowledge Discovery and Data Mining. pp. 211–214. AAAI Press (1997)
15. Kiselev, I., Alhajj, R.: A self-organizing multi-agent system for online unsupervised learning in complex dynamic environments. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence. pp. 1808–1809. AAAI Press (2008)
16. Klusch, M., Lodi, S., Moro, G.: Agent-based distributed data mining: The KDEC scheme. In: Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science). vol. 2586, pp. 104–122 (2003)
17. Klusch, M., Lodi, S., Moro, G.: The role of agents in distributed data mining: Issues and benefits. In: IAT '03: Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology. p. 211. IEEE Computer Society, Washington, DC, USA (2003)
18. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. pp. 281–297 (1967)
19. McBurney, P., Parsons, S., Wooldridge, M.: Desiderata for agent argumentation protocols. In: Castelfranchi, C., Johnson, W.L. (eds.) Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002). pp. 402–409. ACM Press: New York, USA, Bologna, Italy (2002)
20. Moemeng, C., Gorodetsky, V., Zuo, Z., Yang, Y., Zhang, C.: Data Mining and Multi-agent Integration, chap. Agent-Based Distributed Data Mining: A Survey, pp. 47–58 (2009)
21. Park, B.H., Kargupta, H.: Distributed data mining: Algorithms, Systems, and Applications. In: Data Mining Handbook. pp. 341–358. IEA (2002)
22. Provost, F.: Distributed data mining: Scaling up and beyond. In: In Advances in Distributed and Parallel Knowledge Discovery. pp. 3–27. MIT Press (1999)
23. Rao, M.: Clustering analysis and mathematical programming. Journal of the American statistical association 66(345), 622–626 (1971)
24. Reed, J.W., Potok, T.E., Patton, R.M.: A multi-agent system for distributed cluster analysis. In: Proceedings of Third International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'04) W16L Workshop - 26th International Conference on Software Engineering. pp. 152–155. IEE, Edinburgh,Scotland, UK (2004)
25. Younis, O., Fahmy, S.: Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. vol. 1, pp. 629–640 (2004)
26. Zaki, M.J., Ho, C.T. (eds.): Large-Scale Parallel Data Mining, vol. 1759. Springer Verlag (2000)
27. Zaki, M.J., Pan, Y.: Introduction: Recent developments in parallel and distributed data mining. Distributed Parallel Databases 11, 123–127 (2002)

Data Cloud for Distributed Data Mining via Pipelined MapReduce

Wu Zhiang, Cao Jie, Fang Changjian

Jiangsu Provincial Key Laboratory of E-business

Nanjing University of Finance and Economics, Nanjing, P.R. China

e-mail: zawuster@gmail.com, caojie690929@163.com, jselab1999@163.com

Abstract—Distributed data mining (DDM) which often applies autonomous agents is a process to extract globally interesting associations, classifiers, clusters, and other patterns from distributed data. As datasets double in size every year, moving the data repeatedly to distant CPUs brings about high communication cost. In this paper, data cloud is utilized to implement DDM in order to transform moving the data into moving computation. MapReduce is a popular programming model for implementing data-centric distributed computing. Firstly, a kind of cloud system architecture for DDM is proposed. Secondly, a modified MapReduce framework called pipelined MapReduce is presented. We select Apriori as a case study and discuss its implementation within MapReduce framework. Several experiments are conducted at last. Experimental results show that with moderate number of map tasks, the execution time of DDM algorithms (i.e., Apriori) can be reduced remarkably. Performance comparison between traditional and our pipelined MapReduce has shown that the map task and reduce task in our pipelined MapReduce can run in a parallel manner, and our pipelined MapReduce greatly decreases the execution time of DDM algorithm. Data cloud is suitable for a multitude of DDM algorithms and can provide significant speedups.

Keywords: distributed data mining (DDM); Cloud Computing; MapReduce; Apriori; Hadoop

1. Introduction

The last decade has witnessed the successful development of agents and data mining techniques which have been applied to a variety of domains — marketing, weather forecasting, medical diagnosis, and national security [1]. As data mining become more pervasive, the amount of data is increasing larger. The great amount of data is often partitioned into many parts and distributed in many sites. Distributed data mining (DDM) is a process to extract globally interesting associations,

classifiers, clusters, and other patterns from distributed data, where data can be partitioned into many parts either vertically or horizontally [2,3].

Agents are scatted to many sites for handling distributed problem-solving, cooperation and coordination. A high performance DDM system is designed to control agents and exploit the synergy of agents. Traditional DDM system has been designed to take advantage of powerful, but shared pools of CPUs. Generally speaking, data is scattered to the processors, the computation is performed using a message passing, the results are gathered, and the process is repeated by moving new data to the CPUs [4]. As CPU cycles become cheaper and data sets double in size every year, the main challenge for efficient scaling of applications is the location of the data relative to the available computational resources - moving the data repeatedly to distant CPUs is becoming the bottleneck [5].

On the basis of cluster computing, P2P, Grid computing and Web 2.0, cloud computing rapidly emerges as a hot issue in both industrial and academic circles [6]. Cloud computing has been adhering to the belief that moving computation is cheaper than moving data since its birth. Therefore, cloud computing is suitable for solving computation-intensive and data-intensive problems in DDM. MapReduce has emerged as an effective method to implement the *data-centric* belief held by cloud computing. MapReduce has been applied to a multitude of domains for data processing, such as machine learning, satellite data processing, PageRank, scientific data analysis, etc. Since traditional MapReduce stores all middle results in file system and most of DDM applications produce a large amount of middle results, preserving storage for temporary files is extremely expensive and inefficient. Moreover, in traditional MapReduce framework, the reduce task does not start running until all map tasks are finished. This sequential execution manner between map task and reduce task increases the job completion time.

Motivated by the above remarks, this paper proposes a kind of data cloud system architecture for DDM. We make an improvement in the traditional MapReduce framework, namely, pipelined MapReduce framework. The dataflow and the fault tolerance strategy of pipelined MapReduce are addressed in detail. We then discuss the implementation of Apriori algorithm, which is a well-known algorithm for tackling this problem, in MapReduce framework. At last, experimental evaluation of our work is presented.

2. Related Work

In 2007 IBM and Google first proposed cloud computing. Currently, providers such as Amazon, Google, Salesforce, IBM, Microsoft and Sun Microsystems have begun to establish new data centers for hosting cloud computing applications in various locations around the world to provide redundancy and ensure reliability in case of site failures [7]. Data cloud, which is proposed by R. Grossman and Y. Gu in 2008, refers to a class of cloud systems that provide resources and/or data services over the Internet [4].

The Google's MapReduce programming model, which serves for processing large data sets in a massively parallel manner, is a representative technique in cloud computing [8]. Hadoop developed by the Apache Software Foundation is an open source MapReduce framework [9]. The key components of Hadoop are HDFS

(Hadoop Distributed File System) and MapReduce framework. HDFS is a distributed file system designed to run on hardware, and it also manages all files scattered in nodes and provides high throughput of data access. MapReduce provides a programming model for processing large scale datasets in distributed manner. Jobs submitted to Hadoop consist of a map function and a reduce function. Hadoop breaks each job into multiple tasks. Firstly, map tasks process each block of input (typically 64MB) and produce intermediate results, which are key-value pairs. These are saved to disk. Next, reduce tasks fetch the list of intermediate results associated with each key and run it through the reduce function, which produces output. Hadoop is selected as core middleware in data cloud proposed in this paper.

"Market-Basket Analysis" problem, a classic DDM application, is taken as a case study. The supermarket owners may be interested in finding associations among its items purchased together at the check-stand [10]. An example association rule could be that, "90% of customers buying product A also buy product B". Apriori is a seminal algorithm for finding frequent itemsets using candidate generation [11]. It is characterized as a level-wise complete search algorithm using anti-monotony of itemsets, "if an itemset is not frequent, any of its superset is never frequent".

Since Apriori is time consuming and the transaction database is currently distributed in many sites, it is necessary to apply parallel and distributed methods to Apriori. However, the major difficulty lies in that computing support of an itemset should scan the whole database, but each node only stores a split of the whole database. To tackle this difficulty, two kinds of methods have been presented. The first method takes pre-treatment on database to decompose the database into several fragments, and then each node could run totally independently on each fragment of database to compute frequent itemsets using a classical sequential algorithm of Apriori. At last, final results are obtained from these nodes. Although V. Fiolet has suggested three fragmentation methods [10], these methods are time consuming and none of them can be applied to all databases. Furthermore, the preprocessing on database leads to restore data, and data should be moved among clusters (or PC servers), which violates the essence of cloud computing—moving computation is cheaper than moving data. The second method is called *CountDistribution* which is incorporated in this paper. In cloud environment, assuming that the database is initially distributed in a horizontal split, namely non-overlapping subsets of records are randomly stored in clusters (or PC servers). Each node can thus independently get partial supports of the candidates from its local database fragmentation. Reducer then executes reduce function to compute the sum of supports with the same key. Note that only the partial counts need to be communicated, rather than the records of the database. Since the partial counts are represented as <*key, value*> pairs in MapReduce, this method can minimize communication.

3. Data Cloud System Architecture for DDM

Based on campus grid environments we have designed and implemented layered data cloud system architecture as figure 1 depicted. Physical cloud resources along with core middleware form the basis of the system. The user-level middleware aims to provide PaaS (platform as a service) capabilities. The top layer focuses on application services by making use of services provided by the lower layer services. Emerging DDM applications such as social security, enterprise, stock markets and scientific workflows can operate at the highest layer of the architecture. The representative data mining algorithms used by DDM applications such as Apriori, PageRank, kNN and k-Means can be implemented in data cloud system, and some of them even can be implemented in MapReduce framework to improve their performance.

• System level: Physical resources integrated by data cloud system consist of two clusters each with 32 blade servers and several PC servers. The total number of CPU reaches 140.

• Core middleware: This layer is comprised of two sub-layers: VM (virtual machine) and Hadoop. VM management and deployment transparently virtualizes these servers and shares their capacity among virtual instances of servers. These VMs are isolated from each other, which aid in achieving fault tolerant behavior and isolated security context. On the top of VMs, Hadoop framework is deployed, on which Java programs based on MapReduce model can be executed. All data are stored on HDFS. The master node called *JobTracker* is the point of interaction where the user submits jobs, along with location of the input data on the HDFS. The *JobTracker* assigns and distributes the map and reduce tasks to the *TaskTrackers* which assumes the role of worker nodes. The *TaskTracker* performs the task and updates the status to the *JobTracker*. In the MapReduce framework of this architecture, pipeline is added between Mapper and Reducer**.** The new MapReduce framework is called Pipelined MapReduce framework which will be discussed in the next section.

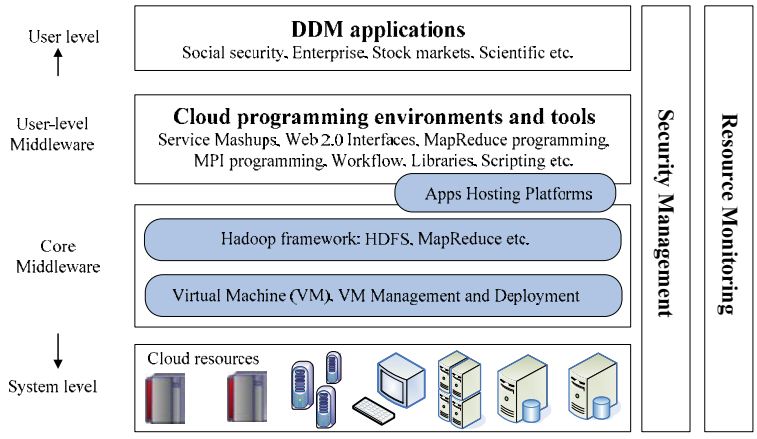


Figure 1. Data cloud system architecture for DDM

• User-level middleware: This layer provides Web 2.0 programming paradigms such as JavaScript with AJAX, Ruby with Ruby on Rail, and PHP etc. Users can utilize Web APIs to create novel browser-based applications. This layer also provides the programming environments and composition tools that facilitate the creation, deployment, and execution of applications in clouds.
• Security management: This module provides authentication and permission control for cloud users. Single sign-on is adopted by cloud system architecture proposed in this paper. Cloud user obtains long-term certificate from Certificate Authority (CA), and user can encrypt this long-term certificate to generate temporary proxy certificate which is used to identify user by data cloud. The deadline of proxy certificate is often short (i.e. 12 hours in our data cloud) to decrease the harm created by various network attacks.
• Resource monitoring: This module monitors all resources in data cloud including clusters, servers, software and jobs. Chukwa, a component of Hadoop, is an open source data collection system for monitoring large distributed systems. Chukwa is built on top of the HDFS and MapReduce framework, and it also includes a flexible and powerful toolkit for displaying, monitoring and analyzing results to make the best use of the collected data.

4. Pipelined MapReduce Framework

In the data cloud system proposed in this paper, we make an improvement in the traditional MapReduce framework, namely, Pipelined MapReduce Framework. In traditional MapReduce framework, the output of each Mapper is managed by the *OutputCollector* instance and stored in an in-memory buffer. The *OutputCollector* is also responsible for spilling this buffer to disk (i.e., HDFS in Hadoop) when the output reaches capacity of memory. The execution of a reduce task includes three phases: *shuffle* phase, *sort* phase and *reduce* phase. The *shuffle* phase fetches intermediate results from each Mapper. *However, a reducer cannot fetch the output of a mapper until JobTracker informs that the mapper is finished.* The output produced by Reducers may be required by the next map step, which is also written to HDFS. There are at least two disadvantages within the traditional MapReduce framework.

(1) HDFS should maintain enough storage for temporary files. Since each file has three copies in HDFS in default manner and most of DDM applications will produce a large amount of middle results, preserving storage for temporary files is extremely expensive and inefficient.

(2) Because Mapper and Reducer execute in a serial manner, and both Mappers and Reducers should spend plenty of time in reading middle data from HDFS, the execution speed of the traditional MapReduce framework is very slow.

To solve these above-mentioned disadvantages, pipeline is added between Mapper and Reducer. Middle data is stored in pipeline files and only final results are written to HDFS. Moreover, when the Mapper is executing, it simultaneously pushes middle data to Reducer via pipeline. Reducers then pull the middle data

synchronously. Pipelined MapReduce Framework makes Mappers and Reducers execute in a parallel manner and also enhance the robustness of the fault tolerance.

A. The dataflow of the Pipelined MapReduce framework

Figure 2 compares the dataflow between traditional and pipelined MapReduce framework. The dataflow on the right depicts the approach utilized by the pipelined MapReduce framework. Each Mapper obtains its input data from HDFS, and when the Mapper is executing, it simultaneously pushes middle data to pipeline. It is noted that when the intermediate data exceeds the memory size, the intermediate data should also be written to HDFS. Each Reducer synchronously pulls data from pipeline and starts running. The middle data produced by the Reducer, which may be required by the next map step, is also pushed to pipeline. Final result produced by the Reducer is written to HDFS.

The precondition that reducers and mappers can run concurrently is that the reduce function of the application is incrementally computable. The applications exist commonly, such as sorting, graph algorithms, Bayes classification, TF-IDF (Term Frequency – Inverse Document Frequency), Apriori, and so on.

The pipeline between Mapper and Reducer is implemented through utilizing TCP socket. Each Reducer contacts every Mapper upon initiation of the job, and opens a socket which will be used to pipeline the output of the map function. As each map output is produced, the Mapper determines which partition the record should be sent to, and immediately sends it via the appropriate socket. A Reducer accepts the pipelined data that it has received from each Mapper and then stores it an in-memory buffer. The Reducer may start running on the basis of these partial data. Once the Reducer learns that all Mappers have completed, it continues to perform the remaining task and writes the output to pipeline or to HDFS.

One practical problem in the above-mentioned method is that when the number of mappers becomes large, each reducer should maintain a large number of TCP connections. To reduce the number of concurrent TCP connections, we restrict the number of connections with mappers at once. The reducer obtains data from the remaining map tasks in the traditional Hadoop manner.

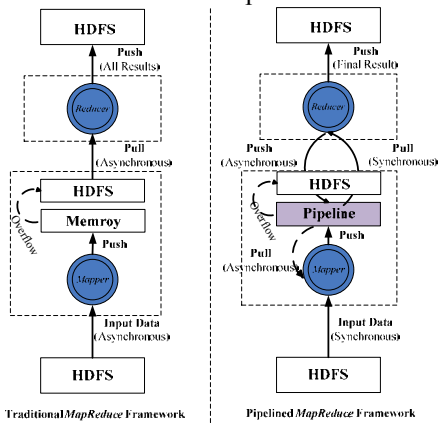


Figure 2. The dataflow comparison between traditional and pipelined MapReduce framework

B. The fault tolerance strategy of the Pipelined MapReduce framework

It is known that the role of cluster node is divided into *master* and *worker*. Both mappers and reducers are *worker* nodes. Although Hadoop could handle master failure, the case unlikely exists [12]. The master detects worker failure via periodic heartbeats. New fault tolerance strategy is implemented in the pipelined MapReduce framework. The reducer treats the output of a pipelined map task as "tentative" until the *JobTracker* informs the reducer that the mapper has committed successfully. The reducer merges together spill files generated by the same uncommitted mapper. Log is added to reducer to record which mapper produced each pipelined spill file. Thus, if a mapper fails, each reducer can ignore any tentative spill file produced by the failed map attempt. The *JobTracker* will then restart a new mapper.

If a Reducer fails and a new copy of the task is started, all the input data that was sent to the failed reducer must be sent to the new reducer. To prevent mappers from discarding their output after sending it to pipeline, mappers should retain their output data until the entire job is completed successfully. This allows the output of mappers to be reproduced if any reducer fails.

5. Case Study

Finding frequent itemsets from a transaction database and deriving association rules is called "Market-Basket Analysis" problem which is one of the most popular data mining problems. This section takes "Market-Basket Analysis" problem as a case study. Although "Market-Basket Analysis" problem has been described in many papers, to make it clear and easy for understanding, we briefly describe it.

Given a transaction database *D*, the number of its records is denoted as |D|. We assume there are *n* distinct items in *D*, denoted as $I=\{i_1,i_2,\ldots,i_n\}$.

Definition 1 support: The support of an itemset X is the percentage of records in the database *D*, which contains this itemset X. The support measures how interesting the itemset is, that is, its frequency in the database. The support of the itemset X can be calculated by equation (1).

$$\text{support}(X)=\frac{D(X)}{|D|} \tag{1}$$

where D(X) is the number of records containing itemset X in database *D*, and |D| is the total number of records of database *D*. The "Market-Basket Analysis" problem is to find out all association rules whose support is greater than the given thresholds. The support threshold is defined by user.

A. Apriori algorithm within the framework of MapReduce

A well-known algorithm for the computation of frequent itemsets is the Apriori algorithm which is used as follows:

- to compute the supports of items, and then to identify frequent items (frequent 1-itemsets)
- to generate candidate 2-itemsets, to count their supports, and then to identify frequent 2-itemsets

- to generate candidate 3-itemsets, to count their supports, and then to identify frequent 3-itemsets, and so on…

To compact the search space of Apriori, the guiding principle "every subset of a frequent itemset has to be frequent" is utilized.

As section 2 described, this paper incorporates *CountDistribution* method to implement Apriori within the framework of MapReduce. In cloud environment, assuming that the database is initially distributed in a horizontal split, namely non-overlapping subsets of records are randomly stored in clusters (or PC servers). These nodes which store the splits of the database execute map function and become *Mappers*. A program implementing the map function of Apriori is sent to all mappers. Each mapper can thus independently get partial supports of the candidates from its local database fragmentation. All partial results from mappers are collected to one or several nodes called *Reducer(s)*. And then, Reducer executes reduce function to compute the sum of supports with the same key and global frequent itemsets F_{k-1} are obtained. Note that only the partial counts need to be communicated, rather than the records of the database. Since the partial counts are represented as <key, value> pairs in MapReduce, this method can minimize communication. Figure 3 depicts the MapReduce framework of Apriori.

Since Apriori algorithm utilizes iteration to obtain all frequent itemsets, it needs to scan database at most $n+1$ times when the maximum size of frequent itemsets is set at F_n. Therefore, *mappers* should be started at most $n+1$ times. Within the framework of MapReduce, though reduce operation also can be run in parallel manner, *reducer* of Apriori is run in a single node because the reduce operation of Apriori only needs to carry out simple count operation. The pseudocode of map and reduce function executed by mappers and reducer respectively for the Apriori is as follows.

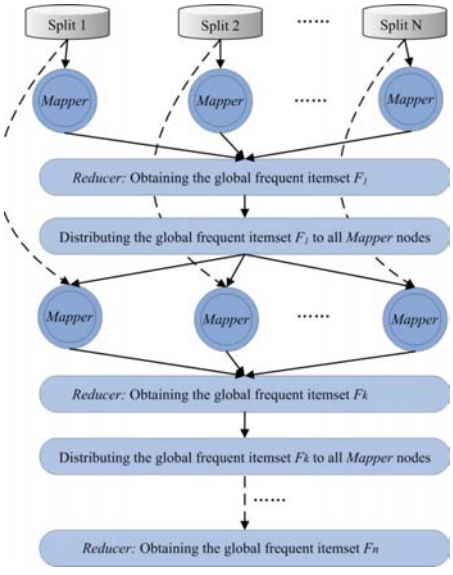


Figure 3. MapReduce framework of Apriori

The map function for the Apriori
Input: the split of database and the last globle frequent itemset F_{k-1}
Output: <key, value> pairs, where key is itemset and value is its corresponding support
1: $Ck \leftarrow$ AprioriGen ($Fk-1$)
2: for each element c in Ck
3: scan the split of database and count support for c
4: generate pair <c,$c.support$>
5: end for
6: return $\cup$ <c,$c.support$>

The reduce function for the Apriori
Input: < c, $c.support$ > pairs generated by *Mapper* nodes
Output: the globle frequent itemset F_k
1: for all <c, $c.support$> pairs with same c
2: compute the sum of their support
3: denote the sum as <c, $sum\_support$>
4: end for
5: $Fk \leftarrow$ {all <c, $sum\_support$> pairs | $c.sum\_support \geqslant min\_sup$}
6: return $\cup Fk$

In the Map function, AprioriGen function generates new candidates denoted as C_k based on the last global frequent itemset F_{k-1}. Each mapper scans its split of database and counts support for all elements in C_k. In the Reduce function, all <key,value> pairs from all mappers are grouped by *key* and supports of the pairs with the same key are added. The pairs whose sum of support is greater than threshold are selected into the current global frequent itemset F_k.

This case study indicates that the MapReduce framework provided by data cloud system architecture exhibits good scalability to other DDM algorithms. We needn't take any pre-treatment on database and needn't move data among clusters. Only map function and reduce function of DDM algorithms need to be provided. Data cloud can automatically move programs containing map function to a multitude of nodes storing the input data.

B. Performance evaluation

We have built a data cloud system based on the architecture shown in figure 1. Machines integrated by this data cloud system consist of two clusters each with 32 blade servers in SEUGrid (Southeast University Grid) and an army of PC servers in our laboratory. Two VMs are created in each PC servers with 2.03GH and 1GB of RAM. Two clusters in SEUGrid utilize Red Hat Enterprise 4.0 and all VMs in PC servers utilize Fedore Core 11, and on the top of Linux, Hadoop 0.20.1 is installed and configured as core middleware.

The transaction database contains two attributes: one is the transaction identifier and the other is the list of items.

In the first experiment, we investigate the effect of the number of *mappers* on the execution time to obtain all frequent itemsets utilizing Apriori. A transaction database is created with 100,000 records and 500 kinds of items. The transaction database is split on average according to the number of *Mapper* nodes. In other words, equal number of records is stored in all mapper nodes. Figure 4 shows the effect of number of *mappers* on execution time. It indicates that with the increase of the number of *mappers*, the execution time decreases. At the beginning, the ex-

ecution time decreases dramatically with the increase of the number of *mappers.* When the number of *mappers* reaches a threshold (i.e. 16 in figure 4), the execution time varies moderately or even increases slightly (i.e. 24, 28 and 32 in figure 4). The reason for the above-mentioned phenomenon is that with the increase of *mappers*, the number of records stored in each *mapper* node decreases. When the number of records reduce to a threshold, the time of scanning the database is hard to further decline. Therefore, the time of *Reducer* and communications is rising, thus playing the major role in the whole execution time.

The second experiment compares pipelined MapReduce with traditional MapReduce. We analyze map progress and reduce progress in an iteration of Apriori (i.e. computing F_k based on F_{k-1}). Hadoop provides support for monitoring the progress of task executions. As each task executes, it is assigned a *progress score* in the range [0,1], based on how much of its input that the task has consumed. There are 80,000 records split on 8 mappers including 500 kinds of items. The first iteration computing F_1 based on the input from HDFS is selected as a statistical object. Figure 5 describes the progress score of map task and reduce task along with the timelapse. The map tasks in both traditional and pipelined MapReduce are same for their input is read from HDFS. There exists obviously difference between the progress of traditional reduce and the progress of pipelined reduce. Traditional reduce task starts after the map task and it consumes more time than pipelined reduce task. However, pipelined reduce task starts immediately after the map task starts, and they are running in a parallel manner (i.e. from 0s to 10s). These results suggest that pipelined MapReduce framework can substantially reduce the response time of a job.

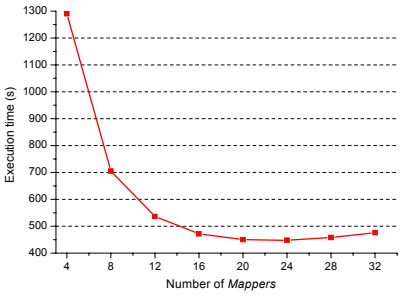


Figure 4. Effect of number of Mappers on execution time

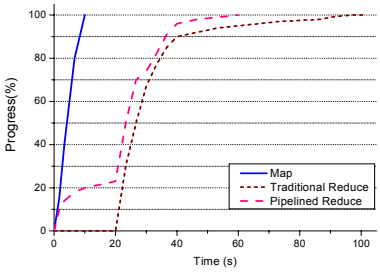


Figure 5. Comparison of map and reduce task completion times between traditional and pipelined MapReduce

In the third experiment, we implement other applications in Hadoop and investigate the speedup as we scale the number of processor cores. The following are brief descriptions of the selected applications:

- WordCount: It counts the frequency of occurrence for each word in a set of files. Mappers process different sections of the input files and return intermediate data that consist of a word (key) and a value of 1 to indicate that the word was found. The reducers add up the values for each word (key).

- Kmeans: It implements the popular Kmeans algorithm to group a set of input data points into clusters [10]. Since Kmeans is iterative, in each iteration, mappers find the distance between each point and each mean and assign the point to the closest cluster. For each point, we emit the cluster ID as the key and the data vector as the value. Reducers gather all points with the same cluster ID, and finds their mean vector.
- Reverse Index: It traverses a set of HTML files, extracts all links, and compiles an index from links to files. Each mapper parses a collection of HTML files. For each link it finds, it outputs an intermediate pair with the link as the key and the file info as the value. The reducer combines all files referencing the same link into a single linked-list.

We evaluate the *speedup* of four algorithms as we scale the number of processor cores used in data cloud. The concept of *speedup* stems from high-performance computing (HPC). *Speedup* is defined as the ratio of the execution time of an algorithm on one node to the execution time of the same algorithm on several nodes. Figure 6 presents the experimental result. Data cloud provides significant speedups for all processor counts and all algorithms. With a large number of cores, the speedup of some applications cannot be improved further (i.e., WordCount, Kmeans, Apriori), where the reason is similar to the analysis in the first experiment. The speedups of Kmeans and Apriori are smaller than other algorithms, because they contain iteration process. Fitting iterative algorithms into the MapReduce framework leads to major overheads compared to sequential code and reduces the overall speedup.

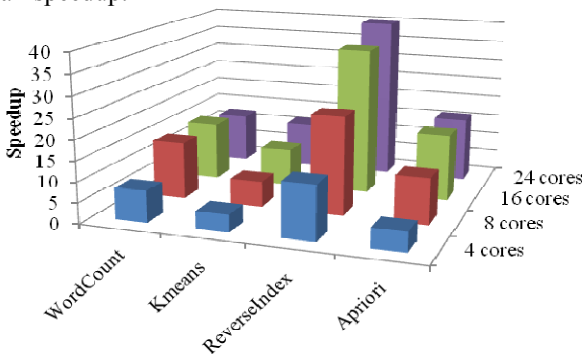


Figure 6. Speedup of four algorithms with increase of processor cores

6. Conclusion

The increasing amount of data has led to concerns regarding the execution efficiency of DDM. The *data-centric* belief held by cloud computing can be utilized to enhance the execution efficiency of DDM. In this paper, we propose a kind of data cloud system architecture for DDM. Pipelined MapReduce framework is presented and implemented in data cloud. We then take Apriori algorithm as a case

study and implement Apriori within the framework of MapReduce. This case study indicates that data cloud system architecture proposed in this paper exhibits good scalability to various DDM algorithms. The fragmentation (or pre-treatment) methods of database needn't to be considered. Users should only provide map function and reduce function of DDM algorithms to data cloud. Data cloud can move computation in terms of the initial data location.

We conduct several experiments in order to evaluate our work. The experimental results indicate that the execution time can be reduced remarkably with moderate number of mappers. Performance comparison between traditional MapReduce and our pipelined MapReduce has shown that: (1) the map task and reduce task in our pipelined MapReduce can run in a parallel manner; (2) our pipelined MapReduce greatly decreases the execution time of DDM algorithm. Data cloud is suitable for a multitude of DDM algorithms and can provide significant speedups.

Acknowledgments This research is supported by National Natural Science Foundation of China under Grants No.71072172, the program for New Century Excellent Talents in university under Grants No.NCET-07-0411, the Natural Science Foundation for key basic research of the Jiangsu Higher Education Institutions of China under Grants No.07KJA52004, Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201and the joint research for Forward Looking Production, Teaching & Research in Jiangsu under Grants No.by20091005.

References

[1] L. Cao, V. Gorodetsky, P.A. Mitkas. Agent Mining: The Synergy of Agents and Data Mining. IEEE Intelligent Systems, Vol. 24, Issue 3, May 2009, pp. 64-72

[2] X. Yi, Y. Zhang. Privacy-preserving naïve Bayes classification on distributed data via semi-trusted mixers. Information Systems, Vol. 34, Issue 3, May 2009, pp. 371-380

[3] L. Cao. Domain-Driven Data Mining: Challenges and Prospects. IEEE Transactions on Knowledge and Data Engineering, Vol. 22, No. 6, pp. 755-769, June 2010

[4] R. Grossman and Y. Gu. Data mining using high performance data clouds: experimental studies using sector and sphere. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 920-927, 2008

[5] A. Szalay, A. Bunn, J. Gray, I. Foster, I. Raicu. The Importance of Data Locality in Distributed Computing Applications. NSF Workflow Workshop 2006

[6] Above the clouds: A Berkeley View of Cloud computing. UCB/EECS-2009-28

[7] Buyya R, Yeo CS, Venugopal S, Broberg J and Brandic I. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6):599–616, 2009

[8] L. Ralf. Google's MapReduce programming model — Revisited. The Journal of Science of Computer Programming. Vol. 70, Issue 1, pp. 1-30, January, 2008

[9] Hadoop. The Apache Software Foundation. http://hadoop.apache.org/core

[10] V. Fiolet and B. Toursel. Distributed Data Mining. Scalable Computing: Practice and Experience. Vol. 6, No. 1, pp. 99-109, 2005

[11] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. Mclachlan, A. Ng, B. Liu, P.S. Yu, Z. Zhou, M. Steinbach, D.J. Hand and D. Steinberg. Top 10 algorithms in data mining. Knowledge and Information Systems. Vol. 14, No. 1, pp. 1-37, 2008

[12] T. White. Hadoop: The Definitive Guide. O' Reilly Publishers, 2010