Expert Systems With Applications 57 (2016) 62–75

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# Applying ant colony hybrid metaheuristics to wrapper verification☆

I. Fernández de Viana*, P.J. Abad, J.L. Álvarez, J.L. Arjona

*Department of Information Technologies, University of Huelva, Campus Universitario de la Rábida, La Rábida, Huelva, Spain*

ARTICLE INFO

*Article history:*
Received 8 September 2015
Revised 24 November 2015
Accepted 13 February 2016
Available online 27 February 2016

*Keywords:*
Wrapper maintenance
Wrapper verification
Enterprise integration
ACO

ABSTRACT

Wrappers are pieces of software used to extract data from websites and structure them for further application processing. Unfortunately, websites are continuously evolving and structural changes happen with no forewarning, which usually results in wrappers working incorrectly. Thus, wrappers maintenance is necessary for detecting whether wrapper is extracting erroneous data. The solution consists of using verification models to detect whether wrapper output is statistically similar to the output produced by the wrapper itself when it was successfully invoked in the past. Current proposals present some weaknesses, as the data used to build these models are supposed to be homogeneous or that the features of this data set can be mapped to an n-dimensional space of independent dimensions when there is a correlation among their features. In this paper, a new verification system based on the Best-Worst Ant System (BWAS) is presented to overcome previous weaknesses. The experimental results show an accuracy improvement of 7.5% over current solutions.



## 1. Introduction

Maintenance is a key challenge in the design of wrappers to extract and structure data from websites (Kushmerick, 1999). Wrappers are used in a lot of real-world scenarios as enterprise information integration, context-aware advertising, database building, business intelligence and competitive intelligence, functional web application testing, opinion mining, or citation databases (Emilio Ferrara & Baumgartner, 2014). Unfortunately, wrappers are not fully resilient to unexpected changes, as websites are dynamic entities that usually undergo changes in search forms, navigational models or the way information is rendered on the screen (de Viana, Hernandez, Jiménez, Rivero, & Sleiman, 2010a). The immediate consequence is that previously defined wrappers are no longer able to successfully extract data, which results in a system that manages corrupted or lost data (Lerman, Minton, & Knoblock, 2003).

To illustrate the problem, Fig. 1 presents a real scenario on Enterprise Information Integration (EII). A metasearch engine (Fig. 1a) uses two wrappers to extract and integrate the information residing in two different websites (Fig. 1b and Fig. 1c) to provide business value-added. These wrappers are been built using induction techniques to learn from a collection of manually annotated web pages as training data (Chidlovskii, Roustant, & Brette, 2006; Ferrara & Baumgartner, 2011; Lerman & Knoblock, 2009; Sarawagi, 2008; Sleiman & Corchuelo, 2012, 2013; Tsourakakis & Paliourast, 2009; Yang et al., 2009). Then, imagine that the designers of a web information provider decide to change the order of paper title and journal name presentation. This change would lead the metasearch engine to render inappropriate data.

Wrapper maintenance (Lerman et al., 2003) consists of two phases: first, wrapper verification, in which a new wrapper output is compared to that produced by the wrapper itself when successfully invoked in the past, and the similarity of new and past outputs is evaluated; and, second, wrapper reconstruction, in which the wrapper is repaired to work on changed pages. Our research focuses on the wrapper verification phase. Several authors have worked on this phase (Fazzinga, Flesca, & Tagarelli, 2011; Kushmerick, 2000b; Lerman et al., 2003; McCann et al., 2005; Tsourakakis & Paliourast, 2009) but their works present some weaknesses as the data used to build these models are supposed to be homogeneous or that the features of this data set can be mapped to an n-dimensional space of independent dimensions when there is a correlation among their features. Some recently wrapper induction approaches based on redundancy (Bronzi, Crescenzi, Merialdo, & Papotti, 2013; Gulhane, Rastogi, Sengamedu, & Tengli, 2010; Shui-Lung Chuang, 2007) or supervised entity extraction (Dalvi, Kumar, & Soliman, 2011; Furche et al., 2014) include verification methods that are specific of the wrapper induction process. Therefore,

☆ Supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E and TIN2010-21744).

* Corresponding author. Tel.: +886 2 27303749; fax: +886 2 27303733.
*E-mail addresses:* i.fviana@dti.uhu.es (I. Fernández de Viana), pedro.abad@dti.uhu.es (P.J. Abad), alvarez@dti.uhu.es (J.L. Álvarez), jose.arjona@dti.uhu.es (J.L. Arjona).

http://dx.doi.org/10.1016/j.eswa.2016.02.022
0957-4174/

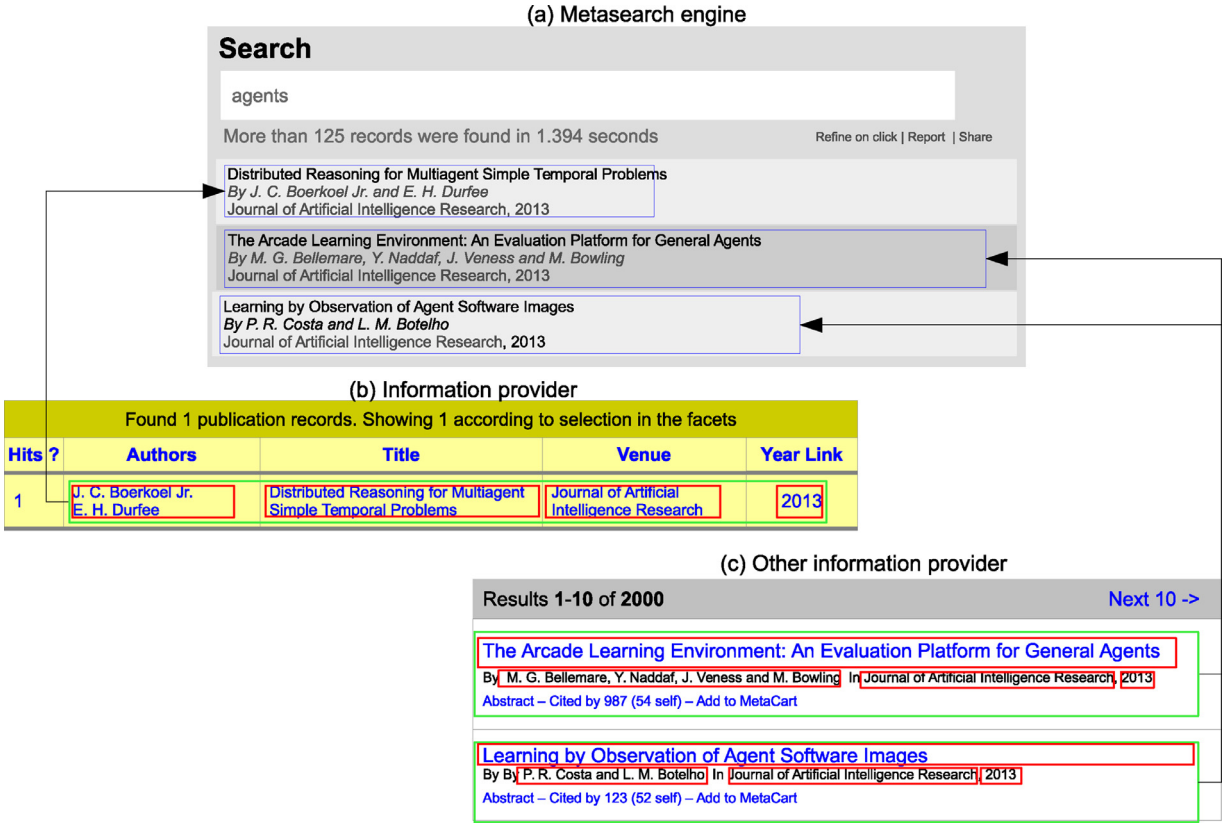


**Fig. 1.** Metasearch engine service, integrating information from other sites.

verification is only valid for those wrappers and are not longer applicable to others.

In this paper, we present a new solution to verifying wrappers. Its novelty lies in the use of Ant Colony Optimization (ACO) metaheuristic (Dorigo & Di Caro, 1999; Dorigo & Stützle, 2004). The term metaheuristic generally refers to optimization algorithms not specifically designed for a specific problem (Liao, Hu, & Tiersch, 2012). Among the different existing ACO algorithms, we use the Best-Worst Ant System (BWAS) by its good performance when solving other problems (Casillas, Cordón, de Viana, & Herrera, 2005; Cordón, de Viana, & Herrera, 2002b, 2002). BWAS is a hybrid algorithm combining ACO and evolutionary metaheuristics. Our experimentation proves that BWAS are well-suited for wrapper verification, overcoming weaknesses and achieving better results than current proposals.

This paper is organized as follows: Section 2 describes the problem of wrapper verification; Section 3 summarizes current wrapper verification proposals and describes their weaknesses. Section 4 briefly introduces ACO and BWAS; Section 5 sets out all aspects related to the application of BWAS algorithm to verify information; Section 6 details the design of the experiments and presents the results obtained. Finally, Section 7 points out some concluding remarks and future work.

## 2. Wrapper verification

Wrappers use a set of extraction rules inferred from an example data set obtained from websites to extract information from web pages. Extraction rules greatly depend on the internal website layout for which they are designed. Therefore, it is necessary to include a new element (the verifier), which is responsible for checking whether wrapper-extracted data are correct. Wrapper verification has been addressed by several authors from different points of view (Kushmerick, 2000b; Lerman & Knoblock, 2009; Lerman et al., 2003; McCann et al., 2005; Tsourakakis & Paliourast, 2009; de Viana, Abad, Álvarez, & Arjona, 2011).

Fig. 2 shows the process of devising and using a verification system. Next two subsections detail each one of these phases. The example, from the websites described in Fig. 1 to extract paper titles, is used to a better understanding of how verifiers are devised and how they work.

### 2.1. Devising verifiers

The process of devising a verifier begins obtaining a wrapper-extracted working set supervised by a human expert to ensure data validity. A working set is given by $WS = \{s_1, s_2, \ldots, s_M\}$, where each $s_i$ is a slot and $M$ is the number of wrapper-extracted slots. Each slot is given by:

$$s_i = (a_i, l_i) \quad with\ i = 1, \ldots, M \tag{1}$$

where $a_i$ is an attribute (a string contained in a web page), $l_i \in \mathcal{L} = \{r_1, r_2, \ldots, r_N\}$ is a label denoting the role of attribute $a_i$, $\mathcal{L}$ is the set of roles, and $N$ is the number of different roles. This way, the wrapper, which extracts information from page in Fig. 1c, adds to the working set the following slots:

$s_5 = (a_5, l_5) = (\text{“}TheArcadeLearningEnvironment : AnEvaluationPlatformforGeneralAgents\text{”}, \text{“}Title\text{”})$

$s_6 = (a_5, l_6) = (\text{“}M.G.Bellemare, Y.Naddaf, J.Veness, M.Bowling\text{”}, \text{“}Autor\text{”})$

$s_7 = (a_7, l_7) = (\text{“}JournalofArtificialIntelligenceResearch\text{”}, \text{“}Venue\text{”})$

$s_8 = (a_8, l_8) = (\text{“}2013\text{”}, \text{“}Year\text{”})$

$s_9 = (a_9, l_9) = (\text{“}LearningbyObservationofAgent$