An On-Line Algorithm for Semantic Forgetting

Heather S. Packer and **Nicholas Gibbins** and **Nicholas R. Jennings**
Intelligence, Agents, Multimedia Group,
School of Electronics and Computer Science,
University of Southampton,
Southampton, SO17 1BJ, UK
{hp07r,nmg,nrj}@ecs.soton.ac.uk

Abstract

Ontologies that evolve through use to support new domain tasks can grow extremely large. Moreover, large ontologies require more resources to use and have slower response times than small ones. To help address this problem, we present an on-line semantic forgetting algorithm that removes ontology fragments containing infrequently used or cheap to relearn concepts. We situate our algorithm in an extension of the widely used RoboCup Rescue platform, which provides simulated tasks to agents. We show that our agents send fewer messages and complete more tasks, and thus achieve a greater degree of success, than other state-of-the-art approaches.

1 Introduction

Evolving an ontology enables it to support tasks that were unforeseen at design time, thus enabling an agent that uses it to adapt to changing requirements. However, evolving ontologies can become large in size, and thus require more resources to host, manage, and use. Moreover, fast response times are crucial for time-critical environments: an ontology that increases in size can diminish response times which can be reduced by removing appropriate concepts from it.

Against this background, we present an on-line algorithm which selects an ontology fragment to remove given the frequency and recency with which its concepts are used and the cost of their acquisition (in terms of time). Specifically, an ontology is reduced in size when it becomes too large to complete a task within a given period of time. Underpinning this is our hypothesis that by removing a fragment the associated costs of using the ontology will be reduced. To evaluate this, we deploy our algorithm within a RoboCup Rescue (RCR) extension, a widely used multi-agent platform for search and rescue simulation [Kitano and Tadokoro, 2001]. The agents (representing the police, fire brigade and ambulances) evolve their ontologies as they encounter unforeseen tasks that require additional knowledge to complete. An agent's ontology must enable the agent to assess and deliberate about its action within a given timeframe.

While we situate our approach using a specific multi-agent system exemplar, our algorithm is a general approach to selecting concepts to forget from an ontology, and can therefore be applied outside of our framework. In AI, this area has been studied under a variety of names such as forgetting and variable elimination [Eiter *et al.*, 2006; Wang *et al.*, 2008]. We provide a general approach for ranking knowledge according to its use and cost, which can be applied to systems that are limited by memory resources to evaluate memory allocation. We also provide a specific approach to select which concepts to remove from an ontology, using the ranking. We note that our forgetting approach does not guarantee an evolving ontology is sound or complete. However, this is not always a requirement, and a 'good enough' answer is appropriate in many cases where a response is required quickly (as described in Section 4). Our approach is agnostic to a specific ontology language, however for simplicity we describe our approach in the context of OWL-Lite ontologies.

Against this background, we advance the state-of-the-art in automatic ontology evolution, with our main contribution, an on-line forgetting algorithm. It rates and weights concepts in an ontology according to their use and acquisition cost. Then, in our evaluation, we compare the performance of our forgetting approach to other state of the art approaches including forgetting single, redundant and subtrees of concepts. Agents using our approach save 99.3% more civilians and 12.4% more of the city, compared with the next best approach.

The rest of the paper is organised as follows. Section 2 presents related work. Sections 3 and 4, introduce our evaluation framework and forgetting algorithm. Sections 5 and 6 detail our evaluation and results. Section 7 concludes.

2 Related Work

The agent community focuses primarily on augmenting an agent's ontology, instead of pruning it. In particular, Bailin and Truszkowski [2002], Afsharchi *et al.* [2006], Wiesman and Roos [2004], and Soh [2002] enable their agents to augment their ontologies with new knowledge, when agents have different domain models representing the same domain. Specifically, Bailin and Truszkowski's approach considers semantically equivalent representations, and Afsharchi *et al.* and Soh focus on the validation of the knowledge to be incorporated into the agent's ontology. Moreover, these approaches augment an agent's ontology with one concept at a time, which increases the overhead cost of retrieving the information. In addition to this work, we have previously

Table 1: An overview of the contents and source of each of the environment ontologies.

Ontology	Domain	Overview	Source	No. of Concepts
EAC Ontology	Fire Brigade	Emergency Action Code (EAC) for dangerous goods	National Chemical Emergency Centre (NCEC): http://the-ncec.com/ncec	1906
HazChem Ontology	Fire Brigade	Hazard Identification Numbers (HIN) for dangerous goods	National Institute for Occupational Safety and Health (NIOSH) www.unece.org/trans/danger/publi/adr/pubdet.htm	1167
Chemical Ontology	Fire Brigade	Chemicals and their EAC and HIN classifications	NCEC: http://the-ncec.com/ncec, and NIOSH www.cdc.gov/niosh/	1800
Vehicle Ontology	Police	Vehicle capabilities, purposes, manufacturers and categorisations	John Dennis Coach Builders: www.johndennisfire.co.uk/	70
HantsFireEngineFleet Ontology	Fire Brigade	Vehicle types, models, manufacturers and registration numbers of the Hampshire fleet of Fire Engines	Hampshire Fire and Rescue Service: www.hantsfire.gov.uk/theservice/sp-and-sr/fleetmanagement.htm	745
Ambulance Ontology	Ambulance	Types of ambulance, their attributes and equipment	American College of Surgeons: www.facs.org/trauma/publications/ambulance.pdf	407
ConstructionVehicles Ontology	Fire Brigade	The attributes and capacity of construction vehicles	Construction Equipment Guide website: www.constructionequipmentguide.com/	114
Triage Ontology	Ambulance	The 5-Category triage system, and the symptoms for each category	The Agency for Healthcare Research and Quality, US department of Health and Human Services: www.ahrq.gov/research/esi/	74
CSI Ontology	Fire Brigade	Chemical Sampling Information (CSI) details chemicals' effects on humans and their organs.	US Department of Labor Occupational Safety and Health Administration: www.osha.gov	3841
Treatment Ontology	Ambulance	Information about burns and broken bones, their symptoms and their treatments	NHS Health Information: www.nhs.uk/chq/pages/Category.aspx?CategoryID=72	481

presented an approach that reduces the cost associated with learning by augmenting a fragment into an ontology [Packer *et al.*, 2010a]. However, while all these approaches allow agents to augment their ontologies, they do not prune them.

However, the Semantic Web community has produced methods that prune ontologies. In particular, an agent could apply the approaches of Eiter *et al.* [2006], Wang *et al.* [2008] or Wang *et al.* [2009] to prune its ontology. All these approaches provide algorithms to remove one concept from an ontology at a time. In contrast to the approach of Wang *et al.* [2008], Eiter *et al.*'s approach requires axioms to first be translated from Description Logic (DL) syntax to rule representations, and translated back to DL syntax, which adds to the resources required by the process, after the expansion of the rules and the removal of a concept. In contrast, [Wang *et al.*, 2008]'s approach can be applied to axioms without need of translation. Eiter *et al.*'s approach has restrictions which limit the use of this technique to OWL-Lite and subspecies of OWL-Lite. These approaches enable an agent to evaluate the knowledge and remove a single concept at a time.

In this work, we chose to use the technique presented by [Wang *et al.*, 2008] to remove concepts from our agent's ontologies because it ensures that there are no inconsistent axioms of an ontology after the removal of a concept. We use this work to remove concepts from an ontology, however our forgetting approach focuses on selecting which concepts to remove, and not on how to remove them from an ontology. This paper builds upon the work presented in our previous paper, [Packer *et al.*, 2010b], with new results.

3 RoboCup OWLRescue Framework

The RoboCup OWLRescue (RCOR) framework extends the RoboCup Rescue (RCR) platform, which models the effects of an earthquake on a virtual city's buildings, civilians and roads [Kitano and Tadokoro, 2001]. In a RCR simulation, buildings may: collapse, possibly with civilians inside; cause road blockages; and ignite. There are three types of specialist agents: ambulance teams recover buried civilians and transfer them to refuges; fire teams extinguish fires; and police force teams clear blocked roads. The goal is to save the lives of as many civilians as possible and to minimise the area of the city which is burnt. The performance of a team is evaluated by analysing the percentage of civilians alive, the state of live civilians, and the average building damage. While RCR is a testbed for the co-ordination of agents, our extension extends the variables associated with each target (civilians, buildings, and blockages) resulting in a set of possible actions an agent can take, where each action has a causal effect.

Our RCOR framework extends buildings to contain (possibly hazardous) chemicals, and extends civilians to have symptoms. Agents have their own ontologies that they can augment, with new information from ontologies in the environment to complete unforeseen tasks (such as extinguishing a fire with a new combination of hazardous chemicals, or treating a civilian with exposure to a rare chemical). *Environment ontologies* describe available resources within the simulation which agents can use, and contain knowledge about vehicles and their ability to deal with fires, building collapses and casualties (see Table 1). These ontologies have been chosen because they are representative of standard industry vocabularies for the domains of interest of RCOR agents. These ontologies encompass the interest areas of the RCOR agents and provide information used in the real world. The RCOR agents can access the environment ontologies by requesting information about concepts and receive fragments representing a desired concept. The agent can then augment its ontology with all the concepts or a subset of concepts depending on its strategy. In order for an agent to retain its core knowledge it uses two ontologies to decide on which actions to take: a Domain Ontology (DO) containing core knowledge; and an Evolving Ontology (EO) which an agent can augment and prune. Maintaining an agent's core knowledge allows agents to maintain their designated speciality:

Fire brigade agents require a different core set of concepts than an ambulance team because of their specialisation, thus, we only remove concepts from an agent's EO. Each agent is allocated a vehicle, which it can exchange if it does not have the necessary equipment for a task.

The RCOR agents can learn about variables encountered while rescuing a target and alternative resources. For example, a police rescue agent can discover an appropriate construction vehicle which can remove a blockage from a road or an ambulance agent can learn about the latest recommendations for treating civilians. It is beneficial for agents to augment their ontology so that they can successfully perform tasks that they could not complete before. In order to simulate real-time the RCR framework allows a team of agents five seconds to perform an action, agents that do not perform an action in this time forfeit their turn. Thus an agent must spend its time efficiently performing actions. In order to do this, our agents maintain a relatively small ontology and send a minimal number of requests for information from the environment ontologies. The next section describes our forgetting approach.

4 The Forgetting Approach

When reasoning in an ontology becomes too slow to use given a specific timeframe, our forgetting approach is invoked. Our approach first evaluates the concepts in the ontology to select which one to remove; second our approach selects a fragment of the concept that is deemed to be the most irrelevant; and third our approach removes the concept so that the ontology remains consistent. These three stages are described in the following sections. We use the following running example.

A fire brigade is tasked with extinguishing a building, that contains particularly toxic chemicals and human exposure results in severe damage to airways. The fire brigade needs to carry additional equipment to treat civilians, and augments its ontology with a fragment containing such equipment. During augmentation the agent learns about intubation equipment, this equipment is usually used by ambulance agents and therefore the fire brigade has only used it once. The agent's response time is waning and in order to improve the response time it decides to reduce the size of its ontology.

4.1 Evaluating Concepts

Once a task agent determines that it needs to forget, it decides which concepts it wants to remove. In order to select concepts to remove our approach considers two factors:

1. How recently and frequently the concept is used to answer queries. This approach aims to reduce the cost of acquiring regularly required concepts so we therefore adopt the Least Recently, Frequently Used value (LRFU) used in [Lee *et al.*, 1997]. This is used in caching scenarios to select concepts to remove from a query agent's ontology. Each time a concept is used; the LRFU increases as do the LRFUs of the concepts which are used to define it.
2. The cost of the original acquisition of the concept. This cost is measured in milliseconds when the concept is augmented. The acquisition value depends on the availability of the concept, and the network bandwidth available to transfer the fragment from another agent.

Both the LRFU and acquisition cost are normalised into a ranking, these two factors are summed (see Equation 1).

$$CFV = LRFU_r + AC_r \quad (1)$$

where CFV is the concept forgetting value which ranks all concepts in an agent's EO, the $LRFU_r$ is the ranking of the LRFU value of a concept, and AC_r is rank for the acquisition value of the concept. A low CFV weighting indicates that the concept has not been used recently, often, and was inexpensive time wise to acquire. A high CFV weighting indicates that the concept has been used recently, frequently and was expensive to acquire. A medium CFV weighting can indicate that the rankings of LRFU and AC is high and is low respectively, or that AC and LRFU is high and low respectively, and as such the likelihood of the concept being forgotten is lower than those with a low CFV weighting. In more detail, the LRFU is calculated for each concept in the agent's EO using Algorithm 1. This algorithm shows how an agent calculates the $LRFU$ value for each of its ontology's concepts, where $concept(EO)$ is a function that holds the set of concepts in an agent's EO, $T = \{\langle t_1, c_u 1\rangle, \ldots, \langle t_n, c_u n\rangle\}$ is a set of tasks where each task is a tuple representing the task (t) and the set of concepts required to complete the task (c_u), all concepts in c_u are a subset of $concept(EO)$, and all concepts in the EO have a $LRFU$ weighting which is represented using a tuple $\langle c, LRFU\rangle$. The LRFU weighting for each concept is calculated over time. After each time period each concept's LRFU is calculated, by increasing the value by 1 if it is used and decaying it exponentially when it is not, so that concepts not used recently have a lower value. In our RoboCup Rescue example, concepts' LRFU weightings are calculated each timestep and are represented by tasks in the Algorithm because an agent has to complete a task per timestep. Depending on the scenario, it may be appropriate to weight the AC or LRFU differently. For example if network bandwidth fluctuates, the acquisition cost may be time-sensitive, and therefore it would be appropriate to weight it lower than LRFU. In our environment available bandwidth does not change, and therefore we do not apply weightings when calculating the CFV.

Algorithm 1 LRFU Algorithm.

Require: $concepts(EO) \neq \emptyset$
Require: $T = \{\langle t_1, c_u 1\rangle, \ldots, \langle t_n, c_u n\rangle\}$ /* T is the set of tasks, where tasks require a set of concepts to complete them. */
Require: $c_u \subset concepts(EO)$
Require: $\forall c \in concepts(EO) = \langle c, lrfu\rangle$

1: **for** $c_u \in T$ **do**
2: **for** $c \in concepts(EO)$ **do**
3: **if** $c \in c_u$ **then**
4: $c = \langle c, lrfu + 1\rangle$ // increment concept's LRFU
5: **else**
6: $c = \langle c, lrfu * \ln 2\rangle$ // exponentially degrade it
7: **end if**
8: **end for**
9: **end for**

Once a concept's LRFU factor has decayed so that the acquisition cost becomes more influential in the weighting,

an agent can determine which concept from a set of concepts that have the same LRFU to forget. It is more likely that concepts will have different acquisition costs due to different agent's network location and bandwidth, than a different LRFU because concepts decay exponentially. Performance wise, it is better for the agent to forget concepts that are inexpensive to acquire because the cost of re-acquiring them is less, compared to concepts that are expensive to acquire. To summarise, the agent selects the concept with the lowest rating in its EO to remove. In our example (see Figure 1), the agent selects the concept labelled *endotrachealTubes*, which is a piece of intubation equipment, because it has the lowest weighting. In the next section, we describe how the agent removes a fragment representing the selected concept.

4.2 Selecting Concepts

Once the agent has selected a concept it desires to forget, it creates a fragment representing that concept. The agent can reduce the memory requirements using a fragment instead of a whole ontology. The fragment is generated using the basic segmentation technique presented in [Seidenberg and Rector, 2006]. In more detail, it first selects the target concept (in our example (see Figure 1) the target concept is *endotracheal-Tubes*) and then selects all the concepts which connect it to the root node. It then selects the target's leaf classes. Finally, it selects concepts linked to the target concept and selects all the concepts which connect them to the root node (described in more detail in [Seidenberg and Rector, 2006]). In order to select concepts to prune from a fragment, the agent selects concepts with a similar CFV weighting so that it can forget more than one concept at a time. This reduces the number of forgetting operations and reduces the costs associated with pruning an ontology (including loading, removing concepts and saving the ontology).

Formally let: l be the capacity limit at which the agent is required to prune concepts from its EO; W be the set of weightings for the concepts contained in the EO, where $W = \{w : C \in concepts(EO) \wedge w = weight(c)\}$ and $c_1 \dots c_n \in concepts(EO)$; f_{o_q,c_t} be the fragment representing the concept to be forgotten, where o_q is the query agent's ontology (where $o_q = DO \cup EO$) and c_t is the concept to be forgotten; $W_{f_{o_q,c_t}} = \{W : C \in f_{o_q,c_t} \wedge w = weight(c)\}$ be the set of concept weightings associated with the concepts contained in f_{o_q,c_t}, where $concepts(f_{o_q,c_t}) = \{c_1, \dots c_n\}$. Using this formal notation we describe how we select the concepts to forget in Algorithm 2, which is run over all concepts in the EO.

Algorithm 2 Lowest Weighted Concept Selection Algorithm.

Function: *getLowestWeightedConcept(concepts):* `returns a concept`
Function: *getWeight(concept):* `returns the weight of concept`
Require: $F_d \leftarrow$ fragment received from merging process
Require: $t \leftarrow 10$
$c_t \leftarrow getLowestWeightedConcept(concepts(EO))$
$w_{c_t} \leftarrow getWeight(c_t)$
$ConceptsToRemove \leftarrow \{c_t\}$
for $c \in concepts(F_d)$ **do**
  if $|w_c - w_{c_t}| \leq t$ **then**
    $ConceptsToRemove \leftarrow ConceptsToRemove \cup \{c\}$
  end if
end for
return $ConceptsToRemove$

In our example, Figure 1 shows the ontology fragment representing concept *endotrachealTubes* which has weight $w_{c_t} = 0.02$, thus our selection algorithm selects the concepts *endotrachealTubes*, *laryngoscopes*, *connellAnotomicMask*, and *intubationEquipment* to forget (these concepts have a grey background). These concepts have not been used recently by the agent so they have a low CFV because the ambulance agents usually intubate civilians. In the next section we describe how the agent removes the selected concepts.

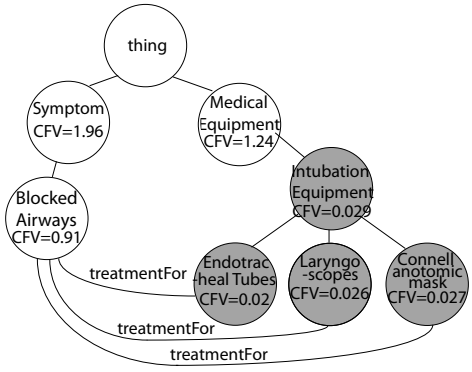

Figure 1: Concepts selected to be forgotten, where the curved lines represent relationships (domain and range restrictions).

4.3 Removing Concepts

After the agent has selected the concepts that it desires to remove, the agent then prunes these from its ontology. In order to prune these concepts from an ontology we use the technique presented in [Wang *et al.*, 2008] so that the ontology remains consistent. For example, we aim to remove concept $Intubation$ from: $MedicalEquipment \sqsubseteq Intubation$, $Intubation \sqsubseteq Laryngoscopes$ which results in $MedicalEquipment \sqsubseteq Laryngoscopes$; $MedicalEquipment \sqsubseteq Intubation$, $Intubation \sqsubseteq Laryngoscopes$ and $Intubation \sqsubseteq EndotrachealTubes$ results in $MedicalEquipment \sqsubseteq Laryngoscopes$ and $MedicalEquipment \sqsubseteq EndotrachealTubes$.

Algorithm 3 Pseudo-code of the RoboCup simulator.

Function: *contains(set, element)* returns true if *set* contains *element*
Require: *simulator* ← RoboCup rescue simulator
Require: *agent* ← RoboCup rescue agent
simulator.generateFires()
simulator.generateBlockades()
simulator.generateCivilians()
for *timestep* ∈ *timesteps* **do**
  target = getFirstTarget()
  targetInfo = agent.getInformation(target)
  if ¬ *contains(agent.ontology, targetInfo)* **then**
    fragments ← *requestFragements(targetInfo)*
    axioms ← *selectAxioms(fragments)*
    agent.ontology.learn(axioms)
  end if
  if ¬ *contains(agent.vehicle, requiredEquipment)* **then**
    travelToCentre()
    changeVehicle()
  end if
  rescue(target)
  simulator.update()
end for

5 Empirical Evaluation

We evaluate our approach, *forget-fragment*, against benchmark forgetting approaches (described below). We analyse their performance: in the number of civilians and buildings saved (see Section 2); and investigate the number of messages and tasks completed. As standard agents have five seconds to complete their actions, and have two thousand timesteps to complete their actions. The pseudo-code in Algorithm 3 provides the basic scenario of the agents in RCOR. In our experiments, we initialise a RCOR scenario where there are ten of each of the ambulance, fire brigade and police agents. The agents use the learning technique presented in [Packer *et al.*, 2010a], and augment their ontologies with unknown concepts or required equipment to rescue their target. This learning technique selects fragments from ontologies about requested concepts, and demonstrated the most efficient forgetting algorithm performance compared to benchmark approaches. We compare our forgetting approach to the following approaches:

Forget Concept removes all concepts and relationships related to the selected concept, where eo^{-c_p} and c_p is the concept to be pruned from the ontology. This technique removes one concept at a time, and is comparable those presented by [Eiter *et al.*, 2006] and [Wang *et al.*, 2008].

Forget Tree extends the above approach by selecting a subtree from the hierarchy of concepts in the agent's EO. The subtree is selected by comparing the weight used for each concept (see Algorithm 4), where $eo^{c_{tree}}$ and c_{tree} is a concept represented by the fragment (which is a subtree) being pruned from the ontology. Removing a connected subtree can result in removing a subtree, branch, or extraction of a subtree.

Algorithm 4 Subtree Selection Algorithm

Function: *parents(concept):* `returns a set of concepts`
Function: *children(concept):* `returns a set of concepts`
Function: *getLowestWeightedConcept(concepts):* `returns a concept`
Function: *getWeight(concept):* `returns the weight of the concept`
Require: $t \leftarrow 10$
Require: $ConceptsToRemove \leftarrow \emptyset$
Require: $CH \leftarrow \emptyset, P \leftarrow \emptyset$
1: $c_t \leftarrow getLowestWeightedConcept(concepts(EO))$
2: $w_{c_t} \leftarrow getWeight(c_t)$
3: $CH \leftarrow children(c_t)$
4: **for** $ch \in CH$ **do**
5: **if** $|w_{ch} - w_{c_t}| \leq t$ **then**
6: $ConceptsToRemove \leftarrow ConceptsToRemove \cup \{ch\}$
7: $CH \leftarrow CH \cup \{children(ch)\}$
8: **end if**
9: **end for**
10: $P \leftarrow parents(c_t)$
11: **for** $p \in P$ **do**
12: **if** $|w_p - w_{c_t}| \leq t$ **then**
13: $ConceptsToRemove \leftarrow ConceptsToRemove \cup \{p\}$
14: $P \leftarrow P \cup \{parents(p)\}$
15: **end if**
16: **end for**
17: **return** $ConceptsToRemove$

Forget Redundant removes concepts that are not used in future queries. Agents have a list of the future queries at the start of the simulation, which have been recorded on a dummy run using the same random seed. This is the only agent that requires a complete list of future queries. This agent is not limited by a capacity.

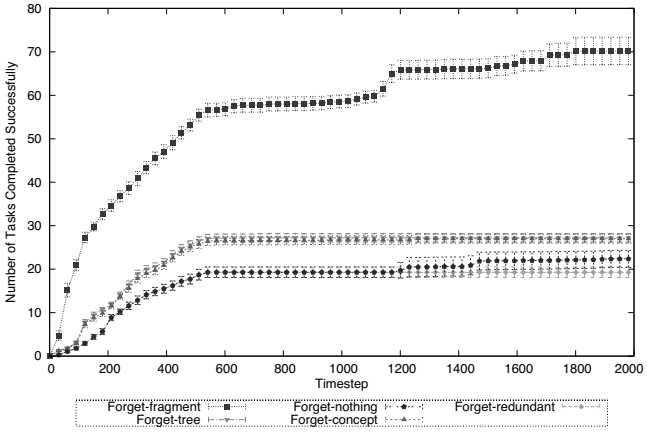

Figure 2: Cumulative average of tasks completed successfully.

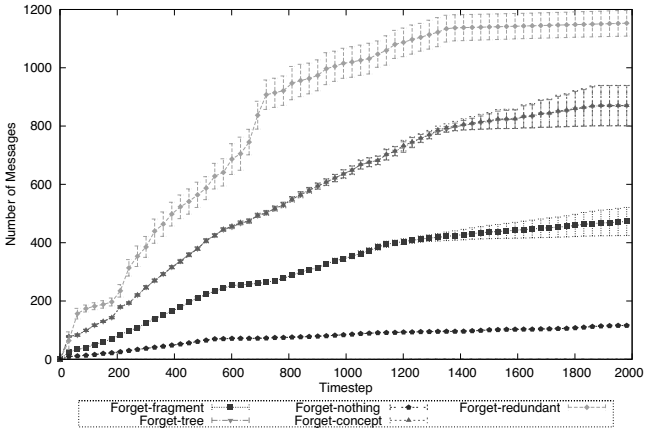

Figure 3: The average number of messages sent.

Forget Nothing does not remove any concepts from the agent's ontology and is not limited by a capacity.

We use these four approaches: **forget-concept**, **forget-tree**, **forget-redundant** and **forget-nothing** as benchmarks for our forgetting approach (**forget-fragment**). These agents adopt their behaviour defined by the sample package in RCR, which enables agents to plan a path through the city and which target to rescue first. In our evaluation we compare the performance of five forgetting approaches with the same 200 scenarios using the standard RCR Kobe map and compare the effectiveness of their approaches for statistical significance. Each scenario is randomly generated by the RCR simulators. The next section presents the results, and analysis of our evaluation.

6 Results

Overall our approach outperforms the other approaches, by saving the highest average number of civilians (99.3% more civilians than the next highest approach (*forget-tree*)) and extinguishes the highest average number of buildings at the end

Table 2: Comparison of average results for each forgetting technique.

Technique	Percentage of City Unburned	Percentage of Civilians in Refuge	Average Number of Tasks Completed Successfully	Average number of Messages per Timestep	Average Total Number of Messages	Number of Times Forgot	Number of Concepts Forgotten per Timestep
Forget-Concept	31.5	29.1	27.0	581.86	870.42	11.0	50.4
Forget-Tree	33.0	29.0	27.21	581.19	869.89	10.3	50.3
Forget-Redundant	31.5	19.8	19.29	853.06	1153.36	50.3	215.1
Forget-Nothing	31.5	1.4	22.38	76.88	116.32	0.0	0.0
Forget-Fragment	37.1	58.0	70.29	311.50	474.21	3.1	44.1

of the simulation (12% more than the next highest approach (*forget-tree*)), see Table 2. It outperforms other approaches because it completes more tasks successfully by spending less time forgetting and more time completing tasks (see Figure 2). Agents that quickly complete tasks can control the spread of the fire, thus reducing the chance that agents will encounter buildings that will burn out. The agents also rank which concepts to forget more accurately, because they encounter and complete more tasks, than the other forgetting approaches and are therefore able to forget the least useful concepts, and reduce the number of messages sent (see Figure 3).

The *forget-concept* and *forget-tree* approaches performed almost identically with regards to the number of messages and tasks completed (see Table 2). This is because the *forget-tree* approach was unable to find trees that were connected by the concept ratings, and subsequently removed the same number of concepts as the *forget-concept* approach. The *forget-redundant* and *forget-nothing* approaches performed the same in regards to the number messages sent, because the *forget-redundant* approach has perfect foresight and does not relearn any concepts. Agents using the *no-collaboration* approach are unable to learn information required for tasks, and thus any task that requires additional information is not possible for the agents to attempt. A consequence is that fires spread in an uncontrolled fashion through the environment, which initially generates more tasks to extinguish buildings, but eventually the city burns out and all of the civilians die.

7 Conclusions

In this paper we present a novel algorithm that selects the most appropriate concepts to remove from an ontology. While this algorithm is evaluated in the search and rescue domain, it is applicable to any scenario where an ontology evolves over time. In order to evaluate our algorithm we developed an extension to the RoboCup Rescue framework and implemented state of the art benchmark approaches. Our evaluation shows that our approach saves 99.3% more civilians and 12.4% more city area compared with the next best approach. For the future, we plan to investigate how to predict which concepts to forget given the outcome of past tasks. We will consider using simple Markov chains to model the concepts required for past tasks and which concepts are forgotten. Our investigation aims to show that we can enable agents to reduce their costs further through using prediction.

References

[Afsharchi *et al.*, 2006] M. Afsharchi, B.H. Far, and J. Denzinger. Ontology-Guided Learning to Improve Communication between Groups of Agents. *In Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan*, pages 923–930, 2006.

[Bailin and Truszkowski, 2002] S.C. Bailin and W. Truszkowski. Ontology Negotiation between Intelligent Information Agents. *The Knowledge Engineering Review*, 17(01):7–19, 2002.

[Eiter *et al.*, 2006] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, and K. Wang. Forgetting in Managing Rules and Ontologies. *In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, Hong Kong, China*, pages 411–419, 2006.

[Kitano and Tadokoro, 2001] H. Kitano and S. Tadokoro. Robocup rescue: A Grand Challenge for Multi-Agent and Intelligent Systems. *AI Magazine*, 22(1):39, 2001.

[Lee *et al.*, 1997] D. Lee, J. Choi, H. Choe, S. Noh, S. Min, and Y. Cho. Implementation and Performance Evaluation of the LRFU Replacement Policy. *In Proceedings of the 23rd Euromicro Conference, Budapest, Hungary*, pages 106–111, 1997.

[Packer *et al.*, 2010a] H. S. Packer, N. Gibbins, and N. R. Jennings. Collaborative learning of ontology fragments by cooperating agents. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Toronto, Canada*, volume 2, pages 89–96, 2010.

[Packer *et al.*, 2010b] H. S. Packer, N. Gibbins, and N. R. Jennings. Forgetting fragments from evolving ontologies. In *International Semantic Web Conference (ISWC), Shanghai, China*, volume 6496. Springer, November 2010.

[Seidenberg and Rector, 2006] J. Seidenberg and A. Rector. Web Ontology Segmentation: Analysis, Classification and Use. *In Proceedings of the 15th International Conference on World Wide Web, Edinburgh, Scotland*, pages 13–22, 2006.

[Soh, 2002] L.K. Soh. Multiagent, Distributed Ontology Learning. *Working Notes of the Second International Joint Conference of Autonomous Agents and Multi-Agent Systems, Workshop on Ontologies in Agent Systems, Bologna, Italy*, 2002.

[Wang *et al.*, 2008] Z. Wang, K. Wang, R. Topor, and J.Z. Pan. Forgetting Concepts in DL-Lite. *In Proceedings of the 5th European Semantic Web Conference, Tenerife, Canary Islands, Spain*, page 245, 2008.

[Wang *et al.*, 2009] K. Wang, Z. Wang, R. Topor, J. Pan, and G. Antoniou. Role Forgetting in ALC Ontologies. *In Proceedings of the 8th International Semantic Web Conference, Washington, DC*, pages 666–681, 2009.

[Wiesman and Roos, 2004] F. Wiesman and N. Roos. Domain Independent Learning of Ontology Mappings. *In Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems, New York, USA*, 2:846–853, 2004.