The 26th annual workshop of the Swedish Artificial Intelligence Society (SAIS)

20–21 May 2010, Uppsala, Sweden

Editor

Roland Bol

Linköping Electronic Conference Proceedings, No. 48
Linköping University Electronic Press
Linköping, Sweden, 2010

ISSN 1650-3686 (print)
http://www.ep.liu.se/ecp/048/
ISSN 1650-3740 (online)

© 2010, The Authors

Conference Organization

Programme Chair

Roland Bol

Program Committee

Marcus Bjäreland
Henrik Boström
Paul Davidsson
Patrick Doherty
Göran Falkman
Peter Funk
Christian Guttmann
Olle Gällmo
Fredrik Heinz
Kai Hübner
Arne Jönsson
Lars Karlsson
Danica Kragic
Jonas Kvarnström
Niklas Lavesson
Jacek Malec
Michael Minock
Mattias Nyberg
Lambert Spaanenburg
Cecilia Sönströd
Ning Xiong

Local Organizatioin

Pierre Flener
Justin Pearson

Table of Contents

Advancements of AI in Healthcare

Intelligent Adherence and Variation Support for Customer Life Cycle Management in Healthcare

Christian Guttmann

christian.guttmann@med.monash.edu.au

Summary

Maintaining contractual relationships and delivering services are key issues in Customer Life Cycle Management (CLCM). Current computer systems assist service providers in managing basic care for customers. This basic care often involves little more than maintaining customer information in a database.

AI technologies can play a valuable role in building advanced systems that monitor, interpret and react to key events during the management of a customer's life cycle. We have built a system that is an example of such an AI technology -- we refer to it as Intelligent Collaborative Care Management (ICCM). ICCM guides a team of providers and customers through critical stages in a customer's life cycle.

This system has been applied to the collaborative care of patients with chronic disease. Managing patient care is difficult, because health care providers and patients must collaboratively achieve goals in a customised care plan. Achieving these goals is strongly correlated with better health outcomes of patients. Regrettably, these outcomes are seldom attained because of uncertainty, incompleteness and bounded resources in this domain (e.g., change of health conditions, change of objectives of providers, unreliable patients and change of governance policies).

ICCM specifies intelligent agents that assist health care providers and patients in
(a) adhering to a care plan and
(b) varying the care plan itself, if required.
This adherence and variation support component in ICCM defines a monitoring-recognition-intervention cycle: monitoring provider and customer behaviour, recognising off-track behaviour, and intervening to (a) move processes back on-track or (b) make changes to the care plan itself.

This presentation is concluded by outlining challenging future directions in this line of research.

CPTV - Generating Personalized TV Schedules

Helmut Simonis

Summary

Today's cable and satellite systems provide a multitude of TV viewing choices for a user, making it difficult to select the best program to watch at any one time. Digital video recording and IPTV add even more options, allowing to time- and sometimes place-shift TV watching. CPTV is a system which generates a personalized viewing schedule for a user, integrated with his current calendar tools. The system is based on three main elements:

- a rule based system to express viewing preferences,
- a content-based recommender system to suggest films based on past viewing selections and information extracted from Wikipedia, and
- a constraint-based scheduler which plans and schedules recording and viewing schedules around existing events and availability time periods of a user.

The application thus combines different AI methods in one Web-based application, and is implemented using the constraint logic programming system ECLiPSe.

Achievements of AI in Linguistics

Anna Sågvall Hein

anna.sagvall-hein@convertus.se

Summary

AI aims at simulating human intelligence by computer. Language is one of the primary expressions of human intelligence. It has been claimed that the acquisition language is the greatest intellectual achievement of man. Natural language processing is, naturally so, one of the major fields of AI, and to a large extent it overlaps with Computational Linguistics, an important and continuously growing field of linguistics.

The fundamental problem of computational linguistics is the modelling of the basic linguistic processes – comprehension, production and learning of language. It includes central AI problems such as perception, communication, knowledge, planning, reasoning and learning. On the application side, information retrieval including text mining and machine translation, currently, seem to dominate.

By tradition, the term Computational Linguistics refers to written language, whereas Speech Technology is used for the analysis and synthesis of spoken language. In later years, Language Technology has emerged as a common denominator of both modes of language. The division into Speech Technology and Computational Linguistics was mainly based on different scientific traditions and methods. Whereas statistical methods dominated in speech technology, so did symbolic processing in computational linguistics. However, the statistical methods originally developed for the analysis of speech were found to be more or less directly applicable to the fairly new paradigm of data-driven machine translation. Below, machine translation will be focused.

Intuitively, translation may be understood as comprehending a source language text and producing an equivalent text in the target language. In terms of AI, or CL, it would imply modelling the comprehension process and the production process, respectively.

Assuming that the comprehension process would result in a complete, language independent meaning representation, an interlingua, this representation might be the starting point of the production process, and there would be no need for a specific translation step. In such an approach, knowledge becomes a central issue. How to identify it and how to represent it? This ideal approach to machine translation, the interlingua approach, has been explored by many researchers with interesting findings, but no viable translation system, as a result. The Google translation service is as far from the interlingua model that one may get, since it does not use any refined knowledge of language at all.

AI researchers in the 70-ies, such as Roger Schank, Terry Winograd and Yorick Wilks, made substantial and innovative contributions to the exploration of language comprehension and production, illustrating the relevance of knowledge, in particular world knowledge, planning, and reasoning in the use of language. Their approaches were exclusively symbolic, excluding, basically,. machine learning of language. Currently, the data-driven approach

dominates the field of machine translation, and attempts are made to combine or complement it with rule-based methods in order to overcome inherent limitations with the approach.

Basic problems of machine translation are due to lexical ambiguity and variation, and grammatical differences in morphology and word order. In the rule-based paradigm, the problems are approached by linguistic analysis sorting out the different uses, hence translation options, of the words. The linguistic analysis also is also the back-bone in creating an appropriate word order. The data-driven, or statistical approach, relies on reuse of large amounts of previously translated text, parallel data.

A problem with the rule-based approach, RBMT, is the coverage of language resources, dictionaries and grammars. Typically, coverage is insufficient and methods have to be found to handle text outside dictionaries and grammars. This is where statistical methods may help.

In statistical machine translation, SMT, the choice of translation alternatives is based on a comparison of the source text with large amounts of parallel data. The fundamental problem is access to sufficiently large amounts of parallel data. The amount needed depends, among other things, on the language-pair in focus. Integrating linguistic data aims at reducing the amount of parallel data needed for quality translation.

Translation quality is a critical issue in machine translation. However, demands on quality vary with the purpose of the translation. Browsing quality, understandability, may be sufficient in certain contexts, whereas publishing quality must approach the quality of human translation. In addition to human evaluation, costly and time-consuming, different metrics for the automatic evaluation of machine translation have been proposed. They assume access to reference data in terms of previous translations of high-quality, typically human translations.

Success factors of machine translation in practice concern adaptation and customisation of the machine translation system to the needs of the user, pre-processing of the source text (language checking, controlled language), and post-processing of the target text. A research issue of great concern is automating the post-processing process.

As an example of machine translation in use, the *Convertus Syllabus Translator*, CST, will be presented. CST translates course syllabi from Swedish to English and is in use at six Swedish universities. The presentation will include the following aspects:

- Background and development
- Methodology
- Translation dictionary
- Spell-checking
- Post-processing interface and translation memory function
- Automatic post-processing
- Scenarios of use
- User feed-back
- Future development and deployment

Finally, further development and prospects of machine translation will brought up for discussion.

Using AI to interpret BI: machine learning for decoding and characterization of brain activity patterns

Malin Björnsdotter[*], Simon Beckmann, Erik Ziegler & Johan Wessberg

Abstract

The appealing properties of artificial intelligence (AI) methods are being increasingly acknowledged by the neuroimaging community, as evidenced by the recent surge of brain activity pattern recognition studies [19]. Supervised learning and classification, in particular, are appreciated tools for localizing and distinguishing intricate brain response patterns and making predictions about otherwise undetectable neural states. Our group refines and applies such methods in order to implement sensitive and dynamic tools for characterization of neurophysiological data. Specifically, we employ support vector machines (SVMs), particle swarm optimization (PSO), independent component analysis (ICA), as well as both genetic and memetic algorithms on functional magnetic resonance imaging (fMRI) and electroencephalography (EEG) data. This paper provides a brief overview of our recent advances in the development and utilization of AI-based analysis, with the particular aspiration to characterize human brain activation patterns produced by touch.

1 Introduction

Machine learning approaches for mining neurophysiological data are rapidly gaining popularity, and justifiably so, as they provide a level of analysis not possible with conventional methods [19]. Commonly used systems for acquiring such brain activity data non-invasively include electroencephalography (EEG), which provides an estimate of electrical currents produced by the brain, and functional magnetic resonance imaging (fMRI), where local blood flow changes are quantified. Classically, the analysis of neurophysiological data relies on descriptive statistical measures where average activity changes in single data points are related to an experimental condition and particular data characteristic of interest (e.g. brain regions in fMRI and frequencies in EEG). Artificial intelligence techniques, in contrast, are capable of distinguishing complex and subtle data patterns, distributed across numerous measuring points, on a single-trial basis. The benefits of AI were acknowledged early for real-time classification of EEG signals in brain-computer interfaces (see e.g. [4] for a review), and more recently for clinical evaluation (see e.g. [9, 28, 24]), as well as a wide range of fMRI analyses (see e.g. [11] and [19]).

Generally, supervised learning techniques are used: a classifier is trained to recognize and decode subtle intrinsic signal patterns correlated to given brain states, such as the fMRI activity produced by a single touch stimulus [2]. Supervised learning methods not only enable real-time single-trial classification (such as brain state tracking [29] or intent decoding [30]), but by virtue of considering information encoded over multiple measuring points they also provide improved condition differentiation sensitivity [27, 19, 13].

In the following sections, we summarize our recent advances in the development and application of such AI-based analysis in both fMRI and EEG studies. In particular, we utilize these methods to characterize human brain activation patterns produced by touch.

2 Functional Magnetic Resonance Imaging

fMRI involves the estimation of local blood flow changes in measuring points, known as voxels,

[*]All authors are with the Department of Physiology, Institute of Neuroscience and Physiology, University of Gothenburg, Sweden. Email: malin.bjornsdotter@neuro.gu.se

which are evenly distributed across the brain volume. Conventional analysis is limited to localization of brain regions which are activated on average by a specific condition using general linear model (GLM) methods, where each voxel is treated independently of any other [16]. Classifier-based approaches, popularly termed multivoxel pattern analysis (MVPA), on the other hand, utilize multiple voxels simultaneously in a multivariate fashion, allowing identification of brain regions containing spatially distributed activity patterns. Numerous studies have demonstrated that by extracting such spatially-encoded information, otherwise non-discernible pattern differences can be identified (e.g. in lie detection [12], the decoding of single visual stimuli – visible [21], as well as invisible [18] – biofeedback [33], and various types of real-time fMRI analysis [23, 14]). In addition, the ability of MVPA to decode single-trial brain states provides an effective tool for observing brain-state changes in real time (somewhat equivocally termed "mind reading"; see e.g. [29, 30]).

A major challenge in MVPA analysis is to identify which voxels (out of hundreds of thousands) are in fact relevant to the classification task. To this end, we have proposed a number of combinatorial optimization techniques based on evolutionary approaches combined with a classifier for fitness evaluation [20]. First we implemented a simple genetic algorithm (GA) to show that only a handful of voxels where sufficient to successfully decode gentle brushing of the forearm (subject mean of 74.3% correct with a linear support vector machine classifier) and that the performance was significantly improved compared to a univariate GLM-based voxel selection scheme (see [1]; Figure 1).

While producing high classification rates by identifying few, representative voxels which were *sufficiently* informative, the simple GA did, however, yield spatially sparse and distributed brain maps which were of limited use for a physiological examination of underlying neural processes. The algorithm was therefore refined to include elements of voxel clustering, with the goal of identifying larger brain regions containing *useful* voxels [8].

In contrast to the simple GA, the clustering algorithm proved to be highly useful in revealing physiologically relevant brain regions. We could, for example, detect which regions of the so-called insular cortex were differentially activated by gentle touch of the forearm and thigh, where conventional GLM methods failed (see [6]; Figure 2).

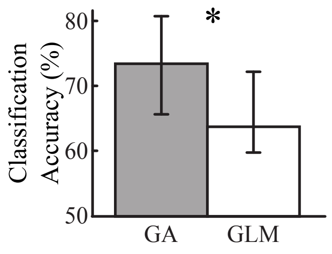


Figure 1: Classification results on fMRI data from six individuals demonstrating that voxels selected by a simple genetic algorithm are highly effective in predicting whether or not the individuals were sensing a tactile stimulation (soft brush) on their forearm. Importantly, a significantly higher classification rate was obtained on the voxels selected by the genetic algorithm in comparison with the conventional general linear model (GLM) approach. Chance classification is 50%.

The clustering GA proved, however, to be erratic when attempting to identify a more complex distribution of multiple voxel clusters. This issue is currently being resolved by incorporating a local search element according to a memetic algorithm (MA) framework, and the improved algorithm is successful in detecting multiple clusters of voxels distributed across the brain volume. Illustratively, the memetic algorithm contrasts the simple GA by being substantially better at detecting *all* useful voxels as opposed to those *sufficient* for good performance (Figure 3). On real data, the memetic algorithm was highly successful in detecting brain regions which could decode finger movement brain patterns learnt from seven subjects and applied to an eighth individual (Figure 4).

The proposed evolutionary algorithms are relatively complex to implement. As a simpler alternative, we are currently exploring particle swarm optimization (PSO). Our first attempt included an implementation which was similar to the simple GA, with no explicit spatial clustering of the voxels [26]. Again, this algorithm provided high brain-state classification results, but yielded maps which were difficult to interpret (Figure 5).

In a second implementation, clustering was forced such that each particle, instead of coding a number of distributed voxels, represented a sin-

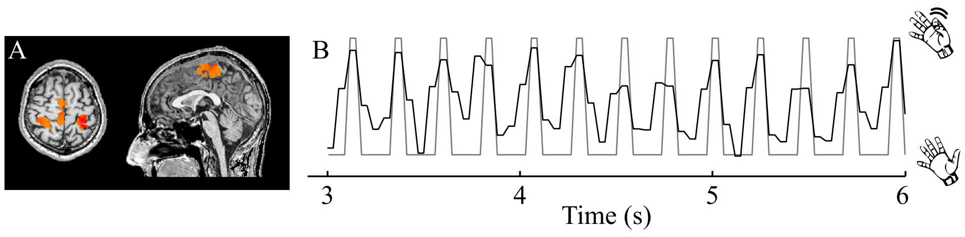


Figure 4: Decoding finger movements from fMRI data in brain regions selected by the memetic algorithm. A) The brain regions identified by the algorithm. B) A temporal trace of the classification performance (thick line). The true brain state is indicated by the thin gray line.

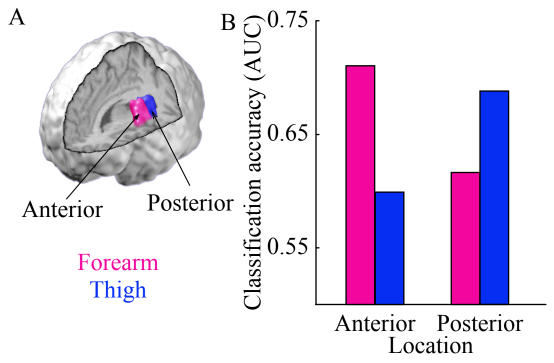


Figure 2: A) Brain regions identified for forearm (purple) and thigh (blue) gentle brushing by the clustering genetic algorithm. B) The brain state decoding accuracies (measured in area under the receiver operating characteristic curve, AUC) obtained when attempting classification of the forearm data in the forearm brain region and vice versa for all permutations of data and regions. The results demonstrate that this patch of the brain, called the insular cortex, is organized in a somatotopic fashion with forearm stimulation projecting anterior to thigh.

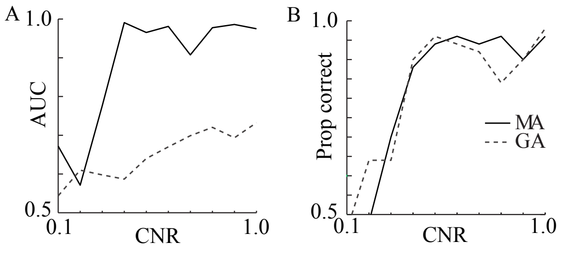


Figure 3: The memetic algorithm (MA) is substantially better than the simple genetic algorithm (GA) in detecting all useful fMRI voxels (A: performance measured in area under the receiver operating characteristic curve, AUC), although both algorithms detect useful voxels sufficient for good classification (B). The methods were evaluated on simulated data of varying contrast-to-noise ratio (CNR) containing useful voxels of known location.

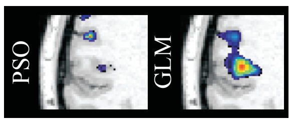


Figure 5: Voxel selection frequency using a simple particle swarm optimization (PSO) algorithm to identify brain areas which are activated by gentle touch, compared to a standard general linear model (GLM) map. The values are scaled to reflect minimum (blue) to maximum (red) map values.

gle spherical cluster of voxels [7]. This approach was inspired by the highly popular "searchlight" method, where such search spheres are sequentially centered on each voxel in the brain [22]. The proposed PSO method was successful in detecting brain areas where the sensation of a soft brush stimulus could be accurately decoded (Figure 6), and was substantially faster than the searchlight (e.g. 6.7 minutes compared to 9 hours to map a whole brain).

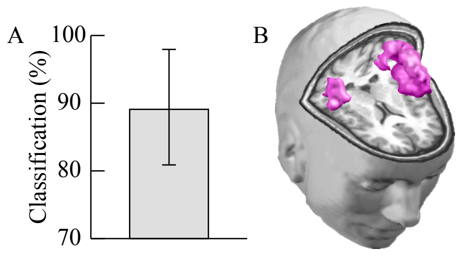


Figure 6: The average classification score (A) and corresponding brain areas (B) detected by the clustering particle swarm optimization algorithm (PSO).

3 Electroencephalography

EEG involves the registration of electrical brain activity at a temporal resolution of milliseconds using electrodes attached to the scalp [15]. The signals are believed to mainly reflect post-synaptic currents in nerve cells, and synchronous activity of thousands of cells is required to produce a measurable potential. With recent advances in the complex problem of localizing signal sources within the brain, EEG has emerged as an excellent tool for both characterizing temporal dynamics and localizing cognitive processing.

Standard EEG analysis involves averaging over numerous events to produce event-related potentials (ERPs) which are subsequently analyzed based on the latency and amplitude of characteristic deflections. The averaging procedure, however, may mask or even eliminate relevant information potentially concealed in complex signal patterns. We therefore explored an EEG analysis approach based on independent component analysis (ICA) combined with supervised learning. ICA blindly decomposes multi-channel EEG data into maximally independent component processes (ICs) that typically express either particularly brain generated EEG activities or some type of non-brain artifacts (e.g. line noise or muscle activity [10]).

We used this approach to identify and characterize differential temporal patterns in the EEG responses to stimulation of the fingerpad with surfaces of varying roughness. Our classifier (a linear support vector machine, SVM) was successfully trained to differentiate between the temporal patterns evoked by the two different textures in some ICs but not in others [3]. For example, an EEG component generated in the contralateral somatosensory cortex with activation peaks at 100 ms after onset (P100) of stimulation significantly differentiated the textures (Figure 7).

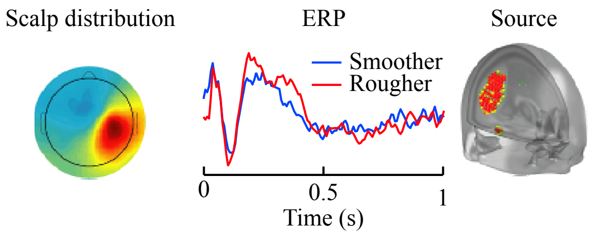


Figure 7: Example of a contralateral somatosensory component where the support vector machine classifier could differentiate two surfaces of different textures. The scalp distribution plot indicates the component's scalp map projection. The component's event-related potentials (ERPs) are plotted for each of the textures (rough spatial period: 1920 μm, smooth: 520 μm) and a source image was constructed to indicate the component's origin.

4 Concluding remarks

AI-based approaches for analyzing brain activity provide a highly appealing complement to conventional statistical methods, enable a deeper understanding of brain function, and promote the development of novel medical techniques. Whereas we primarily use AI for the characterization of brain activation patterns, other brain signal decoding applications include brain-computer interfaces (BCIs), biofeedback [32, 31, 33], real-time signal analysis [14], disease diagnosis [25, 17], enabling prosthesis control, and opening communication channels with locked-in patients [5]. The application of AI concepts in neuroimaging is in its infancy, and further refinement of these algorithms will undoubtedly facilitate our understanding of the human brain.

Acknowledgments

These studies were supported by the Swedish Research Council (grant K2007-63X-3548), the Sahlgrenska University Hospital (grant ALFGBG 3161) and NANOBIOTACT (`nanobiotact.org`, EU-FP6-NMP-033287). We are grateful to Karin Göthner and Linda Lundblad for assistance with data acquisition.

References

[1] M. Åberg and J. Wessberg. Evolutionary optimization of classifiers and features for single trial EEG discrimination. *BioMedical Engineering Online*, 6(32), 2007.

[2] M. S. Beauchamp, S. LaConte, and N. Yasar. Distributed representation of single touches in somatosensory and visual cortex. *Human Brain Mapping*, (in press), 2009.

[3] S. Beckmann, M. Björnsdotter, H. Backlund-Wasling, and J. Wessberg. Brain decoding of texture processing using independent component analysis and support vector machines. pages 287 –292, aug. 2009.

[4] N. Birbaumer. Breaking the silence: brain-computer interfaces (bci) for communication and motor control. *Psychophysiology*, 43(6):517–32, 2006.

[5] N. Birbaumer and L. G. Cohen. Brain-computer interfaces: communication and restoration of movement in paralysis. *Journal of Physiology*, 579(Pt 3):621–636, Mar 2007.

[6] M. Björnsdotter, L. S. Löken, H. Olausson, Å. B. Vallbo, and J. Wessberg. Somatotopic organization of gentle touch processing in the posterior insular cortex. *Journal of Neuroscience*, 29(29):9314–9320, 2009.

[7] M. Björnsdotter and J. Wessberg. Identification and localization of human brain activity patterns using particle swarm optimization. In *Proceedings of the 2nd International Conference on Computer and Automation Engineering*, 2010.

[8] M. Björnsdotter Åberg and J. Wessberg. An evolutionary approach to the identification of informative voxel clusters for brain state discrimination. *IEEE Journal of Selected Topics in Signal Processing*, 2(6):919–928, 2008.

[9] C. Cao, R.L. Tutwiler, and S. Slobounov. Automatic classification of athletes with residual functional deficits following concussion by means of eeg signal using support vector machine. *IEEE Trans Neural Syst Rehabil Eng.*, 16(4):327–35, 2008.

[10] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287 – 314, 1994. Higher Order Statistics.

[11] D. D. Cox and R. L. Savoy. Functional magnetic resonance imaging (fMRI) 'brain reading': detecting and classifying distributed patterns of fMRI activity in human visual cortex. *NeuroImage*, 19(2 Pt 1):261–270, June 2003.

[12] C. Davatzikos, K. Ruparel, Y. Fan, D. G. Shen, M. Acharyya, J. W. Loughead, R. C. Gur, and D. D. Langleben. Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection. *NeuroImage*, 28:663–668, 2005.

[13] F. De Martino, G. Valente, N. Staeren, J. Ashburner, R. Goebel, and E. Formisano. Combining multivariate voxel selection and support vector machines for mapping and classification of fMRI spatial patterns. *NeuroImage*, 43(1):44–58, 2008.

[14] R. C. deCharms. Applications of real-time fMRI. *Nature Reviews Neuroscience*, 9(9):720–729, Sep 2008.

[15] Bruce J. Fisch. *Fisch & Spehlmann's EEG Primer; Basic Principles of Digital and Analog EEG*. Elsevier Science, third edition edition, 1999.

[16] K. J. Friston, A. P. Holmes, K. J. Worsley, J. P. Poline, C. D. Frith, and R. S. J. Frackowiak. Statistical parametric maps in functional imaging: A general linear approach. *Human Brain Mapping*, 2(4):189–210, 1994.

[17] C. H. Y. Fu, J. Mourão-Miranda, S. G. Costafreda, A. Khanna, A. F. Marquand, S. C. R. Williams, and M. J. Brammer. Pattern classification of sad facial processing: toward the development of neurobiological markers in depression. *Biological Psychiatry*, 63(7):656–662, Apr 2008.

[18] J.-D. Haynes and G. Rees. Predicting the orientation of invisible stimuli from activity in human primary visual cortex. *Nature Neuroscience*, 8(5):686–691, April 2005.

[19] J.-D. Haynes and G. Rees. Decoding mental states from brain activity in humans. *Nature Reviews Neuroscience*, 7(7):523–534, July 2006.

[20] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* The MIT Press, April 1992.

[21] Y. Kamitani and F. Tong. Decoding the visual and subjective contents of the human brain. *Nature Neuroscience*, 8(5):679–685, May 2005.

[22] N. Kriegeskorte, R. Goebel, and P. Bandettini. Information-based functional brain mapping. *Proceedings of the National Academy of Science*, 103:3863–3868, 2006.

[23] S. M. LaConte, S. J. Peltier, and X. P. Hu. Real-time fMRI using brain-state classification. *Human Brain Mapping*, 28(10):1033–1044, Oct 2007.

[24] J. Löfhede, N. Löfgren, M. Thordstein, A. Flisberg, I. Kjellmer, and K. Lindecrantz. Classification of burst and suppression in the neonatal electroencephalogram. *J Neural Eng.*, 5(4):402–10, 2008.

[25] A. F. Marquand, J. Mourão-Miranda, M. J. Brammer, A. J. Cleare, and C. H. Y. Fu. Neuroanatomy of verbal working memory as a diagnostic biomarker for depression. *Neuroreport*, 19(15):1507–1511, Oct 2008.

[26] T. Niiniskorpi, M. Björnsdotter Åberg, and J. Wessberg. Particle swarm feature selection for fMRI pattern classification. In P. Encarnação and A. Veloso, editors, *Proceedings of the 2nd BIOSIGNALS*, pages 279–284. INSTICC, 2009.

[27] K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby. Beyond mind-reading: multivoxel pattern analysis of fMRI data. *Trends in Cognitive Sciences*, 10(9):424–430, 2006.

[28] L.M. Patnaik and O.K. Manyam. Epileptic eeg detection using neural networks and post-classification. *Comput Methods Programs Biomed*, 91(4):100–9., 2008.

[29] S. M. Polyn, V. S. Natu, J. D. Cohen, and K. A. Norman. Category-specific cortical activity precedes retrieval during memory search. *Science*, 310(5756):1963–1966, 2005.

[30] Chun Siong Soon, Marcel Brass, Hans-Jochen Heinze, and John-Dylan Haynes. Unconscious determinants of free decisions in the human brain. *Nature Neuroscience*, 11(5):543–545, May 2008.

[31] N. Weiskopf, F. Scharnowski, R. Veit, R. Goebel, N. Birbaumer, and K. Mathiak. Self-regulation of local brain activity using real-time functional magnetic resonance imaging (fMRI). *Journal of Physiology (Paris)*, 98(4-6):357–373, 2004.

[32] N. Weiskopf, R. Veit, M. Erb, K. Mathiak, Grodd W., Goebel R., and Birbaumer N. Physiological self-regulation of regional brain activity using real-time functional magnetic resonance imaging (fMRI): methodology and exemplary data. *NeuroImage*, 19:577–586, 2003.

[33] S. S. Yoo, H. M. O'Leary, T. Fairneny, N. K. Chen, L. P. Panych, H. Park, and F. A. Jolesz. Increasing cortical activity in auditory areas through neurofeedback functional magnetic resonance imaging. *Neuroreport*, 17(12):1273–1278, 2006.

Supervised SOM Based Architecture versus Multilayer Perceptron and RBF Networks

David Gil
Computing Technology and Data Processing
University of Alicante, Spain

Magnus Johnsson
Lund University Cognitive Science
Lund, Sweden

April 21, 2010

Abstract

We address a contrastive study between the well known Multi-Layer Perceptron (MLP) and Radial Basis Function (RBF) neural networks and a SOM based supervised architecture in a number of data classification tasks. Well known databases like Breast Cancer, Parkinson and Iris were used to evaluate the three architectures by constructing confusion matrices. The results are encouraging and indicate that the SOM based supervised architecture generally achieves results as good as the MLP and slightly higher on some measures than the RBF network.

1 Introduction

The use of classifer systems in many areas is increasing gradually. Recent advances in the field of artificial intelligence have led to the emergence of expert systems and Decision Support Systems (DSSs) for economics, linguistics, management science, mathematical modelling, psychology, etc. Artificial Neural Networks (ANNs) have been utilized for improving the classification tasks because of its property called black-box learning. In fact, they are one of the popular methods for classification problems [13] [12].

Compared to most traditional classification approaches, ANNs are nonlinear, nonparametric, and adaptive. They can theoretically approximate any fundamental relationship with arbitrary accuracy. They are ideally suitable for problems where observations are easy to obtain but the data structure or underlying relationship is unknown.

Although there are different types of learning techniques, this paper proposes the study of supervised learning. The learning system may label (classification) a set of vectors choosing one between several categories (classes). There are several types of classifiers that have been used with different degrees of accuracy [28] [21] [20].

Some of our related work in the field of the diagnosis has been developed basically by means of Artificial Neural Networks (ANNs) [9] [10].

The aim of this paper is to test a SOM based supervised architecture ANN and compare it in classification tasks with two other ANNs known as the Multi-Layer Perceptron (MLP) and the Radial Basis Function (RBF) network.

We have compared the SOM based supervised architecture with the MLP and the RBF networks as the MLP and RBF neural networks are the two most widely used within the field of task classification. Moreover, the MLP is purely supervised while the RBF is also a hybrid network with a part of unsupervised learning. Certain similar characteristics of the SOM based supervised architecture and the RBF will allow allow the acquisition of good measures of the efficiency of our network.

The remaining part of the paper is organized as follows: First we give a brief description of the networks MLP, RBF and a longer description of the SOM Based Supervised Architecture; Then we proceed by describing the design of our proposal and the training of the ANNs by the available data; Then we continue by describing the subsequent testing carried out in order to analyse the results; Finally we draw the relevant conclusions.

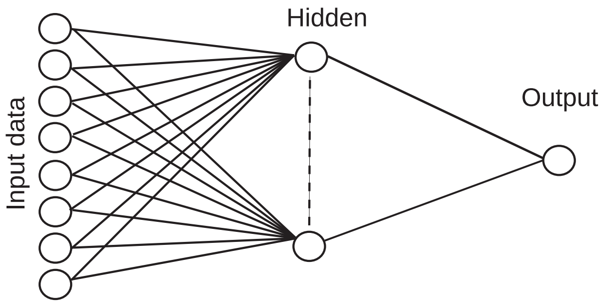


Figure 1: The architecture of the MLP network (input layer, hidden layer and output layer). The input layer represents the input data (the input data is described in section 4.1). The usage of a hidden layer enables the representation of data sets that are not linearly separable. The output layer represents the classification result and it contains as many outputs as the problem has classes, although here only one neuron is shown. The weights and the threshold of the MLP are calculated during an adaptation process.

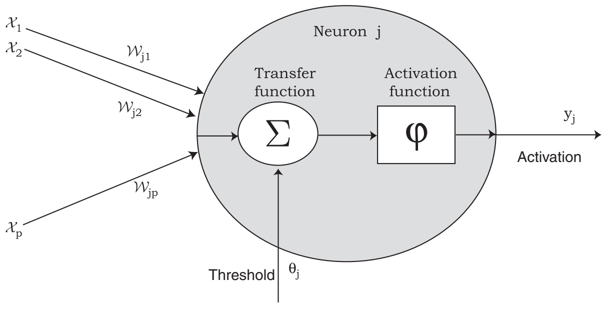


Figure 2: A neuron in the hidden or the output layer in the MLP. In the experimentation section the number of hidden neurons of the MLP will be established.

2 Supervised Architectures

2.1 Multilayer Perceptron

In this study we have used a Multi-Layer Perceptron (MLP) network [24] [11][5]. A typical MLP consists of three layers of neurons: an input layer that receives external inputs, one hidden layer, and an output layer which generates the classification results (see figure 1). Note that unlike other layers, no computation is involved in the input layer. The principle of the network is that when data are presented at the input layer, the network neurons run calculations in the consecutive layers until an output value is obtained at each of the output neurons. This output will indicate the appropriate class for the input data.

Each neuron (see figure 2) in the input and the hidden layers is connected to all neurons in the next layer by weighted connections. The neurons of the hidden layers (see figure 2) compute weighted sums of their inputs and adds a threshold. The resulting sums are used to calculate the activity of the neurons by applying a sigmoid activation function.

This process is defined as follows:

$$\nu_j = \sum_{i=1}^{p} w_{ji} x_i + \theta_j \quad , \qquad y_j = f_j(\nu_j) \tag{1}$$

where ν_j is the linear combination of inputs $x_1, x_2, ..., x_p$, and the threshold θ_j , w_{ji} is the connection weight between the input x_i and the neuron j, and f_j is the activation function of the j_{th} neuron, and y_j is the output. The sigmoid function is a common choice of activation function. It is defined as:

$$f(t) = \frac{1}{1+e^{-t}} \tag{2}$$

A single neuron in the MLP is able to linearly separate its input space into two subspaces by a hyperplane defined by the weights and the treshold. The weights define the direction of this hyperplane whereas the threshold term θ_j offsets it from origo.

The MLP network uses the backpropagation algorithm [25], which is a gradient descent method, for the adaptation of the weights (the backpropagation training parameters are showed in Table 1). This algorithm runs as follows:

All the weight vectors w are initialized with small random values from a pseudorandom sequence generator. Then and until the convergence (i.e. when the error E is below a preset value) we repeat the three basic steps:

- The weight vectors w_i are updated by

$$w(t+1) = w(t) + \Delta w(t) \tag{3}$$

- where

$$\Delta w(t) = -h\partial E(t)/\partial w \qquad (4)$$

- Compute the error E(t+1).

where t is the iteration number, w is the weight vector, and h is the learning rate.

The backpropagation MLP is a supervised ANN. This means the network is presented with input examples as well as the corresponding desired output. The backpropagation algorithm adapts the weights and the thresholds of the neurons in a way that minimizes the error function E

$$E = \frac{1}{2}\sum_{p=1}^{n}(d_p - y_p)^2 \qquad (5)$$

where y_p is the actual output and d_p the desired output for input pattern p.

The minimization of E can be accomplished by gradient descent, i.e. the weights are adjusted to change the value of E in the direction of its negative gradient. The exact updating rules can be calculated by applying derivatives and the chain rule (for the weights between the input and the hidden layer).

2.2 Radial Basis Function Network

In this section, the basic concepts of the Radial Basis Function (RBF) network are described. Radial Basis Function Networks are a type of ANN where the hidden layer is composed of radial-basis functions which are similar to normal distribution curves.

The RBF neural network [3] is generally composed of three layers: input layer, hidden layer and output layer. The input layer feeds the input data to each of the neurons of the hidden layer. The hidden layer differs greatly from other neural networks in that each neuron represents a data cluster with a given radius and which is centred at a particular point in the input space. Each neuron in the hidden layer calculates the distance from the input vector to its own center. The calculated distance is transformed via some basis function and the result is the output from the neuron. The output from the neuron is multiplied by a constant or weighting value and fed into the output layer. The output layer consists of as many classes or outputs as the problem has. It acts to sum the outputs of the previous layer and to yield a final output value [18] [7]. A generic architecture of an RBF network with p input and n hidden neurons is illustrated in figure 3, where x_i are data points, $\varphi(||x - x^i||)$ are the RBFs, x^i are the centres of the basis functions, and w_i are the weights. A very common RBF is the Gaussian RBF:

$$\varphi(||x - x^i||) = exp(-\frac{||x - x^i||^2}{2\sigma_i^2}), i = 1, 2, ..., N \qquad (6)$$

The activity $F(X)$ of the output neuron is given by:

$$F(X) = \sum_{j=1}^{N} w_j \varphi(||x - x^i||) \qquad (7)$$

The learning process used in the RBF network is done in two phases thus calculating the parameters of the hidden layer and output layer (note that the input layer does not perform calculations at all). In this way, we can speed up the learning process considerably compared to the MLP backpropagation. First of all, for the hidden layer, we calculate the number of centres of the basis functions or centroids (figure 3 and 4), using the K-means algorithm. The number of these centroids depend on each case the problem addressed. Table 2 and the explanation in the experimental section help to understand this process. In the second phase we proceed to the training of the output neurons which is an easy task since we know the values. The calculation is then done by simply applying the equation 7.

2.3 SOM Based Supervised Architecture

This section proposes a "Supervised SOM Based Architecture" (see Figure 5). The hidden layer of an ANN is one of the most complex parts to design in an artificial neural network. Here we propose to use a Self-Organizing Map (SOM) as a hidden layer. Some previous works where similar models are implemented are [27] [6] [23].

In [23], Piela represents a very exciting research where a modern approach to imputation is being discussed. It is said that many traditional methods

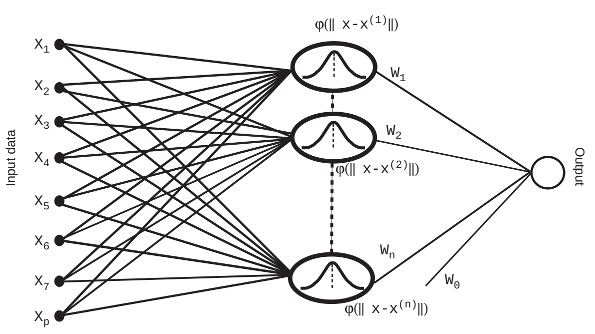


Figure 3: The architecture of the RBF network (input layer, hidden layer and output layer). The input layer represents the input data (the input data is described in the experiment section). The output neuron activity is reflected by equation 7. As in the case of the MLP the output layer of the RBF network consists of as many neurons as the problem has classes, although here only one neuron is shown.

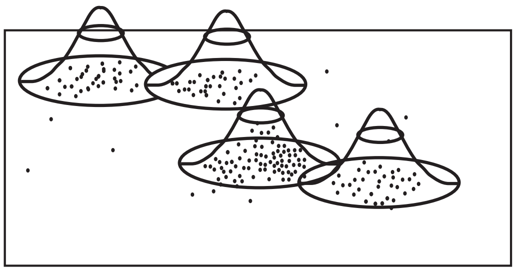


Figure 4: The radial basis functions have a local character since they are functions that reach a level close to their maximum when the input pattern X(n) is close to the centre of the neuron. When the pattern moves away from the centre, the function value is tending to the minimum value. The outputs of the radial basis neural networks are therefore a Gaussian linear combination, where each of the terms in the linear combination is activated for a particular portion of space defined by the input patterns.

of imputation use some kind of classification trying to get observations with missing values into as homogenous groups as possible. SOM is an iterative method for classification and can thus also be used in finding the imputation classes. Therefore, imputations are made within clusters in several ways which can be based on both traditional and neural methods. The main emphasis of this approach is to aid methodological development of knowledge discovery, data analysis, and modelling in general.

The SOM based architecture discussed in this paper consists of two layers (actually two separate but connected neural networks), i.e. a hidden layer and an output layer. The hidden layer consists of a Self-Organizing Map (SOM) [15] which is fully connected with forward connections to the output layer. The output layer consists of a grid of neurons that are adapted by the delta rule to get an activity that converges to the provided desired output.

2.3.1 The Hidden Layer

The hidden layer consists of a version of the SOM that works as follows. It consists of an $I \times J$ grid of a fixed number of neurons and a fixed topology. Each neuron n_{ij} is associated with a weight vector $w_{ij} \in R^n$. All the elements of the weight vectors are initialized by real numbers randomly selected from a uniform distribution between 0 and 1, after which all the weight vectors are normalized, i.e. turned into unit vectors.

At time t each neuron n_{ij} receives an input vector $x(t) \in R^n$.

The net input s_{ij} is calculated using the standard cosine metric

$$s_{ij}(t) = \frac{x(t) \cdot w_{ij}(t)}{||x(t)|| ||w_{ij}(t)||}, \tag{8}$$

The activity y_{ij} in the neuron n_{ij} is calculated by using the softmax function [4]

$$y_{ij}(t) = \frac{(s_{ij}(t))^m}{\max_{uv} (s_{uv}(t))^m} \tag{9}$$

where u and v ranges over the rows and the columns of the neural network and m is the softmax exponent.
The neuron c associated with the weight vector $w_c(t)$ most similar to the input vector $x(t)$, i.e. the neuron with the strongest activation, is selected:

$$c = \arg\max_c\{|x(t) \cdot w_c(t)|\} \tag{10}$$

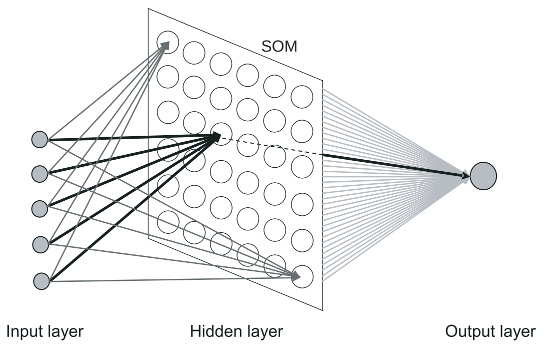


Figure 5: Supervised SOM Based Architecture. For simplicity reasons, the output layer is shown with only one neuron as in the case of the MLP and the RBF networks. However, in reality we use as many neurons in the output layer as the problem has classes. For example, one of the databases in our experiments section, the iris database, has 3 classes and in that case the output layer therefore has three neurons.

The weights w_{ijk} are adapted by

$$w_{ijk}(t+1) = w_{ijk}(t) + \alpha(t)G_{ijc}(t)\left[x_k(t) - w_{ijk}(t)\right] \tag{11}$$

where $0 \leq \alpha(t) \leq 1$ is the adaptation strength with $\alpha(t) \rightarrow 0$ when $t \rightarrow \infty$.

The neighbourhood function is:

$$G_{ijc}(t) = e^{-\frac{||r_c - r_{ij}||}{2\sigma^2(t)}} \tag{12}$$

where $r_c \in R^2$ and $r_{ij} \in R^2$ are location vectors of neurons c and n_{ij}, G is a Gaussian function decreasing with time. σ is the neighbourhood radius which at time t is updated by multiplying σ at time $t-1$ with 0.99 as it is showed in table 3.

All weights $w_{ijk}(t)$ are normalized after each adaptation.

2.3.2 The Output Layer

The output layer consists of an $I \times J$ grid of a fixed number of neurons and a fixed topology. Each neuron n_{ij} is associated with a weight vector $w_{ij} \in R^n$. All the elements of the weight vector are initialized by real numbers randomly selected from a uniform distribution between 0 and 1, after which the weight vector is normalized, i.e. turned into unit vectors.

At time t each neuron n_{ij} receives an input vector $x(t) \in R^n$.

The activity y_{ij} in the neuron n_{ij} is calculated using the standard cosine metric

$$y_{ij}(t) = \frac{x(t) \cdot w_{ij}(t)}{||x(t)||||w_{ij}(t)||}, \tag{13}$$

During the learning phase the weights w_{ijl}, are adapted by

$$w_{ijl}(t+1) = w_{ijl}(t) + \beta x_l(t)\left[d_{ij}(t) - y_{ij}(t)\right] \tag{14}$$

where β is the adaptation strength and $d_{ij}(t)$ is the desired activity for the neuron n_{ij}.

3 Experimentation

3.1 Methods

The databases used in this study are Breast Cancer, Parkinson and Iris. These databases are taken from the University of California at Irvine (UCI) machine learning repository [1] [17] [16] and are used for training and testing in the experiments. The main reason to use these particular datasets is that they are well known to professionals of artificial intelligence.

We have used Matlab and in particular the Neural Network toolbox for our experimentation with MLP and RBF. The reason for using matlab is due to the wide scope of problems addressed and the effectiveness conducted with these [19] [26] [8]. The SOM Based Supervised Architecture has been implemented under Ikaros [2].

The method to evaluate the three methods is to obtain some measures as classification accuracy, sensitivity, specificity, positive predictive value, negative predictive value and a confusion matrix. A confusion matrix [14] contains information about actual and predicted classifications done by a classification system.

Moreover, we have also evaluated the learning process regarding how fast every method learns.

For the construction of the MLP architecture we have proceeded as follows:

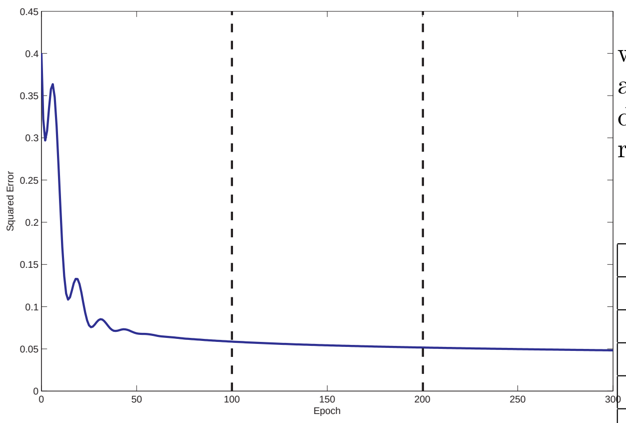


Figure 6: Learning speed in the MLP is slow since it uses the backpropagation method. Generally the backpropagation method always provides a very high precision. The drawbacks are the slowness in learning as well as the risk of over fitting the data learned.

a) Layer 1 corresponds directly to the input vector, that is, all the parameters/fields of the input record.
b) Layer 2 (the hidden layer). The number of hidden neurons for this layer is the most elaborated question in the network's architecture. This number represents a trade off between performance and the risk of over fitting. In fact, the number of neurons in a hidden layer will significantly influence the ability of the network to generalize from the training data to unknown examples [22]. By doing some experiments we discovered that:

- With a low number of neurons for this layer the training and test sets performed badly;
- With a high number of neurons the training set performed good. However there is a high risk of over fitting;
- The optimal solution for this layer has been found to be 24 neurons for Breast Cancer, 12 neurons for Parkinson and 5 neurons for Iris.

c) Layer 3 (the output layer) (Classification). It has two outputs for Breast Cancer and Parkinson and three outputs for Iris.

The learning algorithm used is backpropagation with adaptive learning rate, constant momentum and an optimized algorithm based on the gradient descent method. The backpropagation training parameters are showed in table 1.

Table 1: Backpropagation training parameters.

Parameters	Value
Learning rate	0.01
Adaptive learning rate	0.1
Constant momentum	0.2
Epochs	100-1000-10000
Minimum performance gradient	$\frac{1}{e^{-10}}$

The main parameter we must adjust in order to get a good accuracy with an RBF network is the maximum number of centres. This is a parameter of the center selection algorithm, and is the maximum number of centers/RBFs that is chosen.

We followed the recommendation to set an upper limit between 60% and 70% for the proportion between the number of RBFs and the number of neurons in the input layer. The parameter spread, which is the spread of radial basis functions, helps to construct the hidden layer. The larger the spread is, the smoother the function approximation will be. Spread value represents a compromise between a low value which means low accuracy and a high value with over fitting risk and the possibility that the network may not generalize well.

The RBF network training parameters are shown in table 2.

Table 2: RBF network training parameters.

Parameters	Value
Learning rate	0.01
Adaptive learning rate	0.1
Spread	0.8
Epochs	100-1000-10000
Minimum performance gradient	$\frac{1}{e^{-10}}$

The SOM Based Supervised Architecture training parameters are showed in table 3.

Frequently, the complete data set is divided into two subsets: the training set and the test set. Here, the training set is used to determine the system parameters, and the test set is used to evaluate the diagnosis accuracy and the network general-

Table 3: SOM Based Supervised Architecture.

Parameters	Value
Learning rate	0.1
Learning rate Decay	0.99
Learning rate Minimum	0.01
Neighbourhood Radius (σ)	15
Neighbourhood Decay	0.99
Neighbourhood Minimum	1
Adaptation Strength	0.35
Epochs	100-1000-10000
Minimum performance gradient	$\frac{1}{e^{-10}}$

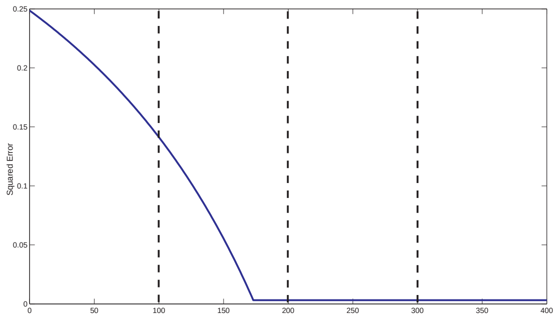


Figure 7: Learning speed in RBF is faster than in the MLP. This is because the RBF network already has some centroids defined due to predefined training which saves time.

ization. Cross-validation has been widely used to assess the generalization of a network. The cross-validation estimate of accuracy determining by the overall number of correct classifications divided by the total number of examples in the dataset.

$$Acc_{cv} = \frac{1}{n} \sum_{x_i \in S} \delta(I(S_i, x_i), y_i) \tag{15}$$

where n is the size of the dataset S, x_i is the example of S, y_i is the target of x_i, and S_i is the probable target of x_i by the classifier. Therefore:

$$\delta(i,j) = \begin{cases} 1 & if \quad i \in N_c(t) \\ 0 & otherwise \end{cases} \tag{16}$$

Specifically, for this study we have applied a five-fold cross-validation method for the performance assessment of every network. The data has been divided in five sets (S1, S2, S3, S4, S5) and the five experiments performed were:
Experiment 1 - Training: S1, S2, S3, S4; Test: S5
Experiment 2 - Training: S1, S2, S3, S5; Test: S4
Experiment 3 - Training: S1, S2, S4, S5; Test: S3
Experiment 4 - Training: S1, S3, S4, S5; Test: S2
Experiment 5 - Training: S2, S3, S4, S5; Test: S1

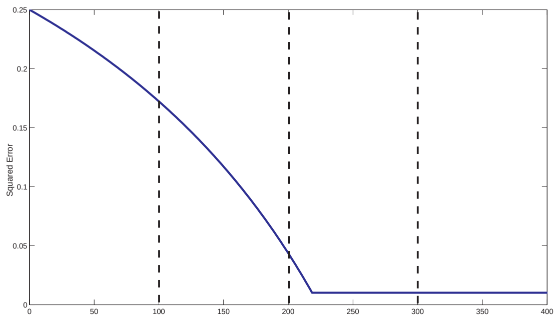


Figure 8: Learning speed in the SOM Based Supervised Architecture is similar as for the RBF network and it is faster than the learning speed of the MLP.

The sets of data used for the process of constructing the model (the training data) were of 565, 195 and 150 registers for Breast Cancer, Parkinson and Iris respectively. The other set of data used to validate the model (the test data) was of 113, 39 and 30 registers also for Breast Cancer, Parkinson and Iris respectively. The test data are chosen randomly from the initial data and the remaining data form the training data. The method is called 5-fold cross validation since this process has been performed five times. The function approximation fits a function using the training set only. Then the function approximation is asked to predict the output values for the data in the testing set. The errors it makes are accumulated to provide the mean absolute test set error, which is used to evaluate the model. The results are presented using confusion matrices.

3.2 Results

Table 4 shows the confusion matrix for all the classifiers with a two classes problem: Breast Cancer database. Classification accuracy, sensitivity, specificity, positive predictive value and negative predictive value can be defined (all the equations 17-21 show 5 values for MLP, RBF and the SOM based architecture respectively) by using the elements of the confusion matrix (table 4).

Table 4: Definition of the confusion matrix with the value for every measure for the MLP, the RBF and the SOM Based Supervised Architecture classifiers with the Breast Cancer database. It has two classes: Possitive (P) and Negative (N). True positive (TP); False negative (FN); False positive (FP); True negative (TN)

	MLP			RBF			SOM	
Act	Predicted			Predicted			Predicted	
	P	N		P	N		P	N
P	184	7		151	56		165	35
	(TP)	(FN)						
N	13	296		46	247		30	268
N	(FP)	(TN)						

$$Classification \quad accuracy(\%) = \frac{TP+TN}{TP+FP+FN+TN} \text{x100} = 96\%, 79.6\%, 86.9\% \quad (17)$$

$$Sensitivity(\%) = \frac{TP}{TP+FN} \text{x100} = 96.3\%, 72.9\%, 82.5\% \quad (18)$$

$$Specificity(\%) = \frac{TN}{FP+TN} \text{x100} = 95.8\%, 84.3\%, 89.9\% \quad (19)$$

$$Positive \quad predictive \quad value(\%) = \frac{TP}{TP+FP} \text{x100} = 93.4\%, 76.6\%, 84.6\% \quad (20)$$

$$Negative \quad predictive \quad value(\%) = \frac{TN}{FN+TN} \text{x100} = 97.7\%, 81.5\%, 88.4\% \quad (21)$$

Table 5 shows the confusion matrix for all the classifiers with a two classes problem: Parkinson database. Classification accuracy, sensitivity, specificity, positive predictive value and negative predictive value can be defined (all the equations 22-26 show 5 values for MLP, RBF and SOM respectively) by using the elements of the confusion matrix (table 5).

$$Classification \quad accuracy(\%) = \frac{TP+TN}{TP+FP+FN+TN} \text{x100} = 84.6\%, 82.1\%, 81.5\% \quad (22)$$

Table 5: Definition of the confusion matrix with the value for every measure for the MLP, RBF and SOM classifiers with the Parkinson database. It has two classes: Possitive (P) and Negative (N). True positive (TP); False negative (FN); False positive (FP); True negative (TN)

	MLP			RBF			SOM	
Actual	Predicted			Predicted			Predicted	
	P	N		P	N		P	N
P	132	15		147	35		129	18
N	15	33		0	13		18	30

$$Sensitivity(\%) = \frac{TP}{TP+FN} \text{x100} = 89.8\%, 80.8\%, 87.8\% \quad (23)$$

$$Specificity(\%) = \frac{TN}{FP+TN} \text{x100} = 68.8\%, 100\%, 62.5\% \quad (24)$$

$$Positive \quad predictive \quad value(\%) = \frac{TP}{TP+FP} \text{x100} = 89.8\%, 100\%, 87.8\% \quad (25)$$

$$Negative \quad predictive \quad value(\%) = \frac{TN}{FN+TN} \text{x100} = 68.8\%, 27.1\%, 62.5\% \quad (26)$$

Table 6 shows the confusion matrix for all the classifiers with a three classes problem: Iris database. Since it is a different classification problem from the two previous examples, with three output, we only show the equation of Classification accuracy (with 3 values for MLP, RBF and SOM respectively) by using the elements of the confusion matrix (table 6).

$$Classification \quad accuracy(\%) = \frac{TP+TN}{TP+FP+FN+TN} \text{x100} = 91.7\%, 73.3\%, 87.6\% \quad (27)$$

4 Conclusion

In this paper we have evaluated the performance of the SOM based supervised architecture. To evaluate the effectiveness of this ANN architecture, we compare it with MLP and RBF networks in classification tasks. The supervised SOM based architecture has similar characteristics as the RBF network,

Table 6: Definition of the confusion matrix with the value for every measure for the MLP, RBF and SOM classifiers with the Iris database. It has three classes: Iris-virginica (A), Iris-versicolor (B) and Iris-setosa (C)

	MLP			RBF			SOM		
	Predicted			Predicted			Predicted		
	A	B	C	A	B	C	A	B	C
A	41	6	0	37	3	10	41	7	0
B	6	39	0	9	42	9	9	43	0
C	3	0	50	4	5	31	0	0	50

and could indeed be seen as an RBF network that automatically finds a suitable number of and suitable locations of RBFs in its hidden layer. Thus a fundamental aspect of this ANN architecture is the use of a SOM as hidden layer. An important aspect of the SOM based architecture is that it helps the designer to get rid of the difficulty and cost of the design of the hidden layers.

The results presented by these three methods (MLP, RBF and SOM based supervised architecture) achieve a high precision level of the confusion matrix regarding the different measurement parameters (accuracy, sensitivity, specificity, positive predictive value and negative predictive value).

With the Breast Cancer database the accuracy of the MLP, RBF and SOM based supervised architecture were very good, especially MLP which showed a high degree of certainty of 96%.

The SOM based architecture accomplish better results than the RBF network. This can be observed not only in the values of the classification accuracy but also in the rest of them. In the case of sensitivity there was a difference of around 10% between the RBF network and the SOM based supervised architecture.

These results as well as those with the Parkinson's and the iris databases are very encouraging because the SOM based supervised architecture is usually better than the RBF and even if its accuracy is a bit lower than the MLP, it learns faster than the latter.

Furthermore, some of the parameters with the SOM based architecture reach very high accuracy such as "Classification accuracy", "Sensitivity" and "Negative predictive value".

The advantages of the supervised SOM architecture are based on both the accuracy, which is not far behind that of the MLP, and especially the faster learning. These benefits will be recommended for use either in problems with a lot of data or with many attributes, where data relationships may be complex. A future line would be to apply this method in such problems as an iterative process leading to features reduction in order to simplify the dependency relationships.

5 Acknowledgment

We want to express our acknowledgements to Asuncion, A. & Newman, D.J, Irvine, CA: University of California, School of Information and Computer Science, who has created the database.

References

[1] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[2] C. Balkenius, J. Morén, B. Johansson, and M. Johnsson. Ikaros: Building cognitive models for robots. *Advanced Engineering Informatics*, 24(1):40 – 48, 2010.

[3] C. Bishop. Improving the generalization properties of radial basis function neural networks. *Neural Computation*, 3(4):579–588, 1991.

[4] CM Bishop. Neural networks for pattern recognition: Oxford University Press. *New York, New York, USA*, 1995.

[5] C.M. Bishop. *Neural networks for pattern recognition*. Oxford Univ Pr, 2005.

[6] Fi-John Chang, Li-Chiu Chang, Huey-Shan Kao, and Gwo-Ru Wu. Assessing the effort of meteorological variables for evaporation estimation by self-organizing map neural network. *Journal of Hydrology*, In Press, Corrected Proof:–, 2010.

[7] B. Choi and J.H. Lee. Comparison of generalization ability on solving differential equations using backpropagation and reformulated radial basis function networks. *Neurocomputing*, 2009.

[8] A.E. Diaz and C.A. Hernandez. Tool for the design and implementation of control systems with neural networks. In *IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009. ICIS 2009*, volume 1, 2009.

[9] D. Gil, M. Johnsson, J.M. Garcia Chamizo, A.S. Paya, and D.R. Fernandez. Application of artificial neural networks in the diagnosis of urological dysfunctions. *Expert Systems with Applications*, 36(3P2):5754–5760, 2009.

[10] David Gil and Magnus Johnsson. Using support vector machines in diagnoses of urological dysfunctions. *Expert Systems with Applications*, 37(6):4713 – 4718, 2010.

[11] S. Haykin. Neural Networks: A Comprehensive Foundation, Englewoods Cliffs, 1998.

[12] S. Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall, 2008.

[13] C.N. Kim, K.H. Yang, and J. Kim. Human decision-making behavior and modeling effects. *Decision Support Systems*, 45(3):517–527, 2008.

[14] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning*, 30(2/3):271–274, 1998.

[15] T. Kohonen. *Self-Organization and Associative Memory.* Springer Verlag, 1988.

[16] M.A. Little, P.E. McSharry, E.J. Hunter, J. Spielman, and L.O. Ramig. Suitability of dysphonia measurements for telemonitoring of Parkinson? s disease. *IEEE transactions on bio-medical engineering*, 2008.

[17] M.A. Little, P.E. McSharry, S.J. Roberts, D.A.E. Costello, and I.M. Moroz. Exploiting Nonlinear recurrence and Fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, 6(1):23, 2007.

[18] J.V. Marcos, R. Hornero, D. Alvarez, F. del Campo, M. Lopez, and C. Zamarron. Radial basis function classifiers to help in the diagnosis of the obstructive sleep apnoea syndrome from nocturnal oximetry. *Medical and Biological Engineering and Computing*, 46(4):323–332, 2008.

[19] I. MathWorks. *MATLAB & SIMULINK Instrument Control Toolbox 2.* The MathWorks Inc., 2007.

[20] S. Mehrabi, M. Maghsoudloo, H. Arabalibeik, R. Noormand, and Y. Nozari. Application of multilayer perceptron and radial basis function neural networks in differentiating between chronic obstructive pulmonary and congestive heart failure diseases. *Expert Systems With Applications*, 36(3P2):6956–6959, 2009.

[21] M.R. Narasinga Rao, GR Sridhar, K. Madhu, and A.A. Rao. A clinical decision support system using multi-layer perceptron neural network to predict quality of life in diabetes. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, 2009.

[22] M. Pal and University of Nottingham (GB). *Factors Influencing the Accuracy of Remote Sensing Classification: A Comparative Study.* University of Nottingham, 2002.

[23] P. Piela. Introduction to Self-Organizing Maps Modelling for Imputation-Techniques and Technology. *Research in Official Statistics*, 2:5–19, 2002.

[24] B.D. Ripley. *Pattern recognition and neural networks.* Cambridge university press, 1996.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[26] F. Temurtas. A comparative study on thyroid disease diagnosis using neural networks. *Expert Systems with Applications*, 36(1):944–949, 2009.

[27] H. Werner and M. Obach. New neural network types estimating the accuracy of response for ecological modelling. *Ecological Modelling*, 146(1-3):289–298, 2001.

[28] H. Yan, Y. Jiang, J. Zheng, C. Peng, and Q. Li. A multilayer perceptron-based medical decision support system for heart disease diagnosis. *Expert Systems with Applications*, 30(2):272–281, 2006.

Stream-Based Reasoning Support for Autonomous Systems [*]

Fredrik Heintz, Jonas Kvarnström, and Patrick Doherty
Dept. of Computer and Information Science, Linköping University,
581 83 Linköping, Sweden
{frehe, jonkv, patdo}@ida.liu.se

For autonomous systems such as unmanned aerial vehicles to successfully perform complex missions, a great deal of embedded reasoning is required at varying levels of abstraction. To support the integration and use of diverse reasoning modules we have developed DyKnow, a stream-based knowledge processing middleware framework. By using streams, DyKnow captures the incremental nature of sensor data and supports the continuous reasoning necessary to react to rapid changes in the environment.

DyKnow has a formal basis and pragmatically deals with many of the architectural issues which arise in autonomous systems. This includes a systematic stream-based method for handling the sense-reasoning gap, caused by the wide difference in abstraction levels between the noisy data generally available from sensors and the symbolic, semantically meaningful information required by many high-level reasoning modules. As concrete examples, stream-based support for anchoring and planning are presented.

Accepted to ECAI'2010

[*]This work is partially supported by grants from the Swedish Foundation for Strategic Research (SSF) Strategic Research Center MOVIII, the Swedish Research Council (VR) Linnaeus Center CADICS, and the Center for Industrial Information Technology CENIIT (projects 06.09 and 10.04).

Making a Map-making Robot:
Using the IKAROS System to Implement the Occupancy Grid Algorithm.

Rasmus Bååth, Birger Johansson
Lund University Cognitive Science
Kungshuset, Lundagård, 222 22 Lund
rasmus.baath@lucs.lu.se, birger.johansson@lucs.lu.se

Abstract

This paper describes an implementation of the *occupancy grid algorithm*, one of the most popular algorithms for robotic mapping. The algorithm is implemented on a robot setup at Lund University Cognitive Science (LUCS), and a number of experiments are conducted where the algorithm is exposed to different kinds of noise. The outcome show that the algorithm performs well given its parameters are tuned right. The conclusion is made that, in spite of its limitations, the occupancy grid map algorithm is a robust algorithm that works well in practice.

1 Introduction

Maps are extremely useful artifacts. A map helps us relate to places we have never been to and shows us the way if we decide we want to go there. For an autonomous robot a map is even more useful as it could, if it is detailed enough, serve as the robot's internal representation of the world. The field of robotic mapping is quite young and started to receive attention first in the early 80s. Since then a lot of effort has gone into constructing robust robotic mapping algorithms, but the challenge is great as the way a human intuitively would build a map can not be directly applicable to a robot. Whereas a human possesses superior vision sensors and can locate herself by identifying landmarks, a robot, most often, only have sensors that approximates the distance to the closest walls. The conditions of robotic mapping actually closer resembles the conditions for a 15th century ship mapping uncharted water. Similar to the ship the robot only knows the approximate distance to the closest obstacles, it could happen that all obstacles are so far away that the robot senses void and it is often difficult for the robot to keep track of its position and heading. As opposed to the ship, a robot using a faulty map will bump into walls in a disgraceful manner, while the ship, on the other hand, might discover America.

A long-standing goal of AI and robotics research has been to construct truly autonomous robot's, capable of reasoning about and interacting with their environment. It is hard to see how this could be realized without general robust mapping algorithms.

1.1 The Approach of this Paper

This paper describes an implementation of a map building algorithm for a robot setup at LUCS. The main characteristics of the robot setup are that the environment is static and that the pose is given, therefore it does not induce all the difficulties mentioned above. The given pose is not without noise but there will never be the problem with cumulative position noise. Even if the problem is eased it is still far from trivial thus interesting in its own right. The setup will be further described in section 2. Given these precondition the *occupancy grid map* algorithm, first described by Elfes and Moravec [3], was chosen. The occupancy grid map algorithm was implemented and a number of experiments were conducted to investigate how it would perform given different types of sensor noise. The results of the experiments are presented in section 3.2.

1.2 The Occupancy Grid Map Algorithm

The occupancy grid map algorithm was developed in the mid 80s by Efes and Moravec and is a *recursive Bayesian estimation* algorithm. Here recursive means that in order integrate an nth sensor reading into a map no history of sensor readings is necessary. This is a useful property which implies that sensor readings can be integrated online and that the space and time complexity is constant with respect to the number of sensor readings. The algorithm is Bayesian because the central update equation is based on *Bayes theorem*:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

which answers the question "what is the probability of A given B", if we know the probabilities $P(B|A), P(A)$ and $P(B)$.

The map data structure is a grid, in 2D or 3D, that represents a region in space. This paper will treat the 2D case, thus the region is a rectangle. The value of each cell of the grid is the estimated probability that the corresponding area in space is occupied. The region corresponding to a cell is always considered completely occupied or completely empty. One can have different definitions regarding whether a region is free or occupied, but often a region is considered occupied if any part of it is occupied.

The algorithm consists of two separate parts: the update equation and a sensor model. The update equation is the basis of the algorithm and does not have to change for different robot setups. The sensor model on the other hand depends on the robot setup and each robot setup requires a customized sensor model. One can construct sensor models in many ways but the basic approach is described in section 1.3.

The computational complexity of the algorithm depends on the implementation of the sensor model. Apart from that, each update loop have time complexity $O(n'm')$, where n' and m' are the number of columns and rows of the grid that are affected by the current sensor reading. The space complexity is $O(nm)$ where n and m are the total number of columns and rows of the grid. An accessible introduction to occupancy grid maps is given by Elfes [2].

The original algorithm is limited in several ways. It requires that the robot's pose is given, thus it can not rely solely on the odometry of the robot. It presumes a static environment or requires sensor readings where dynamic obstacles have been filtered. Finally the area to be mapped has to be specified in advance. This might sound like severe limitations but in many robot setups one can assume a static environment and that there is a way to deduce the robot's pose. The original algorithm has also been successfully extended to deal with e.g. unknown robot poses [7].

1.3 The Inverse Sensor Model

A *sensor model* is a procedure for calculating the probability $P(s_t|m, p_t)$, that is the probability to get sensor reading s_t given map m and pose p_t at time t. Therefore it follows that the procedure for calculating $P(m|s_t, p_t)$ is called an inverse sensor model, that is the probability of m given only one sensor reading. An inverse sensor model can be though of as function $\texttt{ism}(s_t, p_t)$ that returns a grid the size of g where the probabilities of $P(m|s_{t,}p_t)$ are imprinted. There is not only one correct way to construct $\texttt{ism}(s_t, p_t)$ for a given sensor, different approaches have different advantages.

An example of how the output of an inverse sensor model could look is given in figure 1.

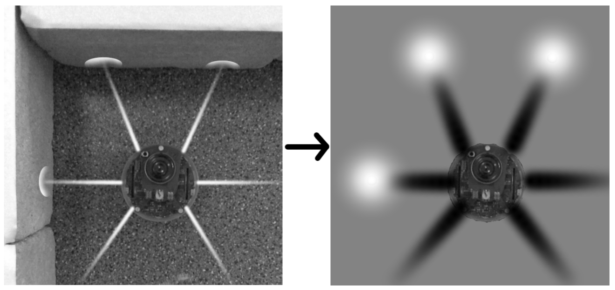


Figure 1: Illustration of an inverse sensor model for a robot equipped with infra-red proximity sensors.

The picture to the left show what the robot senses. The picture to the right is the resulting occupational probabilities. White denotes occupied space,black denotes free space and gray denotes unknown space. Notice how the black strokes fade with the distance to the robot. This indicates that the probability that a sensor detects an obstacle decreases with the distance to the obstacle.

An inverse sensor model can be built by hand or learned, for an example of the first see Elfes and Moravec [3] or the one described in section 2.1.5, for an example of the latter see Thrun et al. [6].

2 Implementation

In order to understand the design choices made a description of the robot setup will first be given, then the implementation will be described. The setup is currently used in the ongoing research regarding robot attention and one purpose of the implementation was that it should be possible to use in this context.

2.1 The Robot Setup

The robot used is the *e-puck*, a small, muffin sized robot developed by École Polytechnique Fédérale de Lausanne (*www.e-puck.org*). Its a differential wheeled robot boosting eight infra-red proximity sensors, a camera, accelerometer and Bluetooth connectivity. The e-puck also have very precise step motors to control its wheels. One problem is that no matter how precise the e-pucks odometry is it can not solely be used to determine the robot's poses. Another problem is the proximity sensors of the e-puck. They have very limited range, roughly 10 cm, and are sensitive with respect to light conditions.

In order to remedy these problems a video camera has been placed in the ceiling of room where the robot experiments take place. The robots movements are restricted to a $2 \times 2\ m^2$ "sandbox" and objects in this area have been given color codes. Robots are wearing bright red plastic cups, the floor, the free space, is dark gray and obstacles are white. Images from the camera are processed in order to extract the poses of the robots and an image where only the obstacles are visible. Given this image and a robot's pose a circle sector is cut out of the image, its center being the robot's position and its direction being the robot's heading. By using this as the robot's sensor reading the robot can be treated as if it had a high resolution proximity sensor. The robots are controlled over Bluetooth link.

2.1.1 Ikaros

The whole system is implemented using Ikaros, a multi-purpose framework developed at LUCS. Ikaros is written in C++ and is intended for, among other things, brain modeling and robot control. The central concept in Ikaros is the *module*, and a system built in Ikaros is a collection of connected module's. An Ikaros module is simply put, a collection of inputs and an algorithm that works on these, the result ending up in a number of outputs. A module's inputs and outputs are defined by an Ikaros control file using an XML based language while the algorithm is implemented in C++.

A module's outputs can be connected to other module's inputs and to build a working system in Ikaros you would specify these connection in a control file. In this control file you could also give arguments to the modules. The data that can be transmitted between modules can only be in one format, that is arrays and matrices of floats. An Ikaros system works in discrete time-steps, so called "*ticks*". Each tick every module receives input and produces output.

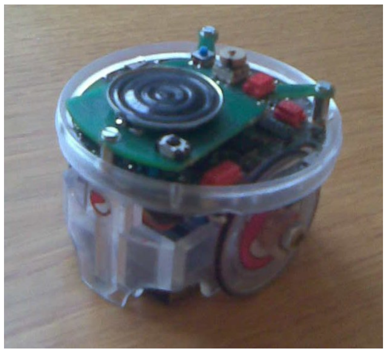


Figure 2: The e-puck.

Ikaros comes with a number of modules, both simple utility modules and more advanced such as several image feature extraction modules. Ikaros also includes a web interface that can display outputs in different ways. For a detailed introduction to Ikaros see Balkenius et al. [1].

2.1.2 Overview of the System

The core of the map drawing system consists of five modules: `Camera`, `Tracker`, `CameraSensor`, `SensorModel` and `OccupancyGridMap`. Further modules could be added to the system, e.g. a path planning module and a robot controller module. The connections between these modules are given in figure 3.

2.1.3 `Camera` and `Tracker`

The `Camera` and `Tracker` modules were already available and will only be described briefly.

The `Camera` module is basically a network camera interface and it is used to fetch images from the camera mounted in the ceiling. It outputs three matrices; `RED`, `GREEN` and `BLUE`, the size of the image, containing the corresponding color intensities of the image.

These matrices are fed into the `Tracker` module that extracts the poses of the robots and the positions of obstacles in the image. It outputs one array `POSITION` with the positions of the robots, one array `HEADING` with the headings of the robots and one matrix `OBSTACLES` with the obstacles extracted from the picture. `POSITION` is of the form $[r1_x, r1_y, r2_x, r2_y \ldots]$ where rn_x and rn_y is the nth robots x and y coordinate receptively. x and y are in the range 0.0 to 1.0 and the origo is in the upper left corner of the image. `HEADING` is of the same form as `POSITION` except for that rn_x and rn_y define a direction vector for the nth robot. The POSITION and `HEADING` will be referred to as the `POSE`. `OBSTACLES` is in the form of an occupancy grid over the area covered by the camera image, where 1.0 denotes an obstacle and 0.0 denotes free space.

2.1.4 `CameraSensor`

The `CameraSensor` module simulates a high resolution proximity sensor. It requires a matrix in the form of Tracker's `OBSTACLES` matrix and an array with the position of a robot as inputs. More specific we

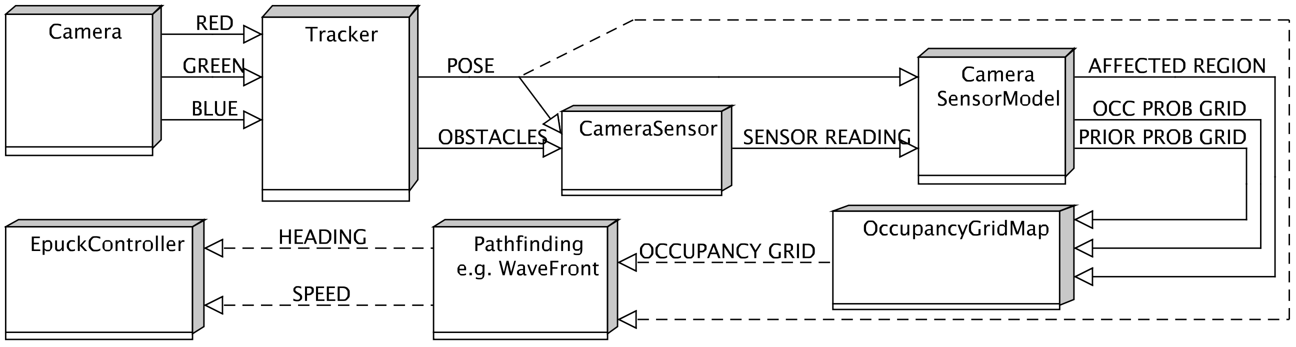


Figure 3: The connections between the modules of the map drawing system, with added path planning and robot control modules.

want to simulate a top mounted stereo camera. The `CameraSensor` module takes arguments specifying he range of the camera and the breadth of the view. Given the pose of the robot a square is cut out of the matrix, this square is rotated and projected onto another matrix representing the `SENSOR READING` of the robot. The `SENSOR READING` shows everything in the cut out square, even obstacles behind walls. Some simple ray-casting will solve this. Rays are shot from the center of the robot to the edge lying on the opposite side of the `SENSOR READING` matrix so that the cells touched by the rays form a circle sector. If a ray hits an obstacle the ray stops and all cells not touched by any ray obtains the value 0.5 indicating it's not part of the sensor reading. `CameraSensor` then outputs `SENSOR READING`.

2.1.5 `CameraSensorModel`

The `CameraSensorModel` is an inverse sensor model tailored to work with the output of the `CameraSensor`. `CameraSensorModel` has two outputs, both required by `OccupancyGridMap`: `AFFECTED GRID REGION` and `OCC PROB GRID`. `OCC PROB GRID` is a matrix the same size as the final occupancy grid that contains the probabilities $P(m|s_t, p_t)$. `AFFECTED GRID REGION` is an array of length four defining a box bounding the area of the occupancy grid that is affected by the `OCC PROB GRID`. The rationale behind this is that `OccupancyGridMap` should not have to update the whole occupancy grid when only a small area of it is affected by the current `SENSOR READING`.

The `SENSOR READING` from `CameraSensor` is already in the format of an occupancy grid, so transforming this into `OCC PROB GRID` in the format the `OccupancyGridMap` module requires, is pretty straight forward. First `OCC PROB GRID` is initialized with $P(m)$, the prior probability, given as an argument to `CameraSensorModel`. Then the `SENSOR READING` is rotated and translated, according to the robot's pose, so that it covers the corresponding area of the `OCC PROB GRID`. The `SENSOR READING` is then imprinted on the `OCC PROB GRID`. The values of `SENSOR READING`; 1.0, 0.5 and 0.0, should not be used directly as they do not correspond to the right probabilities. Instead 0.5 is substituted by the prior probability and 1.0 and 0.0 are substituted by two values `free_prob` and `occ_prob` given as arguments to `CameraSensorModel`. The values of `free_prob` and `occ_prob` should reflect probability that the information in `SENSOR READING` is correct. As the `Camera` and `Tracker` modules are quite exact good values seems to be; `free_prob`$=0.05$ and `occ_prob` $=0.95$. The performance of occupancy grid algorithm depends heavily on these values and they have to be adjusted according to the reliability of `SENSOR READING`. This will be further discussed in section 3.2.

2.1.6 `OccupancyGridMap`

The `OccupancyGridMap` take two inputs in the formats of `OCC PROB GRID` and `AFFECTED GRID REGION`. `OccupancyGridMap` also contains the state of the occupancy grid constructed so far; `MAP GRID`, and the prior probability; `pri_prob`, given as an argument. The `MAP GRID` is initialized by giving each cell the value of `pri_prob`.

The purpose of `OccupancyGridMap` is to update `MAP GRID` using the update equation of the occupancy grid map algorithm. This is done by applying this on all cells in `MAP GRID` that are inside the box defined by `AFFECTED GRID REGION`. Here follows the update equation taken directly from the code:

```
for(int i = affected_grid_region[2];
i <= affected_grid_region[3]; i++)
{
  for(int j = affected_grid_region[0];
  j <= affected_grid_region[1]; j++)
  {
```

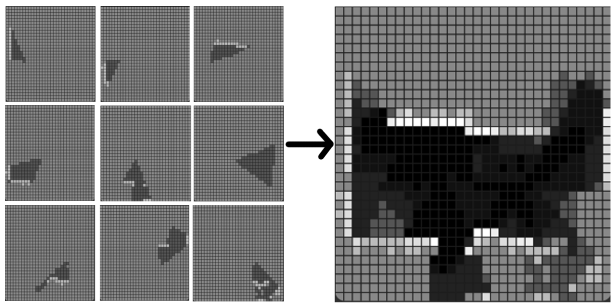


Figure 4: The image to the right shows the probabilities of a number of sensor readings and the image to the left shows the resulting occupancy grid map.

```
        float occ_prob = occ_prob_grid[i][j];
        map_grid[i][j] = 1.0 / (
          1.0 + (1.0 - occ_prob) / occ_prob *
          prior_prob / (1.0 - prior_prob) *
          (1.0 - map_grid[i][j]) / map_grid[i][j]);
      }
    }
```

An example of an how a MAP GRID could look is given in figure 4.

3 Evaluation

The implementation of the occupancy grid algorithm works very well on the robot setup. This is no big surprise as the conditions are ideal, there is practically no sensor noise nor pose uncertainty. In order to investigate how the implementation would handle different conditions a number of experiments were made, where noise was added to the sensor readings. How the implementation reacts to noise is highly dependent on the two parameters of CameraSensorModel; free_prob and occ_prob. Thus for each experiment, except for № 2, three different values of free_prob and occ_prob were used to illustrate this. The following values were used (using the notation [free_prob, occ_prob]): [0.01, 0.99], [0.2, 0.8] and [0.45, 0.55]. These values will be referred to as the *sensor weights,* as they reflect to what degree the occupancy grid map algorithm is persuaded by new sensor readings. All experiments used pri_prob $= 0.5$. The parameters free_prob and occ_prob might seem to be very specific for the CameraSensorModel but any sensor model will have parameters that governs to what degree the sensor readings should be trusted.

3.1 Experiment Setup

The experiments were setup in the following way: A robot was placed in the middle of the 2×2 m^2 "sandbox" and a number of obstacles were placed around it, the result is shown in figure 5 . The "camera" of CameraSensor was given a range of $\sqrt{2}$ m and a breadth of 32°. The robot does not move but each tick the heading of the robot is randomized, in this way the robot will eventually have "seen" the whole "sandbox" visible from the center. Four different experiments were then conducted:

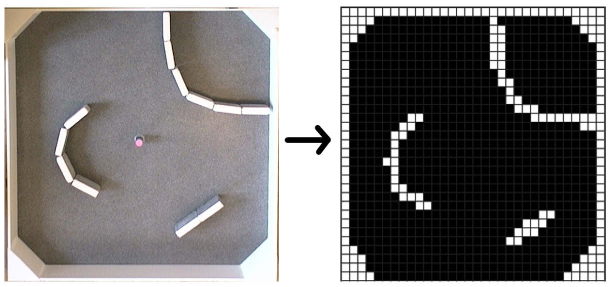


Figure 5: The experiment setup. The real world "sandbox" is to the left and the grid showing the extracted obstacles is to the right.

1. The ideal case. No noise was added, this is to get an measure to compare the other experiments with.

2. Gaussian white noise was added to the OCC PROB GRID of the CameraSensorModel. The noise had a variance of 0.1 and was applied to each cell $\mathtt{OPG}[\mathtt{x},\mathtt{y}]$ in the following way:

$$\mathtt{OPG}[\mathtt{x},\mathtt{y}] = \begin{cases} \text{if } \mathtt{OPG}[\mathtt{x},\mathtt{y}] < \mathtt{pri\_prob} \text{ then} \\ \quad \mathtt{OPG}[\mathtt{x},\mathtt{y}] + \mathtt{abs}(\mathtt{noise}) \\ \text{if } \mathtt{OPG}[\mathtt{x},\mathtt{y}] == \mathtt{pri\_prob} \text{ then} \\ \quad \mathtt{pri\_prob} \\ \text{if } \mathtt{OPG}[\mathtt{x},\mathtt{y}] > \mathtt{pri\_prob} \text{ then} \\ \quad \mathtt{OPG}[\mathtt{x},\mathtt{y}] - \mathtt{abs}(\mathtt{noise}) \end{cases}.$$

   This experiment only uses free_prob=0.0 and occ_prob=1.0.

3. Salt and Pepper noise was added to 40 % of the OCC PROB GRID that represents the current sensor reading. That is, each cell that does no have the value pri_prob is given, by the toss of a coin, one of the values free_prob and occ_prob by a chance of 40%.

4. Gaussian white noise was added to he robot's position given as input to the CameraSensorModel. The noise had a variance of 0.001.

In order to compare the different experiment setups the comparison score measure described in Martin and Moravec [4] was used.

Let I be the ideal map over the same area as a constructed occupancy grid map m. I then only contains the values 1.0, 0.5 and 0.0, where 0.5 indicate that the value of the corresponding cell is unknown. The probability that a cell $m_{x,y}$ represents the same thing as $I_{x,y}$ is $I_{x,y}m_{x,y} + (1 - I_{x,y})(1 - m_{x,y})$. The probability that m represents the same as I is then:

$$\prod_{x,y} (I_{x,y}m_{x,y} + (1 - I_{x,y})(1 - m_{x,y}))$$

A problem is that this value will be very small for large maps. In order to remedy this the log_2 of this value is taken and $|I|$ is added. This results in the following score measure:

$$|I| + log_2 \left(\prod_{x,y} (I_{x,y}m_{x,y} + (1 - I_{x,y})(1 - m_{x,y})) \right)$$

The maximum score of m is $|I|$ minus the number of cells of I that are equal to 0.5. The ideal map was constructed by running experiment № 1 with `free_prob`=0.45 and `occ_prob`=0.55 for 2000 steps. The probabilities of this map was then rounded to the closest of the values 1.0, `pri_prob` and 0.0. Given this ideal map the possible maximum score is 640.

3.2 Results

Generally the implementation performed well in all four experiments but what became obvious is that the choice of sensor weights is important. Each experiment was run for a 1 000 ticks. As all of the experiments contain a randomized component a single run might not produce a characteristic result. To avoid this, each experiment was run ten times and the average of each tick was taken. The result of this is shown in figure 6. When interpretating these charts one should know that a score above 500 corresponds to a reasonably good map. Rather than looking for the sensor weights that eventually results in the best score one should look for the sensor weights that converge fast to a reasonable score. Most often a robot has more use for a good enough map now, that for a perfect map in five minutes. Because of this, the charts only display up to tick 500, even if the maps continue to converge after that.

Experiment № 1

This was the ideal case and as shown in figure 6a the algorithm performs well for both [0.01, 0.99] and [0.2, 0.8]. Even if [0.45, 0.55] surpasses them both eventually, it converges to slow to be practically useful.

Experiment № 2

The outcome of this experiment, as shown in figure 6b, show the strength of the probabilistic approach to robotic mapping. The algorithm handles the noisy sensor readings well and the map converges nearly as fast as [0.2, 0.8] from № 1.

Experiment № 3

Figure 6c show how to high or to low set sensor weight impacts the performance of the algorithm. While [0.2, 0.8] converges nicely, [0.45, 0.55] converges steady but too slow. As [0.01, 0.99] is the most sensible to noise, it converges slowly and never produces a reliable map.

Experiment № 4

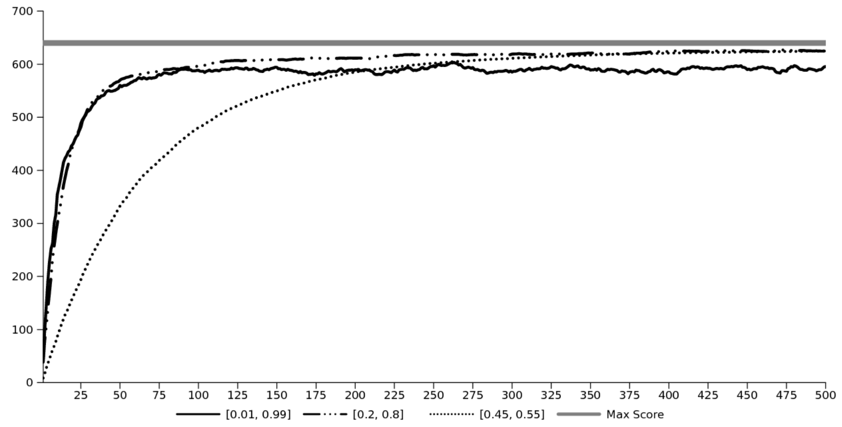
In this last experiment the score measure is a bit misleading. All three choices of sensor weights actually produces acceptable maps. What happens in the case of [0.01, 0.99] is that the edges of the obstacles get slightly displaced, which the score measure penalizes. Even though [0.01, 0.99] of № 3 and № 4 score the same, the map from № 3 is practically unusable, while the map from № 4 is OK.

3.3 Using the Implementation

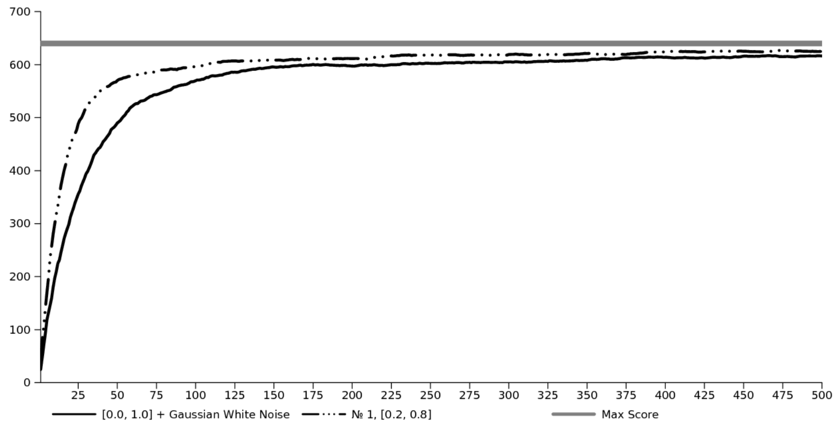
To show that the map drawing implementation can be used in practice an Ikaros system was setup to control an e-puck robot. Basically, this is the system shown in figure 3, including the dashed lines. The goal of the e-puck was to find another e-puck wandering randomly in a maze. The e-puck was not given a path to the other e-puck, only its position. In order to find a path to the other e-puck a wavefront algorithm as described in [5] was used. The e-puck would begin with an empty map, which it would build up gradually as it tried different paths to the other e-puck. Eventually, the map would be complete enough so that the e-puck would find a safe path to the other e-puck.

4 Discussion

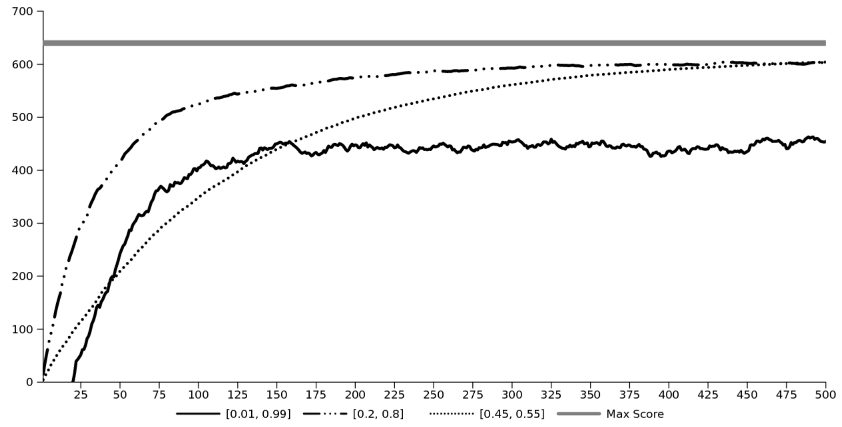
This paper has described an implementation of the occupancy grid map algorithm. This algorithm was implemented to be used with the e-puck robot, using the Ikaros framework. A derivation of the update equation, the basis of the algorithm, was given, as well as a measure for comparing maps. The implementation worked well. This was no surprise as the sensors and the pose tracking system produced very exact information. To investigate how noise would affect the performance of the algorithm a number of



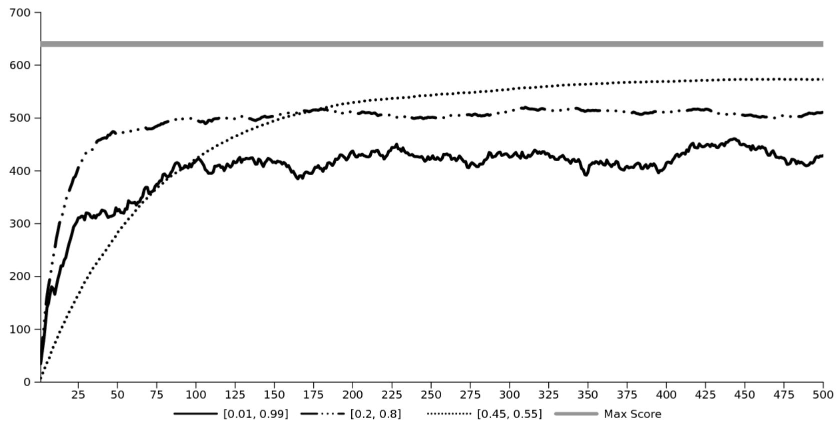
(a) Experiment № 1, the ideal case.



(b) Experiment № 2, gaussian white noise.



(c) Experiment № 3, salt and pepper noise.



(d) Experiment № 4, positional gaussian white noise.

Figure 6: Graphs showing how the estimated map converges to the ideal map given different noise and different sensor weights.

experiments were conducted. Gaussian white noise was applied to the sensors and the pose tracking system, and so called salt and pepper noise was applied to the sensors only. To show that the implementation was usable in practice a system was constructed that made an e-puck draw an occupancy grid map. The e-puck then used this map to find a path to another e-puck wandering randomly.

4.1 Evaluation of the Experiments

Experiment № 1 show that the algorithm works well given ideal preconditions. This is no surprise, but it is important note how the tuning of sensor weights impacts the performance. When the sensor weights are set so that the algorithm put little trust in the sensors, the map converges steadily but unnecessarily slow.

Experiment № 2 and 3 show the strength of the algorithm, its capability to handle independent noise. Both the sensor readings of № 2 and 3 are very noisy, indeed it is often hard for the human eye to separate true obstacles from noise. The algorithm manages this well, given that the sensor weights are set so that the algorithm does not put to much trust in the sensors.

Experiment № 4 show that the algorithm can produce an acceptable map when the position is noisy. The tuning of the sensor weights does not have such an impact as figure 6d might suggest. This is due to the fact that the score measure does not reward correctly identified obstacles that are off by a small distance. One problem with positional noise is that it does not lead to sensor noise that is statistically independent. If the positional noise is to large the algorithm will not be able to handle it no matter how the sensor weights are tuned.

The implementation of the e-puck control system described in section 3.3 worked well in simulation. The two robots steadily moved towards each other, drawing the map and avoiding obstacles as they went along. When trying this with the real robots there were some problems. The `Tracker` module sometimes confused one of the robots for the other one. Also there were some problems communicating with two robots over one Bluetooth connection. Nevertheless, the occupancy grid map algorithm, in combination with the wavefront path planner, always produced a correct path, even if the robot had troubles following it.

4.2 Conclusion

In spite of its limitation the occupancy grid map algorithm is, as this paper has shown, a robust and versatile algorithm. When in need for a robotic mapping algorithm one should have good reasons not to consider using it.

References

[1] C. Balkenius, J. Morén, B. Johansson, and M. Johnsson. Ikaros: Building cognitive models for robots. *Advanced Engineering Informatics*, 24 (1):40–48, 2009.

[2] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. ISSN 0018-9162.

[3] A. Elfes and H. Moravec. High resolution maps fron wide angle sonar. *IEEE International conference on Robotics and Automation*, 1985.

[4] Martin C. Martin and Hans Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

[5] S. Russell and P. Norvig. *Artificial Intelligence - a Modern Approach, 2nd edition.* Prentice Hall, 2002.

[6] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R Murphy, editors, *AI-based Mobile Robots: Case Studies of Successful Robot Systems.* MIT Press, 1998.

[7] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

Integration of the Humanoid Robot Nao inside a Smart Home: A Case Study

Athanasia Louloudi, Ahmed Mosallam, Naresh Marturi, Pieter Janse and Victor Hernandez

School of Science and Technology
Örebro University, Sweden 701 82
`<first name>.<last name>h081@student.oru.se`

April 26, 2010

Abstract

This paper presents a case study demonstrating the integration of the humanoid robotic platform Nao within a Network Robot System (NRS) application. The specific scenario of interest takes place in a smart home environment; the task being that of bringing a can of soda from a fridge to a human user. We use this concrete scenario to evaluate how the performance of such a robot can be affected by being embedded inside an intelligent domestic environment. This study points out that, by cooperating with different components on the network the overall performance of the robot is increased.

Keywords: Network Robotics Systems, Domestic Robots, Nao robot, Smart home, PEIS-Ecology.

1 Introduction

A few years ago, the idea of living with robots, sharing everyday tasks and harmonically coexist in the same environment, seemed to be a distant scenario. However, the use of such technologies, aimed to inhabit our houses and help us with our everyday chores is not a dream any longer. Ongoing research all over the world indicates a trend to develop advanced robotic systems aimed to the service of people in need. According to the International Federation of Robotics, 7.1 million service robots for personal and private use were sold by the end of 2009 and 11.6 million is anticipated to be sold by 2012.[1] The rapid growth in the field of robotics during the last decade provided a strong foundation for smart homes and sensor networks.

A Smart home is a domestic intelligent environment where various components such as a fridge, oven, lights etc., are working together by exchanging information via the same local network. The principal idea behind the smart home concept is to use NRS techniques to integrate different services within the home in an effort to control and monitor the entire living space [1]. NRS can provide robot based services to improve care cost and the quality of life in smart homes. These services are not realized by a single stand-alone robot but by a combination of different elements such as environmental sensors, cameras, laser range scanners and humans communicating and cooperating through a network.

In a stand-alone robot, all the sensorial and computational capabilities are self contained. In the context of a NRS, a stand-alone robot is perceived as part of the ecology itself [2], [3]. Moreover, it can be beneficiated by the flux of information coming from other devices connected to the same network, for example, a

[1] *International Federation of Robotics*, Executive Summary of World Robotics 2009. See `http://www.ifr.org/service-robots/statistics/`

camera mounted to the ceiling can provide a wider view than its own embedded cameras. Network robots are divided into three types: visible robots, unconscious robots, and virtual robots [4], [5]. In this work we consider Nao as a visible robot since it has the role of being the physical interface of the NRS inside the smart home. Furthermore, according to Scopelliti et al., the most preferred robots, are those which are human-friendly in means of appearance, primarily resembling pets or toys [6].

This paper is organized as follows: First, in section 2, the problem formulation is presented, describing the overall tasks and the available tools that endue this work. Then, in section 3 we analyse some specific problems which arose in the implementation of our demonstration, emphasising in particular the contrast between our NRS approach and an alternative, single robot approach. The software architecture follows in section 4. Here, the methodologies and structure followed in order to handle this work are described. Finally concluding remarks close this paper.

2 Problem Formulation

Consider the following scenario:

Neils returns home from a long walk. Soon he enters the living room he taps on the head of Tommy, a humanoid robot and rests on the sofa. Tommy perceives the request, offers to bring a refreshment to Niels and starts moving towards the fridge in the kitchen. While walking, it localizes itself to find its position in the room. When Tommy enters the kitchen, it asks the fridge to open its door and use its gripper to collect a soda can and bring it out. The robot identifies the requested drink, grasps it and returns to deliver the refreshment to Niels.

The above scenario can be decomposed into the following simpler tasks that has to be performed by the robot in order to fulfill the overall goal:

1. Walk towards the fridge
2. Dock the fridge
3. Grasp the drink
4. Carry and hold the drink while walking
5. Deliver the drink to the user

These tasks can be grouped into three modules which are localization, cooperative grasping and the mobility module (explained in later sections of this paper).

The following paragraphs, introduce all the information needed in order to understand the components that surround this project. We describe the NRS infrastructure underlying our smart home, and other details that constitute the robotic platform Nao.

2.1 Available Resources

This section describes all the available tools that can be used in order to successfully accomplish the overall task.

2.1.1 Test Environment

PEIS-Ecology:
The concept of PEIS-Ecology, first introduced by Saffiotti and Broxvall in 2005 [7], is one of the few existing realizations of the notion of network robot system. The name PEIS stands for *physically embedded intelligent systems*. PEIS can be defined as a set of interconnected components, residing in one physical entity which generalizes the notion of robot. Every component that is part of this ecology is called PEIS-component.

The PEIS Ecology model has been implemented in an open source middleware, called the PEIS Kernel to which all the PEIS components are linked. This PEIS-Kernel allows the components to communicate and collaborate with each other in a standardized way. For communication, this middleware establishes a peer-to-peer network and performs dynamic routing of messages between PEIS. All PEIS can cooperate using a *uniform cooperation model*, based on the notion of linking functional components: each participating PEIS can use functionalities from other PEIS in the ecology in order to compensate or to complement its own.

PEIS Home:
The PEIS home[2] is an experimental environment that looks like a typical bachelor apartment which was built to implement the PEIS-Ecology [8]. It consists of a living hall, bedroom and a small kitchen. The PEIS-Home is equipped with communication and computational infrastructure. This study is conducted inside the smart home to make use of various components of PEIS-ecology. Fig. 1 illustrates a few basic views of the home.

The PEIS components available for this work are:

- PEIS-Fridge
  The PEIS-Fridge is a small sized refrigerator with a motorized door, a camera and an attached robotic arm with a gripper. The camera takes images over a shelf in the fridge and in combination with clustering algorithms directs the gripper towards the soda cans. The fridge gripper is able to collect the soda can from the interior space and is able to bring the drink outside.

- PEIS-Cam
  A PEIS-Cam is a component that can provide images from common 2D colour cameras. Several cameras are mounted in discrete areas of the house which can provide a wide view of the living space.

- PEIS-PersonTracking
  This component is a tracking system connected to a set of stereo camera and normal mega pixel camera mounted on the ceiling, capable of tracking multiple persons.

2.1.2 Nao robot

Nao is a humanoid robot equipped with sonar sensors, 2 CMOS cameras and three-fingered robotic hands. It features a multimedia system with 4 microphones and 2 hi-fi speakers for

[2]The PEIS home is developed by the Center for Applied Autonomous Sensor Systems (AASS). See `http://aass.oru.se/~peis/`

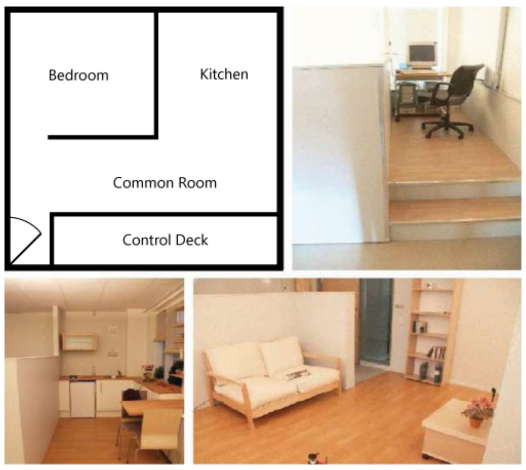


Figure 1: A rough sketch of the PEIS-Home*(upper left)*. A picture of the control deck *(upper right)*. A picture of the kitchen with a fridge located under the workbench *(lower left)*. A picture of the common room*(lower right)*.

voice recognition and text-to-speech synthesis. The built-in functionalities of this robot such as logo detection, mark detection using onboard cameras are loftily adjustable and are used for robot localization. The robotic hands are used for grasping and holding small objects. Nao can carry up to 300g using both hands. [3]

Nao Robocup Edition has 21 degrees of freedom (DOF) whereas Nao Academics Edition has 25 DOF since it is built with two hands with gripping abilities. For this study the academic version of Nao is used.

The Nao is based on Linux and it is a fully programmable robot which uses its own framework called NaoQi. NaoQi, allows the developer to access all the features and functionalities of the robot through an Application Program Interface (API) which also provides the flexibility of executing tasks in sequential order, parallel and event based. The Integrated wireless network card of this robot can be used to exchange information with other devices in the network.

[3]*Nao Documentation.* Aldebaran Robotics. 2009. See `http://academics.aldebaran-robotics.com`

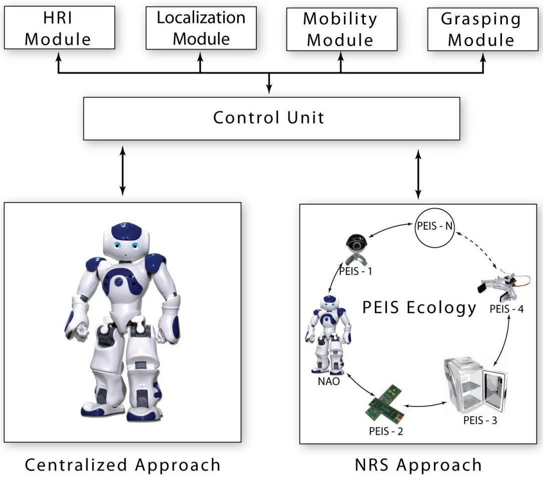


Figure 2: Software Architecture - Centralized and NRS approach

3 The Nao robot inside the PEIS-home

The first attempts to accomplish the scenario explained in section 2 were based on the robot's capabilities (centralized approach shown in Fig. 2), where all the perceptual information of the environment and the tasks are performed by one single agent. Considering that this humanoid platform is quite capable, it was expected to be able to perform successfully the scenario's tasks (localization, grasping and navigation) but several drawbacks revealed through testing. These drawbacks were the main reason for relying on external sources to fulfill the task. The first drawback has to do with localisation. The Nao is equipped with sonars, two cameras and a built-in logo recognition ("ALLandMarkDetection"), a vision module in which Nao recognizes special landmarks with specific patterns on them. Our attempts to leverage these sensory inputs for localisation failed, due to the following reasons:

1. The unreliability of the internal mark detection algorithm led to missing the checkpoint (marks) on the predefined path.
2. Because of the wobbling of Nao while walking the detection of the marks was unreliable.
3. Slight changes in the lighting conditions of the room, could lead to faulty detection or not detection at all.
4. State of the art localisation using stereo vision [12] cannot be implemented using the two onboard cameras since there is no overlapping region between the data acquired by them.

The following section, describes the final implementation. The software architecture will be described and a more detailed illustration of the basic three modules will ensue.

4 Software Architecture

The software architecture is shown in Fig. 2 and was built having in mind the NRS approach which; as explained in [5], has the premise of combining robotics and information communication technologies through a network in order to realize a task or provide services to human users. The software architecture was developed using Aldebaran's NaoQi as a framework and coded in C++. NaoQi gives access to all the features of the robot, like sending commands to the actuators, retrieving information from the robot memory, and managing Wi-Fi connections. This architecture facilates the integration of new components without redesigning the system architecture.

4.1 Control Unit

The control unit is built around a state machine, which provides the logic to arbitrate the interaction between the rest of the software modules and determines the sequence of the actions that should take place in order to achieve the goal described in the introduction of this paper.

Fig. 3 shows the state diagram of the control unit were each state represents an action to be executed and the transitions between states are determined by signals passed to the control unit by the rest of the modules.

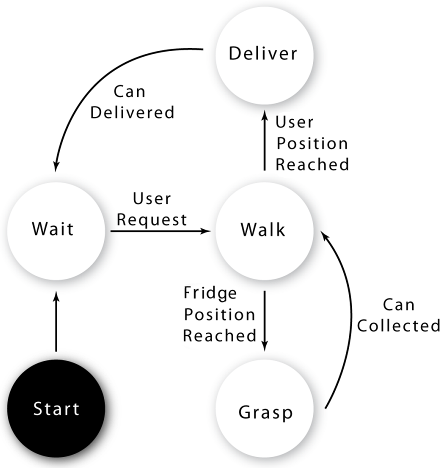


Figure 3: State diagram for the control unit

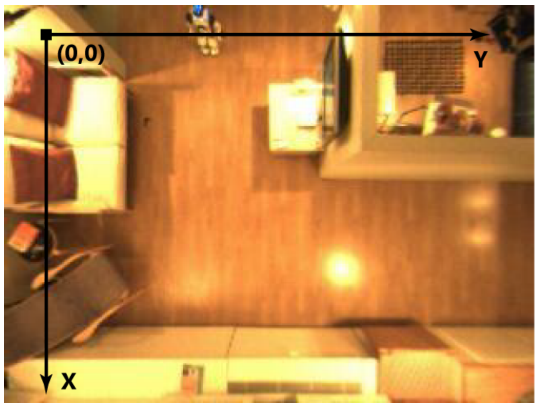


Figure 4: Coordinate frame for the mobility module

4.2 The Mobility Module

The function of this module is to handle the navigation of the Nao robot inside the PEIS home. This module receives as inputs the estimated current pose and the commanded poses through the controller module.

To represent the robots pose in the PEIS home, we have opted for the model suggested in [9], where the degrees of freedom of the joints of the legs and the feet of the robot are ignored and the robot is represented as a rigid body operating in a horizontal plane. The robot pose(RP) is represented as follows:

$$\vec{\mathbf{RP}} = [x, y, \theta]^T \qquad (1)$$

where x and y are the position in the coordinate frame shown in Fig. 4, and θ is the orientation of the robot with respect to the x axis.

The error between the current and the commanded pose is decomposed in an orientation error and a position error, that are later passed through the Nao API's to orient the robot towards the target position and then to move it forward to correct the position error. However, the built in humanoid walking implementation of the Nao robot does not consider the external disturbances that affect the walking schema of the robot. This lack of feedback causes an error between the desired pose and the physical pose of the robot. In order to reduce the impact of this error, the reported pose from the localization module is used to compensate the desired pose of the robot.

The functionality of the mobility module can be summarized in Table 1.

Table 1: PSEUDO-CODE FOR MOBILITY MODULE

```
1  Retrieve the current pose from the localization module.
2  Compute the angle $AT$ between the desired pose and the current pose.
3  Compute the Euclidian distance $DS$ between the desired pose and the current pose.
4  if $AT > anglethreshold$
       Command Nao to turn $AT$ degrees.
5  if ($DS > Distancethreshold$)
       Command Nao to move $DS$ meters forward.
6  When Nao stops moving, repeat steps 1 to 5 to reduce the error in the pose.
7  Wait for the next command from the control unit.
```

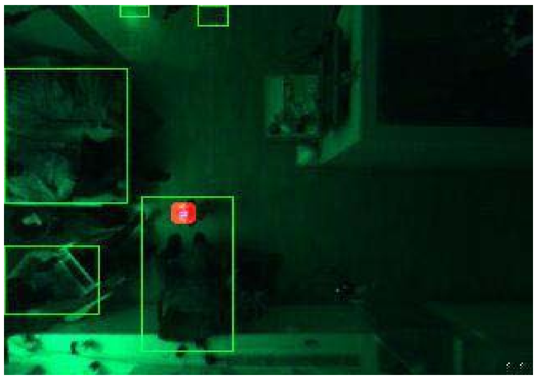


Figure 5: Robot detection and tracking using the background subtraction and color slicing algorithms

4.3 The Localization and Coordination Module

As mentioned before the odometry error between the desired and the actual pose of the robot is present due to external disturbances that affect the robot while walking. In order to reduce the impact of this error, we use an adhoc odometry algorithm that gives the robot a feedback about its actual position. Using already present environmental monitoring cameras placed on the ceiling, background subtraction [10] and color slicing algorithms [11], we could efficiently localize the robot by detecting the blue light on top of its head and provide the mobility module with the actual pose of the robot as shown in Fig. 5. We used this algorithm to calculate the robot position at already known positions and built a lookup table that we used later to interpolate the robot position. The pseudo code in Table 2, presents the basic steps of the algorithm. We did that to decrease the effect of the noisy readings of the robot position due to the robot instability while walking.

The Localization module can be divided into three steps:

1. Background subtraction
2. Color slicing
3. Position lookup table

Table 2: PSEUDO-CODE FOR LOCALIZATION AND COORDINATION MODULE

```
1  Capture background image B”
2  Subtract the new frame from B.
3  for each resulting region:
     if $area > areathreshold$
        add this region to ROI
4  for each ROI:
     for each pixel in the selected ROI
        Calculate the Euclidian distance
        between this pixel and the color
        threshold C”.
          if $C > colorthreshold$
             Set the pixel value to one
          else
             Set the pixel value to zero
5  The region with highest value will be
   the robot pose R”.
6  Calculate the centroid for R and the result will be single pixel P”.
7  The lookup table calculates the robot
   position using P.
```

4.3.1 Background subtraction

In this step the algorithm subtracts each new image from the background image and detects the regions of interest (ROI). ROI area has to be greater than a certain threshold. The threshold has to be set manually according to the ”tolerance” we want to give to our algorithm (a high value of threshold means that only objects with high area value is considered interesting, whereas a lower value of threshold will make the algorithm detect small objects as ROI).

4.3.2 Color slicing

At this point, we consider only the ROI delimited in the previous step. The ROI is processed in the RGB color space, where we calculate the Euclidean distance between the pixel in the ROI and a color of interest. If the distance is less than a certain threshold, the pixel is classified as part of the robot.

4.3.3 Position lookup table

In this final step, we use a lookup table to return the real robot pose. This lookup table provides a mapping from the robot's position in pixels to the real coordinates.

Although this ad-hoc solution is quite simple, the results obtained have significantly reduced the error in the orientation in comparison with the open loop walking schema.

4.4 The Cooperative Grasping Module

The robotic grasping is a very difficult problem, specifically grasping objects that are being seen for the first time through vision. Solving this problem could be through using learning algorithm or building a 3-D model of the object of interests or by using both techniques. In this work we did not apply any learning or modeling approaches, instead we assumed that a specific object is handed in to the robot in a fixed pose. That means whenever the robot reaches the right position for grasping it will start predefined movements in order to grasp with no feedback. The grasping problem was simplified in order to tackle the problems of detecting the presence of the object to be grasped and approaching it. We installed a light source on top of the fridge gripper to be detected in the same way as we detect the Nao robot head. The algorithm detects the robot and the object to be grasped and calculates the Euclidian distance between the robot and the object.

The walking module receives the distance and commands the robot to move. After approaching the grasping point, the robot does the following:

1. Command the fridge to open through the PEIS infrastructure
2. Command the fridge gripper to fetch a certain drink out of the fridge
3. The robot docks the extended fridge gripper to place itself in a position suitable for grasping. This is done by indirect visual servoing through the localisation of the gripper as perceived by the environmental camera
4. The robot then follows predefined arm and hand movements to grasp the object. See Fig. 6

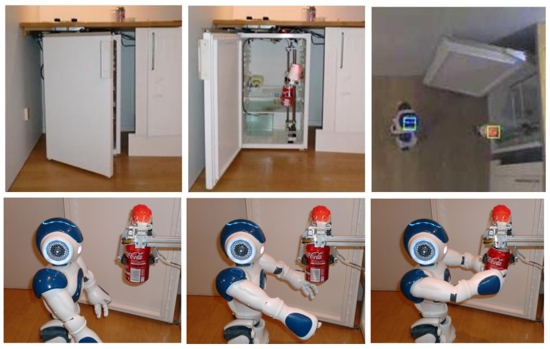


Figure 6: Fridge door opening*(upper left)*, fridge gripper bringing the can out*(upper middle)*, tracked image from the camera *(upper right)*, Pre-defined robot arm movements after reaching the desired position for grasping the object*(bottom left, bottom middle and bottom right)*

The algorithm is explained in Table 3.

4.5 HRI Module

The Human Robot Interface (HRI) module acts as the direct link between the user and the system. Through this module the user is able to request the can of soda by tapping the robots head.

This module will receive the current position (coordinates) of the human user from the PEIS person tracking system. These coordinates in turn will be passed to the mobility module through the control unit as a final destination for the Nao robot.

The position of the user is determined by this module. See Fig. 2. When a centralized approach is used, the task of finding the user is performed by the robot itself (i.e. face recognition), while for the NRS approach, this task is carried out by an external module (i.e. the PEIS person tracker).

Table 3: PSEUDO-CODE FOR COOPERATIVE GRASPING MODULE

```
1   Capture background image B"
2   Subtract the new frame from B.
3   for each resulting region:
        if $area > areathreshold$
            add this region to ROI
4   for each ROI:
        for each pixel in the selected ROI
        Calculate the Euclidian distance
        between this pixel and the color
        thresholds C1 and C2.
            if $C1 > colorthreshold$
                set the pixel value to 1 for the
                robot region RROI
            else if $C2 > colorthreshold$
                Set the pixel value to 1 for the
                object region OROI
            else
                Set the pixel value to 0.
5   In RROI the region with highest value
    will be the robot pose R.
6   In OROI the region with highest value
    will be the object pose O.
7   Calculate the centroid for R and the re-
    sult will be single pixel "PR"
8   Calculate the centroid for R and the re-
    sult will be single pixel "PO"
9   The lookup table calculates the robot
    position and the object position using
    PR and PO.
10  Calculate the Euclidian distance and
    command the robot to approach this
    position.
    if ($Destinationreached$)
        Start the predefined movements of
        the arms.
```

5 System Demonstration

The system was evaluated using two different setups where, inspired by the scenario depicted in section 2, the robot was commanded by the user to follow a predefined path (from the living room to the kitchen) in order to collect and bring back a known object placed in the fridge gripper. Twenty rounds for each setup were performed.

In the first setup, it was required to make the robot locate itself inside the environment without being assisted by any other component inside the PEIS home. We opted for using the on-board vision capabilities of the robot to detect pre defined marks (i.e. Nao Landmark Detection) as shown in Fig. 7 and the marks were placed along the predefined path from the living room to the kitchen and the robot was expected to use them to correct its position and orientation in order to complete one test round. Poor results were obtained due to the noise introduced from embedding the localization module inside one single agent (in this case, the Nao robot), such as the wobbling produced by the walking or in some cases, when the robot missed one mark while moving, the error in the localization was increased in a way that was almost impossible to find the right path to successfully complete the task. This drawbacks lead to a successful rate of only 20% of the rounds.

For the second setup, the robot was assisted by the NRS as described in section 4.3. It was observed that the performance was substantially improved since all the rounds were successfully completed and as a final demonstration of the system, a person who is not familiar with the technical details of this study was taught how to operate and interact with the system.[4] Once again the task was successfully completed. The drawback of this setup is that the system is heavily dependent on the network stability. A failure in the connection or missing information from any of the involved network components may negatively affect the final output.

6 Conclusions

In this work we have successfully integrated a humanoid robot into a NRS to solve a problem in a common day scenario; bringing a can of soda to a human user. This problem involves several tasks such as localization, mobility and

[4]Integrated Project Work demos. See `http://www.youtube.com/user/loutfiamy`

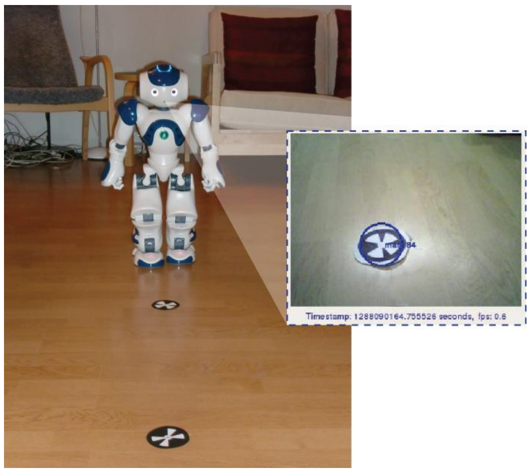


Figure 7: Robot localization using land marks and Land mark detected by Nao*(in sight)*.

cooperative grasping which has to be accomplished in order to successfully achieve the overall goal.

While the Nao is a capable platform; due to the complexity of the problem to be solved and the challenges that home robotics implies, we conclude that the capabilities of the robot can be enhanced and the complexity of the problem can be reduced by decomposing it in simpler tasks executed in cooperation with specialized components in the NRS (i.e. the PEIS Ecology).

The NRS brings also expandability to the Nao itself when new components are added to the network and Nao interacts with them to solve a novel task. The NRS is also beneficiated with the incorporation of Nao, since it became a visible robot that provides different services and acts as an interface between the user and the PEIS Ecology.

Future work may include how to exploit the sensorial capabilities of the Aldebaran's Nao, such as voice and face recognition to allow a more user friendly interface between a smart home and the human user who is requesting different services from the ubiquitous network. This work can also be expanded by using planning and searching techniques to allow the robot to move in non predefined paths and the use of the robot's internal sonars can be used to allow obstacle avoidance.

7 Acknowledgments

This work was supported by AASS (Applied Autonomous Sensory Systems, Örebro University, Sweden) laboratories through the course Integrated Project Work (IPW 2009). Many thanks to all the people at the AASS laboratories for their contributions to this work, especially Dr. Federico Pecora, Dr. Dimitar N. Dimitrov and Todor Stoyanov.

References

[1] A. Sanfeliu, N. Hagita, and A. Saffiotti, "Network robot systems," *Robotics and Autonomous Systems,Elsevier 2008*, 2008.

[2] N. Kubota and K. Nishida, "Cooperative perceptual systems for partner robots based on sensor network," *IJCSNS, International Journal of Computer Science and Network Security*, vol. 6, no. 11, pp. 19–28, 2006.

[3] K. Lynch, I.B.Schwartz, P. Yang, and R. Freeman, "Decentralized environmental modeling by mobile sensor networks," *Robotics, IEEE Transactions on*, vol. 24, pp. 710–724, June 2008.

[4] K. Kemmotsu, T. Tomonaka, S. Shiotani, Y. Koketsu, and M. Iehara, "Recognizing human behaviors with vision sensors in a network robot system," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, (Orlando, FL, United states), pp. 1274 – 1279, 2006.

[5] T. Akimoto and N. Hagita, "Introduction to a network robot system," in *2006 International Symposium on Intelligent Signal Processing and Communications (IEEE Cat. No.06EX1444)*, pp. 91 – 4, 2006.

[6] M. Scopelliti, M. Giuliani, A. D'Amico, and F. Fornara, *I had a robot at home...Peoples' representation of domestic*

robots. In S. Keates, J. Clarkson, P. Langdon and P. Robinson, Designing a more inclusive world. London: Springer Verlag„ 2004.

[7] M. Broxvall, M. Gritti, A. Saffiotti, B. Seo, and Y. Cho, "PEIS ecology: Integrating robots into smart environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, (Orlando, FL), pp. 212–218, 2006. Online at http://www.aass.oru.se/˜asaffio/.

[8] A. Saffiotti and M. Broxvall, "PEIS ecologies: Ambient intelligence meets autonomous robotics," in *Proc of the Int Conf on Smart Objects and Ambient Intelligence (sOc-EUSAI)*, (Grenoble, France), pp. 275–280, 2005. Online at http://www.aass.oru.se/˜asaffio/.

[9] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots.* Bradford Book, 2004.

[10] J. Davis and V. Sharma, "Fusion-based background-subtraction using contour saliency," in *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, (Washington, DC, USA), pp. 162 –182, IEEE Computer Society, 2005.

[11] R. Gonzalez and R.E.Woods, *Digital Image Processing.* Pearson Prentice Hall, 2008.

[12] Don Murray and James J. Little, "Using Real-Time Stereo Vision for Mobile Robot Navigations," *Auton. Robot*, vol. 8, no. 2, pp. 161–171, 2000.

Planning for Loosely Coupled Agents using Partial Order Forward-Chaining

Jonas Kvarnström
Department of Computer and Information Science
Linköping University, SE-58183 Linköping, Sweden (jonkv@ida.liu.se)

Abstract

Partially ordered plan structures are highly suitable for centralized multi-agent planning, where plans should be minimally constrained in terms of precedence between actions performed by different agents. In many cases, however, any given agent will perform its own actions in strict sequence. We take advantage of this fact to develop a hybrid of temporal partial order planning and forward-chaining planning. A sequence of actions is constructed for each agent and linked to other agents' actions by a partially ordered precedence relation as required. When agents are not too tightly coupled, this structure enables the generation of partial but strong information about the state at the end of each agent's action sequence. Such state information can be effectively exploited during search. A prototype planner within this framework has been implemented, using precondition control formulas to guide the search process.

1 Introduction

A major earthquake has struck in the middle of the night, devastating vast parts of the countryside. Injured people are requesting medical assistance, but clearing all roadblocks will take days. There are too few helicopters to immediately transport medical personnel to all known wounded, and calling in pilots will take time. Fortunately, we also have access to a fleet of unmanned aerial vehicles (UAVs) that can rapidly be deployed to send prepared packages of medical supplies to those less seriously wounded. Some are quite small and carry single packages, while others move carriers containing many packages for subsequent distribution. In preparation, a set of ground robots can move packages out of warehouses and possibly onto carriers.

Given the properties of this somewhat dramatic scenario as well as the robotic agents involved, what types of automated planning could we use to generate high-quality plans?

One option would be to rely on distributed cooperative planning techniques, where each agent is responsible for its own actions but also coordinates its local plan with other agents in order to achieve a shared goal.

An alternative is the use of centralized planning, where a single agent generates a complete plan coordinating activities at a higher level before distributing subplans to individual agents. Plan execution then proceeds in a distributed manner, with either centralized or distributed synchronization between agents. This alternative requires all agents to be fully cooperative, which appears reasonable to assume for the application at hand.

Each of these choices has its own advantages. Distributed planning can for example be more flexible, potentially allowing individual agents to renegotiate parts of the plan during execution. Furthermore, it does not require full cooperation between agents. Having a centralized authority can facilitate the generation of high-quality plans and allows a ground operator to approve or modify a complete plan before execution, which may be a requirement in some cases. Thus, each alternative is likely to be better in some situations and is worth pursuing for its own qualities. For the purposes of this paper, we will proceed under the assumption that centralized planning has been chosen.

Returning to the scenario, we see that action durations are not likely to be perfectly predictable, but it is often possible to specify an approximate expected duration for the case where no failures occur during execution. This information should be taken into account during planning by preferring plans that are expected to require less time to execute. We also prefer plans to be minimally constrained in terms of precedence between actions performed by different agents, in order to avoid unnecessary waiting.

To some extent these properties can be treated after plan generation, for example by inferring less constraining precedence relations from a sequential plan [3]. However, this entails hiding important aspects of the domain and of our concept of plan quality from the planner, which can decrease the quality of the final plan. For example, a sequential planner has no concept of which actions can be executed in parallel and might therefore assign all actions to the same agent, as long as this does not lead to using a greater *number* of actions. Parallelizing such plans after the fact is non-trivial: It requires recognizing subplans that can be assigned to other agents in a meaningful manner, selecting suitable agents for reassignment, and generally also replanning for the selected agents, which may not perform tasks in exactly the same manner as the original agent. Similarly, a non-temporal planner might decide to use as few actions as possible even when these actions are very time-consuming. Treating this after plan generation also requires changing the actions in the plan, as opposed to simply adding temporal durations to each action. We would therefore prefer to work directly with a plan structure capable of expressing these aspects, such as a temporal partial-order plan.

Planners generating such plans do not necessarily have to be explicitly aware of agents, as long as they can express mutual exclusion conditions between actions that cannot be executed concurrently. Indeed, the scenario above is quite similar to the standard logistics benchmark domain, where agents are typically modeled as arguments to actions. Therefore it would be possible to generate the required plans using temporal versions of standard partial order causal link (POCL[1]) planners, agent-aware or not.

Partial order causal link planning was initially conceived as a means of increasing the efficiency of plan generation. Through *late commitment*, avoiding "premature commitments to a particular [action] order" [10], less backtracking was required during the search for a plan. Once a solution was generated, the assumption was that it would be executed in sequence by a single agent. Though a number of POCL planners are also able to generate concurrent plans, the desire to delay commitments to action precedence for performance purposes usually remains one of the primary reasons for the use of partial orders.

However, late commitment is far from the only means of improving planning performance. For example, many recent planners build on the use of forward-chaining state-space search, which generates considerably richer information about the state of the world at any given point in a plan compared to POCL planning. This information can then be exploited in state-based heuristics [4, 8] or in the evaluation of domain-specific control formulas [2, 9]. Since the use of state information has led to a number of very successful total-order planners, it would be interesting to also investigate to what extent one can generate rich state information when generating *partially* ordered plans.

We cannot adopt an unmodified version of forward-chaining search, since we do desire flexible plan structures. However, this desire is entirely motivated by the presence of multiple agents whose capability for concurrent plan execution should be utilized to the greatest extent possible. Therefore, an alternative to POCL planning would be to retain partial ordering (and temporal flexibility) between actions executed by *different* agents, while generating the actions for each *individual* agent in sequential temporal order. Each agent-specific action sequence can then be used to generate partial agent-specific states to be used in heuristics or control formulas.

In this paper, we therefore begin our investigations into alternative methods for generating partial-order plans by adopting certain aspects of the standard forward-chaining paradigm in a hybrid *partial order forward-chaining* (POFC) framework that sacrifices some of the positive aspects of late commitment in order to gain the benefits of richer state information.

In Section 2, we discuss the ideas behind the partial order forward-chaining framework and its applicability to centralized multi-agent planning. In Section 3, we go on to present one concrete planner operating within the POFC framework. We then discuss related work in Section 4 and present our conclusions in Section 5.

2 Partial Order Forward-Chaining

Our discussion of the fundamental ideas underlying partial order forward-chaining (POFC) begins with an analysis of common execution constraints for multi-agent plans and how these constraints affect the desired plan structure. We then continue by showing how these ideas and structures allow us to take advantage of richer state information than is generally available to partial order causal link (POCL) planners. This results in a hybrid planning framework taking advantage of certain aspects of forward-chaining in the generation of partial order plans.

A significant degree of variation is possible in terms of the exact plan structures and planning algorithms used within this framework. Consequently, the concepts introduced in this section must be defined at a comparatively high level of abstraction. In the next section we present a detailed definition of one concrete POFC planner, which also provides additional intuitions regarding the high-level framework.

Plan Structures. As noted in the introduction, our interest in the generation of partial order plans is grounded in a desire to support the type of concurrency that is inherent in many execution mechanisms. In particular, centralized planning for multiple agents yields plans that are naturally concurrent: Each agent can generally perform its actions in parallel with other agents, and should not be forced to wait for other agents unless this is required due to causal dependencies, resource limitations, or similar constraints.

In Figure 1, for example, the first two actions of uav4 are independent of the actions of the ground robot robot3. Sequential plans cannot model this fact, since they do not permit concurrency at all. Temporal plans, where each action is assumed to be executed during a specific interval of time, do allow concurrency but are only guaranteed to achieve the goal if actions start and end in the predicted order. Partially ordered plans are more complex to handle, as one must prove during planning that any action order consistent with the partial order will satisfy the goal. On the other hand, the result is a considerably greater degree of flexibility during execution.

However, the fact that there exist actions that can be performed concurrently does not mean that *arbitrary* concur-

[1]We assume a basic familiarity with partial order causal link planning and refer the reader to Weld [12] for an overview of the associated concepts and terminology.

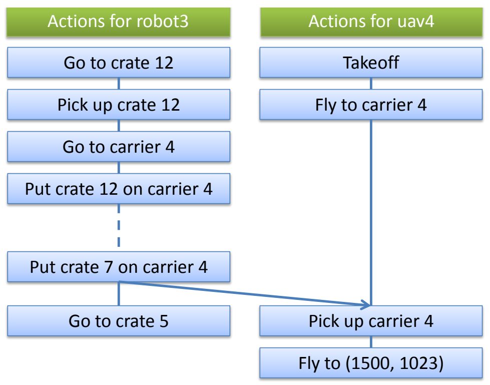


Figure 1: Example POFC plan structure

rency is possible. The actions assigned to any given agent can often only be performed in sequence, or in some cases, through a small and fixed number of sequential *threads* of execution. For example, a UAV would only be able to perform its own flight and package delivery actions in strict sequence and not in parallel, as it cannot be in several places at the same time. Another thread of execution could be used for camera control actions that are also performed sequentially, but in parallel with the flight actions. It is clear that in terms of execution flexibility, the greatest gains come from allowing partial orders across different threads as opposed to allowing partial orders within a particular thread.

The plan structures used in partial order forward-chaining are directly based on this model of execution. Each action is therefore assumed to be explicitly associated with a specific thread belonging to a specific agent. A plan structure then corresponds to one totally ordered action sequence for each thread, with a partial ordering relation between actions belonging to different threads.

Figure 1 shows an example for two threads, one for each of two agents. A UAV flies to a carrier, but is only allowed to pick it up after a ground robot has loaded a number of crates. The ground robot can immediately continue to load another carrier without waiting for the UAV. Depending on the expressivity of the planner, POFC plan structures may also include metric temporal relations between actions, mutual exclusion relations, or other similar information. A concrete plan structure for a specific prototype planner will be formally defined in Section 3.2.

To simplify the remainder of the presentation, we assume without loss of generality that each agent supports a single sequential thread of execution. Multiple threads for each physical agent can be supported either through a minor extension to these definitions or simply by modeling each thread as a separate "virtual" agent.

Sequential Search Order. Since actions for any given agent must be sequentially ordered, it appears natural to also *add* these actions in sequential order during search. In Figure 1, a new action for robot3 would then have to be added strictly after the action of going to crate 5, while a new action for uav4 would have to be added after the action of flying to the position (1500, 1023). Note that this does not prevent a new action from being added before existing actions belonging to *other* agents. For example, the next action added for robot3 could be constrained to occur before uav4 picks up carrier 4, or even before it begins flying to carrier 4.

To some extent this search process results in an earlier commitment to action orderings than in POCL planning. However, the precedence between a new action for one agent and existing actions belonging to other agents only has to be as strong as is required to ensure that preconditions are satisfied and actions do not interfere. Actions belonging to distinct agents can therefore be independent of each other to the same extent as in a standard partial order plan. This allows POFC plans to retain the essential flexibility that is desired for concurrent execution.

State Generation. The more we know about the execution order for a plan, the more information we can infer about the state of the world after any given action in the plan. For example, standard sequential forward-chaining yields a completely defined order of execution. In this case, applying an action in a completely specified world state always yields a new completely specified state. This is very useful when determining which actions are applicable in the next step. Rich state information also facilitates the use of expressive operator specifications as well as state-based heuristics or control formulas [2, 9] guiding forward-chaining search.

Since our plans are partially ordered, we cannot expect to be able to generate complete state information. For any given agent, though, actions are generated in sequential order. POFC planning can therefore be seen as performing a variation of forward-chaining search for each individual agent. We can take advantage of this to generate considerably more state information than is typically available to POCL planners, where one typically aims to restrict action ordering as little as possible and where new actions can be inserted "between" existing actions.

In particular, many state variables are generally associated with a specific agent and are only affected by the agent itself. For example, this holds for the location of an agent (unless some agents actively move others) and for the fact that an ground robot is carrying a particular object (unless one agent can place objects in another agent's gripper). Similarly, agent-specific resources such as fuel or energy levels are rarely directly affected by other agents. Due to the requirement for "local" total ordering, we can easily generate complete information about such "agent-specific" state variables at any point along an agent's action sequence. This information is particularly useful for the agent itself, since actions performed by one agent are likely to depend largely on its own agent-specific variables: Whether it is possible for uav1 to fly to a particular location depends on its own current location, its own fuel level, and its own altitude.

Not all state variables are completely agent-specific.

However, agents are in many cases comparatively loosely coupled [6]: Direct interactions with other agents are relatively few and occur comparatively rarely. For example, a ground robot would require a long sequence of actions to load a set of boxes onto a carrier. Only after this sequence is completed will there be an interaction with the UAV that picks up the carrier. This means that for extended periods of time, agents will mostly act upon and depend upon state variables that are not *currently* affected or required by other agents.

As will be shown in Section 3.5, this also provides opportunities for generating strong and useful partial states by carrying state information from one agent to another along precedence constraints when interactions do occur. In Figure 1, for example, the takeoff action will have little information about the state of the carrier, as one cannot know in advance which actions robot3 will have the time to perform before or during takeoff. However, the action of picking up carrier 4 must occur after the carrier is fully loaded. This information can therefore be carried over from robot3 to uav4 along the cross-agent precedence constraint.

Finally, as in any planner, we also have access to state variables representing static facts such as the locations of stationary objects and the capabilities of individual agents.

Thus, POFC planning enables us to generate quite extensive agent-specific information about the state that will hold after any given action is executed. This information generally will not be total, but is in many cases sufficient to determine whether a particular agent can add a particular action to its sequence. We also expect the information to be useful for the development of new state-based heuristics for POFC planning.

In some cases, additional information that is currently local to another agent will be required in order to determine the executability of an action. Acquiring such information involves the addition of precedence constraints to the plan. Returning once more to Figure 1, picking up a carrier might only be possible if the carrier is fully loaded. We know that carrier 4 is fully loaded after robot3 loads crate 7 onto the carrier. Proving this to be true when uav4 picks up the carrier requires a cross-agent precedence constraint. In the following section, we will present one potential mechanism for generating such constraints.

3 A Prototype POFC Planner

A variety of planning algorithms and search spaces can be realized within the general framework of partial order forward-chaining, differing along several dimensions.

Partial order causal link planners allow actions to be inserted in arbitrary order, without guaranteeing that their preconditions are satisfied or that their effects are compatible with existing actions. Such "flaws" in a plan must be corrected at a later time through the introduction of additional actions and precedence constraints. For example, the planner could insert the action of picking up a fully loaded carrier before adding the actions required for actually loading crates onto the carrier, as long as these actions were then constrained to be executed in the required order.

Similar methods could be used for partial order forward-chaining, with one limitation. It is by definition impossible to insert a new action for one agent before another action belonging to the same agent. However, as mentioned before, a new action can be inserted before an action belonging to *another* agent, allowing previously unsatisfied preconditions of the latter action to be satisfied. For example, the planner could first insert the action of uav4 picking up a fully loaded carrier, and then the actions required for robot3 to load the carrier.

In our initial investigations, we have instead chosen to explore a search space where adding a new action to a plan with a given set of precedence constraints is only permitted if this results in an executable plan without flaws. The preconditions of the new action must be satisfied at the point where it is inserted in the current plan structure, its effects must not interfere with existing actions in the plan, and mutual exclusion relations must be satisfied. In this sense, the planner is closer to forward-chaining planning.

We also choose to achieve goal-directedness through the use of domain-specific precondition control formulas [1, 2, 9] as explained below. This can be very effective due to the comparatively rich state information afforded by the POFC plan structure. Thus, we do not currently make use of means-ends analysis as in standard POCL planning, or state-based domain-independent heuristics as in many forward-chaining planners.

3.1 Domains and Problem Instances

For our first partial order forward-chaining planner, we assume a typed finite-domain state-variable representation of planning domains. State variables will also be called *fluents*. For example, loc(*package*) might be a location-valued fluent taking a package as its only parameter.

An *operator* has a list of typed parameters, where the first parameter always specifies the executing *agent*. For example, the act of flying between two locations may be modeled as the operator fly(*uav*, *from*, *to*), where the *uav* is the executing agent. An *action* is a fully instantiated (grounded) operator. Since finite domains are assumed, any operator is associated with a finite set of actions.

Each operator is associated with a *precondition* formula and a set of *precondition control* formulas, both of which may be disjunctive and quantified. We often use "conditions" to refer to both preconditions and control formulas.

Precondition control represents conditions that are not "physically" required for execution, but should be satisfied for an action to be meaningful for the given domain [1, 2]. For example, flying a fully loaded carrier to a location far from where its packages should be delivered is possible but pointless, and can be prevented using a suitable control formula. Given the search method used in this planner, precondition control will not introduce new subgoals that the planner will attempt to satisfy. Instead, the formulas will be used effectively to prune the search space.

An operator has a strictly positive *duration*, a temporal expression specifying the expected amount of time required to execute the action. The duration may be dependent on the state in which an action is invoked. We currently assume that the true duration of the action is strictly positive and cannot be controlled directly by the executing agent. Apart from this, we assume no knowledge of upper or lower bounds for execution times, though support for such information may be added in the future.

A set of *mutexes* can be associated with every operator[2]. Mutexes are acquired throughout the duration of an action to prevent concurrent use of resources. For example, an action loading a crate onto a carrier may acquire a mutex associated with that crate to ensure that no other agent is allowed to use the same crate at the same time. Mutexes must also be used to prevent actions that are associated with different agents and that have mutually inconsistent effects from being executed in parallel. Thus, we do not model mutual exclusion between actions by deliberately introducing inconsistent effects, as in some planning formalisms.

For simplicity, we initially assume single-step operators, where all *effects* take place in a single effect state. Effects are conjunctive and unconditional, with the expression $\mathsf{f}(\overline{v}) := v$ stating that the fluent $\mathsf{f}(\overline{v})$ has been assigned the value v when execution ends. Both v and all terms in $\overline{v}$ must be either value constants or variables from the formal parameters of the operator. For example, the operator $\mathsf{fly}(uav, from, to)$ may have the effect $\mathsf{loc}(uav) := to$.

For any given problem instance, the *initial state* must provide a complete definition of the values of all fluents. The *goal* is typically conjunctive, but may also be disjunctive. The construct $\mathsf{goal}(\phi)$ can be used in precondition control formulas to test whether ϕ is entailed by the goal. For example, we should only load boxes that must be moved according to the goal.

3.2 Plan Structures

We associate each action a in a plan with an *invocation node inv*(a) where conditions must hold and where mutexes are acquired, and an *effect node eff*(a) where effects take place and mutexes are released. All mutexes belonging to the action are considered to be held by the associated agent in the entire interval of time between its invocation node and its effect node[3]. Invocation nodes and effect nodes are called *plan nodes*.

[2]A mutex is an object that can only be acquired by a single thread, or in this case agent, at any given point in time.

[3]Thus, the planning algorithm must generate a partial order that is sufficiently strong to ensure that no two actions a, a' belonging to distinct agents can hold the same mutex at the same time. This is done at the end of Section 3.6.

A *plan* is then a tuple $\langle A, N, L, O\rangle$ whose components are defined as follows.

- A is the set of actions occurring in the plan.
- N contains one invocation node and one effect node for every action in A.
- L is a set of ground causal links $n_i \xrightarrow{\mathsf{f}=v} n_j$ representing the commitment that the effect node n_i will achieve the condition $\mathsf{f} = v$ for the invocation node n_j.
- O contains a set of ordering constraints on N whose transitive closure is a partial order denoted by $\preceq$, where we define $n_i \prec n_j$ iff $n_i \preceq n_j$ and $n_i \neq n_j$.

For any action a, we implicitly require that $inv(a) \prec eff(a)$: An action is always invoked before it has its effects. Additionally, given that there are no upper or lower bounds on action durations, it is not possible to directly control the time at which an action finishes by any other means than by delaying its invocation. Therefore, if a plan requires $eff(a_1) \prec eff(a_2)$, it is implicitly required that $eff(a_2) \prec inv(a_2)$. Finally, by the definition of partial order forward-chaining, the nodes associated with any given agent must be totally ordered by O.

Similar to standard POCL planning, we assume a special initial action $a_0 \in A$ without conditions or mutexes, whose expected duration is 0 and whose effects provide a complete definition of the initial state. For all other actions $a_i \neq a_0 \in A$, we must have $eff(a_0) \prec inv(a_i)$. Due to the use of forward-chaining techniques instead of means-ends analysis, there is no need for an action whose preconditions represent the goal, as in standard POCL planning.

Note that this plan structure is defined for the expressivity supported by our initial POFC planner. Given different levels of expressivity, different structures may be appropriate. For example, if upper and lower bounds on action durations are supported, a plan may have to include a temporal network or a similar structure, thereby allowing the planner to efficiently query the implicit action precedence constraints that follow from these bounds. As our initial planner has no bounds on durations, precedence can be completely determined by the constraints in O.

3.3 Executable Plans and Solutions

If a partially ordered plan will always be executed sequentially, it can be considered executable if and only if all action sequences satisfying the partial order are executable.

For POFC planners, as well as some POCL planners, the assumption of concurrent execution is fundamental. This leads to the possibility of one agent beginning or finishing executing an action *while* another agent is in the process of executing another action. The need to consider such cases is the reason why our precedence relation is defined relative to invocation and effect nodes, not relative to entire actions. A POFC plan should therefore be considered executable if

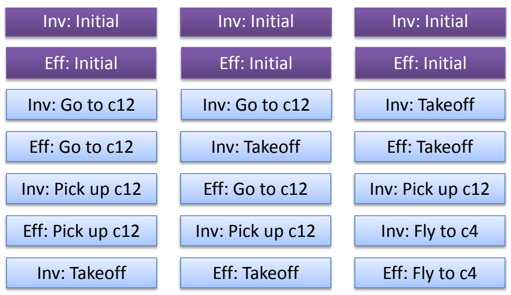


Figure 2: Three node sequences

and only if every *node* sequence satisfying the associated partial order is executable. Figure 2 shows three node sequences compatible with the plan defined in Figure 1, with the addition of the special initial action used in this particular POFC planner.

The executability of a single node sequence is defined in the standard way. Observe that the first node in such a sequence must be the invocation node of the initial action a_0, which has no preconditions or effects. The second node is the effect node of a_0, whose effects completely define the initial state. After this prefix, invocation nodes and effect nodes may alternate, or many nodes of the same type may occur in sequence, depending on the order in which actions are assumed to begin and end. Effect nodes update the current state. For the plan to be executable, an effect node must not have internally inconsistent effects. For example, it must not assign two different values to the same fluent. Invocation nodes contain preconditions and precondition control formulas that must be satisfied in the "current" state. Finally, executability also requires that no mutex is held by more than one agent in the same interval of time.

An executable plan is a *solution* iff every compatible node sequence results in a final state satisfying the goal.

3.4 Search Space

POCL planners add actions first and resolve unsatisfied conditions later, thereby searching through the space of partially ordered (and not necessarily executable) sets of actions. In contrast, forward-chaining planners begin with an empty executable plan, and actions can only be added after being proven executable. Forward-chaining can thus be viewed as searching in the space of *executable plans*, with a single plan modification step consisting of adding one new action at the end of the current plan.

Given the plan structure and expressivity defined above, a similar search space can be used for POFC planning. The initial search node then corresponds to the "empty" executable plan $\langle\{a_0\}, \{inv(a_0), \mathit{eff}(a_0)\}, \emptyset, \{inv(a_0) \prec \mathit{eff}(a_0)\}\rangle$, where a_0 is the initial action whose effects define the initial state. Each child of a search node adds a single new action to the end of one agent's action sequence, together with a set of precedence constraints and causal links ensuring that the plan remains executable.

This claim implies that we do not lose completeness by requiring every intermediate search node to correspond to an *executable* plan, as opposed to an arbitrary action set as in POCL planning. Intuitively, this holds because there can be no circular dependencies between actions, where adding several actions at the same time could lead to a new executable plan but adding any single action is insufficient.

More formally, let $\pi = \langle A, N, L, O\rangle$ be an arbitrary executable plan. Let $a \in A$ be an action which is not guaranteed to precede any other action in the plan (for example, the action of going to crate 5 in Figure 1). In other words, let $a \in A$ be an action such that there exists no other action $b \in A$ where $\mathit{eff}(a) \prec inv(b)$. Such an a must exist, or the precedence relation would be circular and consequently not a partial order, and π would not have been executable.

Since a is not the predecessor of any other action, it cannot have been used to support the preconditions and control formulas of other actions in π. Removing it from the plan will therefore have no negative effects in this respect. Similarly, a cannot be required for mutual exclusion to be satisfied: Removing a can only lead to fewer mutexes being allocated, which can only improve executability.

Consequently, removing a and the associated plan nodes, causal links and precedence constraints from π must lead to a new executable plan π'. We see inductively that any finite executable plan can be reduced to the initial plan through a sequence of such reduction steps, where each step results in an executable plan. Conversely, any executable plan can be constructed from the initial plan through a sequence of action additions, each step resulting in an executable plan.

Since we assume finite domains, solution plans must be of finite size and can be constructed from the initial plan through a finite number of action addition steps.

Given finite domains, the action set must also be finite. Furthermore, when any particular action is added to a plan, there must be a finite number of ways to introduce new precedence constraints and causal links ensuring that the plan remains executable. Any search node must therefore have a finite number of children, and the search space can be searched to any given depth in finite time.

Thus, given a method for generating all valid child nodes (finding all applicable actions), we can incrementally construct and traverse a search space. Given a method for testing goal satisfaction and a complete search method such as iterative deepening or depth first search with cycle detection, we have a complete planner. These issues will be considered in more detail in the following subsections.

3.5 Partial States

Forward-chaining planners find applicable actions by evaluating preconditions in the current completely defined state. For partially ordered plans there is no unique "current" state, and we can rarely infer complete information about the world even for a specific plan node or action.

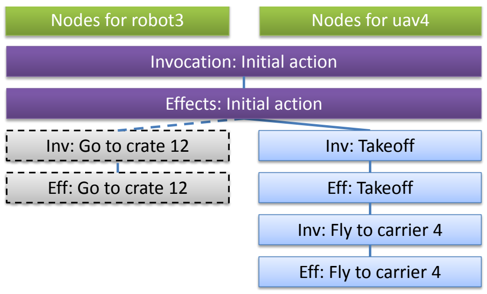


Figure 3: Example POFC plan structure

However, as discussed previously it *is* possible to infer and store *partial* state information for any plan node.

A variety of structures can be used for this purpose, each having its own strengths and weaknesses. For example, associating each plan node with a set of possible states would allow any fact that can be inferred from the current plan to be represented, including arbitrary disjunctive facts such as $\mathsf{at(uav4, pos1)} \vee \mathsf{at(uav5, pos1)}$. However, such structures tend to require considerable space and may take a considerable amount of time to update when new actions are added.

Instead, we currently use partial states represented as a finite set of possible values for each fluent: $\mathsf{f} \in \{v_1, \ldots, v_n\}$. The evaluation procedure defined below resolves as many parts of a formula as possible through this partial state for efficiency. Should this not be sufficient to completely determine the truth or falsity of the formula, the procedure falls back on an explicit traversal of the plan structure for those parts of the formula that remain unknown. This grounds evaluation in the initial state and the explicit effects in the plan for completeness.

The Initial State. The initial plan consists of a single action a_0, whose invocation node is associated with the empty state and whose effect node directly and completely defines the initial state of the planning problem at hand.

Updating States. When a new action is added to a plan, states associated with existing nodes must be incrementally updated to reflect the changes that this might have caused. For example, consider the situation in Figure 3 and assume that robot3 is initially at depot1. Before we add the action of going to crate12, the plan includes no movement for the robot, and the invocation node for takeoff will include the fact that robot3 remains at depot1. When the new action is added, it is temporally unconstrained relative to the takeoff action. Then we only know that when takeoff is invoked, robot3 will be either at depot1 or at crate12.

State updates must generate "sound" states: When a particular node is reached during execution, each fluent must be guaranteed to take on a value included in the state of the node. However, updates do not have to yield the *strongest* information that can be represented in the state structure, since formula evaluation will be able to fall back on explicit plan traversal. Thus, a tradeoff can be made between the strength and the efficiency of the update procedure.

For example, a sound state update procedure could weaken the states of all existing nodes in the plan: If a state claims that $\mathsf{f} \in V$ and the new action has the effects $\mathsf{f} := v$, the state would be modified to claim $\mathsf{f} \in V \cup \{v\}$. On the other hand, it is clear that no effect node can interfere with the states of its own ancestors. In Figure 1, for example, the effect node for the action of picking up carrier 4 has ancestor nodes belonging to robot3 as well as uav4 and cannot interfere with the states of these nodes. Therefore, weakening the states of all *non-ancestors* is sufficient.

Generating New States. When a new plan node n is added, it always has at least one immediate predecessor – a node $p \prec n$ such that there exists no intermediate node where $p \prec n' \prec n$. In Figure 3, for example, the invocation node of going to crate12 has a single immediate predecessor: The effect node of the initial action. If an action is also constrained to start after an action belonging to another agent, it can have multiple immediate predecessors.

Let n be a new node and p one of its immediate predecessors. It is clear that the facts that hold in p will still hold in n except when there is explicit interference from intervening effects. Therefore, taking the state associated with p and "weakening" it with all effects that *may* occur between p and n, in the same manner as in the state update procedure, will result in a new partial state that is valid for n.

For example, let n be the invocation node of going to crate12 in Figure 3, and let p be the effect node of the initial action. We can then generate a state for n by taking the state of p and weakening it with the effects associated with taking off and flying to carrier 4, since these are the only effect nodes that might intervene between p and n.

Now suppose that we apply this procedure to two immediate predecessors p_1 and p_2, resulting in two states s_1 and s_2 both describing facts that must hold at the new node n. If s_1 claims that $\mathsf{f} \in V_1$ and s_2 claims that $\mathsf{f} \in V_2$ for some fluent f, then both of these claims must be true. We therefore know that $\mathsf{f} \in V_1 \cap V_2$. This can be extended to an arbitrary number of immediate predecessors, resulting in stronger state information when a new node is created. Note that given that agents are loosely coupled, a node generally has very few immediate predecessors, which limits the time required for state generation.

Conjoining information from multiple predecessors often results in gaining "new" information that was not previously available for the current agent. For example, if robot3 loads boxes onto a carrier, incrementally updating a total-weight fluent, other agents will only have partial information about this fluent. When uav4 picks up the carrier, this action must have the last load action of robot3 as an immediate predecessor. The UAV thereby gains complete information about weight and can use this efficiently in future actions.

Finally, if the new node is an effect node, its own effects must also be applied to the new state.

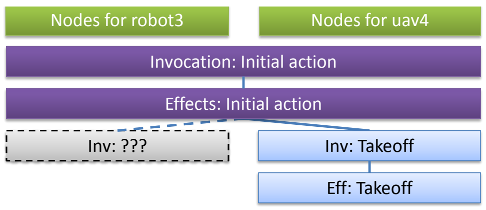


Figure 4: A new invocation node is being created

3.6 Searching for Applicable Actions

When searching for applicable actions, we first determine which agent to use. Several heuristics can be used, such as reusing agents to the greatest extent possible or distributing actions evenly across all available agents. In the latter case, we can calculate the timepoint at which we expect each agent to finish executing its actions in the current plan and test agents in this order. This selection must naturally be a backtrack point, to which the planner can return in order to try a different choice of agent. The same applies to many other choices below.

The intention is then to make use of the previously discussed procedure for generating state information in order to quickly detect most inapplicable actions for the selected agent. For some of the remaining actions, we may have to introduce precedence constraints in order to actively *make* the actions applicable. For example, suppose we are testing whether uav4 can pick up carrier4. This may only be possible if the pickup action is constrained to occur after robot3 loaded the carrier.

Satisfying Preconditions. When a specific agent has been chosen, we create a new invocation node n that is not yet associated with a particular action. This node is appended to the end of the selected agent's node sequence, or after the initial action a_0 if no agent-specific actions have been added previously. In Figure 4, for example, a new invocation node for robot3 is being created. At this point, n will always have a single immediate predecessor.

We then generate a "temporary" partial state s for n according to the procedure discussed previously. This state includes many of the facts that must hold when the new action is invoked, regardless of which action happens to be selected. Consequently it can be used to efficiently separate potential new actions for the current agent into three sets, where preconditions and precondition control formulas are definitely satisfied (A_+), definitely *not* satisfied (A_-), and potentially satisfied ($A_?$), respectively.

For actions in $A_?$, there is currently insufficient information in s to determine whether the required conditions hold. This may be due to incomplete state updates or because it is inherently impossible to determine the value of a particular fluent given the current partial order. For example, robot3 may only be able to move to launchpad1 if uav4 has already taken off, which requires n to be ordered strictly after the effect node of takeoff. Thus, new precedence constraints may be required to ensure that an action is executable, which may give n additional immediate predecessors. This can only strengthen the information previously provided in s, never invalidate it.

Given sufficiently loose coupling, together with the existence of agent-local and static facts, $A_?$ will be comparatively small. Nevertheless, actions in this set must also be handled. For this purpose we define the procedure *make-true*(α, n, π), which recursively determines whether a formula α can be made to hold in n, and if so, which precedence constraints need to be added for this to be the case. Subformulas are evaluated in the partial state of n whenever possible.

The procedure returns a set of *extensions* corresponding to the minimally constraining ways in which the precedence order can be constrained to ensure that α holds in n. Each extension is a tuple $\langle P, C\rangle$ where P is a set of precedence constraints to be added to O and C is a set of causal links to be added to L. Thus, if α is proven false regardless of which precedence constraints are added, $\varnothing$ is returned: There exists no valid extension. If α is proven true without the addition of new constraints, $\{\langle\varnothing, C\rangle\}$ is returned for some suitable set of causal links L. In this case, the state of n can be updated accordingly, providing better information for future formula evaluation.

We will now describe the *make-true* procedure. Certain aspects of the procedure have been simplified below to improve readability while retaining correctness. A number of optimizations to this basic procedure can and have been applied, several of which will be discussed below the main procedure description.

Assume that we call *make-true*(α, n, π), where $\pi = \langle A, N, L, O\rangle$, and let s be the partial state of n.

Let us first consider the base case, where α is the atomic formula $\mathsf{f} = v$. If this is true according to s, we determine which node n' generated the supporting value for f and return $\{\langle\varnothing, \{n' \xrightarrow{\mathsf{f}=v} n\}\rangle\}$. If the formula is false according to s, we return $\varnothing$. Otherwise, s contains insufficient information to determine whether the formula holds. We then find all effect nodes $E = \{e_1, \dots, e_m\}$ in π that assign the value $\mathsf{f} = v$. This set may be empty, in which case we must return $\varnothing$. If $|E| > 0$, then for each effect node $e_i \in E$, we generate all sets $P_{i,j}$ of minimally constraining new precedence constraints that we could use to ensure that the relevant effect cannot be interfered with between e_i and n. Each set $P_{i,j}$, together with the associated causal links, forms one valid extension. The set of all these extensions is returned.

The case where α has the form $\mathsf{f} \neq v$ is handled similarly.

If α is a negated formula $\neg\beta$, the negation is pushed inwards using standard equivalences. For example, *make-true*$(\neg(\beta \wedge \gamma), n, \pi) =$ *make-true*$(\neg\beta \vee \neg\gamma, n, \pi)$.

If α is a conjunction $\beta \wedge \gamma$, then both conjuncts must be satisfied. We first determine how β can be satisfied by recursively calling $E_1 =$ *make-true*(β, n, π). If this returns $\varnothing$, we immediately return $\varnothing$: If we cannot satisfy the first conjunct, we cannot satisfy the conjunction. Otherwise, we need to determine how the extensions in E_1 can be further extended so that γ is also satisfied. For every extension $\langle P_i, C_i \rangle \in E_1$, we let $\pi_i = \langle A, N, L \cup C, O \cup P \rangle$ and call *make-true*(γ, n, π_i). We take the union of all results, remove all extensions that are not minimal in terms of precedence constraints, and return the remaining extensions.

If α is a disjunction $\beta \vee \gamma$, then it is sufficient that one disjunct is satisfied. We therefore calculate *make-true*$(\beta, n, \pi) \cup$ *make-true*(γ, n, π), corresponding to all ways of satisfying either disjunct. We then remove all extensions that are not minimal in terms of precedence constraints and return the remaining extensions.

Finally, if α is a quantified formula, we iterate over the finite set of values in the domain of the quantified variable. Universal quantification can then essentially be considered equivalent to conjunction, while existential quantification is equivalent to disjunction.

This procedure may seem quite complex. However, any POCL planner must also resolve unsupported conditions in a similar manner, searching for existing actions that support the conditions or possibly adding new actions for support. Apart from the order of commitment, the main differences are that the POFC planner uses a partial state to quickly filter out most candidate actions and is restricted to searching for existing actions supporting conditions as opposed to adding new actions.

Similarly, though the evaluation procedure may seem to lead to a combinatorial explosion, recall that we are essentially doing forward search. We must therefore find existing support for all conditions *in the current plan*, which tends to yield a reasonably sized set of consistent extensions.

A number of optimizations can also be applied.

For example, instead of calculating *all* possible extensions in a single call, extensions can be returned incrementally as they are found.

It is possible to store and efficiently update a map associating each fluent with the nodes affecting it, for efficiency when searching for support for an atomic condition.

The evaluation order can be altered so that one always evaluates those parts of a formula that can be resolved in the current partial state before those parts that require support from effect nodes. This is useful in cases such as when the first conjunct in a conjunction is not determined by the partial state but the second conjunct is definitely false.

As a final example, we can structure the process of finding all applicable instances of a particular operator so that large sets of instances can be ruled out in a single evaluation. For example, suppose that flying between two locations is only possible when a UAV is already in the air, represented as the fluent flying(uav). Whether this condition holds depends on the agent but is independent of the locations in question. If a particular UAV is not in the air, there is therefore no need to iterate over all combinations of locations.

Once the evaluation procedure has ensured that preconditions and precondition control formulas will hold, we continue by adding precedence constraints ensuring that no mutex is used twice concurrently. We then ensure that the effects of the new action cannot interfere with existing causal links in the plan. If this entire procedure proceeds, the action was applicable and one of the possible sets of precedence constraints and causal links can be added to the plan.

Finally, we should note that **goal satisfaction** can be tested in a manner equivalent to the *make-true* procedure. The goal formula is then evaluated in a new node having all other nodes as ancestors. Any extension returned by *make-true* corresponds to one possible way in which the current plan can be extended with new precedence constraints to ensure that the goal is satisfied after the execution of all actions.

4 Related work

The ability to create temporal partially ordered plans is far from new. A variety of such planners exist in the literature and could potentially be applied in multi-agent settings. Some of these planners also explicitly focus on multi-agent planning. For example, Boutilier and Brafman [5] focus on modeling concurrent interacting actions, in a sense the opposite of the loosely coupled agents we aim at.

However, the main focus of this paper is to investigate the possibility of taking advantage of certain aspects of *forward-chaining* when generating partially ordered plans for multiple agents. In this area, very little appears to have been done. An extensive search through the literature reveals two primary examples.

First, a multi-agent planner presented by Brenner [7] does combine partial order planning with forward search. However, the planner does not explicitly separate actions by agent and does not keep track of agent-specific states. Instead, it evaluates conjunctive preconditions relative to those value assignments that must hold after *all* actions in the current plan have finished. This is significantly weaker than the evaluation procedure defined in this paper. In fact, as Brenner's evaluation procedure cannot introduce new precedence constraints, the planner is incomplete.

Second, the FLECS planner [11] uses means-ends analysis to add relevant actions. A FLExible Commitment Strategy determines when an action should be moved to the end of a totally ordered plan prefix, allowing its effects to be determined and increasing the amount of state information available to the planner. Actions that have not yet been added to this prefix remain partially ordered.

Though there is some similarity in the combination of total and partial orders, FLECS uses a completely different search space and method for action selection. Also, whereas we strive to generate the weakest partial order possible between actions performed by different actions, any action that FLECS moves to the plan prefix immediately becomes totally ordered relative relative to all other actions.

FLECS therefore does not retain a partial order between actions belonging to distinct agents.

Thus, we have found no existing planners taking advantage of agent-specific forward-chaining in the manner described in this paper.

5 Conclusions

We have presented a hybrid planning framework combining interesting properties of temporal partial order and forward-chaining planning. We have also described one of many possible planners operating within this framework. We view this as an interesting variation of POCL planning worthy of further exploration, and believe that future investigations will show that each framework has its own strengths and applications.

An early prototype implementation of the suggested planner has been developed. As we are still in the exploration phase, the current implementation is written for readability and ease of extension rather than for performance. For example, many data structures can and will be replaced with considerably more efficient ones. Therefore, standard benchmark tests would provide no meaningful information about the strengths of POFC planners as compared to other temporal partial order planners.

However, the basic structure of this particular POFC planning method can also be evaluated by observing search patterns, such as the strength of pruning when precondition control formulas are used. Though a final judgment has to await more extensive testing in multiple domains, initial experiments indicate a pruning strength very similar to that of standard forward-chaining planners based on control formulas for pruning, which is very promising.

Several extensions are planned for the near future, including support for incompletely specified initial states and the generation of conformant plans. We are also very interested in investigating the use of domain-independent heuristics for the new plan structure. Finally, we may develop alternative search procedures more similar to POCL planners in the sense that actions with currently unsupported conditions can be added, resulting in flaws that can be resolved through means-ends analysis.

Acknowledgements

This work is partially supported by grants from the Swedish Research Council (50405001, 50405002) and CENIIT, the Center for Industrial Information Technology (06.09).

References

[1] F. Bacchus and M. Ady. Precondition control. Available at http://www.cs.toronto.edu/~fbacchus/Papers/BApre.pdf, 1999.

[2] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

[3] C. Bäckström. Computational aspects of reordering plans. *Journal of Artificial Intelligence Research*, 9 (99):137, 1998.

[4] B. Bonet and H. Geffner. HSP: Heuristic search planner. *AI Magazine*, 21(2), 2000.

[5] C. Boutilier and R. I. Brafman. Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14:105–136, 2001.

[6] R. I. Brafman and C. Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 28–35, Sydney, Australia, 2008.

[7] M. Brenner. Multiagent planning with partially ordered temporal plans. In *Proc. IJCAI*, 2003.

[8] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[9] J. Kvarnström and P. Doherty. TALplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence*, 30:119–169, June 2000.

[10] E. D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, pages 206–214. Morgan Kaufmann Publishers Inc., 1975.

[11] M. Veloso and P. Stone. FLECS: Planning with a flexible commitment strategy. *Journal of Artificial Intelligence Research*, 3:25–52, 1995.

[12] D. S. Weld. An introduction to least commitment planning. *AI magazine*, 15(4):27, 1994.

Ontology Patterns - Typology and Experiences from Design Pattern Development

Eva Blomqvist [*]
StLab, ISTC-CNR, via Nomentana 56, 00161 Roma, Italy.
e-mail: eva.blomqvist@istc.cnr.it

April 27, 2010

Abstract

Patterns for ontology engineering have received an increased interest during the past few years. Ontology patterns facilitate knowledge reuse and aid the task of engineering application ontologies. Ontology patterns provide general solutions, which are encoded and stored for reuse purposes. This paper first discusses the nature and characteristics of ontology patterns, and presents a typology of ontology patterns that can be used as an informal vocabulary when presenting and reusing patterns. Secondly, we introduce a catalogue of ontology design patterns, for the domain of product development, which were re-engineered from existing knowledge sources, and we especially list some experiences from this re-engineering process. As future work we envision an increased tools support for pattern development and usage, as well as further experiments using the proposed pattern catalogue.

1 Introduction

Ontology engineering, for the Semantic Web and other application fields, is a tedious and error prone process, requiring expertise in knowledge modelling and logical languages. With the emergence of the Semantic Web [3], Linked Data [4], and the introduction of semantic technologies in different application areas, e.g. Content Management Systems[1], the usage of (logically) light-weight application ontologies has drastically increased. Such ontologies are not heavily axiomatized, but provide just a bit of formal semantics to a data set. Thereby web and system developers in these fields have become more and more interested in ontology engineering, e.g. on the Semantic Web it is no longer the case that most ontologies are carefully constructed by knowledge engineers, instead they are drafted by people who have only a brief knowledge of the underlying logical formalisms of ontology languages (such as the Description Logics-based Web Ontology Language, OWL). An example of this trend is the successful series of VoCamps[2] arranged during the past two years, where researchers and practitioners get together and during one or two days draft vocabularies, i.e., small ontologies, to be used for describing some dataset.

To exploit the full benefits of the Semantic Web, and other semantically-enhanced systems, application ontologies therefore need to be easy to construct. Motivated by this development we are focussing on knowledge reuse through *ontology patterns*. Patterns are recurrent solutions to design problems that can be stored and retrieved from catalogues, and reused when engineering ontologies and semantic applications. However, so far there exist no coherent and uniform classification model for ontology patterns, hence, there is no common vocabulary for discussing patterns.

Patterns have so far mainly been available for addressing syntactic representation and logical modelling problems (e.g. the OWL patterns proposed

[*]The research presented in sections 2 and 3 was financed by, and conducted at, a previous affiliation of the author; the Information Engineering research group at Jönköping University, Sweden.

[1]See the IKS project at http://www.iks-project.eu/

[2]http://vocamp.org/wiki/WhatIsVoCamp

by the W3C[3], and logical patterns as in the catalogue maintained by the University of Manchester[4]), and even when present they are still quite scarce and often very general, in terms of domain applicability. General patterns are highly reusable, but in addition hard to match to the problem at hand, and reuse in a correct manner. These drawbacks support the need for pattern catalogues tailored for more specific domains, such as product development application ontologies, which is the target of this paper. The ideal procedure for pattern development is an organic growth of a pattern catalogue from the joint experiences of a community. However, this process may take several years. In order to kick-start the usage of patterns we propose to also re-engineer other patterns (e.g. data model patterns) into ontology patterns. For data model patterns domain specific catalogues already exist and are freely available, although not formalized.

This paper starts by introducing some basic notions of ontology engineering in section 1.1. Section 2 proposes a pattern typology and some general characteristics for describing patterns. In section 3 we address the problem of developing content ontology design patterns (Content ODPs) based on existing knowledge sources, and their initial evaluation and refinement. Related work is mentioned in section 4, and as an introduction to ongoing and future work section 5 briefly presents two tools for ontology pattern development and reuse, before we conclude with the discussion in section 6.

1.1 Ontology Engineering

In this paper we focus on the notion of application ontologies [16], i.e. ontologies built and tailored for a specific application scenario, as opposed to for example general domain ontologies or abstract top-level ontologies. Engineering of application ontologies is a continuous process incorporating the complete life-cycle of an ontology[14]; everything from the description of its intended application, requirements engineering, ontology construction, ontology reuse, to deploying the ontology in the application, maintaining, and evolving it. An important part of ontology engineering is the actual modelling of the ontology, which is the main focus of the patterns in this paper. A set of questions (first proposed by the author in [5], based on an overview of the field) can briefly summarize the problem areas on different abstraction levels that need to be addressed in order to build (model) an ontology:

1. What is the purpose of the ontology; how is it to be applied in a software system?
2. What parts are to form the ontology; how should the architecture be formed?
3. What should the ontology contain; what concepts, relations, and axioms?
4. How should the ontology be represented syntactically?

The first question pertains to the requirements and the interface of the ontology. An application ontology has a well-defined set of requirements and should be tailored to the way it will be used in a software system. The second question concerns the overall architecture of the ontology, e.g. the different knowledge domains to include, the different views needed etc. The third question addresses the detailed design and content of the ontology, i.e., how the individual requirements will be realized inside the overall structure given by the ontology architecture. Finally, the fourth question deals with representation, e.g., the logical language to represent the ontology, efficiency of reasoning, and usability (in terms of understandability by humans).

2 Pattern Characteristics and Typology

We generally define *ontology pattern* as:

Definition 1. *An ontology pattern is a set of ontological elements, structures or construction principles that intend to solve a specific engineering problem and that recurs, either exactly replicated or in an adapted form, within some set of ontologies, or is envisioned to recur within some future set of ontologies.*

Based on this definition a pattern has to provide a solution to a specific engineering problem or category of problems (*problem* focus), and recur in existing solutions or envisioned solutions (*reuse*

[3]http://www.w3.org/2001/sw/BestPractices/OEP/

[4]http://www.gong.manchester.ac.uk/odp/html/index.html

focus). The nature of the 'engineering problem' is not important, however, it has to involve ontologies in some way or another, e.g. it can be any problem where an ontology is seen as the solution or the construction and maintenance of that ontology. These are considered as necessary and sufficient conditions. A pattern is usually also based on 'best practices', but this is not a necessary condition. A pattern that does not follow such best practices, but instead exhibits a common problem or mistake is usually denoted an *anti-pattern.*

2.1 Basic Perspectives on Patterns

Regarding their origin, either patterns are purely experience-based, i.e., produced *bottom-up* through some pattern mining or recognition technique, or they can be carefully designed and constructed, as abstract templates, i.e., produced *top-down.* From the bottom-up perspective, patterns are recurring structures in some set of solutions. In contrast, from the top-down perspective, patterns are templates that represent a consensus view on a specific problem. In ontology engineering both perspectives exist, including hybrid approaches.

2.1.1 Structure or Content Focus

Structural patterns deal with the logical structure of the ontological elements, but not with the actual ontology represented by these. A structural pattern has an empty signature, i.e., on the design level no actual named classes and properties are proposed by the pattern. An example of such a structural pattern (with an empty signature) is the logical macro 'disjoint union' (which in OWL 2 has its own language construct but in OWL 1 could merely be described as an abstract pattern). The pattern expresses the notion of an exhaustive partition of a class into a set of subclasses, which are all mutually disjoint. In contrast, content patterns deal with ontological modelling solutions, hence, they are domain specific, where the scope of the domain depends on their level of ontological abstraction. An example of a content pattern (with a non-empty signature) is the *Situation*[5] pattern, which is a instantiation of the structural n-ary relation pattern within the domain of situations, i.e. it contains a class 'Situation' which is the projection of the n-ary relation, and two properties 'isSettingFor'/'hasSetting' to indicate the things involved in the situation. Since the pattern contains actual named classes and properties it has a non-empty signature, i.e. it proposes a vocabulary for situations. Structural and content design patterns have been discussed in detail in [12].

[5]http://ontologydesignpatterns.org/wiki/Submissions:Situation

2.1.2 Abstraction and Granularity

Granularity is concerned with the scope of the pattern, e.g. treating some small part of an ontology in detail or treating a complete ontology. Whatever level of granularity is chosen, patterns are focused on solving *one specific problem*, on that specific level of granularity. Abstraction is about hiding the details of some structure in order to describe another aspect of the structure in a more convenient manner, thereby 'abstracting' from the details. Abstraction is more than a change of granularity however; a change in abstraction means switching perspective to view completely different aspects, e.g. by using a new form of representation.

2.2 Pattern Typology

We define four levels of abstraction, including applicable levels of granularity, of ontology patterns. A very preliminary proposal for granularity levels was suggested in [7]. In this paper we present a more elaborate classification where (i) the levels have only one differentiating notion, while in [7] abstraction and granularity were used simultaneously, and (ii) the levels have been detailed and given intensional definitions. The abstraction levels, and corresponding levels of granularity, of ontology engineering patterns provide a top-down framework for classifying and describing ontology engineering patterns; a pattern typology. The framework can be illustrated as in Table 1. The levels are described in detail below and each correspond to one of the questions in section 1.1.

2.2.1 Abstraction Level: Application Patterns

To address the first question of ontology engineering, as listed in section 1.1, application patterns are intended to address problems concerned with the

Table 1: Ontology pattern typology.

Abstraction	Granularity
Application patterns	Overall ontology
Architecture patterns	Overall ontology Ontology module
Design patterns	Overall ontology Ontology module Logical elements
Syntactic patterns	Overall ontology Ontology module Logical elements Element syntax

overall scope and purpose of the ontology. There is only one level of granularity possible, i.e. viewing the complete ontology as a unit. In [1] the notion of ontology *application pattern* was defined as follows:

Definition 2. *An ontology application pattern is a software architecture pattern describing a software system that utilises ontologies to create some of its functionality. The pattern describes properties of the ontology, or ontologies, in the system, and the connection between the ontology and the rest of the system.*

Examples of ontology application patterns can be found in approaches such as by Harmelen et al.[24], which define common reasoning patterns that can be realized by ontologies to support some functionality of a software system.

2.2.2 Abstraction Level: Architecture Patterns

To address the second question in section 1.1, architecture patterns are concerned with the overall organisation of the ontology. An ontology *architecture pattern* is defined as:

Definition 3. *An ontology architecture pattern is a pattern describing the overall structure of the ontology, prescribing certain construction principles and restricting the selection of design patterns that can be used to implement the ontology.*

Note that on this level of abstraction, the details of the ontology (concepts and relations) are not considered, nor how to realize the ontology in some logical language. Ontology architecture patterns can be of two different levels of granularity, either treating the complete ontology or only part of the ontology, e.g. one ontology module. To the best of our knowledge, there are so far no ontology architecture patterns proposed in literature.

Analogous to software architectures, also ontologies can benefit from more detailed descriptions of typical architectures. In software engineering a reference architecture can be defined as a set of architecture patterns applied to a specific well-known problem [2], however, note that this is still not a concrete software architecture, i.e. it is a problem decomposition mapped onto abstract components. We analogously define the notion of ontology reference architecture as a domain specific instance of one or more ontology architecture patterns:

Definition 4. *An ontology reference architecture is a domain-specific ontology architecture pattern containing restrictions and requirements on both the structure and content of the complete ontology.*

2.2.3 Abstraction Level: Design Patterns

The third question in section 1.1 concerns the level of design patterns, addressing the actual modelling of the ontology. We define an ontology *design pattern* as:

Definition 5. *An ontology design pattern is a set of ontological elements, structures or construction principles that solve a clearly defined particular modelling problem.*

Ontology design patterns describe solutions on the abstraction level of logical ontology modelling languages, on three levels of granularity; 1) treating individual logical elements, e.g. logical axioms or macros, 2) treating a smaller part of the ontology, e.g. one module, or 3) treating the complete ontology, e.g. restricting the overall structure on the logical level. The latter is not to be confused with the architecture patterns previously described. On the abstraction level of architecture patterns the overall structure is in focus, without considering the logical realisation of that structure on the modelling level. While on the design level, an example on the granularity level dealing with the complete ontology could be 'taxonomy', i.e. restricting the logical constructs of the complete ontology but on the abstraction level of logical constructs.

So far, most of the ontology patterns proposed are on the level of design patterns. For example, the structural patterns developed by the University of Manchester[6] and the W3C[7], and the content patterns collected in the ODP Portal[8].

2.2.4 Abstraction Level: Syntactic Patterns

The final question of section 1.1 concerns representation; e.g. naming of ontology elements or the ontology itself, and how the elements and axioms are represented in some machine processable syntax. We define a *syntactic pattern* as:

Definition 6. *A syntactic ontology pattern is the realisation of other ontology patterns, or parts or combinations of ontology patterns, in a specific representation syntax.*

This definition adds an additional lowest level of granularity for this type of pattern, where single strings and character combinations are the core of the pattern. Most ontology representation syntaxes have their own common constructs, or in software engineering terms, their own language idioms. For example, lexico-syntactic patterns can be considered as syntactic patterns of ontologies, e.g. Hearst patterns [18] representing subclass relations. Other examples are naming conventions [23], e.g. using the camel convention for naming classes and properties, or using verbs for property names.

3 Pattern Catalogue

During the course of our research we have developed a set of content ontology design patterns (Content ODPs), on the granularity level of solving small design problems (modules) within an ontology. This means that the patterns themselves are small ontologies, with additional annotations describing them, such as the general requirements that each pattern solves (usually expressed in the form of Competency Questions [15]). The patterns have been re-engineered from different sources, mainly data model patterns. This section describes the re-engineering process, and the refinement process leading up to the submission of a set of patterns, i.e. a small product development pattern language (c.f. pattern languages in software engineering), to the ODP Portal[9], and the experiences collected during this process.

3.1 Re-engineering

The initial re-engineering steps have already been discussed in [5], hence, we only give a brief overview of the process in this section. The initial set of Content ODPs developed consisted of 26 patterns. The sources were, in addition to data model patterns [17, 21, 20]: software analysis patterns [9], top level ontologies [11], goal structures [22], and cognitive patterns [13]. The set of patterns was constructed with a specific domain in mind, i.e. product development application ontologies (hence not a specific industry domain), intending to create a small pattern language for this domain. Initially a simple translation approach was applied, defining a one-to-one mapping between constructs in the original representation of the sources and ontology element types, e.g. an entity in a data model pattern was mapped to a concept in the ontology language.

The initial set of patterns was tested during the development of a requirements engineering ontology in the context of the research project SEMCO, see [6]. A subset of the patterns (randomly selected) were evaluated together with domain experts. Based on the initial feedback the patterns were updated (details in [6]), and some were removed from the catalogue due to their low understandability. The remaining patterns were translated into OWL, since this language had by this time emerged as the W3C recommendation. This additionally meant removing a few patterns, since not all translated well into OWL, i.e. some inherently assumed an information modelling paradigm not easily translatable into Description Logics.

The remaining patterns were now tested in a rerun of the SEMCO experiment, as well as two other research projects, see [5], and compared to a set of more general patterns from the ODP portal. The interesting conclusion was that even though the

[6]http://www.gong.manchester.ac.uk/odp/html/index.html
[7]http://www.w3.org/2001/sw/BestPractices/OEP/
[8]http://ontologydesignpatterns.org/wiki/Submissions:ContentOPs
[9]http://ontologydesignpatterns.org

patterns in the domain-specific catalogue were intended for the product development domain, some patterns (and parts of patterns) were actually applicable also in new domains (in this case university education and agriculture).

3.2 OWL 'Best Practices' and ODPs

Based on experiences from these experiments a subset of the initial catalogue, 19 patterns, were considered to have been very valuable, hence worth exposing to the ontology pattern community. This set was selected to be published in the ODP Portal. Before publishing however, we decided to apply some of the best practices encoded in other ODPs already present in the portal, in order to (i) increase the model quality (i.e. by applying modelling best practices) of the re-engineered pattern candidates in our catalogue, and to (ii) align our ODPs to existing ones, i.e. reuse existing patterns where applicable instead of redefining constructs.

This process was performed as an ODP-driven ontology evaluation, taking each candidate ODP from our catalogue at a time and identifying (1) modelling issues that did not follow OWL best practices, and (2) overlap with existing ODPs in the ODP portal. For each issue (1) we re-modelled our candidate ODP to apply the existing best practice. This included changes such as adding inverse relations, adding comments and labels, as well as changing the model to remove particularities introduced by the source pattern, e.g. an extra class representing a many-to-many relation in a database setting (in an OWL model this class has no added value). For each issue (2) we replaced the overlapping part of our candidate ODP by importing the existing ODP and if necessary specializing it, i.e. introduce the original terminology rather than the abstract one of the imported ODP.

One major issue we found, related to (1), i.e. modelling best practices, was the size and overlap of our candidate ODPs. In accordance with cognitive principles ODPs should be small enough so that all aspects can be visualized at the same time, and the main intent grasped more or less immediately [12]. Some of our candidate ODPs had between 20 and 40 classes, and covered several aspects, rather than *one* specific modelling problem. Additionally, several candidate ODPs covered common concepts, hence, we decide to extract these common parts.

Table 2: The pattern candidates.

Pattern name	Key terms
Action	action, plan
Analysis Modelling	analysis approach, modelling
Communication Event	event, contact mechanism, relationship
Employee Department	employee, department
Engineering Change	change, impact, status, specification
Information Acquisition	elicit, record, documentation
Organisation	organisation, informal, legal
Part specification	product, part, specfication
Party	organisation, person, classification
Person	name, birth date, gender
Planning Scheduling	plan, allocate, order
Position	party, position, fulfillment
Product	product, good, service
Product Association	product, complement, substitute
Product Category	product, category, classification
Product Feature	feature, applicability, feature interaction
Requirement	requirement, product, internal, customer
Requirement Description	requirement, product, feature, deliverable
Requirements Analysis	analysis, modelling, acquisition
Validation Testing	criteria, strategy, metric
Work Effort	effort, requirement

One such 'new' pattern was the Product pattern, representing information about products. This structure was previously present in several of the other patterns, but in the final revision the product information was collected in a separate pattern, which is then imported by other patterns where needed. Similarly, information about requirements is now represented separately, since it recurred in several patterns. The final set of 21 patterns are listed in Table 3.2. Due to space restrictions we are not able to go into details of all the patterns, however an example is presented in section 3.4. The 'key terms' in the table are concept labels, or parts of labels, from the pattern that can be seen as example keywords indicating the topic of the pattern. These patterns are now being submitted to the ODP Portal repository of patterns, to await scrutiny by the pattern community and the quality committee of the ODP Portal.

3.3 Experiences

From the re-engineering effort, and the subsequent evaluations and refinement of the patterns, we have gained a number of valuable experiences. One experience is related to the nature of the sources of re-engineering. Since none of the sources were expressed in a formal language, but rather descriptions and example models in a book, the direct mapping method that we applied created some less than optimal solutions. For example, the data model sources commonly contained entities such as 'other x', indicating a concept that would contain 'the rest' in an otherwise non-exhaustive partition. Such entities are relevant when transforming for example an ER-model into a relational database, since there needs to be a table storing these 'other x'. While, when using ontologies, this restriction is not present, an instance can simply have the type of the superclass, if this is not defined as an exhaustive partition, hence such 'other x' entities should not be transformed into concepts during re-engineering, but only be seen as an indication of a non-exhaustive partition.

When considering data model patterns they are commonly tailored for a smooth transformation to a relational database, hence they often contain 'extra' entities representing the many-to-many relations that require an additional table in the relational model. This is an additional problem that is not present in OWL/RDF, where data is stored in triples. Thereby, identifying these superfluous entities and avoiding to transform them to concepts is essential for arriving at an appropriate model. Additionally, when using ontologies the definition of certain concepts is more intuitive than when using a relational model, hence concepts such as gender should in an ontology be axiomatized while in a relational model this will most often be a simple character string programmatically restricted to accept certain values. Cases where ontologies give an opportunity to define certain concepts more explicitly than allowed by other models, e.g. OWL axioms instead of literals, should be identified.

Some pattern candidates were, in the last refinement step, split into several patterns, to decrease redundancy. To identify what are the natural borders and relations between patterns is an important step. When studying the existing patterns in the ODP Portal, some reuse opportunities were also discovered, where general patterns, such as 'sequence'[10] and 'partOf'[11], were reusable by the patterns in the catalogue. In the current version of the pattern catalogue, several patterns reuse more general patterns by importing and specializing them. This is important from two respects; (1) to align the ODP candidates to existing well-known Content ODPs, and (2) to reduce redundancy in the catalogue of proposed ODPs by reusing existing solutions instead of re-defining them.

[10]http://ontologydesignpatterns.org/wiki/Submissions:Sequence

[11]http://ontologydesignpatterns.org/wiki/Submissions:PartOf

3.4 Example Pattern

Due to space restrictions we cannot present the complete catalogue in detail, however we here give a representative example, i.e. a Content ODP named **Action**, in order to illustrate the nature of the patterns. It represents the relations between different types of actions, the state of the actions, and defines the relation between a plan and a set of proposed actions. Figure 1 shows a diagram of the Action Content ODP. In addition to the diagrammatic representation Content ODPs are commonly described using a number of catalogue entry fields (c.f. software pattern templates), such as *name*, *intent*, *consequences*, and *building block* (linking to an OWL realization of the pattern).

4 Related Work

In parallel with this work another typology of ontology patterns was developed, as described in [12]. The typologies, and the terminologies, are related and complementary. They represent two perspectives of ontology patterns. The classification schema proposed in this paper takes a top-down approach, aiming to cover the complete development process of an ontology, in terms of the drill-down from highly abstract requirements to how the ontology is represented. In [12] the authors instead take a bottom-up approach, starting from what activities during the ontology life-cycle are actually performed using patterns, or following best practices. Hence, the terminologies are overlapping, and most of the types of [12] can be classified under one of the main categories presented in this paper, as long as the focus is on constructing the actual ontology. Related (non-design) activities are not covered by our approach. Our approach focuses on a coherent classification, where the differentiating notions between levels and subdivisions are uniform, rather than solely related to current practice (as in [12]).

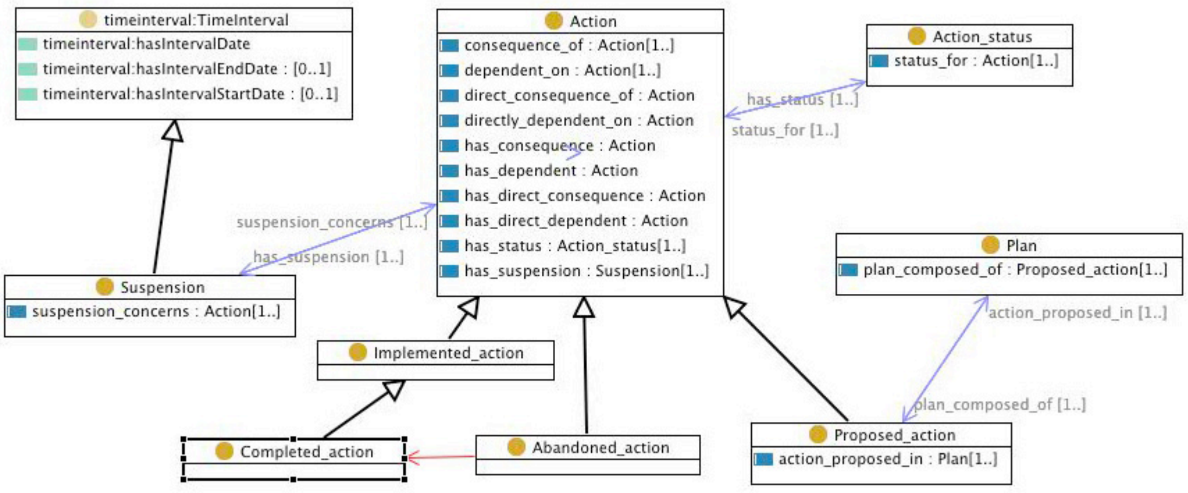

timeinterval:TimeInterval
timeinterval:hasIntervalDate
timeinterval:hasIntervalEndDate : [0..1]
timeinterval:hasIntervalStartDate : [0..1]
Action
consequence_of : Action[1..]
dependent_on : Action[1..]
direct_consequence_of : Action
directly_dependent_on : Action
has_consequence : Action
has_dependent : Action
has_direct_consequence : Action
has_direct_dependent : Action
has_status : Action_status[1..]
has_suspension : Suspension[1..]
Action_status
status_for : Action[1..]
has_status [1..]
status_for [1..]
suspension_concerns [1..]
has_suspension [1..]
Suspension
suspension_concerns : Action[1..]
Plan
plan_composed_of : Proposed_action[1..]
action_proposed_in [1..]
plan_composed_of [1..]
Implemented_action
Completed_action
Abandoned_action
Proposed_action
action_proposed_in : Plan[1..]

Figure 1: The Action Content ODP's graphical representation in UML (based on the OWL model).

The main difference in terminology is concerned with the term 'design'. While [12] denotes all patterns 'ontology *design* patterns', referring to the focus on *designing* ontologies, our notion of ontology design patterns has a narrower scope, and refers to patterns related to the actual logical design of the ontology, i.e. solving logical modelling problems. Also, the notion of ontology architecture pattern is slightly different; in this paper it is used only for patterns that are focused on the overall structure of the ontology, disregarding logical representation, while [12] in addition uses it for logical restrictions on the design level that concern the complete ontology, i.e. mixing two levels of abstraction.

Related work on collecting catalogues of ontology patterns include the ODP Portal[12] where patterns are collected in a Wiki-based interface, letting any community member contribute new patterns and review other users' patterns. Other pattern catalogues also exist, such as the logical patterns collected by the W3C[13] and by the University of Manchester[14]. A recent pattern re-engineering effort was described in [10], concerning the transformation of a component library into ODPs, however, we are not aware of any other similar efforts for bootstrapping ODPs.

5 Current Tool Support

Some ontology patterns are suitable for description as abstract templates that can be reused as general 'ideas' when engineering ontologies, while other types of patterns can be more formally represented and reused as concrete building-blocks. In this section we briefly describe two tools for supporting the development and reuse of ontology patterns; the ODP Portal[15] intending to support the complete range of ontology patterns, and the XD Tools Eclipse plugin mainly focused on support for Content ODPs as reusable OWL building blocks.

The ODP Portal is a semantic wiki portal for collecting, distributing and discussing ontology patterns. Technically it is based on the Semantic Media Wiki[16] framework, and plugins such as the Evaluation Wiki Flow [8]. The portal is divided into areas devoted to different types of patterns, where the most prominent so far is the Content ODP area. Each area contains a catalogue of submissions and a certified catalogue. All members of the portal community are free to upload patterns in the submissions catalogue, while the certified catalogue is moderated by the portal's quality committee consisting of ontology engineering and pattern experts. Patterns are certified through a

[12]http://ontologydesignpatterns.org
[13]http://www.w3.org/2001/sw/BestPractices/OEP/
[14]http://www.gong.manchester.ac.uk/odp/html/index.html

[15]http://www.ontologydesignpatterns.org
[16]http://semantic-mediawiki.org/wiki/Semantic_MediaWiki

peer review process, following a certification request from the pattern author. Important features of the portal also include the possibility to provide open reviews, i.e. any community member can spontaneously review any pattern, and the possibility to post modelling issues and discuss possible solutions and patterns. The portal additionally provides pattern training opportunities, and has been used to support events such as the pattern track at WOP2009[17] (Workshop on Ontology Patterns co-located with ISWC2009).

The eXtreme Design Tools[18] (XD Tools) is an Eclipse plugin supporting the XD methodology for ontology design pattern reuse, as described in [19]. It is compatible with ontology engineering environments such as the NeOn toolkit[19] and TopBraid Composer[20]. The XD Tools currently focuses on supporting the reuse of content ontology design patterns (Content ODPs). Content ODPs are commonly posted as reusable OWL building-blocks (in the ODP Portal, see above) that can be specialized into ontology modules realizing particular requirements of a concrete application ontology. The XD Tools supports the specialization process through a wizard, enforcing best practices such as adding labels and comments to all entities and adding inverse properties when applicable. Additionally XD Tools includes functionality for ontology analysis, based on best practices and 'rules of thumb' for ontology design, as well as an annotation wizard for describing the ontology modules built (based on ontology annotation patterns imported into the tool).

6 Conclusions and Future Work

In this paper we have presented a top-down ontology pattern typology, based on the notion of ontology patterns as a development aid while engineering an application ontology. The main difference to other proposed typologies is the coherence of the different levels, i.e. they are uniformly distinguished. The proposed typology is also consistent with common practice in software engineering where the terminology is defined in a similar way, based on the abstraction level of the patterns. We believe that having a coherent terminology and a set of characteristics with which to describe ontology patterns is essential, and will facilitate communication between both researchers and ontology engineers. Future work with respect to the pattern typology is to further study patterns on the different levels, and exemplify them, e.g. patterns are so far are not present on the architecture level.

We also focused on a specific level, i.e. Content ODPs, and showed how such patterns can be re-engineered from sources similar to ontologies, in order to populate a pattern catalogue. In our case the catalogue was designed for the product development application ontology domain. The re-engineering effort has given us a set of valuable experiences that can be used as guidelines when re-enginereing similar knowledge sources. Future work includes to conduct more evaluation experiments using the patterns in the catalogue. The patterns will also be scrutinized by the ontology patterns community through the ODP Portal.

Finally, according to the motivation presented at the beginning of this paper we need to make it easier to construct good application ontologies, even for non-experts and common web developers. We believe that having patterns on all the levels of the presented typology will ease the building process, however more tool support is still needed. In this paper we presented two examples of current tools that particularly support the reuse of Content ODPs, however in the future ontology patterns on different levels of abstraction can support the introduction of ontology engineering environments more similar to CASE tools in software engineering, where the developer is guided through a structured process and best practices are enforced.

Acknowledgements

This work was supported by Jönköping University, Sweden, through research projects SEMCO (Swedish KK-foundation grant no. 2003/0241), MediaIlLOG (grant from the foundation Carl-Olof och Jenz Hamrins Stiftelse), and DEON (financed by STINT). The tools presented in section 5 were developed by the STLab ISTC-CNR, Italy, in the context of the EU-FP7 IP NeOn.

[17]http://ontologydesignpatterns.org/wiki/WOP2009:Main
[18]Available from: http://stlab.istc.cnr.it/stlab/XDTools
[19]http://www.neon-toolkit.org
[20]http://www.topquadrant.com/products/TB_Composer.html

References

[1] Thomas Albertsen and Eva Blomqvist. Describing ontology applications. In *Proceedings of the 4th European Semantic Web Conference (ESWC07)*, 2007.

[2] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, second edition edition, 2003.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American Magazine*, 2001.

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal On Semantic Web and Information Systems*, 5(3):1–22, 2009.

[5] Eva Blomqvist. *Semi-automatic Ontology Construction based on Patterns*. PhD thesis, Linköping University, Department of Computer and Information Science at the Institute of Technology, 2009.

[6] Eva Blomqvist and Annika Öhgren. Constructing an enterprise ontology for an automotive supplier. In *Engineering Applications of Artificial Intelligence*, volume 21, April 2008.

[7] Eva Blomqvist and Kurt Sandkuhl. Patterns in Ontology Engineering: Classification of Ontology Patterns. In *Proceedings of the International Conference on Enterprise Information Systems 2005*, Miami Beach, Florida, May 24-28 2005.

[8] Enrico Daga, Valentina Presutti, and Alberto Salvati. http://ontologydesignpatterns.org and evaluation wikiflow. In *Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP2008), Rome, Italy, December 15-17,2008*, CEUR Workshop Proceedings, 2008.

[9] Martin Fowler. *Analysis Patterns - Reusable Object Models*. Addison-Wesley, 1997.

[10] Aldo Gangemi and Vinay K. Chaudhri. Representing the component library into ontology design patterns. In *Proc. of the Workshop on Ontology Patterns (WOP 2009)*, volume 516. CEUR Workshop Proceedings,, 2009.

[11] Aldo Gangemi and Peter Mika. Understanding the Semantic Web through Descriptions and Situations. In *Proc. of the International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2003)*, Catania, Italy, 2003.

[12] Aldo Gangemi and Valentina Presutti. Ontology design patterns. In *Handbook on Ontologies, 2nd Ed.*, International Handbooks on Information Systems. Springer, 2009.

[13] Karen Gardner, Alexander Rush, Michael Crist, Robert Konitzer, and Bobbin Teegarden. *Cognitive Patterns - Problem-solving Frameworks for Object Technology*. Cambridge University Press, 1998.

[14] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering*. Springer, 2004.

[15] Michael Gruninger and Mark S. Fox. The role of competency questions in enterprise engineering. In *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, 1994.

[16] Nicola Guarino. Ontology and Information Systems. In *Proceedings of FOIS'98*, pages 3–15, 1998.

[17] David C. Hay. *Data Model Patterns - Conventions of Thought*. Dorset House Publishing, 1996.

[18] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 539–545, Nantes, France, July 1992.

[19] Valentina Presutti, Enrico Daga, Aldo Gangemi, and Eva Blomqvist. extreme design with content ontology design patterns. In Eva Blomqvist, Kurt Sandkuhl, Francois Scharffe, and Vojtech Svatek, editors, *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington D.C., USA, 25 October, 2009*, volume 516. CEUR Workshop Proceedings, 2009.

[20] Len Silverston. *The Data Model Resource Book - A Library of Universal Data Models by Industry Types*, volume 2. John Wiley & Sons, 2001.

[21] Len Silverston. *The Data Model Resource Book - A Library of Universal Data Models for All Enterprises*, volume 1. John Wiley & Sons, 2001.

[22] Alistair Sutcliffe. *The Domain Theory - Patterns for Knowledge and Software Reuse*. Lawrence Erlbaum Associates, 2002.

[23] Ondrej Sváb-Zamazal and Vojtech Svátek. Analysing ontological structures through name pattern tracking. In Aldo Gangemi and Jerome Euzenat, editors, *Proceedings of EKAW 2008*, volume 5268 of *Lecture Notes in Computer Science*, pages 213–228. Springer, 2008.

[24] Frank van Harmelen, Annette ten Teije, and Holger Wache. Knowledge engineering rediscovered: towards reasoning patterns for the semantic web. In *Proceedings of the 5th International Conference on Knowledge Capture (K-CAP 2009), September 1-4, 2009, Redondo Beach, California, USA*, pages 81–88. ACM, 2009.

The INFUSIS Project – Data and Text Mining for *In Silico* Modeling

Henrik Boström[1,2], Ulf Norinder[3], Ulf Johansson[4], Cecilia Sönströd[4], Tuve Löfström[4], Elzbieta Dura[5], Ola Engkvist[6], Sorel Muresan[6], Niklas Blomberg[6]

[1]Informatics Research Centre, University of Skövde, [2]Dept. of Computer and Systems Sciences, Stockholm University, [3]AstraZeneca R&D Södertälje, [4]School of Business and Informatics, University of Borås, [5]Lexware Labs, [6]AstraZeneca R&D Mölndal

Abstract
The INFUSIS project is a three-year collaboration between industry and academia in order to further the development of new effective methods for generating predictive and interpretable models from machine learning and text mining to solve drug discovery problems.

Introduction

One of the most intensive areas of research within the pharmaceutical industry today is to collect and analyze data on absorption, distribution, metabolism, excretion and toxicity (ADMET) [1]. The overall purpose is to learn how various compounds interact with the human body in order to guide drug development projects in the search for promising compounds. Specifically, compounds unsuitable as drug candidates, e.g., due to toxicity, should be detected as early as possible. Hence, a lot of effort is spent on *front loading* the drug development projects, i.e., a substantial amount of analysis is invested in the very early phases of the projects.

Currently, a commonly adopted approach is to leverage large libraries of chemicals (acquired or synthesized to meet stringent quality criteria)) and use high-throughput screening (HTS) to test for biological activity. Promising compounds found in this way become the focus for continued research, which typically leads to further synthesis and screening. Synthesis and screening processes are, however, often time consuming and costly, making it desirable to estimate the biological activity, as well as ADMET properties, before synthesis. When computer software is used for this initial modeling, the procedure is referred to as in silico modeling [1]. If successful, in silico modeling saves much time and investments by excluding non-promising compounds, thus allowing earlier focus on drug candidates with high potential.

Several aspects of *in silico* modeling are investigated in the INFUSIS project[1] (*INformation FUSion for In Silico modeling in pharmaceutical research*), which is a collaboration of University of Skövde, University of Borås, AstraZeneca AB and Lexware Labs, running from January 2009 to December 2011, with funding from the Swedish Knowledge Foundation and the industrial partners. The research problems

[1] www.his.se/infusis

addressed by the project include handling of uncertainty of measurements or descriptors, improve interpretability of predictive models as well as advancing ensemble techniques to improve predictive performance and robustness. The INFUSIS project also tries to improve predictive modeling by fusing information from various sources such as unstructured texts where corpus technology is used to uncover relevant information. The challenges of the addressed problems and some results that have been achieved so far are presented in the following sections.

Handling uncertainty

This part of the project investigates the question: to what extent can information on descriptor value uncertainty be exploited to improve *in silico* modeling. Standard decision tree and forest learning algorithms have been extended with the ability to build models from uncertain data specified by probability distributions rather than specific values. Empirical investigations on selected *in silico* modeling datasets with uncertain data have been undertaken comparing different strategies for representing uncertainty and strategies for handling such distributions during tree building.

Different approaches to handling uncertain numerical features have been explored when using the random forest algorithm for generating predictive models. The two main approaches are: i) sampling from probability distributions prior to tree generation, which does not require any change to the underlying tree learning algorithm, and ii) adjusting the algorithm to allow for handling probability distributions, similar to how missing values typically are handled, i.e., partitions may include fractions of examples. In [2], an experiment with six datasets concerning the prediction of various chemical properties was presented, where 95% confidence intervals were included for one of the 92 numerical features. In total, five approaches to handling uncertain numeric features were compared: ignoring the uncertainty, sampling from distributions that are assumed to be uniform and normal respectively and adjusting tree learning to handle probability distributions that are assumed to be uniform and normal respectively. The experimental results show that all approaches that utilize information on uncertainty indeed outperform the single approach ignoring this, both with respect to accuracy and area under ROC curve. A decomposition of the squared error of the constituent classification trees shows that the highest variance is obtained by ignoring the information on uncertainty, but that this also results in the highest mean squared error of the constituent trees. In [3], a similar experiment was presented on predicting product quality in a casting process.

Future work includes extending the empirical investigation to a larger number of datasets and also to a larger number of uncertain features. Another direction for future work includes investigating the effectiveness of the two main approaches (sampling vs. distributing fractions of examples) also for uncertain categorical features.

Increasing interpretability

When interpretable models are required, additional demands such as brevity, i.e., important relationships are described with as few rules as possible, can be placed on models. An important issue is to develop algorithms that are able to optimize such properties. Furthermore, it is desirable that any parameters of such algorithms are easy to use and that they affect the results in a reasonably predictable way, e.g., allowing users to trade various interpretability properties against each other and also against different accuracy measurements. This project studies the use of *in silico* concept description modelling for drug discovery. The focus has so far been on techniques producing decision trees and ordered rule sets, also called decision lists.

The decision list algorithm Chipper [4], specifically aimed at concept description, has so far been evaluated in two different studies on medicinal chemistry data sets. In [5], three different decision list algorithms (JRip, PART and Chipper) were evaluated on a data set concerning the interaction of molecules with a human gene that regulates heart functioning (hERG). The main results were that decision list algorithms can obtain predictive performance not far from the state-of-the-art method random forests, but also that algorithms focusing on accuracy alone may produce complex decision lists that are very hard to interpret. The experiments also showed that by sacrificing accuracy only to a limited degree, comprehensibility (measured as both model size and classification complexity, i.e., the average number of tests needed for a classification) can be improved remarkably.

In [6], the task studied was how to obtain accurate and comprehensible QSAR models. The data sets used were 8 publicly available medicinal chemistry datasets, with six different feature sets containing up to 1024 attributes. Three techniques (J48 decision trees and JRip and Chipper decision lists) were evaluated on predictive performance, measured as accuracy, and comprehensibility, measured as model size. The results on accuracy showed that J48 obtains superior accuracy, followed by Chipper, and then JRip. On comprehensibility, the results were reversed; JRip obtained the smallest models, followed by Chipper, with J48 producing the largest models. Regarding the effect of feature reduction on accuracy, all techniques were seen to benefit from feature reduction, which almost always resulted in increased accuracy. For model size, however, feature reduction was seen not to be universally beneficial; only J48 produced smaller models for the reduced datasets, while both decision list algorithms actually produced larger models on average. The overall conclusion is that, for these datasets, there exists a definite trade-off between accuracy and interpretability.

Future work consists of a more detailed study of the effect of feature reduction on decision lists and further development of the Chipper algorithm.

Another way of obtaining interpretable models is to generate transparent representations of opaque models, an activity named rule extraction. We have previously developed a rule extraction algorithm based on genetic programming, called G-REX. [7].

A recent addition to G-REX [8] is the ability to explicitly focus on extracting rules for a specific set of instances, similar to transductive learning. Another recent study [9] utilizes the inherent inconsistency of genetic search to form an imaginary ensemble, which is then used as a guide when selecting one specific tree, as the comprehensible model.

Current work includes further development of the G-REX framework, but also hybrid techniques producing comprehensible models. More specifically, we intend to explore the connection between rule extraction and techniques utilizing semi-supervised and transductive learning.

Advancing ensemble techniques

One major open research problem concerns the relationship between ensemble diversity and accuracy, which is not completely understood, especially for classification problems. Furthermore, several different studies show that the correlation between proposed diversity measures and test set accuracy is remarkably low, see e.g., [10,11]. Because of this, there is no widely accepted diversity measure that can be used for ensemble design. Currently, various researchers instead try very different approaches, resulting in a steady stream of highly specialized and quite technical ensemble algorithms.

Naturally, when presenting a novel algorithm, the implicit claim is that the new algorithm, in some aspect, represents the state-of-the-art. Obviously, the most important criterion is predictive performance, typically measured using either accuracy or AUC. A recent study [12], using 32 publicly available data sets from the drug discovery domain, however, showed that several straightforward techniques producing ANN ensembles were more than able to match the performance of the widely acknowledged ensemble techniques GASEN [13] and NegBagg [14]. Especially NegBagg, which is a fairly recent algorithm, was constantly outperformed by most of the standard bagging versions included in the study. Nevertheless, the results for GASEN were even more striking, showing that it was most often detrimental to apply GASEN at all. Or, put in another way, creating an ensemble of all available ANNs was normally a stronger choice than using the subset suggested by GASEN.

This project investigates how diversity measures can be utilized for choosing and combining models to further improve predictive performance. The overall goal is to develop a robust method that effectively incorporates measures of diversity to produce highly accurate ensemble models. Based on the findings in [12], further development of straightforward techniques, which only implicitly target diversity, is prioritized.

Another current study investigates how feature reduction should be applied to ANN ensemble training. Naturally, feature reduction in general has been heavily investigated, but studies targeting feature reduction for classifiers specifically trained to be part of ensembles are quite rare. Our algorithm, aimed at producing “optimal” different feature sets for the base classifiers, is based on genetic search and uses fitness functions combining accuracy and diversity measures.

Finally, it could be noted that more accurate ensembles would probably be beneficial for black-box rule extraction techniques.

Fusing information from multiple sources

Unstructured texts are among the most important additional information sources. Of particular interest are reports with experimental data involving chemical processes important in tracing a certain biochemical activity, e.g., toxicity. Two tasks must be performed in order to obtain relevant data from texts in biochemistry. The first one is a special named entity recognition task: compound names and chemical processes need to be identified in free text. The other one is mapping of the names identified in texts to compounds in some suitable database. In this task, name ambiguity and variability constitute the two chief problems to be addressed [15].

We use text corpus technology tools to uncover relevant information from texts. Culler is an information retrieval system based on natural language processing. It allows versatile and precise data extraction from natural language processed text collections, called corpora. Culler is adapted in the project to allow finding names of chemical compounds. The adapted tool may hence be used to compile new sets of compounds. At the moment there are over 3,000 names of chemical substances available as one concept class in queries in Culler corpora, available at http://bergelmir.iki.his.se/culler/.

One corpus, called Diabetes, is a selection of about 200,000 abstracts on diabetes from PubMed. Patterns of the actual use of chemical nomenclature in research texts have been extracted from this corpus. There are significant differences in how terms are registered in lexicons and how they are actually used [16], making the task of proper identification of chemical compounds in texts a non-trivial task despite availability of large libraries of chemical compounds. Chemlist is the library used in the project [17]. The text sources encompass a broad selection of PubMed abstracts on obesity. The selection counts about 860,000 abstracts and it is currently being turned into a Culler corpus.

Concluding remarks

The INFUSIS project aims to contribute with tools and techniques for data and text analysis to support decision making in the domain of medicinal chemistry. In particular, presented and planned contributions include handling of uncertain data, generating interpretable models, utilizing diversity and feature reduction for ensembles, and using text analysis to compile compound sets related to biochemical activities.

Acknowledgments

This work was supported by the INFUSIS project (www.his.se/infusis) at the University of Skövde, Sweden, in partnership with University of Borås, AstraZeneca, Lexware Labs and the Swedish Knowledge Foundation under grant 2008/0502.

References

[1] H. van de Waterbeemd and E. Gifford, “Admet in silico modelling: towards prediction paradise?” Nat Rev Drug Discov, vol. 2, no. 3, pp 192–204, 2003.

[2] H. Boström and U. Norinder, “Utilizing Information on Uncertainty for In Silico Modeling using Random Forests”, Proc. of the 3rd Skövde Workshop on Information Fusion Topics, pp 59-62, 2009.

[3] C. Dudas and H. Boström, “Using uncertain chemical and thermal data to predict product quality in a casting process”, Proc. of the First ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data, pp 57–61, 2009.

[4] U. Johansson, C. Sönströd, T. Löfström and H. Boström, “Chipper – A Novel Algorithm for Concept Description”, Proc. of the Scandinavian Conference on Artificial Intelligence, pp 133-140, 2008.

[5] C. Sönströd, U. Johansson, U. Norinder, and H. Boström, “Comprehensible Models for Predicting Molecular Interaction with Heart-Regulating Genes”, Proc. of the International Conference on Machine Learning and Applications, pp 559 – 564, 2008.

[6] C. Sönströd, U. Johansson and U. Norinder, “Generating Comprehensible QSAR models”, Proc. of the 3rd Skövde Workshop on Information Fusion Topics, pp 44-48, 2009.

[7] U. Johansson, R. König and L. Niklasson, “Rule Extraction from Trained Neural Networks using Genetic Programming”, Proc. of the International Conference on Artificial Neural Networks, supplementary proceedings, pp 13-16, 2003.

[8] U. Johansson and L. Niklasson, “Evolving Decision Trees Using Oracle Guides”, Proc. of the IEEE Symposium on Computational Intelligence and Data Mining, pp 238-244, 2009.

[9] U. Johansson, R. König, T. Löfström and L. Niklasson, ”Using Imaginary Ensembles to Select GP Classifiers”, EuroGP, 2010, In Press.

[10] L. I. Kuncheva and C. J. Whitaker, “Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy”, Machine Learning, (51):181-207, 2003.

[11] U. Johansson, T. Löfström and L. Niklasson, “The Importance of Diversity in Neural Network Ensembles - An Empirical Investigation”, Proc. of the International Joint Conference on Neural Networks, pp 661-666, 2007.

[12] U. Johansson, T. Löfström and U. Norinder, “Evaluating Ensembles on QSAR Classification”, Proc. of Skövde Workshop on Information Fusion Topics, pp 59-62, 2009.

[13] Z.-H. Zhou, J.-X. Wu and W. Tang. “Ensembling Neural Networks: Many Could Be Better Than All”, Artificial Intelligence, Vol. 137, No. 1-2:239-263, 2002.

[14] M. M. Islam, X. Yao, S. M. Shahriar Nirjon, M. A. Islam and K. Murase, “Bagging and boosting negatively correlated neural networks”. IEEE transactions on systems, man, and cybernetics, Part B: Cybernetics, 38(3):771-84, 2008.

[15] Y. Tsuruoka, J. McNaught, S. Ananiadou, “Normalizing biomedical terms by minimizing ambiguity and variability“, BMC Bioinformatics, Vol. 9, No. Suppl 3, 2008.

[16] E. Dura, O. Engkvist and S. Muresan, “Names of chemical compounds within drug discovery context“, Proc. of the 3rd Skövde Workshop on Information Fusion Topics, pp 55-58, 2009.

[17] K. M. Hettne, R. H. Stierum, M. J. Schuemie, P. J. M. Hendriksen, B. J. A. Schijvenaars, E. M. van Mulligen, J. Kleinjans, and J. A. Kors, “A dictionary to identify small molecules and drugs in free text“, Bioinformatics, September 16, 2009.

Constraint Programming for Random Testing of a Trading System

Roberto Castañeda Lozano
School of Information and Communication Technology
KTH Royal Institute of Technology, Sweden

Winner of SAIS Best AI Master's Thesis Award 2010

Roberto is awarded the prize for his excellent thesis that shows how constraint programming, a classical AI technique, can be used to allow automated random testing of a trading system. The thesis combines a strong theoretical foundation with a thorough empirical evaluation. The developed techniques also lead to the discovery of unknown faults and specification defects in a widely commercially deployed financial trading system.

Abstract

Financial markets use complex computer trading systems whose failures can cause serious economic damage, making reliability a major concern. Automated random testing has been shown to be useful in finding defects in these systems, but its inherent test oracle problem (automatic generation of the expected system output) is a drawback that has typically prevented its application on a larger scale.

Two main tasks have been carried out in this thesis as a solution to the test oracle problem. First, an independent model of a real trading system based on constraint programming, a method for solving combinatorial problems, has been created. Then, the model has been integrated as a true test oracle in automated random tests. The test oracle maintains the expected state of an order book throughout a sequence of random trade order actions, and provides the expected output of every auction triggered in the order book by generating a corresponding constraint program that is solved with the aid of a constraint programming system.

Constraint programming has allowed the development of an inexpensive, yet reliable test oracle. In 500 random test cases, the test oracle has detected two system failures. These failures correspond to defects that had been present for several years without being discovered neither by less complete oracles nor by the application of more systematic testing approaches.

The main contributions of this thesis are: (1) empirical evidence of both the suitability of applying constraint programming to solve the test oracle problem and the effectiveness of true test oracles in random testing, and (2) a first attempt, as far as the author is aware, to model a non-theoretical combinatorial double auction using constraint programming.