Proceedings of IJCAI-97 Workshop: Empirical AI

Toby Walsh (editor)
APES Group, Department of Computer Science
University of Strathclyde, Glasgow
tw@cs.strath.ac.uk

IJCAI-97 workshop. *International Joint Conference on Artificial Intelligence*
Edited by Toby Walsh
Published in 1997 by the Internatical Joint Conference on Artificial Intelligence.

Introduction

Many areas of AI now use large scale experimentation to guide research. Such empirical studies have provided important new results. For example, computational experiments have identified phase transition phenomena in many domains. As another example, progress in several different areas of AI is measured by means of regular competitions on large benchmark libraries (e.g. speech recognition and first order theorem proving).

This one day workshop is open to all members of the AI community and will explore the role of empirical studies across AI. Attention will be given both to the traditional factors found in other empirical sciences (for example, the statistical analysis of results) and to some of the novel features of computational experiments (for example, how to measure performance in a machine or implementation independent way). The call for participation for the workshop identified some possible topics for discussion:

- methodology for computational experiments
- measurement of algorithm performance
- problem sets (e.g. benchmark libraries, hard random problems)
- statistical analysis of computational results
- role of computational models in experiments
- case studies (e.g. novel results obtained via computational experiments)

The domains of interest of the participants cover many different areas of AI including agents, constraint satisfaction, knowledge based systems, machine learning, planning, robotics, satisfiability, scheduling, temporal reasoning, time-tabling and theorem proving.

Invited Speaker

Bart Selman selman@research.att.com

Programme Committee

Bernhard Nebel nebel@informatik.uni-freiburg.de
Christian Suttner suttner@informatik.tu-muenchen.de
Toby Walsh tw@cs.strath.ac.uk

Programme

9.30-11.00, The ambiguous status of experimental computer science.

Invited talk by Bart Selman of AT&T Laboratories. Followed by discussion.

Experimental work in computer science is often met with some skepticism. The argument goes that since we're dealing with well-defined artifacts, we should be able to rigorously *prove* properties and behaviors of computational phenomena. I will show that there is a proper role for experimental work in computer science. In fact, I'll argue that the scientific method of the natural sciences, with its close interaction between experimental and theoretical work, is a sine qua non for scientific progress in computer science.

11.30-12.45, Random problems.

Panel. Jeremy Frank, Ullrich Hustadt, David Poole, Toby Walsh (chair).

Love them, or hate them? Random problems have done so much to invigorate empirical studies. But what *exactly* are the pitfalls of using random problems? Can we surmount these pitfalls? If so, how?

14.00-16.00, Tales from the frontline.

Four invited presentations describing problems and successes in empirical studies.

John Slaney and Sylvie Thiebaux,	Phase Transitions and Optimality: Sense and Nonsense.
Tim Menzies,	Empirical Validation of Knowledge Engineering Methodologies.
Pandu Nayak and Brian Williams,	Overthrowing conventional wisdom about truth maintenance.
Andrea Schaerf,	Scheduling of university courses and sports.

16.30-17.45, Comparing apples with oranges?

Panel. Adele Howe, Pedro Meseguer, Jane Mulligan, Bernhard Nebel (chair).

The comparisons of AI algorithms and systems can often feel like the comparison of apples and oranges. How do we compare algorithms and systems that appear at first sight to have little in common other than CPU time? What exactly do we compare? And how do we chose a suitable set of problems for benchmarking?

17.45-18.00, So long and thanks for the fish.

General discussion led by Toby Walsh.

Do we have any final conclusions? Are there any obvious trends emerging from these workshops on empirical methods in AI? Do we want to make plans for future workshops?

Participants

Anbulagan	anbulagan@utc.fr
Rina Dechter	dechter@ics.uci.edu
Dieter Fensel	dfe@aifb.uni-karlsruhe.de
Jeremy Frank	frank@cs.ucdavis.edu
Alfonso Gerevini	gerevini@ing.unibs.it
Louis Hoebel	hoebel@ai.rl.af.mil
Adele Howe	howe@cs.colostate.edu
Ullrich Hustadt	hustadt@mpi-sb.mpg.de
Mateja Jamnik	matejaj@dai.ed.ac.uk
Kazuya Kaneko	neko@swl.cl.nec.co.jp
Chu Min Li	cli@laria.u-picardie.fr
Tim Menzies	timm@insect.sd.monash.edu.au
Pedro Meseguer	pedro@sinera.iiia.csic.es
Jane Mulligan	mulligan@cs.ubc.ca
Pandurang Nayak	nayak@ptolemy-ethernet.arc.nasa.gov
Bernhard Nebel	nebel@informatik.uni-freiburg.de
David Poole	poole@cs.ubc.ca
Francesca Rossi	rossi@di.unipi.it
Andrea Schaerf	aschaerf@dis.uniroma1.it
Bart Selman	selman@research.att.com
John Slaney	John.Slaney@anu.edu.au
Antje Strohmaier	antje@intellektik.informatik.th-darmstadt.de
Sylvie Thiebaux	thiebaux@irisa.fr
Toby Walsh	tw@cs.strath.ac.uk
Brian Williams	williams@ptolemy-ethernet.arc.nasa.gov

Statements of Interest

Mean Height Verus Width of a Search Treee

Anbulagan & Chu Min Li
LaRIA, Universite de Picardie Jules Verne, Amiens, France

It is well known that random 3-SAT problems are very hard to solve on the average when the ratio of clauses to variables is approximatively equal to 4.25 and some problems are much harder than others. The phenomenon appears to be beyond argument when one uses an algorithm based on the popular Davis-Putnam-Loveland procedure (DPL). One might believe that if a problem is harder to solve by a DPL procedure, it is because the search tree constructed is higher on the average.

The experimental result is of practical interest, because one of objectifs to implement a DPL procedure actually is to try to minimize the mean height of the search trees which is sometimes used to indicate the performance of a DPL procedure. According to our experimentation, we prove that the mean height of a search tree is somewhat irrelevant in the difficulty of random 3-SAT problems when using a DPL procedure and if a problem is harder, it is only because the search tree constructed is wider. Thus, we argue that the essential objectif when designing a DPL procedure is to try to minimize the width (instead of the mean height) of the search tree, which roughly implies branching on the variable allowing to give more and stronger constraints to the subproblems.

We found that the search trees constructed by our highly optimized DPL procedure at a given problem size n (number of variables of initial problem) can be divided into two phases versus their depth. Within the first phase, a node generally has two sons and the number of nodes is increased when descending in the trees; within the second phase, the tendency is reversed and more than a half of nodes are leaves representing a dead end. The widest level is at the mid-depth approximatively equal to n/20. All search trees at a given problem size n have the same mean height and a random 3-SAT problem is harder than another because its associated search tree is substantially wider, showing that if a problem is harder it is because there is no or very few short sequences of assumptions (partial assignments of truth values to variables) allowing to falsify it.

Assumption Hunting as Developing Method for Efficient Problem Solvers

Dieter Fensel
Institut AIFB, University of Karlsruhe

The concept problem-solving method (PSM) is present in a large part of current knowledge-engineering frameworks [3]. In general a PSM describes which reasoning steps and which types of knowledge are needed to perform a task. Such a description should be domain and implementation independent. PSMs are used in a number of ways in knowledge engineering: as a guideline to acquire problem-solving knowledge from an expert, as a description of the main rationale of the reasoning process of the expert and the knowledge-based system (KBS), as a skeletal description of the design model of the KBS, and to enable flexible reasoning by selecting methods during problem solving. During the last years, PSMs have become quite successful in describing the reasoning behavior of KBS. However, there is still no solid theoretical background in characterising the precise competence of PSMs and in providing guidelines for developing reusable PSMs and for adapting these PSMs to application specific circumstances. Recent work tries to achieve both by characterising PSMs in terms of their underlying assumptions.

Reasoning about real-world problems is only possible by introducing assumptions about these problems. Such assumptions are used to properly relate the input and the output of the reasoning process with the actual problem. They are necessary to reduce the complexity of the reasoning task and the development process of the reasoning system. Most problems tackled with KBSs are inherently complex and intractable (cf. [1,4]) and therefore require the introduction of limitations. A PSM can only solve such tasks with reasonable computational effort by introducing assumptions that restrict the complexity of the problem or by strengthening the requirements on domain knowledge [2].

As a consequence the important question arises how to get such assumptions. In this paper, we introduce a systematic approach for constructing such assumptions. We propose a method and a tool for this purpose. The main idea is to use mathematical proofs and analysis of their failure as a systematic means for forming assumptions. A mathematical proof that a PSM solves a given problem usually enforces the introduction of assumptions to close gaps in the line of reasoning of the proof. It can therefore be viewed as a search process for hidden assumptions. Gaps that can be found in a failed proof provide already first characterizations of these assumptions. An assumption that implies the required lemma necessary to close the gap is a possible candidate we are looking for. That is, formulating this lemma as an assumption is a first step in finding and/or constructing assumptions that are necessary to ensure that the competence of a PSM is strong enough to achieve the goals as they are defined by the task.

Using an open goal of a proof directly as an assumption normally leads to very strong assumptions. That is, these assumptions are sufficient to guarantee the correctness of the proof, but they are often neither necessary for the proof nor realistic in the sense that application problems will fulfil them. Therefore, further work is necessary to find improved characterizations for these assumptions. This is achieved by a precise analysis of their role in the completed proof that is used to retrace unnecessary properties of them.

We provide tool support for this process by adapting the Karlsruhe Interactive Verifier (KIV) [5] for our purpose. It is the interactive character of the underlying tactical theorem prover of KIV that makes it suitable for hunting hidden assumptions. Instead of returning with a failure KIV returns with open goals that could not be solved during its proof process.

References

[1] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson: The Computational Complexity of Abduction, Artificial Intelligence, 49: 25-60, 1991.

[2] R. Benjamins, D. Fensel, and R. Straatman: Assumptions of Problem-Solving Methods and Their Role in Knowledge Engineering. In Proceedings of the 12. European Conference on Artificial Intelligence (ECAI-96), Budapest, August 12-16, 1996.

[3] M. David et al. (eds.): Second Generation Expert Systems, Springer-Verlag, 1991.

[4] B. Nebel: Artificial intelligence: A Computational Perspective. In G. Brewka (ed.), Essentials in Knowledge Representation, Springer Verlag, 1996.

[5] W. Reif: The KIV-System: Systematic Construction of Verified Software, Proceedings of the 11th International Conference on Automated Deduction, CADE-92, LNCS 607, Springer-Verlag, 1992.

The Coupon Collectors' Complaint: A Cautionary Note on Random Problem Instance Generation

Jeremy Frank
University of California at Davis, California, USA

1 Introduction

A frequent statement in empirical algorithm studies is: "We generated random problem instances of N variables and C constraints in order to test the effectiveness of our algorithm." In many cases, the authors make an implicit assumption that their problem instances do indeed contain exactly N variables and C constraints. This assumption can extend to the design of algorithms to solve these instances; in many cases the number of variables and constraints appear as parameters to algorithms and no explicit test is performed to actually count the number of variables or constraints. The potential results of this last oversight for empirical studies include incorrect reporting of means and variances for algorithm complexity, and thus a mistaken impression of the performance of algorithms.

In this note I discuss the random generation of SAT problem instances and graphs according to the most common generation algorithms used in empirical studies. I relate this process to the Coupon Collector's problem and show that for small numbers of variables the standard algorithms indeed have the specified number of variables and constraints. I confirm this analysis with some empirical data. Finally, I comment on the issue of random problem generation.

2 Random Problem Instance Generation

Random problem generation has been used to test algorithms for a long time. The issue of poor problem generation is not new; Chvatal and Reed (?) attacked a poorly constructed SAT generator by Goldberg which purported to demonstrate that SAT problems were generally easy to solve. Simply creating sufficiently random numbers can be a problem; Williams (Colin's new book, published yet?) contains an excellent discussion concerning the dangers of poor random number generators both in randomized algorithms and in problem instance generation. Random problem generation most recently became prominent in the empirical algorithms community as a result of Cheeseman et. al. in which they showed that certain problem instance distributions exhibited a sharp threshold in satisfiability. Numerous empirical studies since then have made extensive use of random problem instance generators to demonstrate their results.

```
procedure Uniform3-SAT(Σ,C,N)
    for i=1 to C
      for j=1 to 3
        Select 1 of N − j + 1 variables uniformly
        Assign negative sign with probability 1/2
        Remove selected variable from consideration
      end for
    Store clause
    end for
end
```

Figure 1. Random Problem Generation Algorithm Sketch.

Consider the method for generation of random 3-SAT problem instances shown in figure 1. This algorithm accepts as parameters the number of variables N and the number of clauses C and returns a problem instance Σ. The algorithm generates C clauses by selecting 3 distinct variables from the N possible variables and assigning each a negative sign with probability $\frac{1}{2}$. For SAT problems, it has been shown by Cheeseman et. al. and by Crawford and Auton that problem instances generated using this procedure with about $4.3N$ clauses are on the threshold of unsatisfiability and require the most time to solve on average, and as a result numerous empirical studies employ this algorithm to generate test problem instances. An analagous situation occurs for graph

coloring problems, in which graphs of $4.5V$ edges are on the threshold for 3-colorability. However, these studies assume that each problem instance so generated has exactly N variables. (Clearly each problem instance has exactly C clauses.) Is this assumption well-founded? To answer this question requires a lesson in probability.

3 The Coupon Collector's Problem

The Coupon Collectors problem is as follows. A coupon collector must collect at least one of M coupons in order to win a new car. Each coupon the collector acquires is equally likely to be one of the M coupons. How many coupons must the collector acquire on the average in order to get all M? Motwani and Raghavan discuss this problem and show that, on the average, the collector needs to acquire $M \log M + O(M)$ coupons to have one of each. Furthermore they show that the number of coupons is unlikely to deviate far from it's expectation.

4 Analysis

The implications for random SAT problem instance generation are as follows. We can view the process of generating a random instance as collecting N coupons. Each clause of a 3-SAT problem instance can be viewed as collecting 3 coupons, and so the process of generating an instance with C clauses can be approximately viewed as collecting $3C$ coupons. If $C = cN$, then we collect $3cN$ coupons. So as N grows large we may not collect enough coupons to have one copy of each, since we require $N \log N + O(N)$ on average to collect them all. All is not lost, however, if we look at the relationship between $\log N$ and $3c$. As stated, typically c is near 4.3 and $\log N > 3c \rightarrow N > e^{3c}$. When $c = 4.3$ this means that we need to generate problem instances with over $400,000$ variables in order to have a chance of not having all N variables occur in a problem instance; however if $c = 3$ we need only generate problem instances of 8000 variables in order to run into trouble.

We can also view the situation by deriving exactly the probability that some variable does not appear in a problem instance. We begin by showing that the probability that a variables does not appear in a clause is $\frac{N-3}{N}$. Clauses are generated by selecting 3 variables without replacement from N. The variable is not the first variable in the clause with probability $\frac{N-1}{N}$, it is not the second variable with probabilty $\frac{N-2}{N-1}$ and is not the third with probability $\frac{N-3}{N-2}$; the product of these is $\frac{N-3}{N}$. Clauses are generated independently, and so the probability that the variable is not in any of C clauses is $\frac{N-3}{N}^C$. In fact, the probability distribution for the number of clauses a variable is in can be represented as a binomial distribution with $p = \frac{3}{N}$ (p represent the probability a variable is in) and with C trials.

Now we really want the probability that all variables are in the problem instance, so let $P_v = \frac{N-3}{N}^C$. Thus $1 - P_v$ is the probability that a variable is in a problem instance. The probability that one variable is in a problem instance is independent of the probability of any other variable being in the instance; it only depends on N and C. Hence the probability that all variables are in the problem instance is $(1 - P_v)^N$, and therefore the probability that some variable is not in the problem instance is $1 - (1 - P_v)^N$. For example, with $N = 100$ and $C = 430$ we have $P_v = 2.05 \times 10^{-6}$, resulting in a negligible probability that some variable is not in the formula. Few empirical studies to date have used problem instances with more than 1000 variables. But with $N = 100$ and $C = 300$ we have $P_v = 1.04 \times 10^{-4}$,resulting in $1 - (1 - P_v)^N = 0.001$. Hence 1 in 100 problem instances are likely to have at least one variable missing. For problems with 1000 variables and 3000 clauses we have $P_v = 1.22 \times 10^{-4}$,resulting in $1 - (1 - P_v)^N = 0.011$. Here, 11 problems out of 1000 are likely to have at least one variable missing.

How many variables are likely to be missing? The expected number of missing variables should be accurately represented by NP_v since there are N variables. For $N = 1000$ and $C = 3000$ the expected number of missing variables is 0.12, so even if a variable is missing the expected number of missing variables is quite small.

I conducted an experiment using code to generate random problem instances with 1000 variables and 3000 clauses. I generated 1000 such problem instances and checked the variables to see if any variables were not in a problem instance. I found 905 problem instances with no variable missing, and the number of variables missing varied from 1 to 2. I repeated the experiment for problem instances with 1000 variables and 4300 clauses and found that only 14 problem instances had variables missing and that the number of missing variables was always 1. Why are the results of the experiments slightly different than the probability model? It is possible that the random generation algorithm in the code is not sufficiently robust. It is also possible that the model is slightly wrong; the assumption that the probability that one variable is in a problem instance is independent of the probability of any other variable being in the instance may be incorrect.

A similar analysis can be done for random graph generation algorithms, and indeed the work of Erdös and Whitney discusses the probability that randomly generated graphs are connected.

5 Conclusions

The analysis above shows that, for now, random problem instance generation is unlikely in the extreme to produce problem instances with missing variables except for large problems, and even if variables are missing that there are unlikely to be very many, again except for large problem instances. Most empirical studies to date have analyzed problem instances with fewer than 1000 variables, so there is little reason to suspect the results of these studies. However, it is disturbing that so many empirical studies were done without careful consideration of this problem. For instance, numerous other issues face researchers who use random problem instance generators. This note should serve as a warning to the empirical algorithms community to carefully analyze their generation algorithms, lest they become victims of the Coupon Collector's Complaint.

Acknowledgements

I would like to thank Phil Rogaway of U.C. Davis for asking how many variables a randomly generated problem instance with 100 variables actually has. I would like to thank Chip Martel of U.C. Davis for mentioning the Coupon Collectors' problem and, indirectly, inspiring the title of this note. Finally, I would like to thank John Stutz for helping with the analysis.

Statement of Interest

Alfonso Gerevini
University of Brescia, Italy

I am very interested in all the areas mentioned in the workshop description, since they are all important for my research. My main research interests in artificial intelligence concern constraint-based reasoning (in particular temporal reasoning), planning, and knowledge representation and reasoning in general. The focus of my research is on the computational aspects, especially on the design of efficient reasoning algorithms and systems.

In the last years I have proposed (with L. Schubert) some successful graph-based algorithms and data structures for efficiently reasoning about time (e.g., AIJ-95), and for accelerating partial-order planning (e.g., JAIR-96). Other recent work (with others) concerns algorithms for tractable temporal constraint satisfaction problems (e.g., IJCAI97).

Currently my major research interests concern the development of methods for efficient domain-independent planning. This involves the development of some general techniques that can be used for improving current planners (e.g., UCPOP, Graphlan, Satplan), as well as of new planning representations and algorithms. The experimental analysis of such techniques is an important part of this study. In this context I have recently conducted a large-scale empirical investigation, that was aimed at testing the sensitivity of prominent search strategies for partial-order planning to the order of the operator preconditions. This experimental work was based on solving more than 24,000 planning problems, where the order of the operator preconditions is randomized, and required some weeks of CPU-time.

Statement of Interest

Louis J. Hoebel
Rome Laboratory/C3CA, Rome NY, USA

Temporal reasoning is important to many application areas including robotics, natural language, causal reasoning, scheduling and planning. At FLAIRS-97, I presented a paper that gives a framework for constructing a domain independent tractable temporal reasoner for large scale problems without giving up expressiveness of constraints. The effectiveness of this approach was demonstrated with a series of empirical evaluations using basic control strategies, showing that simple composable control strategies are useful and efficient. The framework and strategies provide a reasonable and useful trade-off between relative completeness and increasing computational performance.

I am also a program manager for the "Evaluation of Intelligent Systems", http://www.ai.rl.af.mil/EIS. A foundation of which is the work of Paul Cohen, note his book on Empical Methods for AI. Large portions of his book and stats package program are included in the web site.

I have also been involved in evaluation (empirical) of planners in both a programmatic and scientific context. A paper on that is "Handbook of Evaluation for the ARPA/Rome Laboratory Planning Initiative" in the Morgan Kaufman Proceeedings of the ARPA/Rome Laboratory Planning Initiative Workshop, Tuscon Ariz. Feb. 1994

Statement of Interest

Adele Howe
Computer Science Department, Colorado State University, Fort Collins, USA

A fundamental practice in empirical AI is the comparison of approaches. A common technique for demonstrating progress is to show that the performance of a new algorithm is superior to the previous best. Unfortunately, all too often these demonstrations amount to comparing apples and oranges, i.e., performance is different but it is difficult to fairly identify which is best.

In performance comparisons, implementations of two or more algorithms are run on a set of test problems, their performance is measured on quantifiable metrics (either standardly accepted or designed to address a hypothesis of the algorithm design), and the metrics are compared statistically. While such experiments provide a clear structure, adequately controlling these experiments to find a fair basis of comparison is complicated by a host of problems, for example:

- different operational goals of algorithm design, e.g., we never directly compared SavvySearch to Meta-Crawler (two meta-search engines on the WWW) because SavvySearch focused on the front-end, how to best select search engines to search, and Meta-Crawler focused on how best to analyze the returned search results.
- high representational demands for coding new problems, e.g., planning systems require domain theories and problem implementations which can be costly to develop.
- benchmark problems are poorly understood or were developed without a specific purpose, e.g., the problem set distributed with the UCPOP planner was contributed by a variety of people over several years.
- test problems or code for existing algorithms are unavailable, e.g., a student told me she could not compare her algorithm to a published one because she was unable to find the author of the previous one.

The first two issues have been partly addressed by community supported efforts, e.g., MUC and the robot competitions. The competitions provided incentive and direction for separate research groups to work on a common goal.

For the third issue, researchers need to develop sharable benchmarks for validating specific hypotheses and to analyze existing benchmarks. For example, we recently conducted a comparison of search techniques in the UCPOP planner in which we systematically modified the standard benchmark problems to control, in small part, for differing skills in building the problems. From this study, we were able to determine which problems were most and least vulnerable to this factor.

For the fourth issue, repositories such as the UC Irvine site for Machine Learning problem sets provide a mechanism for sharing problems. Several groups (mine at CSU, Paul Cohen's at University of Massachusetts and Sterling Research) are developing a repository for evaluation of AI systems in general. The web site, which will be made public in early June, will store tutorials, benchmark problems, code for analysis, examples of experiment designs and pointers to useful sites. A web available statistics package will expedite constructing hypertext research papers that will have the data and analyses embedded for the reader (see http://guaraldi.cs.colostate.edu:4936/ for a prototype).

References

[1] D. Dreilinger and A.E. Howe. Experiences with Selecting Search Engines using Meta-Search. To appear in "ACM Transactions on Information Systems".

[2] S. Rana, A.E. Howe, L.D. Whitley, K. Mathias. A Genetic Algorithm Scheduler: Assessing the Contribution of Search, Heuristics and the Objective Function. To appear in "Engineering Design and Automation Journal".

[3] A.E. Howe and E. Dahlman. "Characterizing Domain Specific Effects in Flaw Selection for Partial Order Planners.", Accepted as poster at IJCAI-97.

[4] A.E. Howe and L.D. Pyeatt. Constructing Transition Models of AI Planner Behavior. In "Proceedings of the 11th Knowledge-Based Software Engineering Conference", September 1996.

[5] A.E. Howe and P.R. Cohen. 1995. Understanding Planner Behavior. In "Artificial Intelligence", Special Issue on Planning. Vol. 76, No. 1-2, pp. 125–166.

Empirical Evaluation of Modal Theorem Provers

Ullrich Hustadt
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Together with Renate Schmidt I'm currently working on modal theorem proving and its empirical evaluation.

There have not been very many evaluations and comparisons of the computational behaviour of modal theorem provers based on different methodologies. Our impression is that empirical performance evaluation is not done seriously in this research area.

In the main program of IJCAI, I will present a paper reporting on our own work comparing the performances of KRIS, KSAT, the Logics Workbench and first-order resolution. Our paper highlights that it is difficult to set up a challenging benchmark suites of randomly generated modal formulae.

Statement of Interest

Kazuya Kaneko
Software Research Laboratory, C&C Research Laboratories, NEC Corporation,
Kawasaki, Japan

My research interests involve scheduling expert systems and combinatorial optimization, especially, general-purpose high-quality assignment algorithms based on constraint satisfaction approach.

Since 1993, I have been working on the development of a constraint relaxation problem solver, COASTOOL (constraint-based assignment and scheduling tool), and a high-school timetabling system on COASTOOL [1,2]. COASTOOL has a constraint-based architecture. Its novel problem-solving method uses an arc-consistency algorithm to generate a high-quality initial assignment, then refines the assignment with a hill-climbing algorithm.

My current research work is improving the problem-solving method to obtain higher quality. One of the improved methods is an improved initialization method that tries to construct a consistent assignment incrementally, applying a repairment algorithm in the case of violation at every assignment [3]. Another improved method is an improved repairment method that escapes from local minima based on two characteristics of high-school timetabling problems; no class can have two or more lessons at a time, and no class has a free time-slot during a week. It tries to repair two lessons of a class at once in similar way of billiards. Namely, it moves a lesson in collision to another time-slot, then it hits another lesson there, push it out to a hole [4].

I evaluated them with large-scale and tightly-constrained practical high-school timetabling problems by COASTOOL experiments. They have more than 1,000 variables and various constraints with characteristics of each high-school. As a result, incorporating the both kinds of improvement dissolved almost all constraint violation for the normal tightly-constrained problems, and 30% of penalty points for the most tightly-constrained problems.

References

[1] M.Yoshikawa, K.Kaneko, T.Yamanouchi, M.Watanabe, "A Constraint- Based High School Scheduling System," IEEE Expert, Feb, 1996, pp.63-72.

[2] M.Yoshikawa, K.Kaneko, Y.Nomura, M.Watanabe, "A Constraint-Based Approach to High School Scheduling Problems: A Case Study," 12th AAAI, vol.2, 1994, pp.1111-1116.

[3] K.Kaneko, M.Yoshikawa, T.Yamanouchi, M.Watanabe, "Proposal and Evaluation of a new Initial Assignment Method for Large High-school Timetabling Problems," 54th Annual convention on IPSJ, vol.2, 1997, pp.321-322 (in Japanese).

[4] K.Kaneko, M.Yoshikawa, T.Yamanouchi, M.Watanabe, "Proposal and Evaluation of Constraint-Relaxation Method in a School Timetabling System," 9th Annual convention on JSAI, 1995, pp.483-486 (in Japanese).

Statement of Interest

Tim Menzies
Department of Software Development, Monash University

Motivations for more empirical work. Situation cognition (SC) is such a motivator. If knowledge is situated, then a symbolic knowledge base will require knowledge mainteance (KM) as the situation changes. That is, SC motivates KM. However, empirical KM assesses SC. If we could demonstrate that the process of change dictacted by weak SC is tamed via the current generation of KM strategies, then the case for strong SC disappears.

Methods for exploring empirical issues. to explore (e.g.) the above issue, we need to record what is changed during KM. Based on a reverse engineering of strategies from requirements engineering (RE) , knowledge-level modeling, knowledge acqusition (KA), case-based reasoning (CBR), software engineering (SE) and machine learning (ML), we say that strategies have been offered to maintain six types of knowledge: terms, assertions, procedures, quality, inconsistency, and behavioural libraries. These strategies perform some combiantion of five knowledge activities: acquisition, operationalisation, faulting, fixing, and perserving. In the case of knowledge acquired from multiple sources, these strategies may also perform two other activities: recognising and handling conflicts. To date, no single strategy maintains all six types of knowledge or performs all seven activities.

Reluctant conclusion the empirical evaluation problem for comparing KA/SE/RE/ML/CBR methodologies is MUCH MUCH BIGGER than I previously thought and CANNOT BE CURRENTLY RESOLVED without research in the six-times-seven space described above.

Statement of Interest

Pedro Meseguer
Institut d'Investigacio en Intelligencia Artificial, CSIC, Bellaterra, Spain

My interest for Empirical AI comes from my research on algorithms and heuristics for CSP and MAX-CSP problems. Given that these problems are intractable in the worst case, algorithms and heuristics are often evaluated empirically. For this reason, we need a fair evaluation methodology which should include both (a) a justified selection of benchmark problems, and (b) a fair comparison of algorithm performance.

Regarding benchmark problems, most of empirical studies use the random CSP model working on the peak of maximum difficulty. This model has represented a significant step forward for better benchmarking (remember test problems in early CSP papers: N-queens and Zebra). However, this model also has some limitations to be taken as a general CSP model, because it assumes (i) same domain size for every variable, (ii) homogeneity in constraint distribution among variable pairs, and (iii) uniformity in constraint tightness. Some CSP classes comply with these assumptions, but not every CSP class. Therefore, conclusions of evaluation studies on random CSPs may not be applicable for every CSP class. Considering real problems, typically they do not comply with (some of) the assumptions mentioned above. Therefore, when solving real problems, some discrepancies might be expected with respect to algorithmic performance established on random CSPs. In addition, real problems are not as hard as having just one single solution on average, so it is debatable whether the peak top is the right point for algorithm testing, or a wider window including the whole peak (and specially the left slope when considering solvable instances) would be more appropriate.

Regarding performance comparison, most of researchers provide the number of consistency tests, as a measure of the search effort, and some researchers provide the number of visited nodes, as a measure of how focused the search is. These parameters try to evaluate the search process, in a way that is machine and implementation-independent. On simple algorithms, these parameters are quite representative of the search work. However, on more sophisticated algorithms keeping some kind of cache of previous results, the number of consistency checks may not be as representative as in the previous case. For this reason, more parameters are of interest such as the number of table lookups performed by the algorithm, differentiating among the different tables kept in memory. As a complementary parameter, the CPU time is also of interest, in order to estimate the average real cost of each parameter (CPU time by consistency check, by table lookup or by visited node).

In summary, I want to attract attention on the limitations of the current random CSP model in order to produce a better, more representative one. Independently, I suggest that more search parameters are needed to measure the search effort required by sophisticated CSP algorithms.

Statement of Interest

Jane Mulligan
Department of Computer Science, University of British Columbia, Canada

The empirical studies and benchmarks that have given us insight into the performance of many types of AI systems in recent years, have only begun to be employed in the intelligent robotics community in the form of mobile robot competitions such as the ROBO-CUP to occur at IJCAI97. This is perhaps because of the relative immaturity of many of the technologies and methods used in these systems.

Rigorous analysis and evaluation of real implemented robotic systems for intelligent tasks are rarely performed. Such systems are often extremely complicated, depending not only on 'interesting' theoretical parameters and models, but on many assumptions and constants which may be set almost arbitrarily. The information required to achieve good task performance includes all of these acknowledged and unacknowledged parameters. In the absence of models and specifications which take the full complexity of such systems into account, the process of elucidating task requirements will inevitably be experimental and cyclical.

If we view any robotic task implementation as a parameterization imposed on the underlying physical computation required, we can evaluate such a system's performance as a function of the parameters and information it uses. We can then compare different task implementations on this basis. Fractional factorial experiments are well understood experimental designs which allow us to establish the statistically significant parameters and parameter interactions for a task parameterization. Factorial experiments give us a starting point for determining which measurements, and their interactions, define the task subspace. Sensitivity analysis and extended complexity models such as information-based (ϵ) complexity give us tools to measure and improve our system models.

We have performed experiments to demonstrate this methodology, based on a part-orienting task implemented in two very different ways. One system uses a 'sensorless' model-based push-orienting method, the other uses a real-time stereo vision system to localize objects in the workspace for sensor-driven part orienting. The parameters used to represent manipulated parts for the two systems are similar, though not identical sets. Through detailed experiments we established the statistically significant parameters and parameter interactions for each system, then applied sensitivity analysis to set optimal parameter values and explore the nature of interactions. Finally, ϵ-complexity was proposed as a cost/performance metric to give us a means of counting the computation and sensing costs to achieve the observed system error rates.

Experimental analysis is a necessary step to understanding integrated intelligent systems which are often large and multifaceted. It reveals aspects of system implementations which cannot easily be predicted in advance, and gives a clear picture of the task requirements, given the strengths and weaknesses of the observed system.

Statement of Interest

P. Pandurang Nayak and Brian C. Williams

NASA Ames Research Center, Moffett Field, California, USA

Experimentation is centrally important in Artificial Intelligence. In a research enterprise dominated by intractable problems, only a series of well-designed experiments can provide evidence for (or against) the efficacy of improved search algorithms.

Experiments have played two important roles in our research:

1. Experimental analysis of algorithms: Conventional wisdom has largely pushed deductive reasoning out of the reactive control loop for nearly a decade. However, our recent experience with Livingstone, a reactive configuration manager for an autonomous system, provides dramatic empirical evidence to the contrary [1], thereby unifying the perceived dichotomy between deduction and reactivity. We achieve a reactive system that performs significant deduction in the sense/response loop by drawing on our past experience at building fast propositional inference algorithms and conflict-based algorithms for model-based diagnosis.
   Conventional wisdom is based largely on the believe that most interesting deductive problems are NP-hard, and hence intractable. However, empirical studies on the phase transition from easily solvable to unsolvable SAT, and the difficulty of generating hard problems hinted that we should reexamine this wisdom. Furthermore, a back of the envelope analysis of the propositional theories describing spacecraft suggested that the SAT problems for real world, causal systems lie far from the phase transition, and hence can be handled efficiently. Our subsequent experiments show that the core search tasks of mode identification and reconfiguration (generalized versions of diagnosis and recovery) when applied to a model of NASA's DS-1 spacecraft involving over 12,000 propositional clauses can be solved in the order of tens to hundreds of milliseconds on a typical desktop workstation. Such experimental evidence has provided key support for the use of Livingstone as part of the Remote Agent, NASA's base-line spacecraft autonomy architecture [2].
2. Providing insights that lead to novel algorithms: The real-time propositional reasoning in Livingstone is achieved by adopting an event driven approach, propagating the effects to the propositional database as sensor readings and states change. A truth maintenance system, in particular the LTMS, offers a natural starting point. While our use of an LTMS in Livingtone has been exceedingly favorable, the stringent performance requirements of real-time leaves room for improvement. Specifically, the main drawback of LTMS algorithms is the need to redo propagations that hold across a context switch (the so-called unouting problem).

In 1984 concerns about the speed of the LTMS lead de Kleer to propose a fundamentally different type of truth maintenance system—the ATMS. Unfortunately despite this initial promise, empirical evidence supporting a consistent benefit of the ATMS on real world problems has been lacking. ATMS research has dwindled with most practitioners reverting back to the simpler LTMS framework.

Revisiting the concerns voiced in 1984, we have taken a completely different approach to the LTMS problem [3]. We carried out a careful analysis of a series of context switch experiments on the above mentioned spacecraft propositional database involving more than 12,000 clauses. This analysis helped us to pinpoint a major source of performance degradation caused by the LTMS's conservative approach to guaranteeing well-founded (i.e., loop-free) support. The identification of this problem allowed us to develop a more aggressive incremental TMS, called the ITMS. We showed empirically that this new algorithm reduces the overhead of processing unchanged consequences by a factor of seven. The experiments were crucial in helping us understand the problem and evaluate our solution.

References

[1] B. Williams and P. Nayak, 1996, A Model-based Approach to Reactive Self-Configuring Systems, In Proceedings of AAAI-96.

[2] B. Pell, D. Bernard, S. Chien, E. Gat, N. Muscettola, P. Nayak, M. Wagner, and B. Williams", 1997, An Autonomous Spacecraft Agent Prototype. In Proceedings of Agents-97.

[3] P. Nayak and B. Williams, 1997, Fast Context Switching in Real-time Propositional Reasoning. In Proceedings of AAAI-97.

Irrelevance in Planning: An Empirical Study

Bernhard Nebel
Albert-Ludwigs-Universität Freiburg

It is traditional wisdom that one should start from the goals when generating a plan in order to focus the plan generation process on potentially relevant actions. However, the GRAPHPLAN system, which is currently the most efficient planning system, builds a "planning graph" by forward chaining. Although this strategy seems to work well, it may lead to problems if the planning task description contains irrelevant information. Whilst some irrelevant information can be filtered out by GRAPHPLAN, most cases of irrelvance are not identified.

With Yannis Dimopoulos and Jana Koehler, I have been analysing the effects arising from "irrelevant" information to planning task descriptions for different types of planners. Based on this analysis, we have proposed a family of heuristics that select relevant information by minimizing the number of initial facts that are used when approximating a plan by backchaining from the goals ignoring anyt conflicts. These heuristics, although not solution-preserving, turn out to be very useful for guiding the planning process, as shown by applying the heuristics to a large number of examples from the literature.

Why Not Random Experiments?

David Poole
Department of Computer Science, University of British Columbia, Canada.

One of the main goals in AI is to determine what structure there is in the world that can be exploited for reasoning. If there was no structure in the world, humans would not be able to survive. It is by exploiting this structure that we are able to reason about which actions will let us eat and reproduce. It is exactly this structure that lets simple reactive policies of insects work, for example. It is such structure that we humans exploit to make sense of the world. If the world was completely chaotic we would not survive.

My claim is that the most crucial question in building intelligence is determining what structure in the world can be exploited computationally to determine what an agent should do based on its observations. We need to map from the sense data into some internal representation (typically a model of the world) into an appropriate action. In order to do this we have to make sense of the world. The traditional approach is to describe the world in terms of variables (propositions) and then determine how these variables interact. In the most general case these variables are stochastically dependent on each other.

In order to model the world, we need to determine (a) what are the variables of interest and (b) how do they interact. One of the most important features of (b) is in determining which variables are independent of each other, which variables are dependent and in what way are they dependent.

In random experiments we can abstract away from the propositions, as it doesn't really matter what the variables mean in order to reason with them (it does when we want to interact with the world in terms of perception and action, but presuming that we can abstract some computational tasks intermediate to perception and action, we can abstract away from the meaning to the symbols for these tasks).

Now, what is wrong with random experiments?

In order to build a random experiment we have to a priori commit to assumptions of independence amongst the variables. At one extreme, this can mean that we make over-simplifying assumptions such as that variables are independent, in which case the results would seem to have little relevance to domains where they aren't independent. At the other extreme, it could mean that we allow allow arbitrary dependence amongst the variables in which case there is no structure to exploit. Intermediate to these extremes, we can allow the designer of the experiment to choose what dependencies they allow. The problem is that a "friendly" experiment designer can choose dependencies to make essentially anything work, and a hostile experiment designer can make anything not work. In such an experimental setup, the design of good algorithms probably depends more on guessing what dependencies amongst the variables the designer has imposed. Either we know this in setting up the experiment, or else the problem is one of second guessing the experiment designer.

"Nature" has already set up experiments, with exactly the structure we need to discover how to exploit. We need to find real domains, real problems and try to determine what structure in these we can exploit. There may be much more structure than we would dare to put into a random experiment and claim it is realistic! Only time will tell.

Non-Homogeneous constraint problems and hidden variables

Francesca Rossi
Dipartimento di Informatica, Università di Pisa, Italy

Often, when a real-life problem is modeled as a constraint problem, there are some parts (consisting of variables and constraints) which are needed to specify all the details of the problem, but which are not interesting when a solution of the problem is requested. It's like when in logic programming we have the variables of the goal, and then many other variables (called hidden variables here) used during the computation of an answer for that goal.

In solving a constraint problem with hidden variables, the presence of the hidden variables can be exploited to make the search for a solution more efficient. In fact, it is enough to be sure that they can be intantiated consistently with the rest of the problem, but we don't need to find any concrete instantiation for them. Therefore, sufficient conditions for the elimination of some hidden variables (and all the constraints attached to them) can be defined, depending on the structure and/or the level of consistency of the constraint problem [1]. For example, if the problem is arc-consistent, then all hidden variables of degree 1 can be eliminated without changing the set of solutions of the problem.

Algorithms which solve constraint problems with hidden variables and exploit these sufficient conditions to eliminate as many hidden variables as posible have been developed and tested on both real (toy) problems and classes of randomly-generates constraint problems. While eliminating hidden variables leads to a shorter search tree, thus providing a gain in the search effort, one has to check whether the search for the hidden variables needs more time than that gained by the elimination, since if this is so then the overall time would be penalized. What emerged is that if the problem is homogeneous (in density and tightness) the hidden variables elimination is convenient in very few cases (only when the density is very small). However, one may note that most real-life problems are not homogeneous, and indeed experiments on classes of non-homogeneous randomly generated constraint problems (obtained by randomly creating holes here and there in a homogeneous constraint problem) showed a great advantage in eliminating the hidden variables. Other experiments, on classes of coloring problems, showed that on this problems the advantage appears only after a threshold in the density of the problem. However, due to the peculiar structure of these problems, it is possible to predict where such a threshold is.

References

[1] "Redundant Hidden Variables in Finite Domain Constraint Problems", F. Rossi, in "Constraint Processing", M.Meyer ed., Springer-Verlag, LNCS 923, 1995.

Statement of Interest

Andrea Schaerf
Dipart. di Informatica e Sistemistica, Universita di Roma "La Sapienza", Rome, Italy.

My research interest is in the solution algorithms (and their implementation) of hard scheduling and planning problems. I am interested in both theoretical problems (e.g. job-shop scheduling) and practical ones (e.g. school timetabling [2]).

For the solution programs, I experimented both constraint satisfaction techniques and local search ones (e.g. simulated annealing, tabu search).

Regarding constraint satisfaction, I made use of the CLP language ECLiPSe. It turned out to be a flexible tool for devising algorithms that produce the exact solution, like branch-and-bound (see [1]). On the other hand (in my experience), it showed to be computationally inadequate for large combinatorial problems which are genuinely hard to solve.

Regarding local search (or neighbourhood search), I am especially focused on the tabu search techinque [3], using which I was able to solve successfully very large timetabling problems. Such technique, which is extremely popular in the Operational Research community, in my opinion does not have in the AI community the attention it deserves. I particular, I believe it can be useful in the empirical AI field for the solution of classical AI problems (e.g. SAT).

I also propose a solution technique for scheduling and constraint satisfaction problems that combines together backtrack-free constructive methods and local search techniques [3]. The technique incrementally constructs the solution, performing a local search on partial solutions each time the construction reaches a dead-end. The new technique has been successfully experimented on two real-life problems: university course scheduling and sport tournament scheduling.

References

[1] Andrea Schaerf, 1996. Scheduling Sport Tournaments using Constraint Logic Programming. Proc. of the 12th European Conf. on Artificial Intelligence (ECAI-96).

[2] Andrea Schaerf, 1996. Tabu Search Techniques for Large High-School Timetabling Problems. Proc. of the 13th American Conf. on Artificial Intelligence (AAAI-96).

[3] Andrea Schaerf, 1997. Combining Local Search and Look-Ahead for Scheduling and Constraint Satisfaction Problems, to appear in Proc. of 15th International Joint Conference on Artificial Intelligence (IJCAI-97).

Statement of Interest

John Slaney
Australian National University, Canberra, Australia

Sylvie Thiebaux
IRISA Rennes, France

During the past 2 years we have worked on a project on optimisation and near-optimisation problems in planning. We have particularly carried out an experimental study of Blocks World (BW) as an example of a non-trivial domain in which near-optimal planning is tractable while optimal planning is NP-hard. Results concerning algorithms for near-optimal planning in this domain were presented at AAAI last year. We have two other sets of results: some mathematical theorems concerning the average properties of BW and some experiments on optimal solution. We intend to present the second of these in the workshop, as a case study illustrating methodological problems as well as results. Closely related problems include finding shortest circuits in graphs, so broadly similar effects are expected for a larger problem class.

We first investigated a conjecture of Selman that the peak of difficulty for the basic BW problem with n blocks would coincide with cases where the number of towers in the initial and goal states is about the square root of n. Our results suggest that this estimate is too low, and that the square root of $2n$ may be closer to the truth. In order to get reliable data, we generated and (optimally) solved some millions of problem instances with 100 blocks, for example. For our experiments, we had to solve the nontrivial problem of generating uniformly distributed random data for the domain.

We next investigated the phase transition associated with the probability of existence of a plan of length k for a problem of size n. The transition is very clear, and scaling with our conjectured limiting value of $n - \sqrt{n}$ we obtained a good crossover as expected. We have no exact method of generating uniformly distributed problem instances with a given minimum plan length k. However, we expect to obtain a reasonable estimate by showing a high correlation between median plan length and the numbers of towers in the initial and goal states. It is clear that the peak of difficulty observed earlier is correlated with an easy-hard-easy pattern in terms of plan length, but less clear exactly how this peak relates to the transition in the probability of existence of a solution of given length. We are currently trying to gain more understanding of this matter.

Our wider interests in empirical AI include work on theorem proving and on reasoning under uncertainty. We intend to present only the BW material in this workshop, but expect to take part in discussion of other presentations as appropriate. We also see it as useful at this point to increase our understanding of the state of the art by direct contact with others working on similar problems.

Statement of Interest

Antje Strohmaier
University of Darmstadt, Germany

My interest in particiapting in this workshops concerns the following aspect: While working on a neural network parallelization of local search algorithms, we did very many experiments with different parameter settings on SAT and CSP problems. There we not only varied one parameter per time, but tried variations of several parameters against each other. The results were interesting and may lead to new insights on how carefully experiments have to be done: We found dependencies between parameters that obviously influence theoretical interpretation of experimental results. Furthermore, it seems that at least for randomly generated problems the cross-over point does not posses that great importance that other result make us believe. In my opinion it is therefore very important to establish useful guidelines for experimental research. These should help us in producing meaningful results as well as judging the importance of reported results. Futhermore by carefully analysing the dependencies at least for random problems, it might be possible to achieve importatn results with a fraction of the computational effort used at the moment.

Some questions, that should be adressed:

- How to find really interesting benchmark problems?
- Is it reasonable to use exceptional difficult problems for the testing of algorithms, when every-day problems occurring in applications may be much easier to solve?
- Which points should be included in a catalogue of universal guidelines?
- How can we guide empirical work such that not every experiment has to be dome by every researcher?
- How can we guide empirical work such that results of different authors remain comparable?

An Epilogue?

Toby Walsh
APES Group, Department of CS, University of Strathclyde, Scotland

As the organizing chair of the workshop, and as the last statement in the notes, I would like to attempt to identify some common themes that appear in these statements of interest.

Empirical studies in AI have an increasingly important rôle to play. This is to be welcomed. AI needs an experimental side as much as, if not more than, other sciences. Indeed, many questions in AI can *only* be answered empirically. For example, it is an empirical question if Horn clauses are adequate to represent models of faulty components. This depends entirely on the sort of faulty components met in practice. Other questions in AI can be answered either experimentally or theoretically. Since theory is often hard, experiments often deliver answers more quickly than theory. But we should not avoid theory just because it is hard. It has difficult for empirical results to equal the clarity and exactness of theoretical results. Finally, some questions in AI can only be answered theoretically. Experiments may suggest that our algorithm finds solutions quickly, but a proof that the problem is NP-complete will, unless P=NP, inject a distinct note of caution into such a result.

Despite the many successes of empirical studies, there is a growing unease apparent in many of the statements of interest. Good experiments require careful design. And unlike other sciences, computational experiments can usually be run with too little ease. Experiments require a sample of problems for our algorithms to test. All too often, only random problems are used. Even if "real" problems are used, the sample size may be small, or the problems contain hidden features that can make them easy to solve. Finally, results will depend on exactly how the algorithms are implemented, and the experiments run. It is often hard to present results in a general way that can be reproduced by other scientists.

We should not be dis-heartened by such observations. Workshops like this are evidence that the research community is aware of the problem that empirical studies present. Standards are evolving fast. Benchmark libraries are common in many areas. Random problems are viewed by many with the respect and caution they deserve. Textbooks have been published (and, no doubt, are being written) passing on valuable experience from the laboratory bench.

I encourage everyone at the workshop to learn from the cautionary tales told today, to set themselves the highest standards in their experimental work, and to lead the field by example. In case you need any more encouragement, I will observe that good experiments will always yield good results, but that poor experiments rarely yield anything.