Integrating Information Gathering and Problem-Solving in Open Environments

Santiago Macho-Gonzalez and Boi Faltings

Artificial Intelligence Laboratory (LIA),
Swiss Federal Institute of Technology (EPFL),
IN-Ecublens, CH-1015 Ecublens, Switzerland,
`boi.faltings@santi.macho@epfl.ch`,
`http://liawww.epfl.ch/`

Abstract. An important realization driving the development of the semantic web is that people are interested in information in order to *solve problems*. Many researchers have considered techniques for turning information sources into a single database which can then be used for problem-solving tasks. We present a method for integrating problem-solving with information gathering and show that it achieves remarkable efficiency gains.

1 Information gathering in open systems

Information gathering is an important practical problem that arises in particular for information agents on the internet. Most recent work such as Infomaster ([1]) or Infosleuth ([2]) has focussed on turning distributed information sources into a single, coherent database that can be used as a basis for solving problems. This is particularly interesting in the context of the semantic web, which provides information in structured form that is easy to assemble.

When considered in an open environment, however, this approach presents two important problems. The first is that there is an unbounded number of information sources, and there is no clear termination criterion that indicates when the database synthesis is complete. The second is that much of the information that is gathered may turn out to be unnecessary to solve the original problem, resulting in low efficiency. Therefore, we believe that large improvements can be obtained by integrating information gathering directly into a problem-solving process. Rather than first obtaining the information and then carrying out problem-solving on that information, we gather information only as required for problem-solving. In this way, we avoid retrieving large amounts of information that plays no part in a solution to our problem, and can instead focus search on those aspects that are critical to the success of problem-solving. Experiments show that we can obtain very significant efficiency gains in this way.

In this paper, we propose *constraint satisfaction* (CSP) as a problem-solving model. Using CSP as a model for information agents has been introduced in [3, 4] and also in [7] and since used in several operational systems such as [5, 6].

2 Open Constraint Satisfaction Problems (OCSF)

Fig. 1. Elements of an open constraint satisfaction problem

We consider the setting shown in Figure 1 which reflects the important elements that occur in an open setting. The problem-solving process is modelled abstractly as the solution of a constraint satisfaction problem. The choices that make up variables domains and relations of the CSP are distributed throughout an unbounded network of information servers $IS_1, IS_2, \dots$, and accessed through a mediator ([8]).

More precisely, the CSP is an *open constraint satisfaction problem*, defined formally as follows:

Definition 1. An open constraint satisfaction problem (*OCSF*) is a possibly unbounded, partially ordered set $\{CSP(0), CSP(1), \dots\}$ of constraint satisfaction problems, where $CSP(i)$ is defined by a tuple $\langle X, C, D(i), R(i) \rangle$ where

- $X = \{x_1, x_2, \dots, x_n\}$ is a set of n variables,
- $C = \{(x_i, x_j, \dots), (x_k, x_l, \dots), \dots\}$ is a set of m constraints, given by the ordered sets of variables they involve,
- $D(i) = \{d_1(i), d_2(i), \dots, d_n(i)\}$ is the set of domains for $CSP(i)$, with $d_k(0) = \{\}$ for all k .
- $R(i) = \{r_1(i), r_2(i), \dots, r_m(i)\}$ is the set of relations for $CSP(i)$, each giving the list of allowed tuples for the variables involved in the corresponding constraints, and satisfying $r_k(0) = \{\}$ for all k .

The set is ordered by the relation $\prec$ where $CSP(i) \prec CSP(j)$ if and only if $(\forall k \in [1..n]) d_k(i) \subseteq d_k(j)$, $(\forall k \in [1..m]) r_k(i) \subseteq r_k(j)$, and either $(\exists k \in [1..n]) d_k(i) \subset d_k(j)$ or $(\exists k \in [1..m]) r_k(i) \subset r_k(j)$.

A solution of an *OCSF* is a combination of value assignments to all variables such that for some i , each value belongs to the corresponding domain and all value

combinations corresponding to constraints belong to the corresponding relations of $CSP(i)$.

Solving an OCSF requires an integration of search and information gathering. It typically starts from a state where all domains are empty, and the first action is to find values that fill the domains and allow the search to start. As long as the available information does not define enough values to make the CSP solvable, the problem solver initiates further information gathering requests to obtain additional values. The process stops as soon as a solution is found.

The problem solver accesses information on the network through a mediator ([8]). The mediator is a broker (as defined in [9]) that has to fulfill two important functions:

- a *directory* or yellow pages, it must locate information servers that can supply values that can be used in the CSP that is being solved.
- an *integrator* that reformulates information from the servers to fit a format required by the CSP.

In order to be generally useable, the mediator must be compatible with any CSP that a user might want to solve. This is achieved through ontologies that define the meaning of CSP variables and values:

Definition 2. An ontological grounding of an open CSP $\langle V, C, D(i), R(i) \rangle$ is a tuple $\langle O, M_V, M_D \rangle$ where:

- $O = \{o_1, \dots, o_m\}$ is a set of ontologies,
- M_V is a mapping $V \rightarrow O \times C(O) \times P(O)$ that maps each variable $v_j, j = 1..n$ into a tuple $\langle o_j, c_k, p_l \rangle$ which means that v_j models the property p_l (sometimes called relation in the literature) of concept c_k in ontology o_j . Note that the mapping need not be bijective, i.e. there can be several variables that map to the same concept and property.
- M_D is a mapping $V \rightarrow O \times 2^{P(O)}$ that maps each variable $v_j, j = 1..n$ into a tuple $\langle o_j, \{c_1, \dots, c_k\} \rangle$ that defines the possible values that could be part of the variable's domain.

The ontological grounding allows the problem solver to define what information is required to fill the different aspects of the problem. It is fixed throughout the problem-solving, and communicated to the mediator either once at the beginning or with each request.

The connection between CSP and information sources is based on its ontological grounding. We make the following assumptions:

- every query for more values refers to some ontology.
- all information sources relevant to an ontology are classified according to this ontology, i.e. an information source that returns values for properties or relations of a concept is indexed under that concept.

We use the technique of [10] to index information sources according to an ontology. In this technique, information servers are classified by the ontological categories of the values they hold.

3 Solving OCSF with information gathering

We now address the algorithms for solving OCSF by search interleaved with information gathering. To keep the algorithms simple, we only consider information gathering for variable domains, not for relations associated with constraints. Techniques exist for transforming discrete CSP into CSP where all relations are a form of equality, so this does not limit generality.

When a CSP has no solution, it is often the case that it contains a smaller subproblem that already has no solution. It will not be possible to create a solution by information gathering unless values are added to variables and relations of that subproblem. This fact can be used to more effectively drive information gathering. The idea is to find a variable that must be part of an unsolvable subproblem as a promising candidate for adding extra values. To develop this into a general and complete algorithm, we need to address two issues: how to identify unsolvable subproblems, and how to select all variables in turn to avoid unbounded information accesses while missing a feasible solution.

The following lemma provides the basis for identifying variables that are part of unsolvable subproblems:

Lemma 1. *Let a CSP be explored by a failed backtrack search algorithm with static variable ordering $(x_1, \dots, x_n)$, and let x_k be the deepest node reached in the search with inconsistency detected at x_k . Then x_k , called the failed variable, is part of every unsolvable subproblem of the CSP involving variables in the set $\{x_1 \dots x_k\}$.*

Proof. In order to reach x_k , the search algorithm has constructed at least one valid assignment to $x_1, \dots, x_{k-1}$, so this set of variables does not contain any unsolvable subproblem. However, there is no consistent assignment to $x_1, \dots, x_k$, so this set does contain unsolvable subproblem(s). Since the only difference is x_k , x_k must be part of all of these unsolvable subproblems. $\square$

On the basis of this proposition, we can use the results of a failed CSP search process to determine for which variable additional values should be collected. These are then passed to the mediator, which will search for relevant information on the network. When there are no additional values for this variable, the mediator returns a **nomore** message, and other variables are then considered. The resulting algorithm **fo-search** (failure-driven open search) is shown in Algorithm 1.

Algorithm 1 makes the assumption that variables are ordered by the index i . It assumes that no consistency techniques are used in the search, although the chronological backtracking can be replaced with backjumping techniques to make it more efficient.

It is possible to show the following theorem which proves completeness of Algorithm 1:

Theorem 1. *Supposed that OCSF is solvable, i.e. by calling **more** on every variable a sufficient number of times we eventually reach an instance $CSP(j)$ such that for all $m \geq j$, $CSP(m)$ contains no unsolvable subproblems. Then Algorithm 1 will eventually terminate with this solution.*

```

1: Function fo-search(X,D,C,R,E)
2:  $i \leftarrow 1, k \leftarrow 1$ 
3: repeat {backtrack search}
4:   if exhausted( $d_i$ ) then {backtrack}
5:      $i \leftarrow i - 1, \text{reset} - \text{values}(d_i)$ 
6:   else
7:      $k \leftarrow \max(k, i), x_i \leftarrow \text{nextvalue}(d_i)$ 
8:     if consistent( $\{x_1, \dots, x_i\}$ ) then {extend assignment}
9:        $i \leftarrow i + 1$ 
10:    if  $i > n$  then
11:      return  $\{x_1, \dots, x_n\}$  as a solution
12:  until  $i = 0$ 
13:  if  $e_k = \text{closed}$  then
14:    if ( $\forall i \in 1..k - 1, e_k = \text{closed}$ ) then
15:      return failure
16:  else
17:     $nv \leftarrow \text{more}(x_k)$ 
18:    if  $nv = \text{nomore}(x_k)$  then
19:       $e_k \leftarrow \text{closed}$ 
20:       $d_k \leftarrow nv \cup d_k$ 
21:  reorder variables so that  $x_k$  becomes  $x_1$ 
22:  fo-search(X,D,C,R,E) {search again}

```

Algorithm 1: *Function **fo-search** for solving OCSP.*

4 Experimental results

We tested the performance of the techniques we described on randomly generated resource allocation problems.

We compare the performance of different combinations of algorithms in the mediator and the problem solver. For the mediator, we consider a **random** selection of any information source that has the right property and concept, and **size** where information sources are additionally ordered by the number of values they carry.

For the problem solver, we consider three algorithms, **OS** for search where variables for information gathering are selected randomly, **FO** for fo-search defined earlier, and **ICSP** for the interactive CSP algorithm defined in [12].

Several metrics for measuring performance have been developed in the field of database selection ([11]). We measure performance by the ratio:

$$R = \frac{\text{Number of variables of the CSP}}{\text{Number of access to IS until a solution is found}}$$

Since each variable must have at least one value, solving the CSP requires at least one information source access per variable, so that the ideal value for R is 1. Smaller values of R mean low efficiency. We consider that R provides a good measure of the relative amount of information gathering effort generated by the two methods, but does not take into account possible parallelization or buffering.

Fig. 2. Efficiency ratio against number of values for several combination of search/mediator algorithms.

Figure 2 plots the efficiency ratio against the average number of values available for each variable for a setting in which there are a total of 12 information servers. The more values there are for each variable, the easier the problem is to solve, and we can see that the average efficiency in general increases with the number of available values. The simulations show that integration produces huge efficiency gains over current methods that access all servers beforehand: their efficiency would only be $1/12$, whereas with integration we obtain efficiency ratios close to the theoretical optimum of 1. Furthermore, the gains would grow as the numbers of servers increases, thus making it a good tool to improve scalability. It can also be seen that the ICSP algorithm, in spite of integrating problem solving with information gathering, is not nearly as good as the search methods we presented here.

5 Related work

Within the CSP community, the work that is closest to ours is *interactive constraint satisfaction* (ICSP), introduced in [12]. Similarly to our work, in ICSP domains are acquired incrementally from external agents. ICSP focuses on settings where value acquisition can be strongly directed using constraints, but does not have an efficient and complete method for deciding what values to gather next. As a result, its performance in an open environment is actually very poor. Using CSP as a model for information agents has been introduced in [3, 4] and also in [7] and since used in several operational systems such as [5, 6]. However, this earlier work does not consider information gathering in a network of servers.

Information integration has been studied by database researchers for a long time ([13]). With the increased use of networking, and particularly the internet, projects such as TSIMMIS ([14]), Infomaster ([1]), and others have addressed dynamic integration of information in order to make the distributed network of information sources appear as a single database. InfoSleuth ([2]) has built a more elaborate infrastructure using information agents that not only integrate information, but can also provide proactive services such as subscriptions and change notifications. Our work assumes the availability of such techniques in order to transform the results of information sources into the format required for problem solving.

Another important topic is how to locate information sources that are capable of answering queries. In this paper, we assume an ontology-based classification similar to that of [10]. However, the matchmaking process can get significantly more complex. The Information Manifold ([15]) has shown that query planning techniques can be used to combine information from different information sources to obtain arbitrary relations, and this would be a useful extension to our work.

Another important issue is actual matchmaking. Decker and Sycara ([9]) investigate the efficiency of middle-agent systems, and Sycara ([16]) elaborates on their use as information agents. Techniques such as LARKS ([17]) show that much more complex matchmaking than ontology-based classification is possible, and it would be possible to derive such criteria from the problem-solver as well.

Recently, researchers in information retrieval have paid more attention to driving information retrieval from the task that users are trying to solve. Systems such as Watson and I2I ([18]) and just-in-time information retrieval ([19]) automatically retrieve information from databases, mail archives and other information sources by matching it with keywords that occur in the current activity of the user - for example, a document being prepared.

6 Conclusions

We have investigated how to integrate information gathering in a constraint satisfaction problem solver in order to increase the efficiency of accessing information servers. Assuming an information agent infrastructure that corresponds well to what is envisaged for the semantic web, we have shown experimentally that such integration leads to significant efficiency gains.

References

1. Genesereth, M. R., Keller, A. M., Duschka, O.: "Infomaster: An Information Integration System", Proceedings of 1997 ACM SIGMOD Conference, May 1997.
2. Jerry Fowler, Brad Perry, Marian Nodine, and Bruce Bargmeyer: Agent-Based Semantic Interoperability in InfoSleuth SIGMOD Record 28:1, March, 1999, pp. 60-67.
3. Marc Torrens, Rainer Weigel and Boi Faltings: " *Distributing Problem Solving on the Web Using Constraint Technology*" Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI'98), pp 42-49, IEEE Press

4. Marc Torrens, Boi Faltings and Pearl Pu: "*Smart Clients: Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems*", *CONSTRAINTS* **7**(1), 2002, pp. 49-69
5. Pearl Pu and Boi Faltings: "*Enriching buyers' experiences: the SmartClient approach*", Proceedings of the CHI2000 conference on Human Factors and Computing Systems, April 2000, pp. 289-296, ACM
6. Craig A. Knoblock, Steve Minton, Jose Luis Ambite, Maria Muslea, Jean Oh and Martin Frank: "*Mixed-Initiative, Multi-source Information Assistants*," Tenth International World Wide Web Conference (WWW10), Hong Kong, 2001.
7. Peter M.D. Gray, Suzanne M. Embury, Kit Y. Hui, Graham J.L. Kemp: "The Evolving Role of Constraints in the Functional Data Model", *Journal of Intelligent Information Systems* **12**, pp. 113-137, 1999.
8. Gio Wiederhold and Michael R. Genesereth: "The Conceptual Basis for Mediation Services" *IEEE Expert*, 12(5), 1997.
9. Keith Decker, Katia Sycara and Mike Williamson: "Middle-Agents for the Internet," *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, Morgan Kaufmann, 1997, pp. 578-583
10. José Luis Ambite and Craig Knoblock: "Flexible and scalable cost-based query planning in mediators: A transformational approach," *Artificial Intelligence* **118**, pp. 115-161, 2000
11. James C. Freanch and Allison L. Powell :Metrics for Evaluating Database Selection Techniques. 2000
12. Rita Cucchiara, Marco Gavanelli, Evelina Lamma, Paola Mello, Michela Milano, and Massimo Piccardi: "Constraint propagation and value acquisition: why we should do it interactively," *Proceedings of the 16th IJCAI*, Morgan Kaufmann, pp.468-477, 1999
13. A. Sheth and J.A. Larson: "Federated Database Systems," *ACM Computing Surveys* **22**(3), 1990
14. S. Chawathe, H. Garcia Molina, J. Hammer, K.Ireland, Y. Papakostantinou, J. Ullman and J. Widom: The TSIMMIS project: Integration of heterogeneous information sources. In *IPSSJ Conference*, Tokyo, Japan, 1994
15. Alon Y. Levy , Anand Rajaraman , Joann J. Ordille: "Querying Heterogeneous Information Sources Using Source Descriptions," *Proceedings of the 22nd VLDB Conference*, Bombay, India, 1996
16. Sycara, K. "In-Context Information Management Through Adaptive Collaboration of Intelligent Agents." In *Intelligent Information Agents: Cooperative, Rational and Adaptive Information Gathering on the Internet*. Matthias Klusch (Ed.), Springer Verlag, 1999.
17. Katia Sycara, Seth Widoff, Matthias Klusch and Jianguo Lu: "LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace." *Autonomous Agents and Multi-Agent Systems*, 5, 173-203, 2002.
18. J. Budzik, S. Bradshaw, X. Fu, and K. Hammond: "Supporting Online Resource Discovery in the Context of Ongoing Tasks with Proactive Assistants," *International Journal of Human-Computer Studies* **56**(1) Jan 2002, pp. 47-74
19. B.J. Rhodes and P. Maes: "Just-in-time information retrieval agents," *IBM Systems Journal* **39**, pp. 685-704, 2000