Retrieval and Reasoning in Distributed Case Bases[†]

M V Nagendra Prasad[1], Victor R. Lesser[1] and Susan E. Lander[2],

[1]Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{nagendra,lesser}@cs.umass.edu

[2]Blackboard Technology Group, Inc.
401 Main Street
Amherst, MA 0l002
lander@bbtech.com

Abstract

The proliferation of electronically available networked information has led researchers to examine the issues involved in developing automated methods for gathering information in response to a query from a user. However, most of this literature deals with locating, gathering and selecting the best response to a query from among a multitude of responses from different repositories or digital libraries. This paper deals with a different model of response to a query, involving composition of mutually related partial responses spread across a network of information repositories. We present a system for cooperative retrieval and composition of a case in which subcases are distributed across different agents in a multi-agent system. From a Gestalt perspective, a good overall case may not be the one derived from the summation of best subcases. Each agent's local view may result in best local cases, which when assembled may not result in the best overall case in terms of global measures. We propose a negotiation-driven case retrieval algorithm as an approach to dynamically resolving inconsistencies between different case pieces during the retrieval process.

Appeared in the Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries, Vol 7, No. 1, March 1996, pp 74 - 87. Also as UMASS CS Technical Report 95-27, 1995.

[†]The work reported here is supported in part by NSF Center for Intelligent Information Retrieval (CIIR), and Office of Naval Research contract N00014-95-1-1198. The content of the information does not necessarily reflect the position or the policy of the U.S. Government and no official endorsement should be inferred.

1 Introduction

The recent explosion in networked information resources has been attracting much attention towards automated methods for gathering information in response to a query from a user [1, 4, 24]. However, most of this literature deals with locating, gathering and selecting the best response to a query from among a multitude of responses from different repositories or digital libraries. In this paper, we deal with a different model of response to a query. No single source of information may contain the complete response to a query; this may necessitate piecing together mutually related partial responses from disparate and possibly heterogeneous sources[24]. For example, consider a multi-agent parametric mechanical design system where each agent has to contribute a component to assemble a complete design. An agent has to obtain its components from manufacturer-specified catalogue case bases that may be distributed across the Internet. For example, it may treat WWW-accessible distributed component library like PARTNET[5, 25] as its case base from which to draw its components. In addition to the constraints arising from the overall design requirements, constraints on a component also arise from the specifications of other components in the design. A response to a query in the form of a specification of design requirements consists of a compatible set of components forming the overall design.

Central to the presentation in this paper is the view that a complex query is presented to a set of agents, each of which is responsible for retrieving information relevant to a part of the query. The agents negotiate to piece together a mutually acceptable response to the query. More specifically, we deal with agents retrieving pieces of a distributed case to assemble a composite case from them. Passive data sources like databases can be transformed into pro-active agents by wrapping them with intelligent interfaces. Agent-based architectures offer concurrency, modularity, robustness, separation of concerns, and other advantages of a distributed system[20].

A multi-agent system comprises a group of intelligent agents working towards a set of common global goals or separate individual goals that interact[3]. In such a system, each of the agents may not be individually capable of achieving the global goal and/or their goals have interactions and hence a need for coordination among the set of agents. Due to its partial view of the problem-solving situation, an agent may have access only to a part of the environment, and communication bandwidth limitations and heterogeneity of representations may limit its view of other agents' state. Given these uncertainties[20], an agent may have to do evidential reasoning and focused negotiation to resolve them to the extent that it can make positive contributions to the on-going problem solving process. More specifically, in a distributed case-based reasoning system (DCBR), each agent's partial view may result in best local cases that when assembled may not result in the best overall case in terms of global measures. This gives rise to a need for the agents to cooperatively access their case bases to retrieve and assemble a good composite case. In our work on Negotiated Search[18][1] and DRESUN[6], we proposed certain general mechanisms for performing distributed search in multi-agent systems. We draw upon our experience with these methods to extend them to case-based systems in a distributed environment.

The rest of the paper is organized as follows. Section 2 presents situations where distributed case bases could arise. Section 3 discusses the Negotiated Retrieval strategy for distributed cases. Section 4 introduces the CBR-TEAM system as our domain to illustrate the algorithm, and provides a trace of the system run on an example. Section 5 presents some empirical results from runs of an

[1] In this paper, we view negotiation as a form of goal-directed communication among agents.

abstract version of CBR-TEAM. Section 6 relates our work to some of the existing literature and Section 7 concludes the paper with a discussion of the implications of the proposed mechanism and future work.

2 Distributed Case Bases

How do distributed case bases arise? A system that performs rote learning by storing good cases, where each agent stores its own local case in its case base, could give rise to a Distributed Case Base (DCB). However, this may not be the only way. In the case of a set of reusable agents[17], each agent could have acquired its own independent problem-solving experiences by participating in different teams of agents. Or a *Case-Knowledge Engineer* could design each of the case bases by giving them episodes from his collection of cases. Perhaps the most important sources of DCBs are distributed digital libraries with mutually related information, like PARTNET on World Wide Web.

A case base is a source of complex data stored in specific formats. Any unstructured database like a text database can also be converted to a case base by generating semantic descriptors characterizing each document in the database. Much of the work in information extraction and text summarization concentrates on generating such descriptors[19]. Given such descriptor generating capabilities, any set of databases with inter-related data can be treated as distributed case bases. Information requirements of many real life applications lead to such distributed case bases. Let us illustrate this with another example: a venture capitalist assembling a team of experts for a startup company (introduced by Neches[23]). We can imagine an automated assistant for the venture capitalist querying a multi-agent system that assembles the team by letting each agent access its own resume database of various experts for a particular aspect of the company: technical, management, sales, etc. In assembling the team, the agents need to consider interactions between the requirements of various experts like for example, all experts willing to relocate to a common region of the country or all technical experts on the team being familiar with a particular computing environment. The distributed case bases in this example are the resume databases for different expertise, with descriptor generators that extract features like *"regions_willing_to_relocate"* and *"familiar_computing_environments"*.

The issue of distributed case bases, to our knowledge, has not been studied in the multi-agent context. Barletta et al.[2] and Redmond et al. [26] deal with distributed cases in single agent systems. In these approaches, each case is divided into subcases or snippets and a snippet is indexed using both global goals and the local context of that snippet within the case. This kind of elaborate engineering in the form of indexing the case pieces using both global and local problem solving contexts may not be feasible for multi-agent CBR systems. The agents may only have a partial view of the global problem solving context and the internal context of a case piece. Case bases for individual agents may be built independently, without the knowledge of the kinds of problem solving systems in which they are going to participate. In this paper, we propose an alternative to using elaborate indexing to avoid interactions when case pieces are re-instantiated in response to a new query. Case pieces are iteratively retrieved and assembled into a case, dynamically resolving any conflicts that arise during the process through negotiation among the participant agents. There are a few other important issues to note here. It may be that even though the cases for individual agents were derived from past problem-solving experience, there could be combinations of these

subcases that may be assembled into a case that the system as a whole has never seen before. In addition, local case integration does not require that the overall case be completely represented at any one node; in certain situations, the distributed case components are integrated only by their mutual consistency.

3 Negotiated Case Retrieval

3.1 Overview

Reasoning about cases drawn from a case base that is a component of a DCB presents an agent with additional uncertainties versus single agent CBR systems. As discussed previously, each agent has to rely on its possibly incomplete local view of problem-solving to retrieve a local case that best contributes to the overall case. This may lead to the retrieval of subcases that cannot be effectively put together or there may be requirements on the solution that cannot be ascertained until the subcases are aggregated. Thus, we propose the negotiated case retrieval (NCR) strategy that needs the agents to augment their local views with constraining information from other agents to achieve the retrieval and assembly of a better overall case. This strategy involves each agent asynchronously executing one of the set of possible operations: *initiate* a seed subcase, *extend* an existing partial case, *merge* existing partial cases or *inform* others about a new partial case.

Initiating a seed subcase involves an agent retrieving a local subcase from its local case base, using the local problem solving state and the relevant portion of the user specification, and forming a seed subcase which can be extended by local cases from other agents to obtain a complete case. An agent intending to extend a subcase from another agent, obtains the subcase's relevant feature values that serve as an anchor for the local case retrieval, the result of which is integrated with the corresponding partial case. Merge is similar to the extend operation. An agent intending to merge one of its chosen partial cases with another agent's partial case obtains the relevant feature values and performs the merge operation. The inform operation involves an agent telling others about the existence of a newly formed partial case that results from the local execution of one of the three previous operators. An extend or merge operation involves checking for any violations of local constraints by the set of feature values from the non-local partial case and the local case or partial case. Detection of such violations leads to an interaction process among the agents by which they negotiate on conflict resolution alternatives. The negotiation process involves an agent communicating feedback to other agents on the causes and possible resolutions for each of the constraint violations. The receiving agents assimilate this feedback, leading to an enhanced view of the global requirements for future operations. Any subsequent initiate, extend or merge is more likely to avoid the same conflicts.

Our problem is cast as a distributed constraint optimization problem[17] implying that not all constraints need to be satisfied in a solution. As many constraints are satisfied as is possible. Constraints have differing amounts of flexibility. Some may be hard, meaning that they must be satisfied in any legal solution. Others may be soft constraints that may be relaxed as and when needed. Figure 1 shows the conceptual view of a two agent DCBR system. Before we formally present our NCR strategy, we need to introduce some notation.

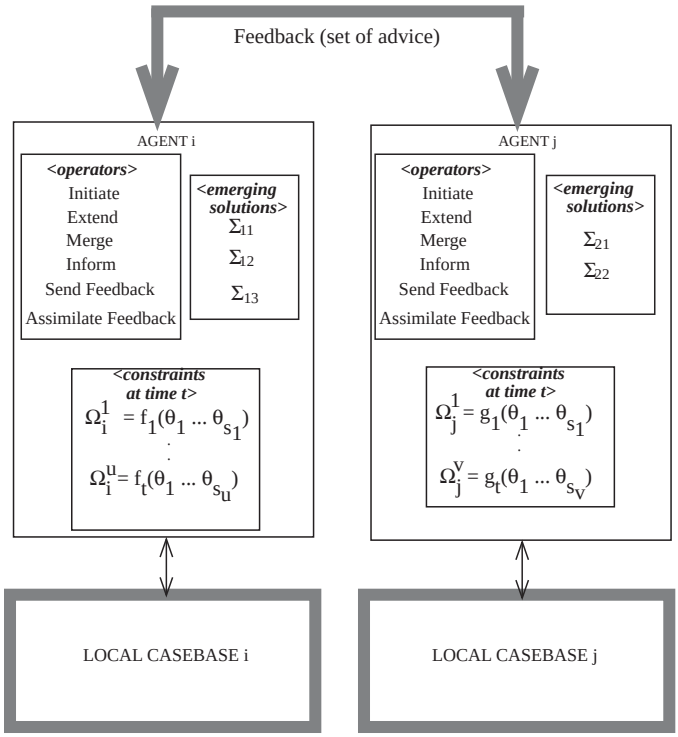

Figure 1: Schematic for Negotiated Case Retrieval

3.2 Notation and Definitions

Let the set of agents be denoted by Υ. Let Ω_i denote the locally known constraint set of agent $\mathcal{A}_i$. Let Ω_i^p, $1 \leq p \leq |\Omega_i|$ represent the p th constraint of $\mathcal{A}_i$.

Let the user-specified query be denoted by Ψ. The part of the query that is relevant to $\mathcal{A}_i$ is denoted by Ψ_i which is cast as a set of constraints $\Omega_{\psi_i} \subset \Omega_i$. In addition, Ω_i has constraints that define domain-specific requirements on any solution. Casting both user specifications and the set of required subcase consistency constraints as a set of constraints for an agent gives us a uniform way to ensure that the user specifications are satisfied and the subcases are mutually acceptable. Subcase consistency constraints arise from the knowledge an agent has about the requirements of the context in which its local subcase can usefully participate[2]. The output of the system is a case, whose subcases are mutually acceptable to the entire set of agents. As discussed previously, the subcases need not all reside at a single physical site and may be integrated just by their mutual consistency.

Let a locally retrieved case u by agent $\mathcal{A}_i$ be denoted by σ_i^u (we simply denote it as σ_i when the name of the case is not important). A partial case s, denoted by Σ_s, is composed of one or more locally retrieved cases (from different agents):

$$\Sigma_s \quad = \quad \sigma_{i_1}^{u_1} \sigma_{i_2}^{u_2} \ldots \sigma_{i_z}^{u_z}$$

[2] For example in TEAM[17], from which our DCBR system draws its inspiration, an agent's sub-design possesses shared features with another agent's sub-design. Pump and Motor agents share a parameter *required-power*. Both agents have to arrive at the same value for the *required-power* parameter. This is equivalent to having a constraint Pump-required-power = Motor-required-power.

where $1 \leq z \leq |\Upsilon|$. The set of features for Σ_s is represented by $\mathcal{F}_{\Sigma_s}$. A partial case represents a partially evolved response to a query. A feature represents an abstract semantic characterization of some particular aspect of a piece of data like a document or a mechanical component specification.

We will introduce a simple example from the ABSTRACT-TEAM system to be discussed in the later sections. Let the agent set be $\{\mathcal{A}_1, \mathcal{A}_2\}$. In ABSTRACT-TEAM, a query is denoted by specifying the ranges of the features f_2 and f_7. For example, Ψ_1 is $0.03 \leq f_2 \leq 2.0$ and Ψ_2 is $10 \leq f_7 \leq 15$. In our example, $\mathcal{A}_1$ retrieves a local case which is simply a set of three features $\{f_1, f_2, f_3\}$ and $\mathcal{A}_2$ retrieves a local case of four features $\{f_4, f_5, f_6, f_7\}$. Ω_1 contains four constraints $\{$C1:$f_2 \geq 0.03$, C2:$f_2 \leq 2.0$, C3:$f_4 \geq 0.55$, C4:$f_4 \leq 0.75$,$\}$ and Ω_2 has three constraints $\{$C5:$f_7 \geq 10.0$, C6:$f_7 \leq 15.0$, C7:$f_2 \leq 0.1\}$.

Let $\mathcal{F}_i^p$ represent the set of features on which a constraint Ω_i^p is defined. Features are the parameters over which the local search in a agent is constrained. The feature set $\mathcal{F}_k^p$ may involve features that are not components of the local subcase. Such features are called non-local features (or non-local parameters). For example, the feature set of constraint C5 is the singleton set $\{ f_7 \}$. Feature f_4 is a non-local parameter for $\mathcal{A}_1$. As mentioned previously, constraints on non-local features, like C4 and C7 arise due to the awareness of an agent about the general context in which its subcase can participate usefully in the overall case [3]. The domain of values for a feature in a constraint can be numerical or categorical or of any other form. A constraint is a relationship between elements of one or more databases and is enforced through inter-relationships between descriptors (represented as feature vectors) of these elements.

A case has a number of associated attributes that involve measures of certain characteristics of a case and are functions of the feature values of a case. Examples of attributes include reliability, quality, uncertainty and cost. In addition to being acceptable to all agents, it is desirable that a case be optimized along the attribute set. These requirements lead to organization of an agent's constraints as *soft* constraints and *hard* constraints where the former set represents solution preferences and the later set represents those constraints that are relaxed only as a result of explicit recognition by the agents that the set is too constrained to lead to a mutually acceptable solution[21]. Relaxing a soft constraint may only involve penalties in terms of loss of optimality in the desirable attributes. For example, relaxing a soft constraint may lead to less robust solutions. Softness of a constraint represents its degree of flexibility. On the other hand, hard constraints are generally not relaxed except with an explicit understanding that the resulting responses satisfy the query specifications only partially or are consistent with only a subset of the agents.

More formally, let a local solution space $\mathcal{S}$ be defined as a region in an *n*-dimensional feature-space $\mathcal{R}^n$. Each point in this space represents a local case of *n* features. The set of hard-constraints in an agent defines $\mathcal{S}$. We can then define subspaces of $\mathcal{S}$ that represent satisfaction of various sets of soft-constraints in addition to the set of hard constraints. Given $\mathcal{S}_c$, a subspace of $\mathcal{S}$ satisfying a set of soft-constraints c and the set of hard constraints, relaxing a constraint $c_x \in c$ leads to a subspace $\mathcal{S}_{c-c_x}$ such that

$$\mathcal{S}_c \subset \mathcal{S}_{c-c_x} \subset \mathcal{S}$$

[3] In a specific problem-solving situation, an agent can substitute for its local features from the problem-specification in its local constraint to get new constraints that are described only on non-local features. Though the example here deals with simple numerical constraints, note that the mechanisms presented in the paper are general enough to deal with constraints of any form.

An agent iteratively identifies those regions in its local solution space $\mathcal{S}$ that satisfy as many of the soft constraints as possible and the relevant non-local requirements of the other agents.

A solution in a DCBR system represents the composition of a number of subcases. Adding a locally retrieved subcase $\sigma_{i_m}^{t_m}$ to a partial case Σ_s to form another partial case $\Sigma_{\tilde{s}}$ is represented as:

$$\Sigma_{\tilde{s}} \quad = \quad \Sigma_s \oplus \sigma_{i_m}^{t_m}$$

A *projection* of a partial case Σ_s onto Ω_k, the constraint set of agent $\mathcal{A}_k$, represents the set of features from Σ_s that participate in some constraint of Ω_k.

$$\Sigma_s^k \quad = \quad \{f | f \in \mathcal{F}_{\Sigma_s} \wedge \exists_{1 \leq p \leq |\Omega_k|} f \in \mathcal{F}_k^p\}$$

An agent $\mathcal{A}_i$ wanting to obtain the projection of a partial case $\Sigma_s = \sigma_{i_1}^{t_1} \sigma_{i_2}^{t_2} \ldots \sigma_{i_z}^{t_z}$ communicates with agents $i_1, i_2, \ldots, i_z$ to obtain those relevant feature values that were not already communicated to the agent previously. An agent $\mathcal{A}_{i_k}$, upon a request for the value of a particular feature in a local case $\sigma_{i_k}^{t_k}$, responds with transmission of the value to the requesting agent. Each agent's organizational knowledge[4] about the distribution of relevant features among the set of agents can aid it in deciding which agents to ask for what features. A projection is the minimum information an agent needs to check the consistency of a partial case against its local constraint set. However, an agent that anticipates and intends to serve the needs of other agents may obtain more information than the projection and pass it on to these other agents, saving them the need to communicate with a number of agents to obtain a projection. We have here a trade-off between the number of communication events versus the amount of information communicated in these events. An agent obtaining more information than a projection increases the number of bytes of communication but can reduce the necessity for some other agents to communicate with a number of agents to obtain their projections and consequently reduces the number of communication events.

For example, in order to check a partial case against Ω_1, $\mathcal{A}_1$ has to obtain its projection by asking $\mathcal{A}_2$ for the value of feature f_4 of the corresponding local case, as f_4 is the only feature participating in some constraint in Ω_1 and is not available locally. Similarly, $\mathcal{A}_2$ asks $\mathcal{A}_1$ for the value of feature f_2 of the corresponding local case to validate a partial case against Ω_2.

3.3 Negotiated Retrieval

Given the above notation, we now discuss agent control. Each agent can opportunistically choose to execute one of the following operators at a particular instance of time.

- *Initiate a seed subcase*:
  Input: Agent $\mathcal{A}_i$ that chooses to initiate a seed subcase using the problem specification Ψ_i, and locally known problem-solving state.

  Output: A partial case $\Sigma_s = \sigma_s^i$

  $\mathcal{A}_i$ retrieves a subcase from its local case base and forms a partial subcase with just one local subcase. It uses the problem specification Ψ_i and the presently available information

[4] Organizational knowledge consists of a specification of general node interaction patterns[9] or static meta-level information about knowledge/case organization in the local databases of the agents.

on the problem-solving state (including previously tried solutions, conflicts they caused and feedback in the form of violated constraints from other agents) to achieve this task[5].

- *Extend an existing partial case*:
  Input: Σ_s^i to $\mathcal{A}_i$

  Output: $\Sigma_{\tilde{s}} = \Sigma_s \oplus \sigma_i^t$ or a failure to extend.

  $\mathcal{A}_i$ decides to extend a partial case Σ_s residing at $\mathcal{A}_k$ and obtains the projection with respect to the local constraints Ω_i. It tries to retrieve a local case based on the information available in Σ_s^i and the locally known problem solving state. $\mathcal{A}_i$ can either succeed or fail to return a subcase or return an infeasible subcase that violates some constraints in the set Ω_i. If any violations are detected due to poor or infeasible values for features then feedback is provided to the relevant agents. The feedback process is discussed in detail below. Note that the extension operation need not assemble all the local cases in $\Sigma_{\tilde{s}}$ at any one physical site. It just needs to record the labels of all the subcases that can be assembled without any local constraint violations with regard to the relevant agents.

- *Merge existing partial cases*:
  Input: Σ_s^i and Σ_t^i to $\mathcal{A}_i$

  Output: $\Sigma_u = \Sigma_s \oplus \Sigma_t$

  An agent $\mathcal{A}_i$ tries to merge two partially assembled composite cases Σ_{s_j} and Σ_{s_k} from agents $\mathcal{A}_j$ and $\mathcal{A}_k$ respectively. Merging involves obtaining projections $\Sigma_{s_j}^i$ and $\Sigma_{s_k}^i$ and testing for violations on the set Ω_i. If any violations are detected due to poor or infeasible values for parameters then feedback is provided to the relevant agents[6].

- *Inform other agents about a partial case*:
  Input: A set of partial cases that are the result of initiate or extend or merge operations at an agent $\mathcal{A}_i$.

  Output: A message sent to other agents.

  An agent $\mathcal{A}_i$ forms a new partial case Σ_u due to the successful execution of an operation like initiate or extend or merge. Other agents have to be informed of the existence of this new partial case so that they can make their decisions on the next operator to execute taking into consideration this new partial case. An agent need not immediately inform all the others of the existence of every new partial case it forms. It can decide to inform the existence of only the "best" ones or to inform once every few time units, all the new partial cases that emerged during the last time window[13]. An agent may also decide to inform others about more than just the existence of the cases. It may inform others about certain features or attributes of the partial cases to aid them in their choice and execution of their operators.

[5] In general, a locally retrieved subcase is re-instantiated in the present context during this operation. Adaptation of the retrieved subcase to the new context could also be performed in some systems.

[6] Though we talk of applying the entire set of constraints at an agent at once, this process could be distributed in time. An agent could apply different subsets of its local constraints at different times and perform evidential reasoning on partially satisfied sets of solutions to decide on further application of constraints like in DRESUN[6].

Agents perform an asynchronous parallel distributed search to obtain a good overall case from case pieces. At a given point in time, there may be more than one developing partial case.

The negotiation process involves an agent delivering feedback to another agent that in turn may decide to accept it. The details of this process are discussed next[7].

A partial case Σ_s is consistent with respect to Ω_k if the feature values of the partial case do not violate any constraint in Ω_k. We denote this by $consistent_{\Omega_k}(\mathcal{F}_{\Sigma_s})$. Two subcases Σ_s and Σ_t are said to be consistent with respect to $\mathcal{A}_k$ if the projections of Σ_s and Σ_t with respect to Ω_k, Σ_s^k and Σ_t^k do not violate any constraints in Ω_k, i.e., $consistent_{\Omega_k}(\mathcal{F}_{\Sigma_s^k} \cup \mathcal{F}_{\Sigma_t^k})$.

In order to gain a deeper understanding of the detected conflicts, each agent $\mathcal{A}_k$ applies a set of predicates $I_k = \{\tau_{1k}, \tau_{2k}, ..., \tau_{nk}\}$ to any detected inconsistent partial case projections $\{\Sigma_s^k, \Sigma_t^k\}$. Associated with each τ_{ik} is a set of advice $\Pi_{\tau_{ik}}$, that can act as a feedback to the agents corresponding to the subcases involved in the detected conflicts.

$$\begin{aligned} \Pi_{\tau_{ik}} &= \{\pi^1_{\tau_{ik}}, \pi^2_{\tau_{ik}}, ..., \pi^m_{\tau_{ik}}\} \\ FEEDBACK &= \bigcup\{\Pi_{\tau_{ik}} / \tau_{ik}(\Sigma_s^k, \Sigma_t^k) \ \ is \ \ true\} \end{aligned}$$

A predicate τ_{jk} characterizes the conflict and the set of advice associated with the predicate suggest ways to resolve it[8]. The set of advice could range from domain independent strategies to highly domain specific ones. For example:

1. If the retrieval is similarity based, i.e., based on numerical measures of "closeness" of the retrieval feature vector to the corresponding feature vector of a retrieved case, then broaden the search by obtaining cases with poorer similarity values.

2. Some CBR systems retrieve a case and use an adaptation strategy to massage the retrieved case to fit the new situation. An agent could advise another agent to modify the retrieved case in a different way — use a different adaptation strategy.

3. Each agent is expected to have some knowledge of the importance of a particular parameter's values and constraints, based on which it can advise another agent to relax a soft constraint involving certain parameters.

4. An agent can advise another agent to change the values or ranges of certain parameters in order to obtain better local solutions.

5. When an agent detects lack of progress either locally or at other agents (based on the projections it receives from those agents) it could advise some of them to relax their hard constraints. This is expected to take the retrieval process to qualitatively different regions of the case base. Just as with soft constraints, the choice of which constraint to relax is based on system-wide knowledge of some sort or on generic strategies each agent possesses.

6. An agent recognizes particular features of the solution space and decides to use a more efficient customized search strategy. It can advise the other agent to play a particular role in

[7] A similar method for generating feedback has been presented in [7].

[8] The predicates and their associated advice are coded into the system by a knowledge engineer. Alternatively, one could appeal to machine learning techniques to let the agents acquire this knowledge through experience.

this customized search. Lander[17] presents a good example of a customized search called *linear compromise* where agents, upon recognizing the linear nature of their solution space, decide to exchange end points and extrapolate between them to find the intersection point as a mutual compromise solution.

For example, in ABSTRACT-TEAM, the feedback is very simple. When an extend or merge violates a constraint, the constraint is communicated as feedback to agents whose local case features include the feature on which the constraint is defined. For example, agent $\mathcal{A}_2$ tries to merge its locally retrieved case ((f_4 = 1.55) (f_5 0.00154) (f_6 = 0.0058) (f_7 = 12)) with $\mathcal{A}_1$'s local case ((f_1 = 5) (f_2 = 0.48) (f_3 = 0.69)) and finds that the constraint C7:$f_2 \leq 0.1$ is violated. For each agent $\mathcal{A}_i$ in ABSTRACT-TEAM, the τ_{ik} is simply the *violated*(Ω_i^k). Each $\Pi_{violated(\Omega_i^k)}$ is $\{\Omega_i^k\}$. In our example, $\mathcal{A}_2$ communicates C7 to $\mathcal{A}_1$.

Thus, in addition to the operators we discussed previously for creating and extending partial cases, agents also have the following two operators for communicating and assimilating feedback to/from other agents.

- *Send FEEDBACK on conflicts*:
  Agent $\mathcal{A}_i$, upon detection of conflicts during a merge or an extend, creates the FEED_BACK set and communicates a subset of it to the relevant agents.

- *Assimilate FEEDBACK from other agents*:
  The process of assimilation enhances an agent's view of the global requirements. Agent $\mathcal{A}_i$, upon receiving a FEEDBACK set, assimilates it into its local constraint set, creating an updated Ω_i. Assimilation may involve processes like relaxing a constraint Ω_i^k, or adding a new constraint to Ω_i. Note that the assimilation of FEEDBACK need not be instantaneous and different advice may be assimilated at different times. For example, advice may be assimilated only after it is repeatedly received for at least '*x*' times. In addition, assimilation may be specialized based on context, i.e., the feedback assimilated is applicable only in specific contexts at the local agent. For example, a particular constraint may be applicable only if certain features are within certain ranges of values[8]. In addition, the assimilation process may also involve transformations where an agent uses the feedback from other agents to generate its own local constraints rather than directly incorporating the feedback. Such a transformation may follow a process similar to that of generating feedback, where a series of transformation operators may be applied to each piece of advice to generate local constraints.

In both the CBR-TEAM and ABSTRACT-TEAM systems to be discussed in the next two sections, the constraints are numerical and apply to a single parameter. The feedback generated from the violation of such a constraint simply consists of that constraint. Thus agent $\mathcal{A}_1$ receives the constraint $f_2 \leq 0.1$ as feedback and the next local case retrieval avoids any case with feature $f_2 \leq 0.1$. However, note that the feedback strategies enumerated above are more general and apply to situations involving more than just single parameter constraint violations. For example, say agent $\mathcal{A}_i$ detects a violation of a constraint $I^2R + I^2/C \leq 1$, where the capacitor value C is a non-local feature. Based on the domain knowledge, $\mathcal{A}_i$ may advise agent $\mathcal{A}_j$ whose local case contains C as its feature, asking it to initiate a case with capacitance in the next range of Farads. When this case is merged at a later time with a local case of $\mathcal{A}_i$, it is unlikely that the same constraint is violated again.

How do the agents decide on which operator to instantiate next and what partial cases on which to apply the chosen operator? This decision is intricately tied to the domain. Domain imperatives dictate the preconditions for each of these operators. These preconditions can either be coded in by the DCBR designer or can be learned[9]. The specifics of domain heuristics for the choice of operators and partial cases to work on at a given time are beyond the scope of this paper.

Termination criteria depend on problem solving requirements. Various termination criteria ranging from simple heuristics to complicated decision theoretic methods have been proposed for multi-agent systems[8]. One simple criteria used in our DCBR systems to be discussed later is as follows: Any agent that detects a case that has subcase contributions by all the relevant agents and has been validated against their local constraints issues a termination message to all the agents. More formally, a complete case $\Sigma_u = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_z}$ is defined as:

$$\forall_{\sigma_{i_m},\sigma_{i_n}\in\Sigma_u} i_m \neq i_n \wedge \forall_{\mathcal{A}_a\in\Upsilon}\exists_{\sigma_{i_m}\in\Sigma_u} i_m = a \wedge consistent_{\Omega_a}(\mathcal{F}_{\Sigma_u})$$

Algorithms similar to those of the negotiated retrieval already exist in the literature on multi-agent systems. The DENEGOT system[21] presents a negotiated planning algorithm suitable to the specific context of distributed planning. The NCR algorithm presented here is a generalization of the negotiated search algorithm presented in TEAM[17] and is more generally applicable to reasoning about inconsistencies in distributed data.

The following section presents a multi-agent system that incorporates a restricted form of negotiated retrieval (i.e., negotiated search as discussed in Lander[17]) and then shows a brief trace from the system. In the section on experiments, we discuss an abstract DCBR system and show some early empirical results on the negotiated retrieval mechanism.

4 CBR-TEAM: A Multi-Agent Design System

We now present a multi-agent system called CBR-TEAM whose core is derived from TEAM[17]. TEAM is a parametric design system that uses a cooperative heterogeneous set of reusable agents, each of which has the capability to produce a component of the overall design that is stored in a centralized blackboard. The TEAM system has six domain agents for the design of six components in a steam condenser and a critic agent that checks for certain features in the assembled design. It is a multi-strategy inferencing system[17] in which different strategies are seamlessly integrated into a design framework. The interactions between the components generate constraints for these strategies leading agents to iteratively negotiate on their results to find an acceptable design. When the components of the individual agents are being assembled, violation of constraints due to mismatches on mutually known parameters leads to information exchange followed by redesign. Agents whose components do not "match" are said to be in conflict. The conflicting set of agents negotiate a resolution which involves a search guided by numerical-valued constraints on the mutually known parameters, or other strategies like linear compromise[17]. CBR-TEAM is a modification of the TEAM system involving three agents, each producing a component of a steam condenser part as shown in Figure 2. This three-component design can be combined with a heat exchanger to get a complete steam condenser.

[9] Nagendra Prasad, Lander and Lesser[22] discuss in detail the issues in learning the order of instantiation of such operators.

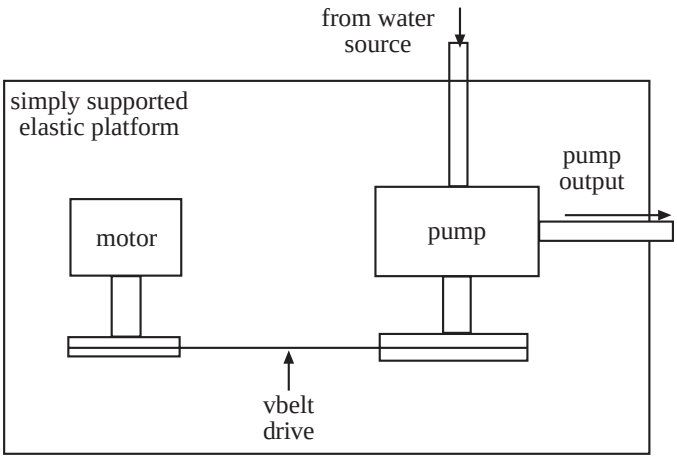

Figure 2: Steam Condenser Component Set produced by CBR-TEAM

The agents in CBR-TEAM retrieve and use suitable members from libraries of manufacturer-specified models for designing their components. These agents use the negotiated retrieval strategy where a new subcase is retrieved by each of the agents in conflict. This retrieval is guided by a set of simple numerical-valued constraints on the shared interface parameters. Interface parameters are those features of a case that are shared by more than one agent. All the relevant agents have to reach an agreement on the values of the shared parameters. We will discuss CBR-TEAM in more detail next.

The user gives a problem specification that consists of minimum head size for the pump in the required design. The agent set $\Upsilon = \{\mathcal{A}_{motor}, \mathcal{A}_{pump}, \mathcal{A}_{Vbelt}\}$ that are responsible for design of the motor, pump and Vbelt components respectively. Each of these agents retrieves a suitable design from a library of manufacturer specified designs. Indexing into a library is based on design requirements and also the requirements of the components of other agents that may interact with the components of this agent. An agent does an *initiate* if it has no partial case to *extend*. If a conflict is detected a *send* operation is performed with the constraint that was violated. The τ_{ik} for agent $\mathcal{A}_i, i \in \{motor, pump, vbelt\}$ is simply *violated(Ω_i^k)*. So any violated constraint on a shared parameter is communicated to the relevant agents. When an agent receives feedback from others, it immediately *assimilates* the feedback. Assimilation involves adding the feedback constraint to the set of local constraints to effect further searches from there on. The retrieved local cases are placed in a central blackboard and hence there is no need for an *inform* operator. Any partial case generated by an agent is completely visible to all the other agents. Thus there is no necessity for an agent to project the relevant portion of a subcase and send it to another agent. During the initial phase of retrieval, the agents may have only partial information on the requirements of other interacting components. So, each of the agents chooses the lowest cost design based on the information available to it. Trying to assemble these components into an overall design may lead to conflicts due to mismatches in the parameters that are shared by two or more components. For example, Vbelt and Pump have required-pump-power as a shared parameter and both $\mathcal{A}_{Vbelt}$ and $\mathcal{A}_{Pump}$ impose their own set of constraints on this parameter. A mismatch on this parameter involves one agent assigning a value to the parameter that violates the constraints in another agent. Each of the agents negotiate with the other agents in conflict to resolve any mismatches. This is done by posting locally generated requirements on the shared parameters that are involved in the

conflict to the relevant agents. The agents then do another round of retrieval using the previous information and the new requirements from other agents to get better cases to be assembled into a design that does not produce the same conflict.

CBR-TEAM does not allow relaxation of hard constraints. It relaxes only soft constraints. Soft constraints are arbitrarily divided into four levels, 1 to 4, with 4 being the level that involves least loss of desirability if a constraint at this level is relaxed. Soft constraints are tied to the cost attribute of a component. Lower cost components satisfy more soft constraints. Violations of any hard constraints, considered to be at level 0, lead to a conflict. Another feature of CBR-TEAM is *unilateral relaxation*. When an agent finds that its progress has been stagnated for a long time, it unilaterally relaxes its local requirements so as to allow retrieval of subcases with poorer similarity to the requirements.

Note that the retrieval process is iterative and can happen many times until a case of required quality is obtained.

Below we give a simplified example trace of CBR-TEAM during a design session (There are a number of other constraints, but we just show an illustrative subset here.). The problem specification, known to all the agents (that understand it) is the minimum head for the pump.

An illustrative subset of constraint set Ω_{motor} for $\mathcal{A}_{motor}$ includes:
1. motor-horse-power $\geq$ pump-required-power, $level\ 0$
2. motor-drive-speed $\geq 1200,\ level\ 0$
3. pump-required-power $\leq 5.0,\ level\ 1$
4. pump-required-power $\leq 4.0,\ level\ 3$

For $\mathcal{A}_{pump}$:
1. motor-horse-power $\geq$ pump-required-power, $level\ 0$
2. pump-required-power $\leq 30,\ level\ 0$

For $\mathcal{A}_{Vbelt}$:
1. motor-drive-speed $\leq 12153.65,\ level\ 0$

Initially, all three agents have a local partial view of the problem solving situation. The agents have certain local constraints on the parameters, and based on these constraints they retrieve a template and develop the best component design. $\mathcal{A}_{pump}$ retrieves a PUMP TEMPLATE “model4-impeller43-wfr1”to initiate a partial design with PUMP “pp_pump_1”. Agent $\mathcal{A}_{motor}$ attempts to extend this by retrieving MOTOR TEMPLATE “motor10” and building MOTOR "mp_motor_1”. However, it detects that this leads to a violation of soft constraints and hence sends feedback information on the detected conflict to all the relevant agents (those that know about the parameters involved in conflict). The agent that detects any conflict on a parameter simply communicates the constraints whose violation led to the detection of that conflict. In this example, $\mathcal{A}_{motor}$ detects and communicates conflict on the *required power* parameter. It sends out the constraint — (pump-required-power $\leq 4.0,\ 3$) to agent $\mathcal{A}_{pump}$. Agent $\mathcal{A}_{Vbelt}$ fails to retrieve any templates based on the local constraints.

The following figures show the designs formed by $\mathcal{A}_{pump}$ and $\mathcal{A}_{motor}$ during this step.

Agent $\mathcal{A}_{pump}$ uses “model4-impeller43-wfr1” to build PUMP “pp_pump_1” which basically consists of slots filled in by parameters from the template and some other slots filled in by functions

for calculating case attributes.

```
PUMP pp_pump_1 an object of class PUMP

MODEL:                "model4-impeller43-wfr1"
WATER-FLOW-RATE:      104.125
MAX-HEAD:             288.63
AVAILABLE-HEAD:       257.5425
PUMP-REQUIRED-POWER:  10.987118
RUN-SPEED-RANGE:      (2700.0 3300.0)
PUMP-RUN-SPEED:       3000
COST:                 228.0625
WEIGHT:               50.503906
EVALUATION:           (:PUMP-AGENT :GOOD)
ACCEPTABILITY:        :ACCEPTABLE
```

Agent $\mathcal{A}_{motor}$ retrieves MOTOR-TEMPLATE "motor10" from the template database to build MOTOR "mp_motor_1".

```
MOTOR "mp_motor_1" an object of class MOTOR

MODEL:                "motor10"
PUMP-REQUIRED-POWER:  10.987118
SPEED-RANGE:          NIL
WEIGHT-RANGE:         NIL
HORSEPOWER:           15.0
MOTOR-DRIVE-SPEED:    2400
COST:                 650.0
WEIGHT:               150
EVALUATION:           (:MOTOR-AGENT :FAIR)
ACCEPTABILITY:        :UNACCEPTABLE
```

$\mathcal{A}_{Vbelt}$ fails to retrieve any templates based on the local constraints. These constraints are not tight enough to retrieve any templates that pass the tests setup by the local filters.

Note that during the time $\mathcal{A}_{motor}$ is extending a partial case initiated by $\mathcal{A}_{pump}$, agents $\mathcal{A}_{pump}$ and $\mathcal{A}_{Vbelt}$ themselves could be initiating other seed subcases due to the asynchronous nature of the distributed search.

Given the increased awareness of the $\mathcal{A}_{motor}$'s requirements, $\mathcal{A}_{pump}$ attempts a new round of retrieval and generates PUMP "pp_pump_4". It takes $\mathcal{A}_{motor}$'s communicated constraints into consideration while doing this. $\mathcal{A}_{motor}$ thus successfully adds its subcase MOTOR "mp_motor_4" and $\mathcal{A}_{Vbelt}$ adds VBELT "vbelt_1" giving the final design.

Agent $\mathcal{A}_{pump}$ retrieves PUMP-TEMPLATE "model4-impeller43-wfr2" from the pump template

database to build PUMP “pp_pump_4”.

```
PUMP "pp_pump_4" an object of class PUMP

MODEL:                "model4-impeller43-wfr2"
WATER-FLOW-RATE:      0.5
MAX-HEAD:             288.63
AVAILABLE-HEAD:       288.63
PUMP-REQUIRED-POWER:  0.1
RUN-SPEED-RANGE:      (2700.0 3300.0)
PUMP-RUN-SPEED:       3000
COST:                 228.0625
WEIGHT:               50.503906
EVALUATION:           (:PUMP-AGENT :GOOD)
ACCEPTABILITY:        :ACCEPTABLE
```

Agent $\mathcal{A}_{motor}$ retrieves MOTOR-TEMPLATE “motor1” from the template database to build MOTOR “mp_motor_4”.

```
MOTOR "mp_motor_4" an object of class MOTOR

MODEL:                "motor1"
PUMP-REQUIRED-POWER:  0.1
SPEED-RANGE:          NIL
WEIGHT-RANGE:         NIL
HORSEPOWER:           1.0
MOTOR-DRIVE-SPEED:    2400
COST:                 100.0
WEIGHT:               33
EVALUATION:           (:MOTOR-AGENT :EXCELLENT)
ACCEPTABILITY:        :ACCEPTABLE
```

$\mathcal{A}_{pump}$ and $\mathcal{A}_{motor}$ have retrieved satisfactory design templates and agent $\mathcal{A}_{Vbelt}$ needs to attempt a retrieval of a Vbelt template. The power parameter from PUMP “pp_pump_4” and the drive speed parameter from MOTOR “mp_motor_4” are the shared parameters for $\mathcal{A}_{Vbelt}$ that are used to determine the retrieval of VBELT-TYPE-TEMPLATE 3VX that in turn is used by the Vbelt agent to design VBELT “vbelt_1”.

```
VBELT vbelt_1  an object of class VBELT

BELT-TYPE:            :3VX
BELT-LENGTH:          25.0
DRIVE-DIAMETER:       2.5
LOAD-DIAMETER:        2.2
NUMBER-OF-BELTS:      1
REQUIRED-POWER:       0.1
MOTOR-DRIVE-SPEED:    2400
LOAD-SPEED:           2727.2727
BELT-FORCE:           7.3234771990515d0
LOAD-PULLEY-WEIGHT:   1.0621228
BELT-LIFE:            334905.70545380004d0
COST:                 34.864
WEIGHT:               2.3162665
EVALUATION:           (:VBELT-AGENT :EXCELLENT)
ACCEPTABILITY:        :ACCEPTABLE
```

This completes the design consisting of PUMP “pp_pump_4”, MOTOR “mp_motor_4” and VBELT “vbelt_1”.

5 Experimental Results

In order to get a better insight into the proposed negotiated retrieval mechanism for distributed case bases, we built an abstract version of CBR-TEAM, called the ABSTRACT-TEAM. It captures the essential aspects of CBR-TEAM, while stripping it of domain-specific complexities. The primary difference between CBR-TEAM and ABSTRACT-TEAM is that the constraints in CBR-TEAM originated out of the steam condenser domain whereas the constraints of ABSTRACT-TEAM were not grounded in any particular domain and were randomly generated. The rest of this section will discuss ABSTRACT-TEAM in some detail and follows it with some experimental results demonstrating the benefits of negotiated retrieval.

Agents in ABSTRACT-TEAM have to assemble a mutually consistent composite case from local cases retrieved from local case bases. Each agent possesses constraints which may be defined on both local and non-local features. Each agent has a local case base that is a list of feature vectors. At this point of our work, the control for negotiated retrieval was intentionally kept simple to avoid obfuscating some of the more important issues that we wanted to gain insight into. During Phase I, each of the agents retrieves its best local case in parallel, based on local constraints derived from problem specification and initiates a seed partial case. In this phase, each agent constrains its local retrieval to avoid cases which violate constraints on local features. During Phase II, an agent informs the other agents about the existence of a new partial case. During Phase III, each agent tries to merge its local partial case with partial cases from the other agents. Each agent obtains the relevant projections from other agents and checks the feature values against its local constraint set for violations. If a violation is detected, the corresponding constraint is conveyed to the relevant agents and is immediately assimilated. A new iteration of negotiated retrieval is then initiated, but this time with an enhanced view of the problem-solving requirements for at least one agent.

For the experiments reported below, we used the example discussed in Section 3. A case of $\mathcal{A}_1$ consisted of three features, and a local case base was built by randomly generating feature vectors and their corresponding costs. Cases for $\mathcal{A}_2$ had four features and a local case base was similarly generated. Constraints that could be communicated to other agents had a representation similar to that in CBR-TEAM; numeric-valued constraints on single features. As of now, $\mathcal{A}_1$ has a single constraint that it could communicate to $\mathcal{A}_2$, and $\mathcal{A}_2$ has two constraints that it can communicate to $\mathcal{A}_1$. ABSTRACT-TEAM in the experiments below had only hard constraints and hence did not deal with optimizing any attributes on a case. Composite case had seven features corresponding to three from $\mathcal{A}_1$ and four from $\mathcal{A}_2$. Problem specification consists of required ranges on certain features[10].

Figure 3 shows the time taken by agents executing negotiated retrieval to obtain a mutually acceptable best case (The exact units of time are unimportant here. We used (get-internal-run-time) provided by Harlequin's Lispworks development environment.). The purpose of this experiment is to see if the negotiated retrieval algorithm scales with the size of the case bases. This issue is particularly important when we are proposing the use of this algorithm on networked digital

[10] Note that our description of ABSTRACT-TEAM does not highlight the fact that in a more general system some of the constraints are emergent - some of the requirements on the solutions emerge as the problem solving progresses. In addition, the constraints exchanged could be context-dependent meaning that they are applicable only in certain contexts that are known only during the problem solving and not a priori. Thus simply exchanging all the constraints at the beginning of problem solving is not a feasible method for the problem solving process described here.

libraries with large volumes of data. If the retrieval times varied linearly with the case base size, then we have satisfactory scalability properties. Thirty runs at each case base size are plotted. The case bases are initialized to be different for each run, i.e., the case bases were populated with different sets of randomly generated feature vectors during each of the runs. Figure 3 shows a linear regression fit to the data. The R^2 value of 0.837 shows that the regression fit explains much of the variation around mean. When we did a second-order polynomial fit for the same data, the coefficient for the second-order term ($case\_base\_size^2$) was small (0.000052) and the probability that it is different from 0 was not significant; indicating that linear regression is a satisfactory model for this data.

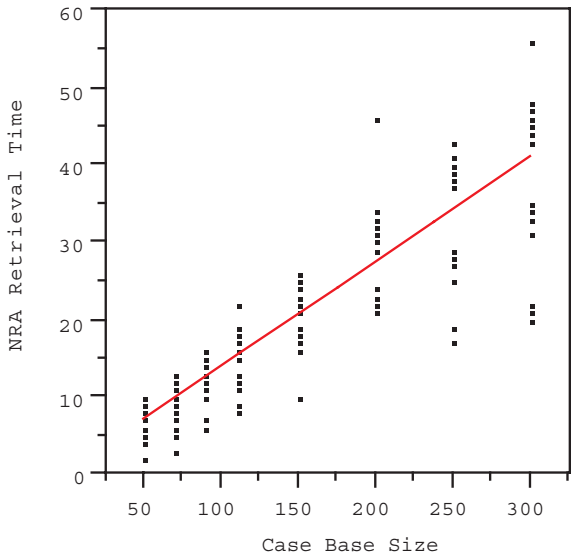

Figure 3: Negotiated Retrieval Time versus Case Base Size

We next compared the negotiated retrieval against a simpler algorithm which involves retrieval and conflict detection but no explicit feedback through negotiation and exchange of constraints. Thus, detection of conflict simply leads to another round of retrieval. This set of experiments was designed to study the effect of negotiation over the retrieval times. We again conducted thirty runs at each case base size, noting the time difference between the simple retrieval algorithm and the negotiated retrieval algorithm for each case base. Figure 4 shows that the difference rises sharply with the case base size with the simple retrieval algorithm taking increasingly larger times compared to negotiated retrieval[11]. The rectangular boxes show interquartile ranges — the top and bottom of a rectangle shows the 75th and 25th quantiles. The median or the 50th quantile is shown as a line in the body of the box.

Our results demonstrate the scalability of the negotiated retrieval algorithm to large-scale applications like digital libraries. Linear rise in the retrieval times is a desirable property when

[11] CBR-TEAM has certain constraints that can be communicated and certain others that cannot be, due to its limited form of representation on communicable constraints. If the detected conflict is due to a non-communicable constraint, then the corresponding agent can only reject a composite solution without being able to give any feedback, just as in the simple retrieval algorithm above. However, if the conflict detected is communicable, then the behavior is similar to negotiated retrieval. Thus, the overall performance of the system will lie somewhere in between the spectrum whose ends are represented by negotiated retrieval and the simple retrieval algorithm.

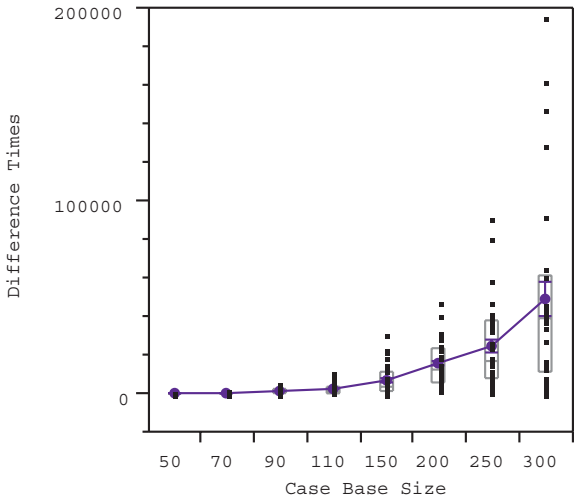

Figure 4: Difference in times of the simple retrieval algorithm and negotiated retrieval

large amount of data is involved. Moreover, the second set of experiments demonstrate the crucial role played by communication and negotiation in this scalability property. The non-linear rise in the difference of times between blind search and negotiated search lead us to conclude that negotiated search is a vast improvement when the amount of data available is large.

6 Related Work

Much of the CBR literature is concerned with case bases of a single agent. There has been some work in breaking a single case into pieces and reasoning with these pieces in a single agent context. MEDIATOR[16] represents large cases monolithically with pieces embedded as parts which can be located within a larger case. A case is indexed both by its own indices and those of its pieces. This sort of representation is different from the distributed case base scenario we discussed in the paper. In a multi-agent system, there may be no monolithic case until the subcases are retrieved and assembled.

Barletta and Mark[2] break cases into pieces where each piece is a sequence of actions used to recover from a hypothesized fault. JULIANA[28] and CELIA[26] implement a distributed case representation that is closer in spirit to the work described here. Cases are broken into pieces called snippets, each of which represents the pursual of a goal. These snippets are linked to other causally related snippets. Following the links can reconstruct a full case. Each snippet has a pointer to the case header, its goal and the context in which it is embedded. During problem solving, an individual snippet can be retrieved and used to pursue a subgoal related to its goal. The next subgoal can be pursued using a snippet from an entirely different case. In all these systems, both the global context and the internal context of a piece in a case are carefully extracted and used as indices for that piece to avoid inconsistencies among pieces participating in a problem solving run. However, in multi-agent systems the agent case bases may be developed in disparate situations making it impractical to follow this strategy. So negotiated retrieval tries to deal with incompatibilities between pieces by detecting and resolving them at the retrieval time.

Rissland et al.[27] discuss a CBR system called FRANK for report generation. Based on user's preferences and requirements, a report type is selected. Each report type is associated with groups of strategies which act as indices into a library of plans[12]. A plan is selected and instantiated to generate a report. Among other things, FRANK includes various types of responses to a failure to retrieve an adequate set of plans. It exploits information about failures to select better alternatives.The system tries alternative plans under the present group of strategies for report generation. Failing that, the systems tries alternative grouping of strategies. If this leads to a failure, the system changes the report type that it is trying to generate. This sort of broadening of allowed strategies upon failure is similar to the broadening of retrieval in our algorithm.

The negotiated retrieval algorithm benefits from a long history of thought in DAI and Multi-agent systems. Lesser[20] discusses the importance of interactions among subgoals and data distributed across a set of agents in distributed problem solving. Selectively sharing relevant partial results of a local search or data can enhance the global problem solving quality and efficiency without paying a large price for communication. In some recent studies, Decker and Lesser[12] further quantify these observations and discuss various trade-offs involved in communication versus enhanced problem solving efficiency. However, much of the past work in DAI has primarily focussed on exploiting interactions among subgoals to resolve control uncertainty. This paper deals with explicit detection of inconsistencies among the local solutions and subsequent focussed resolution of these inconsistencies[7]. In negotiated retrieval, feedback on the causes of conflicts detected leads to an improved retrieval in subsequent rounds where these conflicts are avoided. An agent that assimilates feedback from other agents enhances its view of global problem solving requirements leading to an improved retrieval process. Literature on multi-agent systems contains algorithms similar to those of the negotiated retrieval. The DENEGOT system[21] presents a distributed negotiation algorithm for distributed planning and TEAM presents a negotiated search algorithm for distributed search among a set of heterogeneous agents[17]. Negotiated retrieval is specifically tailored to distributed case bases with strategies for broadening and deepening a search in a case base not necessarily applicable to planning domains dealt with in DENEGOT. In addition, negotiating over conflicts in subcases can be qualitatively different from negotiations over plans in DENEGOT. For example, detection of interactions in DENEGOT involves elaborate reasoning about the local plans. In Case Based Systems, elaborate reasoning about cases may be infeasible because cases often implicitly encompass a lot of knowledge about subgoal interactions. Thus we resort to weak declarative constraint based knowledge to reason about possible interactions between subcases. The strategies in negotiated retrieval include the strategies in negotiated search as a subset. In this sense, negotiated retrieval is more general than negotiated search. In addition, negotiated search as discussed in Lander[17] imposes a sequentiality in search by giving an agent only two options — either extend an existing solution or initiate a new solution which other agents sequentially extend.

7 Conclusion & Future Work

This paper makes an initial foray into methods for performing retrieval of cases distributed across multiple agents. Constraints emerge dynamically, as a result of the on-going problem solving

[12] The size of a case base in most traditional CBR systems is around a few hundred cases.

activity. Negotiated retrieval performs focused exchange of information on these constraints to achieve a more coordinated distributed case retrieval. Even though the negotiated retrieval algorithm was introduced in the context of distributed case bases, we view the mechanisms involved there in as being more general. We believe that with the growing importance of information on the Internet, retrieval of mutually related pieces of information will play a central role in the needs of net users. Examples include assembly of a team of experts for a product design based on the advertised resumes of various experts, generation of a legal argumentation discourse on various aspects of a case based on distributed legal case bases, etc. In fact, negotiated case retrieval can be applicable to any information sources that can generate semantically meaningful descriptors for their content. We are in the process of further formalizing the properties of this algorithm and have plans to deploy it on an Internet-based information retrieval application.

8 Acknowledgments

The first author would like to thank Qiegang Long for providing the initial encouragement to work on the ideas presented here and also for providing constructive criticism at various stages of the work. Our thanks go to Robert St. Amant for readily helping us with the arduous task of interpreting our results. We would like to thank Prof. Edward Fox and Prof. Su-Shing Chen, whose extensive comments vastly improved the readability of the paper. We also would like to thank J. Daniels, K. Decker, D. Nieman, T. Oates, Z. Rubinstien and D. Skalak for reading draft versions of the paper and providing useful inputs.

References

[1] Arens,Y., Chee, C. Y., Hsu, C., and Knoblock, C. A., "Retrieving and integrating data from multiple information sources", in *International Journal on Intelligent and Cooperative Information Systems*, 2 (2), 1993, pp. 127-158.

[2] R. Barletta and W. Mark., "Breaking cases into pieces", In *Proceedings of Case-Based Reasoning Workshop*, St. Paul, MN., 1988, pp 12-17.

[3] A. H. Bond and L. Gasser, Eds., *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, 1988.

[4] M. C. Bowman, P. B. Danzig, U. Manber, and M. F. Schwartz, "Scalable Internet Resource Discovery: Research Problems and Approaches", *Communications of the ACM*, 37(8), 1994, pp 98 - 107, cntd on 114.

[5] D. R. Brown, R. Mecklenburg, D. L. Crandall, K. Y. Hwang, and R. Haddad, "Distributed Component Information in Engineering Design", 1995, submitted for review.

[6] N. Carver, and V. Lesser., "Sophisticated Cooperation in FA/C Distributed Problem Solving Systems", Proceedings of AAAI-91, 1991, AAAI Press/MIT Press, pp 191–198.

[7] N. Carver, Q. Long, and V. R. Lesser, "Reasoning About Inconsistency in Cooperative Distributed Problem Solving", Unpublished memo, Department of Computer Science, University of Massachusetts, Amherst, 1993.

[8] S. E. Conry, K. Kuwabara, V. R. Lesser, and R. A. Meyer, "Multistage Negotiation for Distributed Constraint Satisfaction", *IEEE Systems, Man, and Cybernetics*, 21(6), pp 1462-1477.

[9] D. D. Corkill and V. R. Lesser, "The use of meta-level control for coordination in a distributed problem solving network", in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp 748-756, Karlsruhe, FRG, 1983.

[10] D. D. Corkill, *A Framework for Organizational Self-design in Distributed Problem-solving Networks*, Ph.D. Dissertation, Dept. of Computer Science, University of Massachusetts, Amherst, 1983.

[11] K. Decker and V. R. Lesser, "Generalizing the Partial Global Planning Algorithm", *International Journal of Intelligent and Cooperative Information Systems*, 1(2), 1992, pp 319 – 346.

[12] K. Decker and V. R. Lesser, " Quantitative Modeling of Complex Computational Task Environments", in *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp 217-224, Washington, 1993.

[13] Durfee, E.H., Lesser, V.R. and Corkill, D.D., "Coherent Cooperation Among Communicating Problem Solvers," *IEEE Transactions on Computers*, Vol. 36, No. 11, November 1987, pp. 1275-1291.

[14] E. H. Durfee and V. R. Lesser, "Partial Global Planning: A coordination framework for distributed hypothesis formation", *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167-1183, September 1991.

[15] Kolodner, J. L., *Case-Based Reasoning*, Morgan Kaufmann Pub., San Mateo, CA, 1993.

[16] Kolodner, J. L. and Simpson, R. L., , "The MEDIATOR: Analysis of an early case-based problem solver", *Cognitive Science* 13(4), 1989, pp 507-549.

[17] Lander, S. E., *Negotiated Search in Heterogeneous Multi-Agent Systems*, Ph.D. thesis, Department of Computer Science, University of Massachusetts, Amherst, 1993.

[18] Lander, S. E and Lesser, V. R., Understanding the Role of Negotiation in Distributed Search Among Heterogeneous Agents, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Chambery, France, 1993, pp 438 - 444.

[19] Lehnert, W. and Sundheim, B., "A Performance Evaluation of Text-Analysis Technologies," *AI Magazine*, 1991, pp 81-94.

[20] Lesser, V. R., "A retrospective view of FA/C distributed problem solving", *IEEE Systems, Man, and Cybernetics*, 21(6), pp 1346-1363.

[21] Moehlman, T., Lesser, V. R., and Buteau, B., "Decentralized Negotiation: An Approach to the Distributed Planning Problem," *Group decision and Negotiation*, 1(2), Kluwer Academic Publishers, 1992, pp 161-192.

[22] Nagendraprasad, M. V., Lesser, V. R., and Lander, S. E., "Learning Organizational Roles in a Multi-agent System", TR 95-35, Department of Computer Science, University of Massachusetts, Amherst, 1995.

[23] Robert Neches, *Private communications*, January, 1995.

[24] Tim Oates, M V NagendraPrasad, V. R. Lesser, "Cooperative Information Gathering: A Distributed Problem Solving Approach", Technical Report 94-66, Dept. of Computer Science, University of Massachusetts, Amherst, 1994.

[25] *Partnet: The Distributed Component Information System*, http://part.net

[26] Redmond, M.A., "Distributed cases for case-based reasoning: Facilitating use of multiple cases", In Proceedings of AAAI-90, Cambridge, MA, AAAI Press/MIT Press, 1990, pp 304-309.

[27] Rissland, E., Daniels, J., Rubinstein, B., and Skalak, D., "Case-Based Diagnostic Analysis in a Blackboard Architecture", in the *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp 66-72, Washington, 1993.

[28] Shinn, H.S., "The Role of mapping in analogical transfer", In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Northvale, NJ:Erlbaum, 1988.