Quantitative Organizational Models for Large-Scale Agent Systems *

Bryan Horling and Victor Lesser

University of Massachusetts
Amherst, MA 01003-9264
{bhorling,lesser}@cs.umass.edu

Abstract. As the scale and scope of multi-agent systems grow, it becomes increasingly important to design and manage the manner in which the participants interact. The potential for bottlenecks, intractably large sets of coordination partners, and shared bounded resources can make individual and high-level goals difficult to achieve. To address these problems, many large systems employ an additional layer of structuring, known as an organizational design, that assigns agents particular and different roles, responsibilities and peers. These additional constraints allow agents to operate effectively within a large-scale system, with little or no sacrifice in utility. Different designs applied to the same problem will have different performance characteristics, therefore it is important to understand and model the behavior of candidate designs. In this paper, we will introduce a domain-independent organizational design representation capable of modeling and predicting the quantitative performance characteristics of agent organizations. This representation can support the selection of an appropriate design given a particular operational context. We will demonstrate how the language can be used to represent complex interactions, and show modeling techniques that can address the combinatorics of large-scale agent systems.

1 Introduction

Many of the decisions made in multi-agent system design, and in computational systems in general, are predicated on the idea that one wishes to minimize the "bad" characteristics of the system while maximizing the "good". This practice manifests itself in blanket, axiomatic objectives such as "minimizing communication", "reducing uncertainty", and "maximizing profit". While these are worthy, abstract goals that have critical practical and research importance, when a system is deployed and situated in context such ideals may no longer have the same level of relevance. Consider the underlying issues that drive these objectives. Why should communication be minimized? Why do we care about the combinatorics of a particular technique? Why should centralization be avoided? In each case, we presume the existence of some limiting factor,

* This material is based upon work supported by the National Science Foundation under Grant No. IIS-9988784. This material is also based upon work supported by the National Science Foundation Engineering Research Centers Program under NSF Award No. EEC-0313747. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

some bounded resource which motivates these objectives. However, when the system is placed in a particular context where these bounds can be quantified, the intangible nature of these blanket statements is no longer sufficient. For example, if ample communication bandwidth is available and additional utility may be derived by using it, then a strategy which always minimizes communication may lead to a solution which fails to reach its potential. If a particular resource is bounded, and the qualitative side effects of using some or all of that resource are the same, then the system should exploit it as best it can in service of satisfying or maximizing the system's specified goals.

Because of this, it our belief that any real-world system must be tailored to the environment in which it exists, if it is to make effective use of the resources and flexibility available to it. We will explore this tailoring through the system's *organizational design*. The notion of an organizational design is used in many different fields, and generally refers to how members of a society act and relate with one another. This is true of multi-agent systems, where the organizational design of a system can include a description of what types of agents exist in the environment, what roles they take on, and guide how they act both independently and with one another. More generally, if we assume an entity has a set of possible choices to make during its operation, the organizational design will identify a particular subset of those choices that should actually be considered at runtime. By working with this typically smaller set, the entity's decision process is facilitated. This additional structure becomes increasingly important the system scales in number and scope [1]. Imagine how difficult it would be for a large human organization, such as a corporation or government, to function if individuals lacked job descriptions and long-term peer relationships. Agents in large and massively-scaled systems face similar challenges, and can derive similar benefits from an explicit organizational design.

In previous work, we demonstrated that the organizational design of a multi-agent system has a measurable, quantifiable effect on the performance of the system as a whole [7]. Intuitively, changing the manner in which agents interact or the pattern that those interactions take on can change how the system behaves from both global and local perspectives. We also demonstrated that it was possible to analytically model and predict those effects in a realistic, complex domain. In this paper, we will continue with this line of reasoning, and present a generic, domain-independent language capable of capturing these types of organization effects. This organizational design modeling language (ODML) incorporates quantitative information in the form of mathematical expressions, which are used to predict the characteristics of an organization.

The immediate benefits of such a language are twofold. First, by incorporating quantitative information about the environment, resources, agents, tasks, goals, or any other object relevant to the system's performance, candidate organizations may be tailored and evaluated in a context-specific way. Second, once a suitable model has been found, it can serve as an explicit organizational representation, guiding agents' local decisions in a manner consistent with global objectives. The longer-term benefits of the organizational model include being able to make predictions about runtime performance, which can be used to isolate and diagnose system failures and deficiencies. This same information can also be used to support adaptation of the system, by incorporating learned knowledge into the existing model and analyzing the resulting structure.

In the following section, we will provide an example of the type of organizational decisions we are concerned with, and how they can affect system performance. We will continue by introducing ODML, and show how it can be used to model those decisions and support organizational reasoning. We will conclude by discussing strategies that can be used to cope with the combinatorics inherent in the models of large-scale systems.

2 Information Retrieval Domain

We will frame our discussion within an information retrieval domain, inspired by work presented by Zhang in [12]. A general peer-to-peer information retrieval system is composed of a number of interconnected databases, controlled by entities which we will refer to as agents. Queries are first received by individual members of the network. An appropriate set of information sources must then be discovered that can address the query, after which the query is routed and processed to produce a response for the user. The information necessary for responding to a particular query may be distributed across the network, which can cause an undirected retrieval process to be time consuming, costly, or ineffective, particularly when the number of sources is large.

Zhang proposes that a structured, hierarchical organization can be used to address this problem. Content in the network is arranged in hierarchies, allowing queries to quickly propagate to data sources, and results be efficiently routed and incrementally aggregated back to a single agent in the network. At the top level of the hierarchy are a set of mediators. Each mediator is responsible for providing a concise and accurate description, known as a collection signature, of the data available in its hierarchy. A hierarchy forms below an mediator, which manages a collection of information sources. An information source may be an individual database, or an aggregator agent which manages other sources. Mediators are also responsible for handling the user queries, by first using the collection signatures of other mediators to compare data sources, and then routing the query to those sources that seem most appropriate. A graphical depiction of a simple organization in this style will be seen later in Figure 1b.

This organizational design provides several advantages. The use of collection signatures to model the contents of a number of individual sources can dramatically reduce the number of agents that must be searched and queried. The use of hierarchies introduces an element of parallelism into the query distribution process. These same hierarchies also distribute the communication and processing load of the response through the use of information aggregation and consolidation.

At the same time, if the structures are poorly designed, they can also lead to inefficiencies. A single collection signature, which must be bounded by size to be efficiently used, can become unacceptably imprecise if the set of sources it models is large or extremely diverse. This can cause data sources to be overlooked, potentially reducing the response quality. Whenever a hierarchy is used, there also exists a tension between the width and height of the structure. Very wide structures can lead to bottlenecks, as particular individuals with high in-degree may become overwhelmed by the number of interactions. Very tall structures can be slow or unresponsive, as the long path length from root to leaf increases latency. The collection signature generation process may

also be affected by the tree height, as when abstraction is used at intermediate nodes, causing the signatures of tall hierarchies to incur additional imprecision.

Additional constraints and characteristics exist in the system that exist independent of the organization that is employed, but are relevant to the organization selection process. The communication and processing loads of individual agents are bounded. There may be quality or response time constraints imposed at a high level by the designer. Queries may arrive at regular rate, or at least be probabilistically predictable. Individual databases will vary in size, scope and content. Each of these aspects may affect performance in a non-trivial way.

The problem then, is to determine the most appropriate organization of agents and databases, given the desired characteristics of the system, the provided characteristics of the environment and the tradeoffs we have presented here. For example, how tall should the aggregation hierarchies be? How many nodes should be searched to answer a query? How many mediators should be created? How should these various roles be mapped to actual agents? In the following section, we will introduce our organizational modeling language, and show how these questions can be answered by embedded the relationships described above in such an organizational model.

3 Organizational Representation

The organizational model, as we have described it, must serve in several different capacities. At design time, it should be possible to use the structure to create and evaluate not just a single organizational instance, but an entire family of organizational possibilities. At runtime, it should accurately describe the current organization. In both cases, the model must be sufficiently descriptive and quantitative that one can evaluate the organization, and rank alternatives according along some specified criteria. Below, we enumerate the desired capabilities and characteristics the modeling language should possess to satisfy these requirements:

1. Represent the scope of organizational possibilities, by identifying general classes of organizations and the parameters which influence their behavior.
2. Represent the current organizational structure. This would include roles, interactions and associations (e.g. coalitions, teams). Different flows in the organization, such as communication and resources, should be represented.
3. Allow deductive analysis by quantitatively describing the relevant characteristics exhibited by the structure, and the manner in which those characteristics interact. For example, both communication overhead and the effect that overhead has on work load should be representable.
4. Identify which parameters and characteristics are under deliberate control, and which are derived from external factors.
5. Define thresholds and constraints, and the possible consequences of exceeding those thresholds.

Several different organizational representation schemes have been developed by researchers in the past [2, 6, 10, 3, 9, 4], however none of these meets all the requirements outlined above. Consequently, we have designed the Organizational Design Modeling

Language (ODML) to meet our needs. Conceptually, ODML models exist in two distinct forms that share a common representational definition. The first acts as a template, that expresses the range of organizational possibilities by explicitly encoding the organizational decisions that must be made. The second is an organizational instance, created from the template by making specific choices for those decisions. Because the instance form is an instantiation of the template, individual entities in the instance are related to their original abstract specification. This relationship allows one to explore the space of changes that might be made to an organization at runtime.

Formally, an ODML template specification $\mathcal{O}$ is defined as follows:

$$\begin{aligned}\mathcal{O} &= \{\mathcal{N}, H, C, K, M, V\} \\ \mathcal{N} &= \{N_0, N_1, \ldots, N_n\} \\ N &= \{t, \bar{p}, I, H, C, K, M, V\}\end{aligned}$$

The foundation of the ODML template specification is the set $\mathcal{N}$ of *node templates*, each of which corresponds to a particular physical or logical entity which might exist in the organization. For example, in our information retrieval scenario there would be nodes corresponding to mediators, aggregators, databases, agents and the environment, among other things. Each node N contains a number of elements, defined below:

- t The node's *type*. This label must be unique within the set of template nodes that make up the organization.
- $\bar{p}$ A list of $parameters$ which must be passed to the node's template when an instance of the node is created. These are analogous to the parameters one might pass to an object constructor. Each parameter is specified with a type and local name.
- I The set of node types that this node has an *is-a* relation with, with the conventional object-oriented inheritance semantics. If we assume that a node's $I = \{a, b\}$, an instance of the node will also be an instance of a and b, possessing the characteristics of all three node type.
- H The set of node types that this node has a *has-a* relation with. If we assume that $H = \{a, b\}$, an instance of the node will possess some number of instances of both a and b. It is through this type of relationship that the primary organizational decomposition is formed.
- C A set of $constants$, which represent quantified characteristics associated with the node. Constants may be defined with numeric constants (e.g. 42), or mathematical expressions (e.g. $x + y$).
- K A set of $constraints$. An organization is considered valid if all of its constraints are satisfied.
- M A set of $modifiers$, which can affect (e.g. mathematically change) a value contained by a node.
- V A set of $variables$, representing decisions that must be made when the node is instantiated. Each variable is associated with a range of values it can take on.

The top-level organization node $\mathcal{O}$ also contains the elements H, C, K, M, V, providing a location for the designer to embed global information and constraints. Collectively, we will refer to H, C, K, M as a node's *fields*, and the quantitative state of a

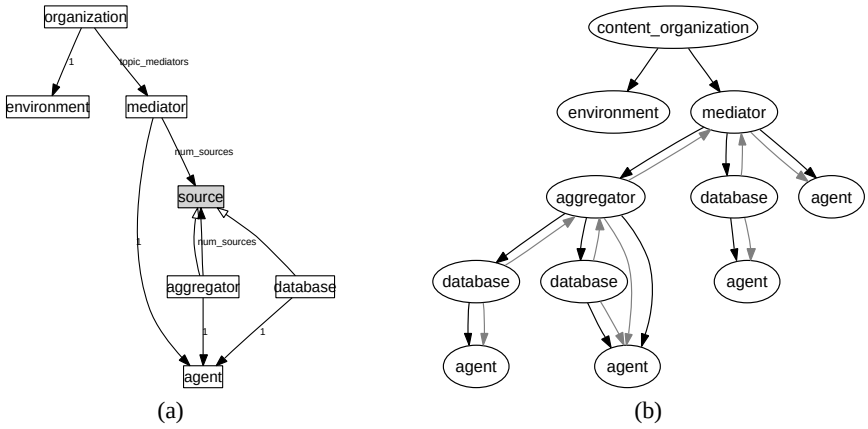

Fig. 1. a) An ODML template structure for the information retrieval domain. Vertices represent nodes. Solid edges are has-a relations, and hollow edges are is-a relations. Edge labels reflect the magnitude of the has-a relation. b) A small organizational instance produced from that template. Black edges indicate has-a relationships, gray edges reflect value modifiers that span nodes.

field as its *value*. For example, the constant field *total load* might be defined with the expression *total_load = work_load + communication_load* and have a value of 0.9 for an agent in a particular organization. Note that the term "constant" may be misleading. While the expression defining *total load* is fixed, the value for *total load* produced by that equation may change through the application of modifiers or due to changes in fields or values that the expression is dependent on.

The different aspects of ODML are best explained with an example. For clarity, we will represent particular nodes, or fields that reside the nodes, in italics. Space precludes showing the raw, textual model constructed for the information retrieval environment, however, a graph showing some aspects of the model can be seen in Figure 1a. Vertices in the graph, such as *mediator* and *database*, represent nodes. Directed edges with a solid arrow represent has-a relations, and the corresponding label indicates the magnitude of that relation. For example, each *aggregator* node has a single *agent*, while *organization* has a number of *mediators* specified by the variable *topic_mediators*. A hollow-arrow edge represents an is-a relation, so both *aggregator* and *database* are instances of *source*. Shaded nodes, such as *source* are abstract, and cannot be instantiated. This particular template also demonstrates the ability to design recursive models in ODML, because *aggregator* has both is-a and has-a relationships with *source*.

The heart of any ODML model exists in the expressions encoded within nodes' fields. A portion of the fields contained by the *aggregator* node are shown in Table 1. Each field may contain an arbitrary mathematical equation, combining local and nonlocal information to calculate new local values. These expressions provide a way for the designer to represent how different characteristics of the node may be computed. For example, the *data_size* of an aggregator is the sum of the *data_size* values for each of its *sources*. Similarly, the aggregator's *agent.work_load* is affected by the aggregator role's

Table 1. A portion of the *aggregator* node's fields.

Constants	*query_rate = manager.query_rate* *response_time = max(sources.response_time) + agent.work_load* *data_size = forallsum(sources.data_size)* *communication_load = query_rate* $\times$ *env.query_communication_load* *work_load = query_rate* $\times$ *env.query_work_load*
Constraints	None (modeled by corresponding agent node)
Modifiers	*manager.response_rate += response_rate / num_sources* *agent.communication_load += communication_load* *agent.work_load += work_load*
Variables	*num_sources* = $\{2, 3, 4\}$

work_load, which is itself derived from the local *query_rate* that models how frequently the aggregator is asked to retrieve information. In this way, the characteristics of one node may affect or be affected by those of another. The resulting web of equations allows one to model important concepts such as information flow, control flow, and the effects of interactions. By propagating data through these expressions, the model can predict the characteristics of both individual nodes and the organization as a whole.

As mentioned earlier, ODML templates can describe a family of organizations, where individual members of such a family represent different decision paths through the template. For example, *topic_mediators* is a variable in this template. It can be assigned different values, which will result in organizations that have different numbers of mediators. In addition, because both *aggregator* and *database* are possible instances of *source*, mediators with the same number of sources can be further differentiated by the types of sources they manage. A complete set of decisions applied to the template will produce a particular organizational instance, such as the one shown in Figure 1b. Again, space precludes showing the complete specification for the instance, which includes much more quantitative information, but the graph depicts the relationship between entities in the structure. In this example, a single mediator manages two sources, one of which is an aggregator hierarchy, while the other is a simple database. The example instance also shows a single agent taking on two different roles, as both an aggregator and database in the left subtree. The environment node is used to capture information outside of the scope of other nodes, such as the query rate expected by the system, or the communication bandwidth available to the participants.

Like the working system it represents, there are many facets to the model we present here. Although each can be modeled as a particular, distinct characteristic of the system, they may interact through coexistence in nodes' fields. The tensions that arise in the resulting object embody the tradeoffs and decisions that must be made when designing the organization. We will discuss some of the interesting aspects below in more detail.

Roles The main portion of the organization is divided into mediator, aggregator and database nodes. In this model, these nodes do not represent particular agents by themselves. Instead, each represents a role that may exist in the organization, that is assigned a particular agent through a has-a relationship. Separating these two concepts allows the creation of more complex organizations, where agents may be assigned multiple roles,

possessing the capabilities, constraints and responsibilities of each. We also believe this separation facilitates the modeling process, by clearly identifying the individual factors which contribute to (in this case) an agent's overall place in the organization [8].

Constraints The notion of bounded rationality manifests itself in this domain within the agents. Specifically, each agent has a finite amount of processor cycles and bandwidth at its disposal. There are both "soft" effects caused by increased load, and "hard" load constraints that may not be violated. An example of the former is the increased time needed to finish any individual task as the local processing load increases. The latter occurs when the agent can no longer keep up with the requests it receives. In this case, the local work queue will grow without bound, causing an untenable situation.

Both these effects are present in the model. The relevant high level metric is the mediator's *response_time*, which represents the average length of time from query to response. As mentioned above, each role has a set of responsibilities which affect the agent it is assigned to. The model specifies the *work_load* and *communication_load* incurred by the role, which are then propagated to the agent with a pair of modifiers. During instantiation, the agent will then receive the cumulative effects of each of its roles. To model the soft effect of work delay, the *response_time* for each role is dependent on its agent's *work_load*. As the load increases, so will the delay for that particular agent. Because the response rate for an aggregator or mediator is dictated by its slowest source, this can potentially affect the performance of the entire hierarchy. The hard constraint is modeled solely within the agent, which has a pair of constraints that ensure a satisfying agent will not be assigned too much work (on average).

Task Environment The load incurred by a mediator at runtime, and by relation any sources beneath it, will be dependent on the number of queries that mediator is asked to service. This value depends on a number of factors, including the mediator's perceived value, the average number of queries arriving in the system, the number and value of competing mediators, and how many mediators are used to answer the query. To estimate this probability, we first determine the *rank* of each mediator. This ranking reflects the relative *perceived_response_size* of the mediator, which is an estimate of how good a response the mediator is expected to return based on its collection signature. For example, a mediator with many sources of information will have a higher rank than one with just a few, because its "larger" signature will cause it to be selected more frequently to answer queries. We distinguish this perceived size from the mediator's *actual_response_size*, the true quality of its response, to model the effect that signature imprecision can have on search efficiency, as mentioned in Section 2. By this definition, a mediator with higher rank will be selected over those with lower, so by determining the rank we can begin to determine the individual mediator's *query_probability*. With this, that mediator's *query_rate* can be determined, which is propagated to its *sources* so that they may estimate their individual work loads.

Utility Function A key evaluation criteria used by [12] is information recall. This metric, defined as the ratio of relevant documents retrieved to the total number of relevant documents available, objectively quantifies the quality of the query response. The mediators' *query_probability* and *actual_response_size*, along with the total amount of relevant information in the environment, can be used to determine the average information recall for the organization. A secondary metric, the response time, gives the

average amount of time the system requires to answer a query. We have previously implied how the response time of an individual agent is determined, by incorporating work load data from the roles it has been assigned. The response time of an individual mediated hierarchy ties these values together, along with the number of queries received by the mediator, the height of the tree and communication latency. The value is generated incrementally, and propagated up the tree to the mediator.

These two metrics are combined by the *organization* node in its *utility* field, which is typically used to compare and rank candidate instances. In this case, recall is more important than response time, so a multiplicative factor is applied to the recall value, after which the response time is subtracted out. This will generally favor quality over speed, but instances with equal recall will be differentiated by their response time. An arbitrary utility function could be substituted here as needs dictate.

Together, the decisions paths embodied in the template represent a wide range of possible organizations. The characteristics we have described above capture the distinguishing features that allow instances generated from the template to accurately predict how that organization will behave at runtime. We can therefore define the search space of an ODML template as the ranked set of possible organizations it produces. An appropriate organization can be found by finding the valid instances in this set (i.e. those with no unsatisfied constraints), and selecting one with sufficient utility. The optimal organization can be found be selecting the one with the highest utility.

4 Coping With Scale

As noted earlier, structured organizations become increasingly important as the number of participants grows, because of increased difficulty when coordinating, forming consensus, managing shared resources and discovering appropriate partners, among other things. It is therefore critical that any organizational representation be able to model the relevant effects of larger systems. Unfortunately, finding valid organizations from an ODML template can be very difficult, as the search space can be multiply-exponential in the template size. If we are to design organizations for hundreds, thousands or even millions of agents, different modeling techniques must be employed.

The complexity of an ODML template is derived from the number of decisions that must be made when using the structure to generate organizations, which determines the number of candidate organizations that may be derived from that template. When the number of agents that can be in the final organization increases, the number of agent-role assignments will usually increase accordingly, as may the size and number of the less tangible organizational structures such as hierarchies, teams, resource pools, etc. This potentially large number of candidate organizations can make finding the optimal, or even an appropriate, organization a difficult process.

We can begin to address the scale problem through changes to the model itself. By altering the template, one can limit the number of decisions that must be made when interpreting the template, thereby making the number of decisions less dependent on the number of agents in the system. This will reduce the number of candidate organizations, which will shrink the organizational space that must be searched.

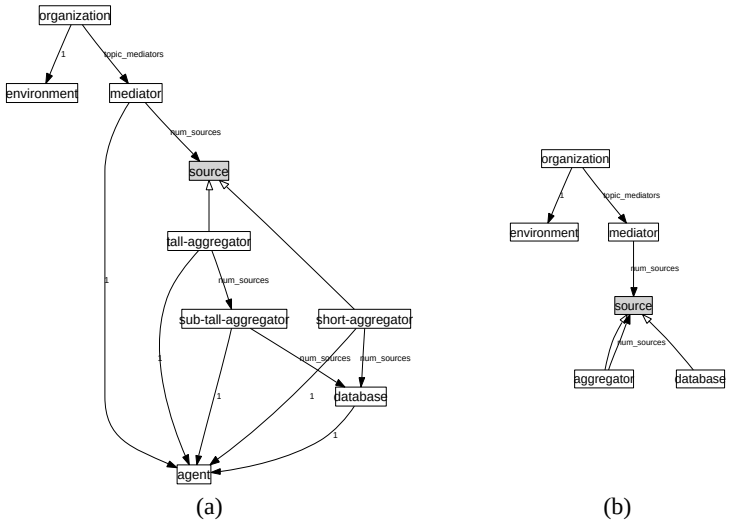

Fig. 2. Two information retrieval templates, derived from Figure 1a. a) Incorporates homogeneity, by limiting aggregator selection to two distinct choices. b) Incorporates abstraction, by eliminating the assignment of roles to distinct agents.

4.1 Homogeneity

Enforcing a certain amount of homogeneity, or at least similarity, within an organizational structure can dramatically reduce the number of decisions which must be made by eliminating organizational choices. For example, we might change our template such that all of a mediator's sources must have the same form, e.g. that they are all single level aggregators. We can improve on this strategy by exploiting ODML's inheritance rules to embed multiple distinct alternatives, rather than providing only a single choice. For example, consider the template in Figure 2a. In this model, we have defined two distinct source types, a *tall-aggregator* that has two levels, and a *short-aggregator* that has just one. Both contain a total of four databases, but they will have different performance characteristics because of their different structure. In this new template, candidate organizations may contain either or both alternatives, while other permutations of the aggregator hierarchy have been eliminated.

We view the use of homogeneity as an iterative process best exploited during the design phase. Typically, one would begin by creating a very general template, capable of producing almost all feasible organizations. As variations are generated and compared, it is common that particular organizational characteristics will define certain classes of structures or substructures. Simple examples of this include the "tall" and "short" varieties we have identified in the information retrieval domain. The members of a particular variety may be similar enough that a single representative structure can stand in for the entire class with only minimal loss of utility. For example, there are a vast number of short and wide aggregator hierarchies that have only minor differences in form and function. In Figure 2a we replaced this large number of choices with a sin-

gle *short-aggregator*, which will certainly reduce the organizational search space, and hopefully not limit the quality of the final organization. Such classes can serve as the foundation for a reduction process that captures the notion of homogeneity, by replacing a potentially complicated set of decisions with a set of predefined structures. Optimally, one could incorporating a representative from each distinguished class, producing a template with a smaller candidate search set but negligible loss of potential utility.

4.2 Abstraction

A different way to reduce the decision complexity of a model is to use abstraction to reduce elements of the structure to their simplest form. Unnecessary or optional details may be removed or captured with a probabilistic representation to eliminate branches of the template which would otherwise add to the decision process. As with homogeneity, this practice can potentially lead to an undesirable loss of expressivity in the model, but with care an appropriate compromise can usually be found.

An example of this approach, particularly relevant to decomposition-based representations such as ODML, is to truncate the model at some point higher than the level actually used by the running system. This is already used in the example models in some respects because the internal decision making processes of agents are not represented. A more typical example of this technique is to not model down to the level of assigning roles to individual entities or agents, as shown in Figure 2b. Organizations derived from this template will specify what roles exist, and where they are located in the organizational structure, but leave them otherwise unbound. This technique is analogous to those presented by Durfee in [5], which used team-level abstraction to leave specific agent assignments unbound during coordination, also to reduce complexity. If agents were heterogeneous or permitted to take on multiple roles, this can reduce the search space exponentially. Even if agents were homogeneous, in a hierarchical structure this can cut the size of instances in half, which simplifies analysis and reduces memory consumption. The precision lost in this instance stems from the details that were previously stored within individual agent nodes. For example, it is more difficult to validate an individual agent's communication or work loads. Generic agent nodes can be retained to compensate for this loss of detail, but one will not be able to predict how the combined effects of multiple roles affect the agent or its performance within the organization.

The further implication of using this technique arises from the fact that the resulting organizational instance will no longer completely specify how it should be applied to a set of resources and agents. Decisions that were previously made during the design process must now be made by an axillary process or at runtime. In the example above, roles must be assigned to specific agents before the system can function. A second process must take the agent population and map them to the nodes proscribed by the selected organizational instance, which is itself a search process [11]. Although this late binding requires additional analysis after the design phase, our belief is that it also fosters increased context-sensitivity by providing a framework to support dynamic allocation. For example, assume that the *mediator* role not been bound to a particular agent at design time. At runtime, when the actual number and types of databases are known (as opposed to the statistical averages used in our models), the organizational design can be inspected to determine what resources that role requires and what burdens it will

Table 2. Results from organizational search in small-scale information retrieval templates. Number of agents and utility are given for the optimal found organization.

Template	Decisions	Valid Organizations	Agents	Utility
Baseline	14,380,508	8539	9	6.978
Homogeneous	2,329,951	6785	9	6.978
Abstract	4947	15	9	6.978
Homogeneous + Abstract (a)	1309	6	9	6.978
Homogeneous + Abstract (b)	1050	4	8	6.975

place on the agent it is assigned to. That entity model, coupled with the new information obtained at runtime can be used to select an appropriate agent to fill that role.

4.3 Scalability Technique Examples

The exact amount of search space reduction that is observed using these techniques is dependent on the particular manner in which the template changes are carried out. Some approaches will clearly be better than others in terms of space complexity and achievable utility, and we have shown how hybrid strategies that use a limited set of decisions can help offset the drawbacks associated with model reduction.

To demonstrate the effectiveness of these techniques, we created two sets of templates for the information retrieval domain. The small set allowed up to five databases, up to two mediators, and the aggregators could have two, three or four sources. The larger design allowed up to 100 databases, with up to five mediators, each of which could have one to four sources. The number of agents was unbounded. The source node types are the same hierarchies discussed in Section 2, with a single level height restriction in the small scenario, and a three-level restriction in the large (i.e. up to two aggregators with a database leaf). Five templates were created for each scenario. The baseline template is shown in Figure 1a. The second template, using the homogeneous technique, is similar to that shown in Figure 2a. The third employs abstraction, by not assigning a particular agent to each role. Alternatively, one could also view this model as creating a new agent for each role. The fourth and fifth templates incorporate both the homogeneous and abstract techniques. Slightly different modeling changes were made in each, to demonstrate how the search space may be affected by these techniques.

Node constraints specify a maximum communication and work load for individual agents, as well as a minimum average response recall for the organization as a whole. All valid structures must satisfy those constraints. Utility was calculated primarily with the response recall, with ties broken by the average expected response time. Other environmental and behavioral values were the same between templates. A complete, depth-first search was performed for each template, and the optimal instances generated compared. Although the structures themselves are different, the expressions underlying the utility function they employ are the same, therefore the computed utility of the structures the different templates produce remain directly comparable.

The results from the small scenario are shown in Table 2. The table shows the number of decisions made during the entire search, the number of valid organizations that

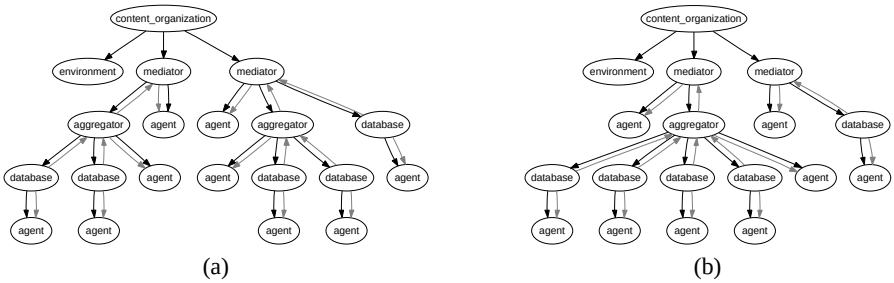

Fig. 3. Optimal organization instances produced by the Homogeneous + Abstract (a) and (b) templates for the small-scale scenario.

were found, and the number of agent and utility of the optimal structure. The most dramatic reductions in search space occurred using abstraction, which reduced the number of valid organizations by two orders of magnitude. The reduction in decisions that were made was even greater. As we will discuss below, the number of possible assignments of agents to roles can be quite large even for small organizations, so avoiding this process results in a tremendous savings in complexity. Because there was no predefined limit on the number of agents used, and no additional costs were incurred by using more agents, the optimal organizations for both the small and large scenarios use as many agents as possible. This distributes the load, which in turn decreases the time needed to answer a query, which ultimately causes such organizations to have higher utility.

The organizations produced by the two Homogeneous + Abstract templates in the small scenario can be seen in Figure 3. Template (a) allowed single hierarchies of two databases, while (b) used hierarchies of four. Although both structure have the same information recall, the optimal organization arising from template (b) had slightly lower utility. As seen in in Figure 3, the mediators in (a) are more evenly balanced than those in (b). This imbalance lead to an increase in average response time, which resulted in the slightly lower expected utility. This demonstrates the intuitive fact that different modifications will result in different search space modifications, and it is possible to lose an optimally valued solution in the process. Template (a) retained an optimally valued organization, while it was lost through the modifications to (b).

The optimal organization for three of the five large-scale database scenarios were simply too difficult to compute. Consider that, if agents may take on multiple roles, a single candidate structure containing 100 roles has 100^{100} possible assignments of agents to those roles. Even if agents may take on only a single role, there may be 100! permutations if the agents are distinguishable. Organizations produced from the structures used in this scenario may contain more than 100 roles, and there are billions of possible structures. Finding the optimal structure in such a large space is intractable. However, by incorporating the concepts of homogeneity and abstraction into the original model, we were able to design valid organizations in a reasonable amount of time. A quantitative summary of the search process for the remaining structures which employ both the abstraction and homogeneity techniques is shown in Table 3. Lacking a base-

Table 3. Results from organizational search in large-scale information retrieval templates. Number of agents and utility are given for the optimal found organization.

Template	Decisions	Valid Organizations	Agents	Utility
Baseline	Intractable, $\approx 7.4 \times 10^{301}$ candidates			
Homogeneous	Intractable, $\approx 2.2 \times 10^{108}$ candidates			
Abstract	Intractable, $\approx 7.4 \times 10^{201}$ candidates			
Homogeneous + Abstract (a)	52,792,143	473	135	14.561
Homogeneous + Abstract (b)	44,057,638	264,293	116	10.489

line comparison, we cannot state that the optimal organizations that were found had the optimal utility originally achievable. However, many different organizations were found, all of which meet or exceed the constraints specified by the original model.

The particular organizations obtained by the two remaining templates are of minor significance here, more important is the fact that we were able to use generic models to find appropriate organizations for systems incorporating more than 100 agents. These results demonstrate how generic modeling techniques can be used to reduce the complexity of an organizational search process. If suitable modifications are made, one can vastly reduce the search space with minimal reduction to utility, although the amount of reduction is clearly affected by the skill of the expert making the changes.

5 Conclusions and Future Work

All of the organizational characteristics described in Section 3 have been implemented in models created with ODML's domain-independent language. In doing so, we have produced an artifact that both describes the breadth of possible organizational alternatives, and gives the tools necessary to evaluate and compare those alternatives in a concrete, quantifiable manner. This provides the foundation upon which automated organizational design technology can be built.

We have also demonstrated how relatively simple optimizations to organizational models can be used to make organizational design for large scale multi-agent systems more realistic. Unfortunately, despite such modifications, the search space of organization models remains generally intractable with scale, and the exhaustive search approach we used quickly becomes inappropriate. Because of this, it is important to couple techniques such as those presented here with better search strategies that can make more effective decisions during the analysis. We are currently exploring the use of equivalence classes and gradient estimation as a way of bounding the search. We intend to pursue other techniques in future work.

ODML models have been used to design organizations for the information retrieval domain presented here, as well as a distributed sensor network domain described in [7]. Our hope is that the flexibility and generality of the language will allow it to be suitable for many other types of agent systems. We have tested the organizations produced by such models, and shown that the predictions they make correspond correctly to observed behaviors. A rigorous empirical comparison currently underway.

We believe that structured, explicit organization design will be a critical element in large-scale multi-agent systems. In practice, implicit organizations already exist in almost all agent systems – it is common for agents to take on different roles, interact with only a subset of the population, and have a variety of different interrelationships. By making the construction and representation of the organization an explicit, principled act it is possible to make hidden inefficiencies apparent and take full advantage of the resources at hand.

References

1. D. D. Corkill and S. E. Lander. Diversity in Agent Organizations. *Object Magazine*, 8(4):41–47, May 1998.
2. K. Decker and V. R. Lesser. Quantitative Modeling of Complex Environments. *International Journal of Intelligent Systems in Accounting, Finance and Management. Special Issue on Mathematical and Computational Models and Characteristics of Agent Behaviour.*, 2:215–234, January 1993.
3. S. DeLoach. Modeling organizational rules in the multi-agent systems engineering methodology. In *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, pages 1–15. Springer-Verlag, 2002.
4. V. Dignum, J. Vazquez-Salceda, and F. Dignum. A model of almost everything: Norms, structure and ontologies in agent organizations. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-agent Systems*, 2004.
5. E. H. Durfee and T. A. Montgomery. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1363–1378, 1991.
6. M. S. Fox. Organization structuring: Designing large complex software. Computer Science Technical Report CMU-CS-79-155, Carnegie-Mellon University, December 1979.
7. B. Horling, R. Mailler, and V. Lesser. A Case Study of Organizational Effects in a Distributed Sensor Network. In *Proceedings of the International Conference on Intelligent Agent Technology (IAT 2004)*, Beijing, China, September 2004.
8. B. Horling, R. Mailler, J. Shen, R. Vincent, and V. Lesser. Using Autonomy, Organizational Design and Negotiation in a Distributed Sensor Network. In V. Lesser, C. Ortiz, and M. Tambe, editors, *Distributed Sensor Networks: A multiagent perspective*, pages 139–183. Kluwer Academic Publishers, 2003.
9. J. F. Hübner, J. S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Proceedings of the Brazilian Symposium on Artificial Intelligence (SBIA'02)*, pages 118–128, 2002.
10. H. E. Pattison, D. D. Corkill, and V. R. Lesser. Instantiating Descriptions of Organizational Structures. *Distributed Artificial Intelligence, Research Notes in Artificial Intelligence*, I:59–96, 1987.
11. M. Sims, C. Goldman, and V. Lesser. Self-Organization through Bottom-up Coalition Formation. In *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003)*, pages 867–874, Melbourne, AUS, July 2003. ACM Press.
12. H. Zhang and V. Lesser. A Dynamically Formed Hierarchical Agent Organization for a Distributed Content Sharing System . *Proceedings of the International Conference on Intelligent Agent Technology (IAT 2004)*, September 2004.