Measurement Techniques for Multiagent Systems

Robert N. Lass, Evan A. Sultanik, William C. Regli
Drexel University
Department of Computer Science
College of Engineering
3141 Chestnut Street
Philadelphia, PA 19104
{urlass, eas28, regli}@cs.drexel.edu

ABSTRACT

Multiagent Systems (MAS) are a software paradigm for building large scale intelligent distributed systems. Increasingly, these systems are being deployed on handheld computing devices, or on non-traditional networks such as mobile ad-hoc networks and satellite links. These systems present new challenges for computer scientists in describing the performance of a system and analyzing competing systems. This paper surveys existing metrics that can be used to describe MASes and related components, and provides a framework for analyzing MASes with a case study using DCOPolis, a distributed constraint reasoning system.

1. INTRODUCTION

An agent is a situated computational process with one or more of the following properties: autonomy, pro-activity and interactivity. A multiagent system (MAS) is a system with one or more agents. MASes are a software paradigm for building large scale intelligent distributed systems. Increasingly, these systems are being deployed on handheld computing devices, or on non-traditional networks such as mobile ad-hoc networks and satellite links. These systems present new challenges for computer scientists in describing the performance of a system and analyzing competing systems.

Much of the research in this area is entirely theoretical, in the sense that no examples of large-scale systems of this type exist. As a result, most work utilizes simulators or metrics that have not been validated against real-world results. Furthermore, there is a lack of standard terminology or even an agreed upon set of functions that a MAS must provide. Hopefully the recently published Agent Systems Reference Model [7] will provide this in the same way that the OSI reference model has for the field of computer networking.

This is not to say that the simulators or metrics have no value. So many variables exist that comparing a fielded system of this type against another fielded system is not a straightforward task. In some cases the researchers may not even have access to hardware or enough experience to successfully run experiments with real systems [43].

Problems with current methods of evaluating decentralized systems are discussed at length in [23]. Specifically, the authors claim that current practices have a tendency to be inappropriately generalized, to use technically inappropriate but "standard" evaluation techniques, and to focus too heavily on feasible systems. Generalization is caused by only evaluating the performance of the system in a small portion of the environmental and workload space. Standard evaluation techniques bias research towards systems that perform well with regard to those techniques. The difficulty of establishing new methods may cause systems to be evaluated at points that are not commensurate with their intended use. As a result of these three points, research may become ossified: increasing the difficulty of making new discoveries. Lastly, the authors discuss robustness: focusing on a few evaluation points may not uncover behavior that may occur in a more dynamic environment.

The main contributions of this paper are a procedure for developing and testing frameworks and procedures for MASes to avoid these problems, and to present results from an example application of this procedure to a framework for distributed constraint optimization. As advocated in [23], the framework separates the implementation of the algorithms being studied from the platform (simulator, real 802.11 network, *etc.*) to allow code to be written once and then tested in the simulator or run as part of a real system. The latter also allows the simulation data, as well as algorithm metrics to be verified.

2. MULTIAGENT SYSTEM MODEL

Models describe relationships between components of a system to facilitate reasoning about the system. Many abstract models of MASes have been written. In this paper, we use the model derived in the Agent Systems Reference Model (ASRM), and classify metrics based on the layer of this model to which they are applied. We believe that this model is more relevant than others for applied researchers because the various components of the model were informed by reverse engineering of existing multiagent systems, rather than theories about how multiagent systems ought to operate. In addition, the ASRM was not written by a single research group, but a collection of people from industry, government and academia.

2.1 Abstract Model of a MAS

An abstract representation of a MAS is shown in Fig. 1,

PerMIS '08 August 19–21, 2008, Gaithersburg, MD, USA

and is taken from the Agent Systems Reference Model (ASRM) [7]. At the top of the diagram are *agents*, represented as triangles. Conceptually, an agent is a process with a sensor interface that determines the state of the world. It gives information about the world to a controller, performs some computation, and may result in the effector taking some action to modify the world. A thermostat could be taken as simple example: the sensor consists of a thermometer, the controller decides whether or not to turn the air-conditioner or heater on, and the effector is the air conditioner or heater interface.

The agent is supported by an *agent framework*. An agent framework is the software components that support agent execution. In some agent systems, the agent framework may be trivial, if the agents run natively on the platform (as opposed to in a virtual machine or some other local execution environment). Most agent systems, however, are based on a framework that supports key functionality agents commonly use, such as services for migration, agent messaging, and matchmaking. Examples of such systems are JADE [27], Cougaar [14], and A-Globe [22].

Under the framework is the *platform*. The platform consists of all non-agent software present, such as the operating system, databases, networking software or window managers. As depicted in Figure 1, each platform may have multiple frameworks on top.

The platform executes on a computing device, or a *host*. This is the physical computing platform on which the software is executing. A host may have multiple platforms executing on it. These hosts are distributed in the physical world, which is the bottom layer in the figure.

To summarize, measurement can take place at four layers in the model: agent, framework, platform, and environment/host. In addition, *system* measurements that cover the whole system (*i.e.* all of the components functioning together) can be taken. Within each of these layers, there are different levels and classifications of components to be measured, such as:

- **Framework:** The OSI [52] layer 7 application protocol could be analyzed, the memory footprint, cpu usage, and other framework related metrics.
- **Platform:** Except for in the trivial case where the agents run directly on the platform, the OSI [52] layers 2–6 occur within the platform. Measurement could occur at any of these levels. This means the performance of 802.11, Internet Protocol (IP), Transmission Control Protocol (TCP), Session Initiation Protocol (SIP), and Secure Sockets Layer (SSL) may all be measured, each of which is at a different OSI layer.
- **Environment:** This layer is primarily composed of the OSI layer one.

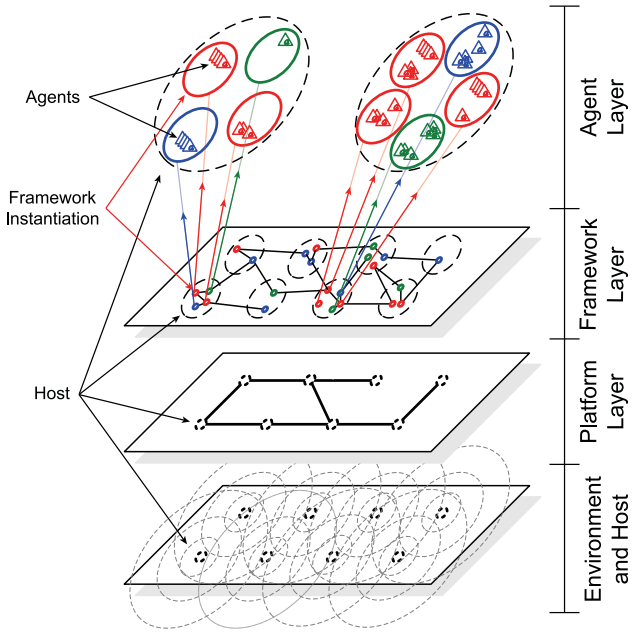

Figure 1: An abstract representation of a multiagent system. This diagram shows all of the different agents, frameworks, platforms (along with their non-agent software), and hosts needed to deliver the functionality of the system. Different colored circles represent different types of frameworks. The framework layer links show the logical connections that exist between hosts, the platform layer shows the communications connections that exist between hosts, and the environment layer shows the state of the physical medium (in this case, radio signals) used for communications between hosts.

3. METRICS SURVEY

3.1 Meta-Metrics

There are a number of *types* of metrics that can be applied at each layer of the model. They are generally classified based on their purpose, or based on the domain of the values they may take on.

3.1.1 Effectiveness vs Performance

Measures of Effectiveness (MoE) quantify the system's ability to complete its task in a given environment. In some cases this could be a binary value (the system succeeds or it fails) while in other cases it could be a range of values (the system saved x% of the hostages).

Measures of Performance (MoP) are quantitative measures of some system characteristic, such as bandwidth required, power consumed, communications range or time to perform some task. They do not describe the quality of the solution, but the quality of obtaining the solution.

Often, these two types of measures will be combined to say something about the performance required to achieve a level of effectiveness with a system. For example, one might produce a graph showing the trade off between time and solution quality for a certain system.

3.1.2 Data Classification

The most widely adopted method of data classification divides data into one of four different categories [47]. Nominal measurements are labels that are assigned to data. Ordinal measurements are rankings; greater than and less than can be applied to the measurements, but meaningful arithmetic transformations are not possible. Interval measurements are also numbers, but the difference between them has meaning. Ratio measurements are the same as interval measurements, except that there is a known zero point.

3.2 Agents and Frameworks

An agent is a situated computational process, and for our purposes agents are the components that achieve the desired functionality of the system at the highest level. The framework is a part of the system that provides functionality to the agents. Depending on the system, due to the way in which agents and frameworks are differentiated, metrics that are relevant for the agents in one system may be relevant to the framework in another system, and *vice versa*. Therefore, such metrics are categorized together herein.

A method for comparing ontology matching algorithms that is compatible with the accepted criteria of recall and precision is proposed in [17]. The author states that this is more accurate as it takes into account semantics, not just syntax.

Quantifiable measures for autonomy are described in [5]. The autonomy metric is a number between 0 and 1, and is always defined with respect to a goal and an agent. The number is calculated by determining the percentage of decisions that are made by the agent to reach the goal. For example, consider an agent a_1 working to achieve goal g_1. If one out of every four decisions used to reach g_1 were made by a_1, the autonomy metric with respect to a_1 and g_1 is 0.25. If the agents vote on decisions, then the metric is the weight that this agents vote has on the final decision. For example, if five agents each cast votes of equal weight for each decision, each agent's autonomy metric is 0.2.

For conflict resolution, metrics such as those used by the distributed constraint reasoning community may be used. Cycle-based runtime (CBR) [16], NCCC [36] and ENCCC [42] are three popular metrics based on logical clocks that may be extended to measure virtually any asynchronous decision process.

3.3 Platform

3.3.1 Distributed Systems

Many MAS are distributed systems, and it follows that techniques for analyzing distributed systems can be applied to MAS analysis. Hollingsworth summarized metrics [25] and techniques for evaluating the performance of distributed systems [24]. Lynch gives an overview of many widely used distributed algorithms, along with their analysis in [34]. The reader is referred to these three publications.

3.3.2 Networking

Networking is major component of most realistic MASes, and very diverse and active research in networking metrics is ongoing. First is a brief overview of two types of networking metrics, connectivity and capacity, followed by a discussion of MANETs.

Connectivity.

Connectivity refers to the state of the communications links between computers in a network. Often, this is represented as a graph with the nodes representing the computers and the edges representing communications links. If a computer can communicate with another computer, there is a communications link between them.

The volatility of these links depends on the type of network. On traditional wired networks, the communications links between computers are relatively static. On a MANET, the links between nodes change as the nodes move spatially.

The performance of the network at OSI layer three, specifically the routing protocol, is also critical. In [28], three different ad-hoc routing protocols are each tested under three different scenarios. In these experiments, the scenarios differed in terms of the network load.

Capacity.

There are several ways of measuring cross-layer network performance. First, by using a program such as `netperf` [29], throughput can be measured for a given network topology and configuration.

MANET.

Various metrics specifically for evaluating MANETs are described by [15], mainly at the four lowest layers of the OSI model. End-to-end throughput and delay, route acquisition time, percentage out-of-order delivery and efficiency. The latter is a general term describing the overhead involved in sending data. Three example efficiency ratios are given: bits transmitted to bits delivered, control bits transmitted to data bits delivered and control and data packets transmitted to data packets delivered. It also describes different contexts under which a MANET may operate. These contexts include network size, connectivity, topological rate of change, link capacity, fraction of unidirectional links, traffic patterns, mobility and the fraction and frequency of sleeping nodes. When evaluating the performance of a MANET system, it is important to note the context under which the researcher performs the evaluation.

3.4 Environment / Host

These metrics describe some aspect of the environment in which the system was tested are environmental measures. In the case of a robot, this might be the physical world. In the case of a software agent, this is the services, users, and other agents with which it interacts.

In [1], the quantification of the complexity of a test environment is proposed. The intent is to allow one to measure the task performance of an agent with respect to the tests' complexities. A more specific example of the complexity of a test environment is given in [38], which describes three metrics for describing the "traverseability" of terrain by a robot. Two are for roughness and one is for "crossability."

Since humans can be part of the environment in which agents operate, it may be useful to describe the level of interaction agent(s) have with them. A classification system for autonomous systems is proposed in [26], based on the complexity of the mission, independence from humans and difficulty of the operational environment. Terms and other metrics for autonomous systems are also defined in this work.

3.5 System

System metrics are overall metrics that measure something about the system as a whole. When comparing different systems for a task, often the evaluator wants a brief summary they can present to others on the overall performance or effectiveness of the system, making these metrics some of the most relevant.

An approach to evaluating performance in surveillance systems is presented in [20], along with domain specific metrics are also proposed.

There is much research in the literature on evaluating the effectiveness of robots, which have many similarities with MAS. A brief overview of metrics related to human-robot interaction is provided by [46]. The robots' performance usually cannot be measured in terms of optimality, as they have to deal with a messy environment, making it difficult to objectively assess them. Often this means that the robots are given a number of tasks that are taken to be representative and evaluated based on how well they are able to complete these tasks. For example, [4] deals with autonomous robots in a disaster scenario. The authors propose metrics that award points based on how well the robots are able to map their environment and find disaster victims. In a hybrid human-robot system, one method of analysis is to measure the effect of the robot on the human user's effectiveness at a task [9]. Other works on evaluating human-robot interaction include [21], which presents a framework for evaluating performance and testing procedures for human-robot teams, and [39] which evaluates a number of metrics for how well robots help humans complete tasks.

Another human-machine hybrid system is the integrated automobile crash warning system, presented in [18]. Three metrics describing effectiveness were given based on the warnings the system gave: percent true, percent false and percent missed. A measure of performance (see next section) was also used describing how far before an area of danger a driver will be able to stop.

A discussion of endurance testing for robots and a recommendation for all safety, security and rescue (SSR) robots to undergo endurance testing is given in [31]. Using WEKA[50], statistical analysis was performed on the failure data collected to determine the causes of fault.

The results of a long term experimental use of robots in a joint effort between Swedish academic and military organizations was described in [33]. It included a qualitative analysis of the users attitude towards the robots before, during and after the study.

When the optimal or actual solution is known, one way to evaluate effectiveness is to compare the optimal or actual solution to the solution produced by the MAS. In [19], the authors present a new approach to analysis of road recognition algorithms. In this approach, the feature extraction results are compared to actual features from a National Institute of Standards and Technology (NIST) database. The feature search trees were also used to describe the computational complexity of the search.

When it's not clear what the optimal solution is, or when there are many ways to describe the effectiveness of the system's solution, several metrics may be needed. In [45], the experimenters set up a slalom course, and wanted to measure the performance of a hybrid human-robot system at navigating the course. There is no single metric that describes how well the system performed, so a number of performance measures were recorded such as time to navigate the course, gates passed through, symbols seen, and the results of a human user's survey. The latter covered efficiency, effectiveness and user satisfaction.

Some of this work is more general. For example, in [30], the authors proposes a general effectiveness metric $P = A - B$, where A is the success metric, B is the failure metric and P is the combined performance metric. This type of metric works with a wide variety of systems that have some notion of success and failure. Similarly, in [11], the authors proposes an information theoretic metric for evaluating the quality of the amount of information processed by an intelligent system.

If a single performance metric is desired, a number of metrics can be combined. In [49], the authors propose a number of performance metrics for the Mars Rover and a formula for generating a composite performance score. The scores are combined using a technique inspired by information theory, described in [41].

A definition of performance, scalability and stability in terms of multiagent systems, and an example of analyzing a MAS for these factors is presented in [32]. In general, performance is computational cost and throughput (computational complexity and message complexity), scalability is the rate at which the overhead increases as the agent population increases, and stability is whether or not there is an equilibrium point that the system will return to after perturbations. An example of analyzing these factors is given for a MAS that solves a standard contract net [44] problem.

In many cases, a number of different metrics are needed to get a sense of the performance of the system. An example of this is [13], which investigates benchmarks for UGVs in terms of reconfiguration, communications and adaptation and learning. Another is [8], in which four metrics (2 MOE, 2 MOP) are used to evaluate an algorithm for transforming disparate data sets to a common coordinate system: convergence (MOP), speed (MOP), translation error (MOE) and rotation error (MOE).

4. ANALYSIS FRAMEWORK FOR MULTI-AGENT SYSTEMS

This section presents a framework for applying these metrics to a decision making task. There are three main components: selection, collection, and application. First the evaluator decides which metrics to use, which must be grounded in some overall goal for the system. Next, the metrics are

collected by performing experiments. Finally, the metrics are applied to the original goals to determine if the system meets the goal or perhaps if one system performs better than another.

4.1 Selection

There are an infinite number of metrics that could be applied to a system, and an infinite number of ways to apply them. How does a researcher go about deciding which metrics need to be measured for his or her system?

The Goal, Question, Metric (GQM) [6] approach for evaluation was developed during a series of projects at the NASA Software Engineering Lab. This technique is intended to provide a focus to investigation and evaluation of systems.

In this approach the evaluator first chooses goals for different products, processes, and / or resources. There are four parts to a goal: the purpose, the issue, the object and the viewpoint. The example given in [6] is "Improve (purpose) the timeliness of (issue) change request processing (object) from the managers viewpoint (viewpoint)."

Next, the evaluator selects questions, usually with quantifiable answers, that must to be answered to understand if the system meets the goal. Each goal may need multiple questions.

Finally, the metric is a set of data associated with the questions that can be subjective (depends on the point of view, such as ease of use of a UI) or objective (independent of the point of view, such as program size). This data is used to answer the questions, which in turn informs the evaluator about the goals.

As an example scenario to illustrate how the GQM approach works, consider evaluating two systems for solving a Distributed Constraint Reasoning (DCR) problem [51] on a MANET. First, we must decide on a goal, such as "Select (purpose) the system (object) providing the lowest average runtime in a bandwidth constrained environment (issue) from the point of view of the last agent to converge on a solution (viewpoint)." There is still some fuzziness to the statement, but the scope is narrower. For example, this goal is not concerned with the networking cost of a system, the amount of information an algorithm leaks to other agents, or memory utilization. There are usually multiple goals in a real evaluation, but for the rest of this example we will only look at this single goal.

The next step is to select questions that allow to characterize the object with respect to the goal, such as "How long does the system take to converge with test data A?" "How long does the system take to converge with test data B?"

Alternatively, there may be metrics or tests that are commonly used in the domain in which the system operates. For example, [31], describes a specific type of test that the authors recommend performing on a certain class of robots.

4.2 Collection

From the questions chosen in the previous section we need to select a set of metrics to collect that will allow us to answer them. In the example questions we selected we were only concerned with time to completion. So, we need to collect runtime information for the system. This could be a sophisticated solution, such as instrumenting the code to record timing information, or it could be something more informal such as having the user time it with a stopwatch.

Papers describing practices for conducting research, methods for analyzing data are classified here as "empirical methods." Some of them are general, such as [3] which provides some "rules of thumb" to keep in mind when comparing algorithms, and an example of the application of each of those rules. One widely cited resource for empirical methods for MASes is [12].

4.3 Application

In our example scenario, we asked the questions "How long does the system take to converge with test data A?" and "How long does the system take to converge with the test data B?" to help us meet the goal "Select the system providing the lowest average runtime from the point of view of the last agent to converge on a solution." Al, all that is left is to compare the runtime of each system to determine which is the lowest. If the runtime using both sets of test data is lower for one system, clearly that is the system to select. If one system has a lower runtime for test data A and another has a lower runtime for test data B, then we have to either decide which data set is most similar to the use the system will see once deployed, or we need to create new goals and re-assess the system.

5. CASE STUDY: DCOPOLIS

A large class of multiagent coordination and distributed resource allocation problems can be modeled as distributed constraint reasoning (DCR) problems. DCR has generated a lot of interest in the constraint programming community and a number of algorithms have been developed to solve DCR problems [37, 35, 40, 10]. A formal treatment of DCOP is outside of the scope of this paper, and the reader is referred to [51] for an introduction to the topic.

Informally, DCR is a method for agents to collaboratively solve constraint reasoning problems distributedly with only local information. The four main components of a DCR problem are variables, domains, agents and constraints. Each *agent* has a set of *variables*, to which it must assign *values*. Each variable has an associated *domain*, which is the set of all possible value assignments to the variable. *Constraints* are a set of functions that specify the cost of any set of partial variable assignments. Finally, each agent is assigned one or more variables for which it is responsible for value assignment. DCOP algorithms work by exchanging messages between agents, who give each other just enough information to allow each agent to make a globally optimal variable assignment.

DCOPolis [48] is a framework for comparing *and deploying* DCR software in heterogeneous environments. DCOPolis has three key points:

1. The communications platform, DCR algorithm, and problems instances are all modular and may be swapped for a truly comprehensive analysis of algorithmic performance;

2. DCOPolis contributes to comparative analysis of DCR algorithms by allowing different state-of-the-art algorithms to run in the same simulator under the same conditions or to be deployed on "real" hardware in "real" scenarios; and

3. DCOPolis introduces a new form of distributed algorithm simulation that shows promise of accurate prediction of real-world runtime.

DCOPolis has three primary abstract components: *problems*, *algorithms*, and *platforms*. The main function of DCOPolis is to provide an interface through which the three components can interact. By writing a new instance of any of these components that properly adheres to DCOPolis' API, any algorithm should be able to solve any instance of any problem while running on any platform—even without prior knowledge of such. This makes implementation and testing of new algorithms and platforms trivial.

In keeping with the example given in Section 4 ("Select (purpose) the system (object) providing the lowest average runtime in a bandwidth constrained environment (issue) from the point of view of the last agent to converge on a solution (viewpoint)."), let us apply the framework to DCOPolis running two different algorithms.

- **Agent:** DCOPolis agents are instantiated with a local view of the problem and then assign values to their variables, send messages to other agents, and change the assigned values based on the messages they received from other agents. Here, we need to record the time each agent takes to converge upon a solution.
- **Framework:** In this case, the framework is what the ASRM refers to as a "NULL framework." The functionality is contained within the agents, which interact directly with the platform through the Java Virtual Machine. There is nothing to be measured here.
- **Platform:** Any of the metrics in Section 3.3.2 can measure the performance of the network at the platform layer. Then an estimation of the system's performance were the bandwidth to drop below our test environment's could be made. Also at this level, the metric in [2] could also be used to determine which bottlenecks could be optimized, if we were interested in improving as well as comparing the systems.
- **Environment:** Our goal stated that we must be concerned with bandwidth constrained environment. The main thing to measure here is available bandwidth.

6. CONCLUSION

MAS are complicated systems made up a number of interconnected components. Measuring these systems presents new challenges, especially when these systems are deployed in dynamic environments such as mobile ad-hoc networks. This paper surveyed a number of metrics that can be used to measure these types of systems, as well as a general framework for analyzing MASes and its application to an example MAS, DCOPolis.

7. REFERENCES

[1] Michael L. Anderson. A flexible approach to quantifying various dimensions of environmental complexity. In *2004 Performance Metrics for Intelligent Systems Workshop*, 2004.

[2] Thomas E. Anderson and Edward D. Lazowska. Quartz: a tool for tuning parallel program performance. In *Proceedings of the 1990 ACM SIGMETRICS conference on measurement and modeling of computer systems*, pages 115 – 125, New York, NY, USA, 1990. ACM Press.

[3] S. Balakirsky and T. R. Kramer. Comparing algorithms: Rules of thumb and an example. In *2004 Performance Metrics for Intelligent Systems Workshop*, 2004.

[4] S. Balakirsky, C. Scrapper, S. Carpin, and M. Lewis. Usarsim: Providing a framework for multi-robot performance evaluation. In *2006 Performance Metrics for Intelligent Systems Workshop*, 2006.

[5] KS Barber and CE Martin. Agent autonomy: Specification, measurement, and dynamic adjustment. *Proceedings of the Autonomy Control Software Workshop at Autonomous Agents (Agents2019 99)*, pages 8–15, 1999.

[6] V. Basili, G. Caldiera, , and H.D. Rombach. *The Goal Question Metric Approach*, pages 528–532. John Wiley and Sons, Inc., 1994.

[7] Brandon Bloom, Christopher J. Dugan, Tedd Gimber, Bernard Goren, Andrew Hight, Moshe Kam, Joseph B. Kopena, Robert N. Lass, Israel Mayk, Spiros Mancoridis, Pragnesh Jay Modi, William M. Mongan, William C. Regli, Randy Reitmeyer, Jeff K. Salvage, Evan A. Sultanik, and Todd Urness. *Agent Systems Reference Model*. Drexel University, Philadelphia, PA, 2006. http://gicl.cs.drexel.edu/people/regli/reference_model-v1a.pdf.

[8] Bruce Brendle. 3d data registrnation based on human perception. In *2006 Performance Metrics for Intelligent Systems Workshop*, 2006.

[9] Jennifer L. Burke, Robin R. Murphy, Dawn R. Riddle, and Thomas Fincannon. Task performance metrcis in human-robot interaction: Taking a systems approach. In *2004 Performance Metrics for Intelligent Systems Workshop*, 2004.

[10] Anton Chechetka and Katia Sycara. No-commitment branch and bound search for distributed constraint optimization. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1427–1429, New York, NY, USA, 2006. ACM Press.

[11] Rama Chellappa and Amit K. Roy Chowdhury. An information theoretic evaluation criterion for 3d reconstruction algorithms. In *2004 Performance Metrics for Intelligent Systems Workshop*, 2004.

[12] Paul R. Cohen. *Empirical methods for artificial intelligence*. MIT Press Cambridge, MA, USA, 1995.

[13] Sesh Commuri, Yushan Li, Dean Hougen, and Rafael Fierro. Evaluating intelligence in unmanned ground vehicle teams. In *2004 Performance Metrics for Intelligent Systems Workshop*, 2004.

[14] BBN Corporation. Cognitive Agent Architecture (Cougaar). http://www.cougaar.org/.

[15] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501, January 1999.

[16] John Davin and Pragnesh Jay Modi. Impact of problem centralization in distributed constraint optimization algorithms. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1057–1063, New York, NY, USA, 2005. ACM Press.

[17] Jérôme Euzenat. emantic precision and recall for

ontology alignment evaluation. In *IJCAI '07: Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2007.

[18] Jack J. Ference, Sandor Szabo, and Wassim G. Najm. Performance evaluation of integrated vehicle-based safety systems. In *Performance Metrics for Intelligent Systems Workshop*, pages 85 – 89. NIST Special Publication, 2006.

[19] M. Foedisch, C. Schlenoff, and R. Madhavan. Performance analysis of symbolic road recognition for on-road driving. In *2006 Performance Metrics for Intelligent Systems Workshop*, 2006.

[20] Michael Freed, Robert Harris, and Michael Shafto. Measuring autonomous uav surveillence. In *2004 Performance Metrics for Intelligent Systems Workshop*, 2004.

[21] A. Freedy, J. McDonough, R. Jacobs, E. Freedy, S. Thayer, and G. Weltman. A mixed initiative human-robots team performance assessment system for use in operational and training environments. In *2004 Performance Metrics for Intelligent Systems Workshop*, 2004.

[22] Agent Technology Group. A-globe. http://agents.felk.cvut.cz/aglobe/.

[23] Andreas Haeberlen, Alan Mislove, Ansley Post, and Peter Druschel. Fallacies in evaluating decentralized systems.

[24] Jeffrey K. Hollingsworth, James Lumpp, and Barton P. Miller. Techniques for performance measurement of parellel programs. In *Parallel Computers: Theory and Practice*. IEEE Press, 1995.

[25] Jeffrey K. Hollingsworth and Barton P. Miller. Parallel program performance metrics: A comparison and validation. In *Proceedings of the 1992 ACM/IEEE conference on Supercomputing*, pages 4 – 13, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[26] Hui-Min Huang. The autonomy levels for unmanned systems alfus framework. In *Performance Metrics for Intelligent Systems Workshop*, pages 47 – 51. NIST Special Publication, 2006.

[27] Telocom Italia. Java Agent DEvelopment Framework (JADE). http://jade.tilab.com/.

[28] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 195–206, New York, NY, USA, 1999. ACM Press.

[29] Rick Jones. Netperf. http://www.netperf.org/netperf/NetperfPage.html.

[30] Balajee Kannan and Lynne E. Parker. Fault tolerance based metrics for evaluating system performance in multi-robot teams. In *2006 Performance Metrics for Intelligent Systems Workshop*, 2006.

[31] Jeffrey A. Kramer and Robin R. Murphy. Endurance testing for safety, security and rescue robots. In *Performance Metrics for Intelligent Systems Workshop*, pages 247 – 254. NIST Special Publication, 2006.

[32] Lyndon Lee, Hyacinth Nwana, Divine Ndumu, and Phillipe De Wilde. The Stability, Scalability and Performance of Multi-agent Systems. *BT Technology Journal*, 16(3):94–103, 1998.

[33] C. Lundberg, H.I. Christensen, and R. Reinhold. Intellectual performance using dynamical expert knowledge in seismic environments. In *2006 Performance Metrics for Intelligent Systems Workshop*, 2006.

[34] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[35] Roger Mailler and Victor Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 438–445, Washington, DC, USA, 2004. IEEE Computer Society.

[36] A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms, 2002.

[37] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 161–168, New York, NY, USA, 2003. ACM Press.

[38] V. Molino, R. Madhavan, E. Messina, T. Downs, A. Jacoff, and S. Balakirsky. Treversability metrics for urban search and rescue robots on rough terrain. In *2006 Performance Metrics for Intelligent Systems Workshop*, 2006.

[39] Dan R. Olsen. and Michael A. Goodrich. Metrics for evaluating human-robot interactions. *Proceedings of PERMIS*, 2003, 2003.

[40] Adrian Petcu and Boi Faltings. A distributed, complete method for multi-agent constraint optimization. In *CP 2004 - Fifth International Workshop on Distributed Constraint Reasoning (DCR2004)*, Toronto, Canada, September 2004.

[41] Guillermo Rodriguez and Charles R. Weisbin. A New Method to Evaluate Human-Robot System Performance. *Autonomous Robots*, 14(2):165–178, 2003.

[42] Marius Silaghi, Robert N. Lass, Evan Sultanik, William C. Regli, Toshihiro Matsui, and Makoto Yokoo. Constant Cost of the Computation-Unit in Efficiency Graphs. In *The Tenth Annual Workshop on Distributed Constraint Reasoning*, May 2008.

[43] Emin Gun Sirer. Sextant deployment (accessed 10/31/2007). http://www.cs.cornell.edu/People/egs/sextant/deployment.php.

[44] Reid G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. In *IEEE Transactions on Computers*, volume C-29 12, pages 1004 – 1113, december 1980.

[45] Brian Stanton, Brian Antonishek, and Jean Scholtz. Development of an evaluation method for acceptable usability. In *Performance Metrics for Intelligent Systems Workshop*, pages 263 – 267. NIST Special Publication, 2006.

[46] Aaron Steinfeld, Terrance Fong, David Kaber, Michael

Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich. Common metrics for human-robot interaction. *ACM SIGCHI/SIGART Human-Robot Interaction*, pages 33–40, 2006.

[47] S. S. Stevens. On the theory of scales of measurement. *Science*, 1946.

[48] Evan A. Sultanik, Robert N. Lass, and William C. Regli. DCOPolis: A framework for simulating and deploying distributed constraint optimization algorithms. In *The Ninth Annual Workshop on Distributed Constraint Reasoning*, September 2007.

[49] Edward Tunstel. Performance metrics for operational mars rovers. In *Performance Metrics for Intelligent Systems Workshop*, pages 69 – 76. NIST Special Publication, 2006.

[50] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[51] M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998.

[52] Herbert Zimmerman. OSI reference model—the ISO model of architecture for open system interconnection. *IEEE Transactions on Communications*, 28(4):425–432, April 1980.