Chapter XXV

Grid Enabled Surrogate Modeling

Dirk Gorissen

Ghent University, Belgium

Tom Dhaene

Ghent University, Belgium

Piet Demeester

Ghent University, Belgium

Jan Broeckhove

University of Antwerp, Belgium

ABSTRACT

The simulation and optimization of complex systems is a very time consuming and computationally intensive task. Therefore, global surrogate modeling methods are often used for the efficient exploration of the design space, as they reduce the number of simulations needed. However, constructing such surrogate models (or metamodels) is often done in a straightforward, sequential fashion. In contrast, we present a framework that can leverage the use of compute clusters and grids in order to decrease the model generation time by efficiently running simulations in parallel. We describe the integration between surrogate modeling and grid computing on three levels: resource level, scheduling level and service level. Our approach is illustrated with a simple example from aerodynamics.

INTRODUCTION

Computer based simulation has become an integral part of the engineering design process. Rather than building real world prototypes and

performing experiments, application scientists can build a computational model and simulate the physical processes at a fraction of the original cost. However, despite the steady growth of computing power, the computational cost to perform

these complex, high-fidelity simulations are still enormous. A simulation may take many minutes, hours, days or even weeks (Gu, 2001; Lin et al., 2005; Qian et al., 2006). This is especially evident for routine tasks such as optimization, sensitivity analysis and design space exploration as noted below:

"...it is reported that it takes Ford Motor Company about 36-160 hrs to run one crash simulation. For a two-variable optimization problem, assuming on average 50 iterations are needed by optimization and assuming each iteration needs one crash simulation, the total computation time would be 75 days to 11 months, which is unacceptable in practice" (Wang and Shan, 2007, p1).

Consequently, scientists have turned towards upfront approximation methods to reduce simulation times. The basic approach is to construct a simplified approximation of the computationally expensive simulator, which is then used in place of the original code to facilitate Multi-Objective Design Optimization (MDO), design space exploration, reliability analysis, and so on (Simpson, 2004). Since the approximation model acts as surrogate for the original code, it is referred to as a *surrogate model* or *metamodel*.

While the time needed for one evaluation of the original simulator is typically in the order of minutes or hours, the surrogate function, due to its compact mathematical notation, can be evaluated in the order of milliseconds. However, in order to construct an accurate surrogate one still requires evaluations of the original objective function, thus cost remains an issue. The focus of this paper is to discuss one technique to reduce this cost even further using distributed computing. By intelligently running simulations in parallel, the "wall-clock" execution time, in order to come to an acceptable surrogate model can be considerably reduced.

We present a framework that integrates the automated building of surrogate models with

the distributed evaluation of the simulator. This integration occurs on multiple levels: resource level, scheduling level and the service level. Each of these will be detailed below.

BACKGROUND

Surrogate Modeling

Surrogate models play a significant role in many disciplines (hydrology, automotive industry, robotics, ...) where they help bridge the gap between simulation and understanding. The principal reason driving their use is that the simulator is too time consuming to run for a large number of simulations. A second reason is when simulating large scale systems, for example: a full-wave simulation of an electronic circuit board. Electro-magnetic modeling of the whole board in one run is almost intractable. Instead the board is modeled as a collection of small, compact, accurate surrogates that represent different functional components (capacitors, resistors, etc.) on the board.

There are a huge number of different surrogate model types available, with applications in domains ranging from medicine, ecology, economics to aerodynamics. Depending on the domain, popular model types include Radial Basis Function (RBF) models, Rational Functions, Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Kriging models (Wang and Shan, 2007).

An important aspect of surrogate modeling is sample selection. Since data is computationally expensive to obtain, it is impossible to use traditional, one-shot, full factorial or space filling designs. Data points must be selected iteratively, there where the information gain will be the greatest (Kleijnen, 2005). A sampling function is needed that minimizes the number of sample points selected in each iteration, yet maximizes the information gain of each sampling step. This process is called adaptive sampling, but is also

known as active learning, Optimal Experimental Design (OED), and sequential design.

Grid Computing

Almost two decades old, grid computing has become an established computing paradigm that integrates heterogeneous resources (computers, networks, data archives, instruments, etc.) in an interoperable virtual environment crossing both geographical and institutional boundaries (Berman, 2003). These resources are coordinated to provide transparent, dependable, pervasive and consistent computing support to a wide range of applications. These applications can perform either distributed computing, high throughput computing, on-demand computing, data-intensive computing, collaborative computing or multimedia computing.

A grid computing user interacts with the raw grid resources through a software layer, referred to as the middleware. The middleware is responsible for managing the grid resources (access control, job scheduling, resource registration and discovery, etc.), abstracting away the details and presenting the user with a consistent, virtual computer to work with. A large number of general purpose middleware have been developed, examples include: Globus, Unicore, Legion, JGrid and VgrADS.

SURrogate MOdeling (SUMO) Toolbox

The SUMO Toolbox (Gorissen, 2006) is an adaptive tool that integrates different modeling approaches and implements a fully automated, global surrogate model construction algorithm. Given a simulation engine, the toolbox automatically generates a surrogate model within the pre-defined accuracy and time limits set by the user. However, at the same time keeping in mind that there is no such thing as a 'one-size-fits-all', different problems need to be modeled differently.

Therefore the toolbox was designed to be modular and extensible but not be too cumbersome to use or configure. Different plugins are supported: model types (neural networks, Kriging, splines, ...), model parameter optimization algorithms (BFGS, GA, PSO, ...), adaptive sample selection (density based, gradient based, ...), and sample evaluation (local, on a cluster or grid). The behavior of each component is configurable through a central XML configuration file and components can easily be added, removed or replaced by custom, problem-specific, implementations. This is illustrated in Figure 1.

The SUMO toolbox has been successfully applied to a very wide range of fields ranging from combustion modeling in metallurgy to structural mechanics modeling in the car industry. Its success primarily due to its flexibility, self tuning implementation, and its ease of integration into the larger computational science and engineering pipeline.

It is instructive to go through the control loop of the toolbox: First an initial set of samples is chosen according to some experimental design. Based on this initial set, one or more surrogate models are constructed and their parameters optimized according to an optimization algorithm (e.g., pat-

Figure 1. SUMO toolbox plugins

tern search). Models are assigned a score based on one or more measures (e.g., cross validation) and the model parameter optimization continues until no further improvement is possible. The models are then ranked according to their score and new samples are selected based on the top k models. The model parameter optimization resumes and the whole process repeats itself until one of the following three conditions is satisfied: (1) the maximum number of samples has been reached, (2) the maximum allowed time has been exceeded, or (3) the user required accuracy has been met.

MAIN FOCUS

Integrating Grid Computing and Surrogate Modeling

Research efforts that integrate surrogate modeling and design space exploration with grid computing can be divided into two categories: those catered towards design optimization and those geared towards the building of standalone global surrogate models.

The first category is by far the most populous: there are many grid-enabled optimization frameworks which can be applied to different problem domains. The most notable are Nimrod/O (Abramson, 2001), DAKOTA (Giunta and Eldred, 2000), GEODISE (Eres, 2005), and the work by Y. S. Ong et al (Ng, 2005).

Since these projects are tailored towards optimization (local models), they are not concerned with creating a surrogate that can be used on its own (global models). Research efforts that do build replacement surrogate models exist (Hendrickx and Dhaene, 2005; Weiss, 2005; Busby, 2007), but fail to include concepts of distributed computing. Thus the repetitive process of evaluating the objective function while constructing the surrogate is done sequentially, nor is there any tie-in with the sample selection process. We were unable to find evidence of other projects that tackle this.

Integration: Resource Level

When constructing a global surrogate model for an expensive simulation engine the largest computational bottleneck is performing the necessary simulations. An obvious step is to harness the power of the grid and run the simulations in parallel. By distributing the evaluation back-end of the toolbox, (the *Sample Evaluator*) the overall run time of the SUMO Toolbox can be significantly reduced.

A high level design diagram is shown in Figure 2.

The Sample Evaluator (SE) can be seen as a kind of Application Aware Scheduler (AAS) that forms the glue between the modeler and the middleware. It is responsible for translating modeler requests (i.e., evaluations of data points) into middleware specific jobs (e.g., <Task> tags in the case of APST, jdl files in the case of LCG), polling for results, and returning them to the modeler. The SE is implemented as a set of Java interfaces and base classes that are sub classed for each of the supported middlewares. This point is worth emphasizing. There are a large number of distributed middlewares available and their usage varies significantly between institutions. Therefore it is important not to restrict the integration to one particular middleware. The SE must be designed in such a way that different backends can be easily added and replaced (i.e., the SE acts as a meta-scheduler). At the same time the SE should hide as much of the complexity of the grid as possible. Besides a few configuration options, the user should not have to worry about the intricacies of each job authentication or submission system.

How this works in practice is illustrated in Figure 3. There is a separate delegate for each step in the sample evaluation process. The workflow is as follows: The modeler selects a number of data points that need to be simulated and sends them to the SE. The SE passes them on to the distributed backend (specified in the configuration file, in

Figure 2. High level components

Figure 3. SGE sample evaluator backend

this case Sun Grid Engine (SGE)) that translates each data point into a middleware specific job request, stages the necessary input files, and submits it to the grid. A poller object is then instantiated which monitors the job state (finished, failed, running, waiting,). If a completed job is detected, the poller notifies a result processor object which takes care of retrieving the results and returning them back to the modeler. If a job

failure is detected, the simulation is automatically re-submitted k times before regarding the job as permanently failed.

Thus, in sum, adding support for a new middleware (e.g., Condor) means providing a new {Distributed Backend, Poller, Result Processor} triplet.

Integration: Scheduling Level

Simply running simulations in parallel already gives a significant performance improvement. Performance can be improved even further if the intelligence of the SE is increased. It should be realized that not all data points are equally important, a partial ordering exists. For example, data points lying close to interesting features (e.g., extrema, domain boundaries), or far from other data points have a higher priority. These priorities are assigned by the sample selection algorithm and dynamically managed by the input queue which is a priority queue. Consequently, the priorities are reflected in the scheduling decisions made by the SE and distributed backend (simulations with higher priority are done first). The priority queue can have different management policies. For example, a policy can be to let the priority decrease with time unless interest in a sample point is renewed.

Additionally, the number of simulations to run (as requested by the modeler) is not fixed but changes dynamically. This number is calculated based on average time needed for modeling, the average duration of a single simulation, and the number of compute nodes available at that point in time. In this way the underlying resources are always used optimally.

Eventually this scheme will be extended to directly integrate the SE with the grid information system used (e.g., Ganglia). Combining knowledge on **real-time** system load, network traffic, ... with data point priorities would allow the SE to achieve an optimal *job-host* mapping (i.e., the data points with the highest priority should be scheduled on the fastest nodes).

Integration: Service Level

The previous two subsections exemplify the main reasons for traditionally turning to grid computing: computational power. However, the past few years Service Oriented Architectures

(SOA) have become an increasingly popular (if not dominant) way to think about the grid. In this regard the grid is a heterogeneous collection of services, where each service provides access to a particular resource. Examples include services providing access to: a printer, a high performance numerical library, storage space, or CPU power. Users can connect to these services using standard technologies such as Jini or SOAP and use them as part of complicated workflows.

In this sense, "automated construction of surrogate models" is a prime example of a service that a scientist or engineer can use to delegate surrogate model construction to. The advantages are obvious: there are no setup or maintenance costs and interfacing is straightforward. Thus surrogate model construction can easily be integrated into the larger engineering design process, enhancing productivity.

Example Application

Test Problem

As an intuitive example to illustrate the discussion above we present the following application. The goal is to model the aerodynamic properties (lift, drag) of a winged box kite given 4 geometric parameters: h_1, h_2, w_1, w_2 . This model can then be used for optimization, visualization, or as part of an educational kite design tool.

The kite and the different parameters are shown in Figure 4. We shall use this example to illustrate integration on the resource and scheduling level.

The example is interesting since it is easy to grasp and explain, but complex enough since the relationships between the variables are not immediately obvious. Additionally it is a good example, although it is somewhat academic, of what a surrogate model is used for. For example, an engineer may be interested in what parameter assignment gives the optimal lift/drag ratio. Instead of having to solve the full flow equations

Figure 4. Winged box kite

for each assignment a global surrogate model can be constructed that gives insight into the relationships between the different parameters (design space exploration) and help identify promising optima. The simulation code is freely available from NASA¹, written in Java, and was only slightly modified by the authors. The running time is quite short, about one minute, but long enough to be useful and illustrative.

Experimental Setup

The simulation code is distributed on a shared pool of 256 Sun Fire V20z nodes, running SUSE Linux and Matlab R2007a, administered by Sun Grid Engine (SGE) and accessible through a remote head node (meaning the toolbox has to tunnel job submissions through SSH). The resource pool is available through a number of submission queues and the toolbox dynamically switches to the queue with highest availability. The SUMO toolbox starts with an initial Latin hypercube sample distribution of 50 samples (in the 4D design space) and runs up to a maximum of 1000. The number of samples selected each iteration is selected dynamically and samples with higher priorities are scheduled first. For this example, a fully sequential approach would require 16-19 hours of computation time.

As a surrogate model type Kriging models are used. Initially developed within Geo-statistics, Kriging models (also referred to as Gaussian Process (GP) models) have become particularly popular in recent years due properties such as natural uncertainty estimation, interpolation, and fast training times. It is for these reasons that Kriging is used here, though other surrogate model types could have been used just as well.

The quality of a surrogate model is determined by k -fold cross validation (using a root mean square error function). To improve the modeling process the input parameters were scaled to $[-1,1]$.

Results

Figure 5 shows the final Kriging surrogate for the lift (for discrete values of w_1, w_2). A full discussion of the model is out of scope for this paper. In sum we can say the model agrees with intuition: The most lift occurs when the area of the kite is the largest (high h_1, w_2) and the least lift where the area is smallest. Interestingly though, the lift also increases for high h_2 , low h_1 , and $w_2 = -1$. This could point to a potential inaccuracy of the model or a subtlety of the simulation code. More input from a domain expert would be needed in order to fully understand the relationships.

The total simulation time for the experiment was only 1.2 hours with an average availability of 20 nodes. Compared with a purely sequential approach this results in a speedup factor of about 15². The speedup is not the only quality metric. Recall that the SUMO Toolbox handles all interfacing with the grid automatically and transparently (even through ssh). Besides some initial configuration (user credentials, executable path, ...) no user intervention is required.

FUTURE TRENDS

The application of grid computing concepts and application aware scheduling to global surrogate

Figure 5. Final Kriging model for the lift (w_1, w_2 are plotted at 3 discrete intervals: -1, 0, 1)

modeling is still an open field. Most of the work so far has been done on the lowest level of integration with already promising results (Gorissen, 2006). The problem here is interfacing with the different middleware in a flexible, extensible manner. Each middleware has its own characteristics and semantics which makes it difficult to support different middleware in a transparent way. Luckily, work on different meta-schedulers (e.g., Grid Way, ProActive) and standardization efforts (CoG kit, DRMAA, ...) is underway to tackle this problem. The future should bring promising results in this respect.

Concerning the higher levels of integration, the authors know of no other related work. So much still remains to be done. It will be interesting to see how the problems of priority management integration with resource information will be tackled. In this respect there is a vast body of knowledge available from the scheduling communities in operating systems, distributed systems and grid-economics that surrogate model researches can leverage.

Finally, integration at the service level will play a very important role in bringing advanced surrogate model construction techniques closer to the scientists and engineers that need them. Easy, standardized (programmatic) access to a surrogate model construction service will greatly help engineers to faster explore, optimize and understand the design space of their problem

without all the usual installation, configuration and interfacing costs.

CONCLUSION

Due to the computational complexity of current simulation codes, the use of *global* surrogate modeling techniques has become standard practice among scientists and engineers alike. However, these techniques are still very much applied in a one shot manner (collect data, do a regression, evaluate the model). More adaptive, integrated approaches have been described in literature (Farhang-Mehr and Azarm, 2005; Busby, 2007) but have found little or no use outside the labs that designed them. In addition, the authors found no evidence of the use of distributed computing in the global surrogate model construction process, thus expensive simulator evaluations are still being performed sequentially. Even if parallelization is used, integration with the sample selection and scheduling layer is lacking.

In this chapter we have made the case for the use of distributed computing while building global surrogate models. The use of grid computing allows one to run expensive simulations in parallel (resource level), optimally make use of available resources (scheduling level) and make surrogate modeling algorithms easily available to the engineers that need them (service level).

We have presented an adaptive framework based on the SUMO Toolbox that achieves this and can be downloaded from <http://www.sumo.intec.ugent.be>.

REFERENCES

- Abramson, D., Lewis, A., Peachey, T., & Fletcher, C. (2001). An automatic design optimization tool and its application to computational fluid dynamics. In *Proceedings of the 2001 ACM/IEEE conference on supercomputing* (pp. 25–25).
- Berman, F., Fox, G., & Hey, T. (2003). The Grid: past, present, future. In F. Berman, G. Fox, & T. Hey (Eds.), *Grid computing - making the global infrastructure a reality*. John Wiley & Sons.
- Busby, D., Farmer, C. L., & Iske, A. (2007). Hierarchical nonlinear approximation for experimental design and statistical data fitting. *SIAM Journal on Scientific Computing*, 29(1), 49–69.
- Eres, M. H., Pound, G. E., Jiao, Z., Wason, J. L., Xu, F., Keane, A. J., et al. (2005). Implementation and utilization of a grid-enabled problem solving environment in Matlab. *Future Generation Comp. System*, 21(6), 920–929.
- Farhang-Mehr, A., & Azarm, S. (2005). Bayesian meta-modeling of engineering design simulations: a sequential approach with adaptation to irregularities in the response behavior. *International Journal for Numerical Methods in Engineering*, 62(15), 2104–2126.
- Giunta, A., & Eldred, M. (2000). Implementation of a trust region model management strategy in the DAKOTA optimization toolkit. In *Proceedings of the 8th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization*. Long Beach, CA.
- Gorissen, D., Hendrickx, W., Crombecq, K., & Dhaene, T. (2006). Integrating grid computing and metamodeling. In *Proceedings of 6th IEEE/ACM international symposium on cluster computing and the grid (CCGrid 2006)* (p. 185–192). Singapore.
- Gu, L. (2001). A comparison of polynomial based regression models in vehicle safety analysis. Paper DETC/DAC-21063, *Proceedings of DETC'01 ASME 2001 Design Engineering Technical Engineering Conferences and the Computers and Information in Engineering Conference*, Pittsburgh, PA, 9–12 September.
- Hendrickx, W., & Dhaene, T. (2005). Multivariate modelling of complex simulation-based systems. *Proceedings of the IEEE NDS 2005 conference* (pp. 212–216).
- Kleijnen, J. P., Sanchez, S. M., Lucas, T. W., & Cioppa, T. M. (2005). State-of-the-art review: A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17(3), 263–289.
- Lin, Y.-C., Fregly, B., Haftka, R., & Queipo, N. (2005). Surrogate-based contact modeling for efficient dynamic simulation with deformable anatomic joints. In *Proceedings of the tenth international symposium on computer simulation in biomechanics* (p. 23–24).
- Ng, H.-K., Ong, Y.-S., Hung, T., & Lee, B.-S. (2005). Grid enabled optimization. In *Advances in Grid Computing - EGC 2005* (LNCS, 296–304).
- Qian, Z., Seepersad, C. C., Joseph, V. R., Allen, J. K., & Wu, C. F. J. (2006). Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design*, 128(4), 668–677.
- Simpson, T. W., Booker, A. J., Ghosh, D., Giunta, A. A., Koch, P. N., & Yang, R.-J. (2004). Approximation methods in multidisciplinary analysis and optimization: A panel discussion. *Structural and Multidisciplinary Optimization*, 27(5), 302–313.
- Wang, G. G., & Shan, S. (2007). Review of meta-modeling techniques in support of engineering

design optimization. *Journal of Mechanical Design*, 129(4), 370-380.

Weiss, L., Amon, C., Finger, S., Miller, E., Romero, D. & Verdinelli, I. (2005). Bayesian computer-aided experimental design of heterogeneous scaffolds for tissue engineering. *Computer Aided Design*, 37, 1127-1139.

KEY TERMS AND DEFINITIONS

Experimental Design: The theory of Design of Experiments (DOE) describes methods and algorithms for optimally selecting data points from an n dimensional parameter space. A simple example, say you have to select 1000 points from a 3-dimensional space ($n=3$). This can be done randomly, using a full factorial design (an equal number of points in every dimension, i.e., $10 \times 10 \times 10$) or according to a Latin hypercube. These are 3 basic examples of an experimental design.

Meta-Scheduler: A meta-scheduler is a software layer that abstracts the details of different grid middlewares. In this way a client can support multiple submission systems while only having to deal with one protocol (that used by the abstraction layer). An example of a meta-scheduler is GridWay (www.gridway.org).

Middleware: The middleware is responsible for managing the grid resources (access control, job scheduling, resource registration and discovery, etc.), abstracting away the details and presenting the user with a consistent, virtual computer to work with. Examples of middlewares include: Globus, Unicore, Legion and Triana.

Sequential Design: For a high number of dimensions, $n > 3$, it quickly becomes impossible to use traditional space filling experimental

designs since the number of points needed grows exponentially. Instead data points must be chosen iteratively and intelligently, there where the information gain is the highest. This process is known as sequential design, adaptive sampling or active learning.

Service Oriented Architecture (SOA): SOA represents an architectural model in which functionality is decomposed into small, distinct units (services), which can be distributed over a network and can be combined together and reused to create applications.

Surrogate Model: This is a model that approximates a more complex, higher order model and used in place of the complex model (hence the term *surrogate*). The reason is usually that the complex model is too computationally expensive to use directly, hence the need for a faster approximation. It is also known as a *response surface model* or a *metamodel*.

Workflow: A workflow is a set of tasks (=nodes) that process data in a structured and systematic manner. In the case that each node is implemented as a service, a workflow describes how the services interact and exchange data in order to solve a higher level problem.

ENDNOTES

¹ <http://www.grc.nasa.gov/WWW/K-12/InteractProgs/index.htm>

² Using a middleware incurs an extra overhead (submission, staging, polling, ...) which explains a speedup < 20

PUBLISHING ACADEMIC EXCELLENCE SINCE 1988
 Publisher of Innovative Books, Journals, Cases, and
 Databases in Science, Technology, and Medical Informatics

[BOOKS](#) | [BOOK SERIES](#) | [JOURNALS](#) | [PROCEEDINGS](#) | [TEACHING CASES](#) | [PAY-PER-VIEW](#) | [REFERENCE](#) | [E-RESOURCES](#) | [ABOUT IGI](#)
[BECOME AN AUTHOR/EDITOR](#) | [MAILING LIST](#) | [HOW TO ORDER](#) | [LIBRARY SUGGESTION](#) | [EXAMINATION REQUESTS/COURSE ADOPTION](#) | [DISTRIBUTORS](#)

IGI Online Bookstore Reference Exact Search

About This Book

- Description & Key Features
- Topics Covered
- Accolades
- Contributors
- Editorial Advisory Board
- Table of Contents
- Preface
- About the Editors
- View the Brochure
- View the Excerpt

- [Reference Home Page](#)
- [Recommend to your Library](#)

Authoritative -
Innovative -
Comprehensive

Take Advantage of the Pre-Publication Discount by Ordering this Book Today!

Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications

Edited By: **Emmanuel Udoh**, Indiana University-Purdue University, USA; **Frank Zhiqiang Wang**, Cranfield University, UK

Description:

Today, many institutions and corporations are involved in various grid researches that involve improvement in concepts, protocols, applications, methods, and tools. To help capture the rapid growth in grid concepts and applications and chart the future direction in this field, a resource compiling current advances in grid computing is essential.

The **Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications** provides a compendium of terms, definitions, and explanations of concepts, issues, processes, and trends in grid technology. As the first book collection of its kind, this reference features current articles presenting comprehensive coverage of grid technologies valuable to any researcher, practitioner, or academician in grid computing.

Key Features:

- 30 authoritative contributions by the world's leading experts in grid technologies and utility computing
- Comprehensive coverage of each specific topic, highlighting recent trends and describing the latest advances in the field
- More than 650 references to existing literature and research on grid technologies and utility computing
- A compendium of over 200 key terms with detailed definitions
- Organized by topic and indexed, making it a convenient method of reference for all IT/IS scholars and professionals
- Cross-referencing of key terms, figures, and information pertinent to grid technologies and utility computing

ISBN: **978-1-60566-184-1**
 Hard Cover
 Publisher: **Information Science Reference**
 Copyright: **May 2009**
 Pages: **396**
 List Price: **\$265.00**
 Pre-Pub Price: **\$245.00**

[add to cart](#)

Perpetual Access:
\$400.00
[add to cart](#)

Print + Perpetual Access:
\$530.00
[add to cart](#)

PRE-PUBLICATION
PRICE IS GOOD UNTIL
JULY 1, 2009

**Information Science
REFERENCE**
The premier reference source for information science & technology research