A Grid-Computing Framework for Quadratic Programming under Uncertainty

Ankur Kulkarni Albert Rossi Jay Alameda Uday V. Shanbhag

February 8, 2007

***Abstract*— Mathematical programming under uncertainty concerns the optimal management of resources when there is stochasticity in the data. Specifically, this requires making a (first-period) decision before the realization of uncertainty. In addition, recourse-based formulations react to the randomness by reacting through (second-period) *recourse* decisions. Such formulations allow a decomposition into a master-worker framework. We consider the solution of such problems by developing grid-computing extensions of two algorithms for a class of problems with quadratic objective functions and linear constraints. A description of the framework for implementing these algorithms on the TeraGrid is provided. Some preliminary computational experience on a serial platform is reported.**

I. Introduction

In this paper, we focus on the stochastic quadratic program [13], [14], [3]:

$$\begin{array}{lll} \text{SQP } \min_{x,y^\omega} & \frac{1}{2}x^TQx + c^Tx + \mathbb{E}_\omega[\frac{1}{2}(y^\omega)^TD_\omega y_\omega + d_\omega^T y_\omega] \\ & Ax = b \\ \text{s/t} & A_\omega x + By_\omega = b_\omega \\ & x, y_\omega \geq 0, \end{array}$$

where ω is defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. In addition, we assume that Q and D_ω are positive definite for all $\omega \in \Omega$ and A, A_ω and B are of full row-rank for all $\omega \in \Omega$. Furthermore, we assume that the sample-space Ω is finite with ω taking on $K(= |\Omega|)$ realizations.

The structure of SQP implies that we make a first-stage decision x and then make recourse decisions y_ω based on the realization of random variable ω. The recourse decisions come at a cost and have to stay feasible with respect to a set of random constraints. Such a formulation has been employed for a variety of problems, ranging from asset-management to inventory control to network design [10], [3] with uncertanty often lying in the specification of demand or price. [1] Such problems belong to the broader class of decision-making problems called stochastic programs [6], [2], [3], [10] and may take a variety of formulations. Our interest is in the two-stage structure in which we make decisions before the uncertainty reveals itself and subsequently make recourse decisions, after the uncertainty is revealed.

The aim of this paper is to communicate our efforts to solve SQP on a grid-computing environment. This entails extending existing middleware structures to accommodate specific concerns (such as asynchronicity) as they relate to such problems.

This paper is organized in the following fashion. In section II, we introduce two decomposition-based approaches for solving the stochastic quadratic program. Section III describes the grid-based implementation while section IV provides some initial computational results. We conclude in section V.

II. Description of Algorithm

We suggest two algorithms for solving the stochastic quadratic program, both of which are provably convergent. These algorithms are based on the **L-shaped method** for solving stochastic linear programs [15]. Chapter 5 in [3] provides a complete explanation of the L-shaped method for stochastic linear programs. The L-shaped method for quadratic programs uses the same idea to reduce the SQP above to a problem of the kind shown below.

$$\begin{array}{lll} \text{SQP} & \min_{x,\theta} & \frac{1}{2}x^TQx + c^Tx + \theta \\ & & Ax = b \\ & \text{s/t} & \theta \geq \mathcal{Q}(x) \\ & & x, \theta \geq 0. \end{array}$$

where $\mathcal{Q}(x) = \mathbb{E}_\omega \mathcal{Q}(x;\omega)$, where $\mathcal{Q}(x;\omega)$ is defined as

$$\min_{y_\omega} \left[\tfrac{1}{2}(y_\omega)^TD_\omega y_\omega + d_\omega^T y_\omega | A_\omega x + By_\omega = b_\omega, y_\omega \geq 0\right]. \quad (1)$$

The L-shaped method approximates the nonlinear function $\mathcal{Q}(x)$ by a series of cuts (linear inequality constraints) [3]. For stochastic quadratic programs, the recourse function $\mathcal{Q}(x)$ can be shown to be convex for all x in the domain [13]. The approximations to $\mathcal{Q}(x)$ are achieved by means drawing a series of tangents to $\mathcal{Q}(x)$ and creating thus an outer linear approximation of $\mathcal{Q}(x)$. In effect, we replace this function by a successively more accurate set of linear constraints. The resulting quadratic program, called the master problem, computes x. The determination of the cuts requires the solution of K quadratic programs which may be solved in parallel. We describe the computational architecture next.

Ankur Kulkarni and Uday V. Shanbhag are with the Department of Industrial and Enterprise Systems Engineering at the University of Illinois at Urbana-Champaign, e-mail akulkar3@uiuc.edu and udaybag@uiuc.edu

Albert Rossi and Jay Alameda are with the Middleware Research Group, National Center for Supercomputing Applications (NCSA), e-mail jalameda@ncsa.uiuc.edu and arossi@ncsa.uiuc.edu

[1] An example of a recourse-decision in the context of inventory management would be to store excess production, under an assumption of random demand.

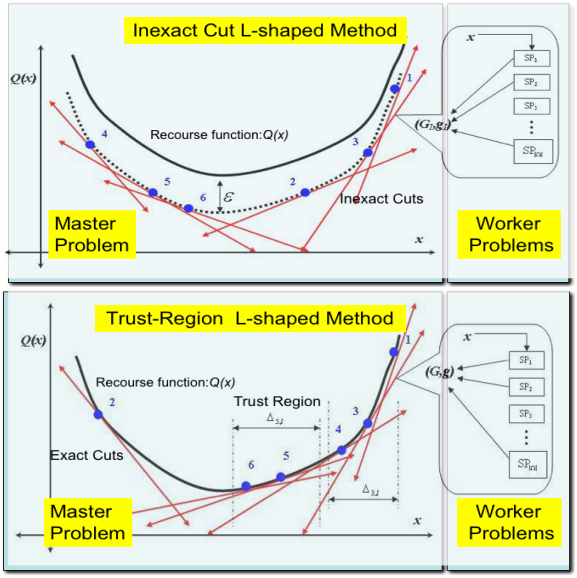

Fig. 1. Inexact-Cut (Upper) and Trust-region (lower) L-shaped Decomposition Methods

A. Master-worker structure

The minimization problem in (1) is completely decomposable into scenario-based subproblems. Thus the SQP can now be decomposed into a **master problem** in (x, θ) and K **subproblems** in y^{ω}, each parameterized by x and ω, making it amenable for solving on a computational grid. For both algorithms that we present here, the master problem is assigned to one node of the cluster, the master node. Each worker gets assigned a set of subproblems parametrized by the subproblem numbers. This assignment is kept constant throughout the execution of the algorithm.

B. L-shaped Method with Inexact Cuts

For explaining our algorithms for solving the SQP, it is useful to define the dual of the ω^{th} subproblem as

$$\max_{\pi_\omega, z_\omega} \left[(b_\omega - A_\omega)^T z_\omega - \frac{1}{2}\pi_\omega^T D_\omega \pi_\omega | - D_\omega \pi_\omega + B^T z_\omega \leq d_\omega \right]$$

Following on the lines of Zakeri et al. [16], the L-shaped method was extended to stochastic QPs by Shanbhag et al. [13], [14] by using inexact cuts. Suppose each dual problem is solved to feasibility, with an optimality tolerance of ϵ. The resulting constraint is an ϵ-inexact cut $(G_I^T x + g_I + \epsilon > \mathcal{Q}(x))$, where $G_I := \mathbb{E}_\omega \left[-z_\omega^T A_\omega \right]$ and $g_I := \mathbb{E}_\omega \left[z_\omega^T b_\omega - \frac{1}{2}{\pi_\omega}^T D_\omega \pi_\omega \right]$.
The algorithm:

- Set $n = 1$ and choose an update parameter u. Upper bound $= \infty$ and lower bound $= -\infty$. Pick ϵ_n
- Solve the master problem exactly to get (x_n, θ_n). Update the lower bound.
- Using x_n, solve all K subproblem-duals to ϵ_n tolerance
- Add cut (G_n^I, g_n^I) to the family of optimality cuts
- Update the upper bound. Check for termination
- $\epsilon_{n+1} = \epsilon_n / u$, $n = n + 1$

The inexact cuts form a weaker outer-linearization of $\mathcal{Q}(x)$(relative to what would be obtained from using exact cuts). The ϵ tolerance is reduced steadily as the iterations proceed. It can be shown that as $n \to \infty, \theta_n \to \mathcal{Q}(x_n)$, and $x_n \to x^*$ [13], [14].

C. Trust-Region L-shaped Method

Linderoth and Wright have discussed a trust region based approach to solving stochastic linear programs [12]. This idea was extended to stochastic quadratic programs by Kulkarni and Shanbhag [11]. This method is essentially an extension of the L-shaped method with a trust region appended to the constraints of the master problem as follows.

$$\begin{aligned} \text{Master-TR } \min_{x,\theta} \quad & \tfrac{1}{2} x^T Q x + c^T x + \theta \\ \text{s/t} \quad & Ax = b \\ & \theta \geq G_i^T x + g_i \quad i = 1, ..., m \\ & -\Delta_{n,l} e \leq x - x_n \leq \Delta_{n,l} e \\ & x, \theta \geq 0. \end{aligned}$$

where (G_i, g_i) is the set of cuts that approximate $\mathcal{Q}(x)$, n is the *major iterate* index, l is the *minor iterate* index and $\Delta_{n,l}$ is the box-shaped trust region around x_n.
The algorithm [11]:

- Choose starting point x_0, initial trust region $\Delta_{0,0}$, initial approximation (G_0, g_0)
- Solve Master-TR to get minor iterate $x_{n,l}$
- Solve all subproblems duals to optimality using $x_{n,l}$
- Add cut (G_n, g_n) to the family of optimality cuts
- Check if the *sufficient descent* condition [11] is satisfied.
  - If not, update model to get $m_{n,l+1}$. Update Δ_n
  - If yes, get new major iterate $x_{n+1} = x_{n,l}$. Update model m_{k+1}, and Δ_{k+1}
- Terminate if termination condition is satisfied

III. Grid-based Parallel Implementation

Computational grids are geographically-distributed heterogenous networked computing resources. For instance, the TeraGrid has more than 11,000 processors that it can access with a total computational power of more than 45 Teraflops while the National Center for Supercomputing Applications (NCSA) has a grid with 1774 processors. Grid-computing in the context of optimization problems has largely been restricted to Condor-based implementations, which we describe next. We follow a different path, leveraging the middleware research done in NCSA. This is described in section III-B.

A. Condor-based Implementations

Most of the attempts to use grid-based methods to solve large-scale optimization problems have used the Condor system. In this system, a set of collections of processors are managed. These processors may be nodes of a supercomputer or could merely be accessible PCs. Condor processes communicate through PVM [7] and a set of shared files. The implementation of a Master-Worker paradigm on computational grids is achieved through a set of runtime libraries.

There has been significant effort by several researchers to extend optimization algorithms to the grid-computing framework. We discuss two successful implementations using the master-worker framework (MW) [9], [8] in brief:

1) **Stochastic Linear Programming:** Linderoth et al. [12] extend a decomposition method[15] for stochastic linear programming to the grid-computing framework. An *asyncrhronous* variant of the algorithm is developed and implemented on the Condor system. The authors aim to solve a problem in which the stochasticity is given by 10,000,000 scenarios. If the problem were to be solved as a large-scale linear program, the resulting size of the matrix would be of the order of 10^{22}. By decomposing the problem over 1,345 processors, the total CPU time was 1.03 years while the real time (after parallelizing) was approximately 32 hours.
2) **Quadratic Assignment Problems:** The quadratic assignment problem is a optimization problem with discrete variables. In [1], the authors set out to solve NUG30, an instance of the problem that would possibly take between 5-10 years on a fast workstation. Through the use of grid-computing methods, the problem was solved in less than 7 days by using approximately 655 machines.

B. Middleware at NCSA

A general method for running distributed processes would be to construct scripts that are run on each of the available nodes. By sending across input data in a file-based form, the scripts can be executed. The jobs are then monitored and the output is moved back to the originating processor when the jobs have completed. Such an approach tends to place a significant responsibility on the user and relies significantly on robust scripts which are often not reusable. Our approach relies on the following middleware:

- VIZIER: This service allows one to ensure that the configuration for every node is automatically controlled every time it is used. Furthermore configurational changes may be restored and reused.
- TROLL: This service launches jobs and is also responsible for monitoring the status of every job. It also determines the requirements of each user.
- ELF: The ELF utility allows for every application to be wrapped in an XML wrapper. This allows for flexibility in the type of application being executed and would render a Matlab program indistinguishable from a C executable. Such a wrapper would also manage input, output, execution and error messages.
- SIEGE: The user remains protected from the complexity of the inner workings of the system by working entirely through the SIEGE utility. The service-oriented architecture that has been used in the system has the desktop client, Siege, which interacts with the Troll ensemble broker stack and Vizier information services, to deploy applications controlled by the remote application container, ELF, which implements our own scripting language, OgreScript. The system is tied together with the notification systems, currently using the Java Messaging Service (JMS) [JMS] channel ActiveMQ [ActiveMQ], and a metadata system.

Such a system is modular, generalizable and can be extended easily to work with systems like Condor. It has a more robust methodology for notifying the use of errors and has a comprehensive (SIEGE) interface for usage.

TABLE I
COMPUTATIONAL EFFORT BY SIZE OF DISTRIBUTION (n = NUMBER OF PROBLEMS PER WORKER)

$\|\Omega\|$	$n=1$	$n=10$	$n=25$	$n=50$	$n=100$	$n=\|\Omega\|$
400	31	135	307	595	1170	4062
625	40	166	376	726	1426	7087
900	49	188	420.5	808	1583	11714
1000	60	267	612	1187	2337	14046
1225	59	226	503	966	1891	18067
1331	73	262	577	1102	2152	22485
1681	973	1286	1807	2676	4413	32800
2187	190	730	1630	3130	6130	99631
6561	595	1954	4219	7994	15544	834482
15625	1617	7416	17081	33190	65406	3912376
16384	1735	4255	8455	15455	29455	2580980
19683	1831	6448	14143	26968	52618	4128329
32768	2667	7752	16227	30352	58602	8581700
46656	2570	9270	20438	39050	76275	18686359

IV. COMPUTATIONAL EXPERIENCE

In table I, we provide some preliminary computational results for a large-scale stochastic quadratic program [5] in which the number of scenarios is varied from 400 to close to 50,000. The current set of results are obtained by using the exact-cut L-shaped method [15] on a single processor machine (Intel Xeon CPU, 3.40GHz 64-bit processor, 2GB of RAM) using KNITRO (http://www.ziena.com/ [4]) as the quadratic programming solver. The first column of table 1 shows the number of scenarios in the second-stage distribution while n specifies the number of scenarios allocated to each worker. In effect, $n = 1$ implies that there are assumed to be as many workers as there are scenarios (the best-case situation) while $n = |\Omega|$ implies that all the scenarios are computed on a single worker. The interspersed columns show similar results for higher levels of granularity. Since, the current implementation is on a single node, we have provided an indication of how parallelization would impact the computational effort. An important characteristic of the behavior of the algorithm is that the effort grows linearly with size (and not faster).

V. SUMMARY AND FUTURE RESEARCH

This paper considers the solution of a two-period stochastic programming problem. Direct solutions of such problems is impossible since the sizes run well into the hundreds of thousands in variables and constraints. Instead, we describe two decomposition methods of which a variant of one has been applied and tested on a serial platform. Our next step would be to extend the implementation to the TeraGrid,

focusing specifically on making algorithmic enhancements to allow for asynchronous worker computations.

VI. Acknowledgements

The authors would like to thank Richard Waltz and Todd Plantenga of Ziena Optimization Inc. for providing us with the nonlinear programming solver KNITRO.

References

[1] K. Anstreicher, N. Brixius, J.-P. Goux, and J. Linderoth, *Solving large quadratic assignment problems on computational grids*, Math. Program., 91 (2002), pp. 563–588. ISMP 2000, Part 1 (Atlanta, GA).

[2] E. M. L. Beale, *On minimizing a convex function subject to linear inequalities*, J. Roy. Statist. Soc. Ser. B., 17 (1955), pp. 173–184; discussion, 194–203. (Symposium on linear programming.).

[3] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming: Springer Series in Operations Research*, Springer, 1997.

[4] R. H. Byrd, J. Nocedal, and R. A. Waltz, *KNITRO: An integrated package for nonlinear optimization*, in Large-scale nonlinear optimization, vol. 83 of Nonconvex Optim. Appl., Springer, New York, 2006, pp. 35–59.

[5] X. Chen and R. S. Womersley, *Random test problems and parallel methods for quadratic programs and quadratic stochastic programs*, Optim. Methods Softw., 13 (2000), pp. 275–306.

[6] G. B. Dantzig, *Linear programming under uncertainty*, Management Sci., 1 (1955), pp. 197–206.

[7] A. Geist, A. Beguelin, J. Dongarra, R. Manchek, and V. Sunderam, *PVM:Parallel Virtual Machine*, The MIT Press, Cambridge, MA, 1994.

[8] J. Goux, J. Linderoth, and M. Yoder, *Metacomputing and the master-worker paradigm*, Tech. Rep. Preprint MCS/ANL-P792-0200, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., February 2000.

[9] J.-P. Goux, S. Kulkarni, J. Linderoth, and M. Yoder, *Master-worker : An enabling framework for master-worker applications on the computational grid*, Cluster Computing, 4 (2001), pp. 63–70.

[10] G. Infanger, *Planning Under Uncertainty*, Boyd and Fraser Publishing Co., 1994.

[11] A. Kulkarni and U. Shanbhag, *Decomposition methods for convex programming under uncertainty*, To be submitted.

[12] J. Linderoth and S. Wright, *Decomposition algorithms for stochastic programming on a computational grid*, Comput. Optim. Appl., 24 (2003), pp. 207–250. Stochastic programming.

[13] U. V. Shanbhag, *Decomposition and Sampling Methods for Stochastic Equilibrium Problems*, PhD thesis, Department of Management Science and Engineering (Operations Research), Stanford University, 2006.

[14] U. V. Shanbhag, G. Infanger, and P. W. Glynn, *An interior sampling method for stochastic quadratic programs*, To be submitted, (2006).

[15] R. M. Van Slyke and R. Wets, *L-shaped linear programs with applications to optimal control and stochastic programming*, SIAM J. Appl. Math., 17 (1969), pp. 638–663.

[16] G. Zakeri, A. B. Philpott, and D. M. Ryan, *Inexact cuts in Benders decomposition*, SIAM J. Optim., 10 (2000), pp. 643–657.