Parallel Processing Approaches for Multi Disciplinary Optimization Algorithms

Florin B. Manolache
Dept. of Mathematical Sciences,
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Sorin Costiner
United Technologies Co.
UTRC, 411 Silver Ln. East Hartford,
CT 06108, USA

Abstract

Multi Disciplinary Optimization (MDO) problems are often encountered in many industrial areas. MDO refers to the optimization of systems of subsystems (disciplines). The optimization of the full system is often reduced to a hierarchy of optimizations at subsystem and system levels. This paper presents a concise survey of major MDO approaches and discusses opportunities of parallel processing (PP) at four algorithmic levels of the approaches, i.e., the subsystem solver level, the subsystem optimization level, the full system optimization level, and the sequence of problems level. Advantages of different PP implementations of the MDO approaches are outlined. Special emphasis is put on vertical PP processing, where one thread treats a hierarchical structure (e.g., a full system evaluation), inter-thread communication is low, and processor loads are uniform.

1 Introduction

With the increase of computer power, multi disciplinary optimization (MDO), that usually refers to the optimization of systems of subsystems, became largely used in industry, ([1], [2], [27]). Common industrial examples of subsystems in a large system are: departments in a company; different disciplines in an analysis code, e.g., flow and structures; parts or components of a CAD assembly; physical subsystems of a complex system; subsystems of equations describing different models in a complex model etc. The systems and subsystems are usually simulated by analysis codes. The optimization of such a system may be a time consuming task. Classical optimization approaches are often replaced by more efficient hierarchies of optimizations that couple optimizations of individual subsystems into a full system optimization. Parallel processing (PP) can be used at analysis level or at different optimization levels ([3]).

Without restricting the generality of the presentation, it will be assumed that the systems and subsystems are modeled by systems of equations, that usually are PDEs, algebraic, or integral equations. Such systems will be denoted further by

$$f(p, x, y, z) = 0 \tag{1}$$

where f may represent the physics of the system and may be simulated or solved by specialized analysis codes that could be of black box type. The variables x and y are the optimization parameters or design variables. The vector x denotes system level parameters or shared parameters. e.g., that are common to two or more subsystems. The vector y contains subsystem parameters or local parameters, i.e., $y = (y_1, ..., y_n)$ where each vector y_i contains the optimization parameters that belong only to the subsystem S_i. The vector p defines a parametric class of problems, i.e., for each p value there is one system of type $f_p(x, y, z) = 0$, and an optimization problem has to be solved with an objective $Q_p(x, y, z)$. The vector p may define a sequence of optimization problems, e.g., for trade off studies, architecture optimization, or p may define operating regimes of the full system, i.e., are multi-point type parameters. They are used by all subsystems but are not optimization parameters. The vector z represents state variables computed by solving equation (1) for given p, x, y, hence z can be regarded as a function of $p,\ x,\ y$. A part of the z variables have a special role, they are the subsystem inputs and outputs. These parameters are called in different ways, such as input/output variables, coupling variables, communication variables, or subsystem interactions. The couplings do not allow individual subsystems to be optimized independently for a full system optimization, and may be involved in serious efficiency challenges for PP implementations. The parameters may be continuous or discrete, and are constrained by bounds:

$$p_L \le p \le p_U,\ x_L \le x \le x_U,\ y_L \le y \le y_U,\ z_L \le z \le z_U \tag{2}$$

The optimization of a given system requires minimizing an objective Q:

$$\min Q(p, x, y, z) \tag{3}$$

The optimization in (3) is performed over a set of parameters $x,\ y$ satisfying the constraints (1), (2) and

$$h(p, x, y, z) \le 0 \tag{4}$$

The functions h in (4) often impose constraints on the outputs, (e.g., the outputs should be physically realistic). An array (p, x, y, z) is called feasible if it satisfies (1,2,4), i.e., it is a solution of the system (1) and satisfies constraints (2,4).

To distinguish between variables or functions associated to different subsystems S_i , they will get indices i such as $f_i(p_i, x_i, y_i, z_i),\ \ Q_i,\ \ h_i$.

The full MDO problem to be solved is (1-4), where f incorporates all systems of equations modeling the subsystems, for all variables and all constraints (4).

The functions f, Q and h can be nonlinear, hence the problem (1-4) may involve nonlinear optimization approaches. The constrained optimization problem (1-4) may present local minima, hence a global optimization for finding a global minimum or for finding multiple good local minima is required.

Multi-objective versions of the problem (1-4) can be considered as well, in which case Q in (3) denotes a vector of objectives to be optimized ([7]).

Classical optimization approaches for solving problems of form (1-4), use a single optimizer that calls a full system solver for each fixed set of parameters (p, x, y) ([5]). These "All-at-Once" approaches, are often inefficient due to the prohibitively long computing time that may be required by full system solvers. This motivated the development of the more efficient MDO techniques that decompose the system, and call individual subsystem solvers. MDO techniques may handle very large hierarchical systems that could not be treated by All-at-Once approaches. MDO techniques may benefit a lot from parallel processing (PP) that can be used in multiple ways in particular MDO implementations as discussed further (see also [12], [22]).

2 Parallel Algorithm Design Considerations

This section discusses conditions to be met by PP architectures and algorithms to attain increased efficiency. These conditions are further used to choose some MDO PP implementations against others.

Two paradigms were imposed by the hardware architecture in the parallel processing world: (a) the SMP computer has a number of processors with uniform access speed to the entire memory, the communication between threads being done through shared memory and the synchronization through semaphores; (b) the network of computers that is composed of processors which have a much faster access to a certain local memory segment, the communication and the synchronization between threads being done through messaging.

Despite fundamental differences in the way the two paradigms work, the conditions to have an efficient parallel algorithm boil down to attaining:

- minimal communication between threads - realized through a good partitioning of the algorithm;
- efficient use of the processor time - realized by a good allocation of the computing time on different processors and good synchronization policies between threads.

It is well known, ([15]), that the parallel algorithm performance is highly dependent on the configuration of the hardware it runs on. However, the practice shows that an algorithm cannot be fine tuned to become efficient unless the above requirements are fulfilled.

Complex industrial design tasks, which usually imply optimization, can be parallelized and practically solved only if they are split in weakly coupled subtasks layered on several levels. Imitating that structure, the algorithm to solve the task can be programmed in the same level structured way in a parallel implementation. This means that both the parallel processing structure and the algorithm structure are reflections of the task structure. The main PP virtue of the MDO approach is that it offers clear indication about how to do the partitioning of the algorithm such that the two conditions are satisfied, i.e., communication between threads is minimized and the use of processor time is optimized.

3 Hierarchical Parallel Processing in MDO Approaches

This section describes a hierarchy of four PP levels for the problem (1-4). The levels are:

1) the subsystem analysis level (or solver level) where systems $f_i = 0$ are solved for each subsystem S_i;

2) the subsystem optimization level where objectives Q_i are optimized for parameters y_i;

3) the system optimization level, where the objective Q is optimized for parameters x; and

4) the sequence of optimizations level, e.g., at a multi-point optimization level, where a set of optimizations is performed for different values of p.

Traditional PP implementations refer to the first two levels. Levels 3 and 4 refer particularly to MDO problems and usually involve a hierarchy of optimizations. In principle, one may use PP at all four levels, but, contrary to common practice, this may not be the most efficient most robust, generic, and easy to code approach. For example, using PP at level 1 in global optimization MDO problems may not be efficient even when a very large number of processors is available. This paper discusses the four level PP for major MDO approaches and emphasizes the efficiency of using PP at levels 3 and 4 instead of using PP at levels 1 and 2.

In [11], PP continuous optimization issues are discussed, including MDO optimization. The MDO discussed there uses a single optimizer for the whole system, although multiple analysis subsystems are considered. We refer to that paper for details of treating subsystem interactions. In this paper, multiple optimizers are considered, e.g., at subsystem, system, and sequence of optimizations (multi-point) levels.

The description of the four PP levels follows.

3.1 PP at the Subsystem Analysis Level

On the subsystem analysis level, the systems of equations $f_i(p, x, y_i, z_i) = 0$ of the different subsystems S_i are solved for z_i using PP. The p, x, y_i are fixed. Most techniques discussed in literature may be used here, e.g., parallel algorithms for systems of algebraic equations, domain decomposition techniques, etc. The use of PP at this level is usually inefficient due to the large inter-thread communication overhead. An algorithm using PP at this level may not be robust. If one of the threads is interrupted, the entire computation may have to be reconsidered due to the strong inter-dependence between threads. All available processors may be used much more efficient at higher levels.

3.2 PP at the Subsystem Optimization Level

On the subsystem optimization level, the subsystems S_i are optimized for the objectives $Q_i(p, x, y_i, z_i)$ for the local parameters y_i (keeping x and p fixed). The many useful PP techniques and discussions from [11] can be applied here. A subsystem optimization may be local or global, it may be based on derivatives , e.g., quasi Newton approaches, or on derivative free approaches, e.g., pattern searches, ([11], [22]). The gradients are usually approximated by finite differences that may be computed in parallel and do not need inter-thread interaction. Line searches and pattern searches may be performed in parallel too.

Global optimization of subsystems may be deterministic or stochastic, ([16], [22], [29]). Most of these approaches are easy to parallelize at very low inter-thread communication. E.g., deterministic grid searches can be performed in parallel, and stochastic searches based on generation of random samples over the domain (e.g., genetic algorithms, simulated annealing, cluster techniques) may be performed in parallel too, for any number of processors. Branch and bound type techniques can also be efficiently applied in parallel, ([4]).

3.3 PP at the System Optimization Level

On the system optimization level, the full system is optimized for the objective $Q(p, x, y, z)$ and parameters x, by calling subsystem optimizations. The p are fixed. The PP techniques for optimizing the system are the same as for the subsystems. The parameters x usually are common to several subsystems. In a generic MDO, the system optimizer sends the x to subsystems that get optimized by local parameters (y_i for subsystem S_i) keeping the system parameters x fixed. The results of the subsystem optimizations are used further for system optimization. During the system optimization, the subsystems may be analyzed or optimized in parallel. If a global optimization is performed at system level, it may be inefficient to optimize different subsystems by different processors since the optimization times may differ strongly. Instead, one may perform full system evaluations by different processors for different parameter values x. In this way, the processors get loaded uniformly.

3.4 PP at the Sequence of Optimizations Level

On the sequence of optimizations level, e.g., at a multi-point level, optimizations are performed calling system optimizations for different p values. For multi-point optimization, ideally, a system should be optimal for each value of the point (operation/regime) parameter p. This is usually not possible, so one has to define a compound objective to be optimized, using different p values, and the resulting problem is again a global optimization problem. This optimization should call multiple full system optimizations, hence it can substantially use PP in both local and global optimizations. The number of optimization levels can be increased, each level calling optimizations from the lower levels. On each level, the same optimization techniques as on system level are used.

4 PP for Several Generic Classes of MDO Approaches

MDO approaches can be classified in many ways, ([1], [2], [3], [6], [8], [27]). This section mentions generic classes of MDO techniques and discusses PP implementations for them. For all classes, PP can be applied at the four mentioned levels, as described in the previous section. In addition, different classes have specific features that allow particular PP for higher efficiency. A given MDO implementation may belong to several classes and may combine different PP techniques.

Sequential Optimization, Simultaneous Optimization, and Hierarchical Optimization are generic and the most often used MDO approaches. In a sequential optimization of a system, the different subsystems are optimized one by one, each optimized subsystem sending its outputs to the other subsystems. In a simultaneous optimization all subsystems are optimized in parallel and the outputs are sent all at a time. Hierarchical optimization approaches couple subsystem optimizations in a hierarchical level structure, optimizations on higher levels being based on results of optimization on lover levels. The system optimization described in previous section is an example of hierarchical optimization. Sequential and simultaneous approaches are efficient especially for subsystems with weak interactions, i.e., changes in one subsystem do not influence strongly another subsystem. They may be less efficient when subsystems are strongly coupled. Hierarchical optimization can be used to address subsystem interactions at system level in a robust way. Sequential, simultaneous and hierarchical approaches are often combined together. PP can be directly applied for simultaneous optimization, e.g., each processor optimizing a different subsystem. Sequential and simultaneous optimizations can be easily combined, for example, in a sequence of simultaneous optimizations, several groups of subsystems can be processed sequentially while all subsystems in a group are processed simultaneously. PP may be used in many ways in a hierarchical optimization, in addition to the different possible PP treatments of simultaneous or sequential optimization on different levels. Generally speaking, PP in MDO approaches may involve two main types of parallel computations:
1) horizontal threads or horizontal parallel computations performed on one of the levels, and
2) vertical threads or vertical parallel computations, where a hierarchical computation involving more levels is assigned to one thread. Sequential and simul-

taneous computations on a single level would require horizontal threads. The horizontal PP usually requires much more communication than vertical PP. One can perform simultaneously a set of vertical threads with little communication.

Combinations of Global and Local Optimization MDO approaches perform global or local optimizations at different levels of the MDO. While many optimization approaches combine global optimization algorithms with local optimization algorithms for one given problem, (see for example [5], [14], [16], [19], [24], [25]), in MDO different optimization techniques may be applied for different problems corresponding to different subsystems. PP can be efficiently used in the often encountered case when global optimization is performed at system level while local optimization is performed at subsystem level. Each thread could evaluate the full system for different values of x (the same for p in multi-point formulations).

Multi-Objective MDO approaches are MDO approaches where the system (or some of the subsystems) have multiple objectives $q_j,\ j = 1, ..., m$, ([7]). All q_j are considered here objectives of the same subsystem S_i, for a fixed i (i.e., q_j are the components of the vector objective Q_i), as opposed to the notation Q_i that denotes objectives of different subsystem S_i. The objectives q_j can be lumped together in one compound objective to reduce the problem to a classical single objective optimization, e.g., by a weighted sum of objectives by factors w_j:

$$q = \sum_j (w_j\ q_j) \tag{5}$$

Different solutions (x, y, z) can be obtained for different weight values w_j. PP can be used by allocating to each processor an optimization problem with a different weight value. In goal programming, only one of the objectives q_k is optimized while the other ones are used as constraints, e.g., set to fixed goal values $q_j = c_j$ or $q_j \leq c_j$. PP can be used efficiently choosing different goal sets $\{c_j\}$ for different processors. For multi-objective optimization, an optimal solution $u = (p, x, y, z)$ is a point such that in a neighborhood of u there is no other point v which is better than u with respect to all objectives, i.e., $q_j(v) < q_j(u)$ cannot hold for all objectives q_j. This means, that any other v is worse than u with respect to at least one objective. In the space of objectives $(q_j(u))_{j=1,...,m}$, the optimal solutions lie on a boundary P, called Pareto frontier, of the set $S = \{(q_j(u))_{j=1,...,m} \| u\ in\ U\}$ where U is the full parameter set. People usually wish to investigate the Pareto frontier P. PP is a good option for this task, since multiple points on the frontier can be found in parallel by different processors, one often used technique being the mentioned optimization of the weighted sum of objectives that produces one point on the frontier for each fixed set of weights.

Surrogate Models MDO approaches combine optimizations of original high complexity models f with approximations (surrogates) f' of f, where f' can be computed in shorter time than f ([9], [10], [17], [18], [26]). For example, the full system is often represented by surrogates, such as response surfaces, reduced order models, linearizations, coarse models, while subsystems may be represented by high fidelity models. Global optimization may be performed using low fidelity models, while local optimization may be used for high fidelity models to improve the solutions found by low fidelity model optimizations. PP can be used very efficiently for many parallel evaluations of the surrogates, e.g., in a hierarchical grid search, on a sequence of coarser and finer grids. PP can also be used to generate many surrogates simultaneously, e.g., to generate local response surfaces in different design regions. The interaction between such computations is minimal.

Sensitivity based MDO approaches use sensitivities in the design of MDO optimizers, (e.g., how variables influence objectives, constraints, states, and outputs, [23], [28]). Sensitivity based approaches may be very efficient, may incorporate reduced models based on sensitivities, and may use the same objective or compatible objectives for all subsystems (which eliminate the conflicts of different subsystem optimizations). For example, in the BLISS approach, ([28]), sensitivities of the system objective Q with respect to the subsystem parameters y are computed, DQ/Dy, and used to define linearized subsystem objectives that are used to optimize the subsystems. Then DQ/Dx are computed and used to linearize the system objective for a system optimization. The special input/output coupling variables, that are part of z, and their derivatives with respect to x, y, Dz/Dy, Dz/Dx, are used for the sensitivity computations applying the chain rule. For this type of approaches, PP can be used efficiently for the sensitivity approximations by finite differences.

Discrepancy Minimization MDO approaches, in a simplified explanation, reduce equality constraints to minimization formulations, e.g., an equation constraint as $a = b$ could be replaced by a residual minimization $min(a-b)^2$. Collaborative optimization, is an MDO approach from this class, ([21], [26]) that eliminates direct subsystem interactions. During optimization, the system sends the system variables x to a subsystem S_i. The S_i, in addition to its local optimization parameters, y_i, gets optimization parameters x_i that are local versions of the system parameters x. The optimization of S_i has an objective to minimize the discrepancy between x and x_i, e.g., to minimize $Q_i(x_i, y_i) = (x - x_i)^2$ with constraints given by f_i and h_i. This may be interpreted as: the system sets a task x that has to be satisfied as well as possible by the subsystem S_i. The system optimization minimizes $Q(x)$ with additional constraints $Q_i = 0$ or $Q_i < \epsilon$. Subsystem interactions may be treated by minimization as well, e.g., instead of setting $z_i = z_j$ meaning that the output from S_i equals the input from S_j, one may define an optimization $min(z_i - z_j)^2$. Particular for this class of approaches, PP can be used to optimize the subsystem interactions. In actual PP implementations, each processor should solve the problem associated with one subsystem, or a group of subsystems. The relation and the communication between threads should follow the hierarchy of the particular MDO algorithm.

5 Conclusions and Discussion

Four MDO computational levels have been separated, that correspond to: the subsystem analysis, the subsystem optimization, the system optimization, and the multi-point optimization. PP options that are applicable at different computational levels have been described, and their efficiency has been discussed for major classes of MDO approaches. MDO codes for complex industrial optimization tasks can be efficiently implemented using PP as long as the communication between threads is kept at low levels and the processor time is effectively used. To achieve this, MDO approaches should decompose the full system into weakly interacting subsystems. PP algorithms can be efficiently implemented for such MDO structures since the communication between threads is low for weakly coupled subsystems if every thread is solving a different subsystem. This is a system level (level 3) PP implementation where each thread should handle a hierarchy of optimizations and analysis tasks. Such a hierarchical PP should be more efficient than a classical single level PP approach that is heavily applied to the analysis codes or to a subsystem optimization (on levels 1 and 2). Single level approaches often require important communication between threads. They are often hard and unnatural to program because they artificially split subsystems. By contrast, the hierarchical approaches follow the natural, physical structure of the system, avoiding most of the above problems.

A crucial issue is the robustness of PP algorithms with respect to thread failures. A hierarchical PP approach can be made robust much easier than the single level PP approaches. While in the hierarchical approach a thread failure may be easily rescued by restarting the thread, in a single level approach it may be hard to implement a rescue, and one may need to rerun the whole task.

Summarizing the above discussion, MDO PP approaches involve two main types of PP computations: horizontally, on each level, (as in levels 1 and 2) and vertically where a hierarchical computation involving more levels is assigned to one thread (as in levels 3 and 4). The horizontal PP usually requires much more inter-thread communication and unbalanced processor loads than the vertical PP. Heavy and complicated communication takes place at analysis level or at subsystem interaction level in horizontal PP. On the other side, different vertical threads in a PP approach usually have little communication between them. For example, in global optimization, each vertical component (thread) may evaluate the whole system for another set of parameters. Such vertical threads have practically no communication between them. The same argument holds for large classes of MDO applications where the described vertical PP MDO implementations may be efficient. Such applications include: multi-point optimization, multi-objective optimization, trade off studies, feasibility studies, sensitivity studies, designs of experiments, uncertainty analysis, architecture optimization, safety studies, and robust design.

References

[1] N.M. Alexandrov, R.M. Lewis, Comparative Properties of Collaborative Optimization and Other Approaches to MDO, ASMO UK/ISSMO Conf. On Eng. Design Optimization, MCB Press, 1999.

[2] N.M. Alexandrov, M.Y. Hussaini, eds., Multidisciplinary Design Optimization: State of the Art, Philadelphia, SIAM 1997.

[3] R.J. Balling, J. Sobieszczanski, Optimization of Coupled Systems: A Critical Overview of Approaches, AIAA J., Vol. 34, No. 1, pp6-17, Jan. 1996.

[4] J. Clausen, Parallel Branch and Bound - Principles and Personal Experiences, in A. Migdalas, P.M. Pardalos, S. Storoy, Parallel Computing in Optimization, Kluwer, 1997.

[5] COCONUT Project, Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of the Art, 2001; http://www.mat.univie.ac.at/ neum/glopt/coconut/

[6] E.J. Cramer, J.E. Dennis, P.D. Frank, R.M. Lewis, G.R. Shubin, Problem Formulation for Multidisciplinary Optimization, Rice University Report CRPC-TR93334, 1993, presented at the AIAA Symposium on Multidisciplinary Design Optimization, 1993.

[7] I. Das, J.E. Dennis, Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Multicriteria Optimization Problems, SIAM. J. OPT, Vol. 8. No. 3. pp. 631-657, 1998.

[8] J.E. Dennis, R.M. Lewis, problem Formulations and Other Optimization Issues in Multidisciplinary Optimization, Rice University Report CRPC-TR94469, 1994.

[9] J. Dennis, A. Booker, P. Frank, D. Serafini, V. Torczon, Optimization Using Surrogate Objectives on a Helicopter Test Example, in Optimal Design edited by J. Burns and E. Cliff, SIAM, Philadelphia 1998.

[10] J.E. Dennis, A. Booker, P. Frank, D. Serafini, V. Torczon, M. Trosset, A Rigorous Framework for Optimization of Expensive Functions by Surrogates, Structural Optimization 17(1), pp. 1-13, 1999.

[11] J.E.Dennis, Z.Wu, Parallel Continuous Optimization, Available as Rice CAAM-TR00-01, (Submitted for Publication), 2001.

[12] J. Eddy, K.A. Hacker, K.E. Lewis, Solving Computationally Expensive Optimization Problems using Hybrid Methods in Parallel Computing Environments, Aoo-40114, AIAA-2000-4864, Presented at the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach CA, 2000.

[13] M.S. Eldred, W.E. Hart, B.D. Schimel, B.G. van Bloemen Waanders, Multilevel Parallelism for Optimization on MP Computers: Theory and Experiment, A00-40079, AIAA-2000-4818, Presented at the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach CA, 2000.

[14] C.A. Floudas, Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications, Oxford University Press, 1995.

[15] I. Foster, Designing and Building Parallel Programs, ACM Press, 1995.

[16] R. Horst, P.M. Pardalos, Handbook of Global Optimization, Kluwer, 1995.

[17] M.G. Hutchison, W.H. Mason, B. Grossman, R.T. Haftka, Aerodynamic optimization of an HSCT configuration using variable-complexity modeling, AIAA PAPER 93-0101, 1993.

[18] D.R. Jones, M. Schonlau, W.J. Welch, Efficient Global Optimization of Expensive Black-Box Functions, J of Global Optimization 13; 455-492, 1998.

[19] A.G.H.R. Kan, G.T. Timmer, Global Optimization, in G.L. Nemhauser, A.G.H. Rinnooy Kan, M.J. Todd, Optimization, North-Holland 1989.

[20] P.N. Koch, B. Wujek, O. Golovidov, A Multi-Stage, Parallel Implementation of Probabilistic Design Optimization in an MDO Framework, A00-40069, AIAA-2000-4805, Presented at the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach CA, 2000.

[21] I. Kroo, V. Manning, Collaborative Optimization: Status and Directions, IAA 2000-4721, 6-8 Sept, 2000.

[22] A. Migdalas, P.M. Pardalos, S. Storoy, Parallel Computing in Optimization, Kluwer, 1997.

[23] J.R. Muñoz, M.R. von Spakovsky, The Use of Decomposition for the Large Scale Synthesis/Design Optimization of Highly Coupled, Highly Dynamic Energy Systems Part I - Theory, International Journal of Applied Thermodynamics , Vol. 3. 2001.

[24] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, John Wiley and Sons, 1988.

[25] G.L. Nemhauser, A.G.H. Rinnooy Kan, M.J. Todd, Optimization, North-Holland 1989.

[26] I.P. Sobieski, I.M. Kroo, Collaborative Optimization using Response Surface Estimation, IAA-98-0915, 1998.

[27] J. Sobieszczanski-Sobieski, R.T. Haftka, Multidisciplinary Aerospace Design Optimization:, Survey of Recent Developments, AIAA 96-0711, 1996.

[28] J. Sobieszczanski-Sobieski, M.S. Emiley, J.S. Agte, R.R.Jr. Sandusky, Bi-Level Integrated System Synthesis (BLISS), NASA/TM-1998-208715 1998.

[29] H. Vladimirou, S.A. Zenios, Parallel Algorithms for Large-Scale Stochastic Programming, in A. Migdalas, P.M. Pardalos, S. Storoy, Parallel Computing in Optimization, Kluwer, 1997.