SANDIA REPORT
SAND97–2052 • UC–705
Unlimited Release
Printed August 1997

Analysis Driven Mechanical Redesign

Arlo L. Ames, Richard H. Robison

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SF2900Q(8-81)

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND 97-2052
Unlimited Release
Printed August 1997

Distribution
Category UC-705

Analysis Driven Mechanical Redesign

Arlo L. Ames
Richard H. Robison

Advanced Engineering and Manufacturing
Software Development
Intelligent Systems and Robotics Center
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-1010

Abstract

This paper documents an effort to use a constrained nonlinear optimization package (OptdesX) to drive a feature-based mechanical design system (Pro/Engineer) in an optimization loop. Optimizations performed in this manner can maximally respect the design intent built into the model, and eliminate the need to propagate optimization results back to design, a flaw of most current optimization systems. A prototype system was built to demonstrate the capability; use of the prototype uncovered a variety of issues that should be addressed to productionize this kind of capability.

Acknowledgements

This report documents work performed by Richard Robison, while he was an employee at Sandia National Laboratories. This report was principally written by Arlo Ames after Richard's departure from the Labs when he was unavailable to provide input.

We would like to thank Alan Parkinson of Brigham Young University for providing OptdesX and research in large scale optimization techniques.

Thanks to Robert Lafarge, Jill Rivera, and David Hickerson for reviewing the paper and Joella Archibeque for editing and preparation help.

Table of Contents

Introduction 1
Problem Statement 1
Definition of Domains 1
  Optimization 1
  Feature-Based Design 2
Connecting Pro/Engineer and OptdesX in an Optimization Loop 2
  Design Variables 2
  Design Variable Tolerance 3
  Constraint and Objective Functions 3
The Prototype System 3
  Interfacing Concerns 3
  Troublesome Models 3
    Unusable geometry 3
    Unchangable Features. 4
    Changed Model Topology 4
    Regeneration failures 4
  Objective and Constraint Function Evaluation 5
    Evaluations Within Pro/Develop 5
    Evaluations Within Pro/Engineer, But Outside of Pro/Develop 5
    Evaluations External to Pro/Engineer 6
  Dealing with Large Numbers of Design Variables 6
  Discrete Variables 6
  Specifying Design/Analysis/Manufacturing Constraints 7
Results 7
Impact 7
Future Directions 7
References 7
Distribution 9

Introduction

This report documents the results and recommendations of the Analysis-Driven Mechanical Redesign LDRD project.

The principal accomplishments of this project are: an understanding of how to integrate constrained nonlinear optimizers (such as OptdesX) with feature-based modelers (such as Pro/Engineer) in an optimization loop, a prototype system that proves the concept and provides a development testbed that will be useful for developing a fieldable system, and advanced work in developing the necessary tools for large-scale analyses. We will first document what we have learned about the requirements for the overall design process, then describe the prototype system, followed by a discussion of advanced development activities.

Problem Statement

The demand for rapid and accurate part redesign is increasing with the push to develop new processes that minimize design problems using computer modeling prior to manufacture. Current development projects, such as Sandia's FastCast [1], SmartWeld [2] and A-Primed [3], are implementing new design, analysis and manfacturing processes based on parametric solid modeling to produce high quality parts with reduced development times. Making correct decisions early in the conceptual design process ultimately reduces cycle times and production costs [4,5]. Although many product realization activities require varying degrees of design optimization, few projects devote the necessary resources to this activity due to high costs and long turn-around times required to achieve an optimal design solution. A few commercially available approaches for performing optimization exist, but such approaches are typically limited in domain, and they don't modify the design model directly, leading to difficulty in propagating design intent to the optimizer and propagating optimization results back to the design. The purpose of this project is to develop a capability for easily performing constrained nonlinear optimization of parametric designs that modifies the original part model using any computational tools available for performing analysis.

Definition of Domains

In this section, we describe the domains of optimization and feature-based design.

Optimization

Optimization is the process of seeking a best, or optimal solution to a design problem [6]. The fundamental requirement for optimization is the *objective function* $f(\bar{x})$, which is evaluated to determine the performance of the system being optimized. The objective function is a function of one or more *design variables*, x_i, which are varied to produce design changes. Design variables are real-valued, and may be either continuous (having any real value) or discrete (having real values from a specific collection, e.g. from the set of diameters of commercially available drill bits). In addition to the objective function, we define a set of *constraint functions*, $g_i(\bar{x})$, which constrain the space of feasible designs. A special form of constraint function is expressed as *upper and lower bounds* on individual design variables.

The constrained nonlinear optimization package chosen for this project is OptdesX [7]. The process of optimizing designs varies with the optimization algorithm, but involves modifying design variables, then evaluating objective and constraint functions to search for feasible, optimal solutions. Due to the hill-climbing nature of nonlinear optimization,

gradient evaluations are common. These can be performed by performing objective function evaluations directly for various values of design variables, or, less expensively, by evaluating explicitly written gradient functions, when they can be provided. OptdesX has a capability for producing "robust" designs by ensuring that the final result of optimization is sufficiently far from constraints to account for possible variation in design variables.

Feature-Based Design

Feature-based design systems, such as Pro/Engineer[8], model designs in terms of an ordered list of *features* (such as holes, slots, and pockets) which are evaluated to produce a boundary-representation solid model. The feature representation is a constructive representation; each feature adds to the description of the model by adding or removing material from the object.

Each feature has *defining geometry* and *dimensions*. Defining geometry establishes the shape of the feature; dimensions establish sizes and positions of the defining geometry relative to the part. Dimensions have *parameters* which establish the values of lengths and angles. Each parameter might have an associated *tolerance*, which establishes minimum and maximum variation. *Relations* define relationships between the values of parameters. The Pro/Engineer model of relations partitions parameters into sets of dependent and independent parameters. The values of dependent parameters are derived by evaluating relations (e.g. d1 = d2 * d3, where d1 is a dependent parameter). Independent parameters do not depend on any relations; their values are user-supplied.

A design may represent a *part* or an *assembly*. A part is the smallest unit of mass that is individually considered in mechanical design, and can be of arbitrary geometric complexity. Assemblies are collections of references to parts and other assemblies; each reference has an associated transformation for placing the referenced part or assembly within the coordinate frame of the containing assembly. An assembly may contain more than one instance of any given part. Assemblies describe groups of parts that work together to perform a mechanical function.

Pro/Engineer allows applications to be written using a programmer's interface called Pro/Develop [9].

Connecting Pro/Engineer and OptdesX in an Optimization Loop

Optimization packages such as OptdesX permit definition of optimization problems in terms of an objective function, design variables, and constraints. Our problem is to define mappings between objects in the design system (Pro/Engineer) and objects in the optimization package (OptdesX), and to explore the implications of working in this combined system.

Design Variables

Mapping Pro/Engineer design parameters to OptdesX design variables is a straightforward process. The Pro/Engineer features must be searched to find the complete set of design parameters. The set of relations for the design is then searched to determine which of the design variables are independent and which are dependent. The set of independent design parameters constitutes the set of potential design parameters for optimization.

Design Variable Tolerance

In order to compute robust designs, OptdesX requires information about the variation expected in each design variable. Pro/Engineer provides for tolerances to be specified for each design variable. These design tolerances, if specified, can be simply transcribed to OptdesX for computation of robust design.

Constraint and Objective Functions

Functions for evaluating constraints and objective function are not explicitly present in the Pro/Engineer framework. Traditionally, in the domain of optimization, these functions are hand crafted for each optimization activity being undertaken. To this traditional approach, the design system adds solid model representations of objects, and functions for performing geometric evaluations of those solid representations (e.g. mass properties, finite element analysis, etc.). Producing the geometric model for evaluation requires that Pro/Engineer regenerate the model for a given set of parameters.

The Prototype System

Given a basis for interaction between Pro/Engineer and OptdesX, a prototype system was developed in which the packages were proven to work together. A number of issues were raised in developing the prototype; they are discussed here.

Interfacing Concerns

Connecting the codes required establishing a mechanism for OptdesX to control Pro/Engineer as a part of the optimization loop. The programmer's interfaces to the packages treat application code as subservient; that is, each package expects to be started by a user, not by some other application. Since Pro/Engineer and OptdesX both expected to initiate the application code, it was necessary to develop a layer of indirection, a program that we will call HOOK. We start OptdesX, which in turn starts up HOOK, passing it a socket identifier to enable communication. HOOK starts Pro/Engineer through a fork call, passing it the socket identifier. Pro/Engineer starts the Pro/Develop application, passing it the socket identifier (and communicating with it through a different socket), and the actual OptdesX/Pro/Engineer interface starts. The Pro/Develop application services queries from our OptdesX application code, and executes the necessary Pro/Engineer functions.

The need for HOOK has been eliminated since the prototype was developed, since Pro/Develop included a function to permit an application to start Pro/Engineer (through the function pro_start_pro_engineer). Current programmer's interfaces to geometric modelers frequently omit such functionality, so such an approach might be necessary with other modelers.

Troublesome Models

It is possible, using the typical design tools in Pro/Engineer, to produce models that are difficult to use in design optimization.

Unusable geometry

There is great variety in the quality of geometry produced by solid modeling systems. Gaps and degeneracies that are allowed in one modeler can cause other solid modeling packages to crash. Pro/Engineer tends to produce geometry that is less precise and contains more degeneracies than many other packages can tolerate. These problem geometries are generally invisible to the user of the CAD system. The principal symptom of the problem is that downstream non-Pro/Engineer applications may appear to be brittle: they work well for some models and crash on others. Ongoing LDRD work in "Design

Simplification" [10] is producing algorithms that search geometry for problems areas and suggests approaches to eliminate problem geometries.

Unchangable Features.

It is possible for a Pro/Engineer model to contain geometry that cannot be changed. Unchangable features can be created by importing geometry from a non-Pro/E source (the imported geometry is a single feature with no parameters) and by built-in Pro/E functionality, such as creating a part that is a mirror of another part (the mirror has no parameters, and can only be modified by changing the part it was mirrored from). Models of this type can be detected by examination of feature definitions. Any faces that are created by features that have no parameters are likely uneditable. If features that are important to the optimization are uneditable, the model will have to be redefined in different, editable terms. Uneditable geometry can be detected automatically by searching the features that produced each face of the model to determine whether parameters are present to enable editing. Such an algorithm can be built based on techniques developed in [10].

Changed Model Topology

Changing parameters can cause features to interact differently. For example, a hole can be moved in such a way that a new intersection between it and another feature occurs. Taken to an extreme, the hole may completely cease to intersect the part. Such changes in topology can play havoc with downstream applications (e.g. hexahedral finite element mesh generators that rely on certain types of topology to succeed). Moreover, changes of this kind reflect a fundamental change in the design. Changes in model topology must be controlled.

Like much Pro/Engineer functionality, regeneration relies on user input to be verified. The user function Info:Geom Check can be used to check whether certain classes of topology problem occur. When running in an optimization loop, such checks are difficult to implement. To satisfy the need for constraining model topology, we developed software to compare solid models to locate topology changes.

Controlling model topology changes is perhaps easiest to implement by defining a constraint function that returns one value for unchanging topology, and another value if the topology changes. Such a two-valued step function might be difficult for many optimization algorithms to support, as typical constraint functions are continuous. This area requires further exploration.

Regeneration failures.

Certain models have proven to be somewhat brittle to design change. A feature-based model relies on the ability to resolve the set of geometric relationships between a feature's construction geometry and the remainder of the solid. Many geometric relationships (e.g. tangency), taken in combination with other relationships, can only be resolved over finite parameter ranges. Outside of these ranges, the modeler fails to regenerate.

As with topology changes, regeneration failures can be handled with a step-wise constraint function with the same caveats.

Regeneration failures cannot be prevented, but unlike topology changes, might be addressed by design standards. Certain construction schemes appear to be more prone to regeneration failure than others. This is another opportunity for further work.

Objective and Constraint Function Evaluation

Including the design system in the optimization loop provides the capability of writing objective and constraint functions in terms of functions that evaluate solid models. There are three categories of functions that evaluate solid models: functions accessible from within Pro/Develop, functions available within Pro/Engineer but not from Pro/Develop, and functions that are executed outside of the design system altogether.

Evaluations Within Pro/Develop

Evaluating functions within Pro/Develop is straightforward. Any function that can be called for evaluating the geometry (e.g. mass properties calculation, surface area, etc.) can simply be executed, and expected to return either an answer or an error code. Unfortunately, Pro/Develop provides direct access to relatively few evaluators.

Evaluations Within Pro/Engineer, But Outside of Pro/Develop

Executing functionality that is internal to Pro/Engineer, but outside of the scope of Pro/Develop functionality, is more difficult. An example constraint in this category would be that a part is NC machinable. Proof of machinability might require (among other things) that Pro/Manufacture be executed to produce an NC plan for the current combination of parameters.

In principle, executing any Pro/Engineer functionality from within a Pro/Develop application should be straightforward. Such is not the case. As stated above, a significant portion of Pro/Engineer functionality is not accessable from Pro/Develop. Thus, we need a different mechanism to access other categories of functionality.

It is possible to use pro_load_cmd_sequence to state that buttons in the menus should be pressed, but this access is rather limited. pro_load_cmd_sequence only permits 256 characters of type-ahead (the names of buttons are approximately 10 characters, so this presents a limit of 25 buttons), and the command sequence is not executed until the Pro/Develop application returns control to Pro/Engineer. The typeahead limitation can be circumvented by playing a trail file. Maintaining control through the Pro/Develop application requires that the trail file contain, as its last entry, button pushes to reinvoke the optimizer interface, and that the optimizer interface be able to continue execution where it left off before the button-press function was called (This is a very long go-to executed through two different languages and execution contexts).

Providing error handling in such a context is difficult. Pro/Engineer relies heavily on visual inspection to verify that actions are performed correctly, and executing through a trail.txt file provides no clear means of detecting processing errors. While it is possible to operate Pro/Engineer in this manner, it is much less desirable than direct functional access through a mechanism like Pro/Develop.

An additional difficulty is in interpreting the results of using such functionality. Given Pro/Engineer's GUI (Graphical User Interface) orientation, most Pro/Engineer functionality relies on a user to view a graphic picture of results to understand the result of performing an operation. Such an interface is difficult to use in an automated setting such as optimization. For some operations, writing a data file, which is evaluated by an external application, is an option.

Evaluations External to Pro/Engineer

Executing functions external to Pro/Engineer requires exporting geometry to whatever format the external program requires, initiating the external application, handling errors that might occur in the external application and extracting values to be used for evaluating constraint and objective functions. Exporting geometry involves translation, which may be difficult (especially for systems of limited accuracy and that commonly produce degeneracies). Our work in design simplification has uncovered a variety of difficulties with Pro/Engineer geometries. In working with external evaluations, it is crucial to select tools that produce exact, nondegenerate geometry, or be prepared to deal with a variety of translation difficulties.

There are a variety of tools for initiating external applications (e.g. CORBA, sockets), and they should be chosen to maximize ability to handle errors. Our experiments with simple fork calls were successful. External applications are simpler to execute than the within-Pro/Engineer-outside-Pro/Develop cases discussed above, as they can be directly executed without the long go-to. The potential for difficulties as experienced within Pro/Engineer exists, and varies with the application selected.

Dealing with Large Numbers of Design Variables

Mapping variables from real design space into an optimizer is not as simple as implied earlier. A typical design can have hundreds or thousands of design variables, most of which have nothing to do with the optimization at hand. Many of the dimension parameters required by a design system such as Pro/E are an artifact of the manner in which the design was constructed, and might be constant for an entire product or product line. In a part with holes, each hole feature may have its own parameter defining diameter, and all of those diameters might be the same, even if no explicit relations exist to constrain them. The part may have some fixed geometry that is required if the part is to mate with other parts in the assembly; that geometry might be defined in terms of many parameters, even though few of them will actually be varied.

Selecting a few design variables from the many can be a daunting task without significantly improved tools. Selection tools need to include the ability to select or deselect variables associated with a specific feature or group of features, part, or assembly, with selection occurring graphically, by name, by layer, or using any other selection mechanism natural to the design system. Activities in this area were begun.

Optimization algorithms have traditionally dealt with a relatively small number of design variables. A large scale optimization algorithm should be able to discover the few design variables that are most crucial to the design automatically, given sufficiently clever algorithms and fast machines. Work to develop large scale optimization algorithms, including sparse matrix representation and design of algorithms to traverse these large spaces, was initiated for inclusion in OptdesX. At present, these algorithms are still under development.

Discrete Variables

Many design variables occur only in discrete steps. In order to support such variables, it is necessary to have an optimizer that can work with discrete variables, and have some way to state that a variable is discrete in the design system. Many optimizers can support mixed continuous/discrete variable optimization, but no mechanism currently exists within Pro/E to state that a variable is discrete. With Pro/Develop, it is possible to develop an interface for stating such information about design parameters. Family tables provide some related functionality, but more is required.

Specifying Design/Analysis/Manufacturing Constraints

Constraining the optimizer is a problem that deserves special consideration. The optimizer provides means for directly specifying functions that relate design parameters, but this is insufficient for constraining mechanical designs. A typical manufacturing constraint might be that wall thickness be greater than some value; constraining that parameter by relating design parameters can result in $O(n^2)$ constraint functions for n design parameters. Wall thickness can much more easily be expressed as a geometric evaluation of the part, and minimum wall thickness constraint expressed in terms of that geometric evaluation.

Design systems currently fail to represent the constraints required for functionality, manufacturability, or most other "ilities". Emerging analysis functions can provide a means for evaluating such constraints, and will drive the need to specify the constraints within the modeler. Our work in Liaisons [11] has investigated mechanisms that can be useful in defining relationships necessary for specifying functionality constraints.

Results

The prototype system successfully interfaced OptdesX and Pro/Engineer. The system traversed Pro/Engineer models and transcribed parameters to optimization variables in Pro/Engineer. A number of simple objective and constraint functions were written, using each kind of functional access. A number of simple solid models were successfully optimized using the prototype.

After raising many concerns related to larger-scale use, development began on addressing these issues. At the time the project terminated, the bulk of these developments remain incomplete.

Impact

This work has examined the ramifications of connecting two current tools, OptdesX and Pro/Engineer, in an optimization loop. The prototype tool has been shown to execute well for optimizing small designs, and should be scalable to address the many concerns raised during its development. The prototype can be used, in its present state, by expert users. The concerns raised in this work have been forwarded to Defense Programs work in Design to Analysis, and are actively being investigated.

Future Directions

Future directions include work in determining the scalability of feature-based models, improved interface, interfacing to a wide class of analysis capabilities, large-scale optimization and direct specification of design/analysis/manufacturing constraints in the design model so they can be used as optimization constraints.

References

[1] Sikorski, R., *Fastcast Toolkit Integration*, Investment Casting Institute, Las Vegas NV, Oct. 1992.

[2] Kelly, J.C., *SmartWeld Proof of Feasibility Demonstration*, Sandia Presentation, Nov. 1993.

[3] Ashby, M.R., *A-Primed Project Plan, Part 1*, Sandia Project Plan, October 8, 1993.

[4] Huthwaite, B., *Concurrent Engineering Handbook*, Institute for Competitive Design, 1993.

[5] Dieter, G.E., *Engineering Design*, McGraw-Hill, 1983.

[6] Reklaitis, G., et al, Engineering Optimization, John Wiley and Sons, 1983.

[7] OptdesX™, A Software System for Optimal Engineering Design, Users Manual, Release 2.0, Design Synthesis, Inc., Orem, UT.

[8] Pro/Engineer Fundamentals, Release 16, Parametric Technology Corporation, Waltham, MA, 1995.

[9] Pro/Develop Reference Guide, Release 16, Parametric Technology Corporation, Waltham, MA, 1995.

[10] Ames, A.L., et al, "Design Simplification, Sandia Report, in preparation.

[11] Ames, A.L., et al, "Liaison Based Assembly Design", Sandia Report SAND96-3004, 1996.

Distribution

Internal Distribution:

1	MS 1002	P. J. Eicker, 9600	
1	MS 1010	M. E. Olson, 9622	
10	MS 1010	A. L. Ames, 9622	
1	MS 0188	LDRD Office, 4523	
1	MS 9018	Central Technical Files, 8940-2	
5	MS 0899	Technical Library, 4916	
2	MS 0619	Review & Approval Desk, 12690	For DOE/OSTI
6		RMSEL Library	

External Distribution:

Richard Robison
Structural Research Dynamics Corporation
2000 Eastman Drive
Milford, OH 45150-2740