Experience in Combining AI and Optimization in Different Engineering Domains

S.J. Fenves, I.E. Grossmann

EDRC 06-103-91

Experience in Combining AI and Optimization in Different Engineering Domains

Steven J. Fenves and Ignacio E. Grossmann
Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The course *Synthesis of Engineering Systems* has been offered by the Engineering Design Research Center over the last three years. In this interdisciplinary course students are exposed to two major paradigms for design synthesis: mathematical programming and knowledge based systems. The former emphasizes the mathematical formulation of optimization problems that involve discrete and continuous variables for the selection of topologies and parameters in engineering systems. The latter emphasizes representations and search techniques for processing qualitative knowledge for synthesis of designs by hierarchical decomposition.

In the fall of 1990 the two authors taught this course and assigned a term project in which the students applied both synthesis approaches to a design problem in their domain. Most of the projects dealt with civil, chemical and mechanical engineering problems, reflecting the disciplinary background of the students. Overall the quality of these projects was very good and they showed a number of different schemes for combining AI and optimization. For instance, some used AI techniques for preliminary screening or as critics of mathematical optimization, while others showed a direct comparison between the approaches.

This paper summarizes the experience gained in the course, illustrates representative student projects, discusses the major results and conclusions, and provides a perspective for future research needs and educational approaches in this area.

1. Course overview

Motivation. The course *Synthesis of Engineering Systems* is a first year graduate level course offered by the Engineering Design Research Center (EDRC) at Carnegie Mellon. It is one of four interdisciplinary courses taught by members of the EDRC. The courses are administered by the Carnegie Institute of Technology (the college of engineering at Carnegie Mellon) and accepted by all engineering departments. The overall motivation of the suite of courses is twofold: (1) to provide a rapid introduction for new graduate students and research assistants to the common methodologies underlying EDRCs research activities; and (2) to foster an interdisciplinary setting for problem-solving the reflects the multi-disciplinary nature of engineering design processes.

The synthesis course is intended to present, compare and contrast a set of generic, domain-independent synthesis methodologies. For the purposes of the course, synthesis is defined as the generation, selection and integration of systems, subsystems and components satisfying global objectives and requirements as well as constraints due to physical lrfWs governing the behavior of the systems. Synthesis is thus viewed as a generic design activity in which abstract descriptions of artifacts (objectives and requirements) are *elaborated* into more detailed descriptions (structure and behavior of the system, etc. synthesized). The methodologies of interest are those that systematically generate alternatives and identify improved designs.

The development and application of synthesis models is presented from two perspectives: the mathematical programming approach and the knowledge-based approach. The potential benefits for the combination and integration of these two approaches has been recognized recently by a number of researchers (e.g. see Glover, 1986; Simon, 1987). The two components of the course are summarized in the following subsections.

Mathematical programming component. This component presents optimization concepts and algorithms arising in Operations Research that can be applied to the synthesis of engineering systems. The context for the application of these techniques is for the case when synthesis problems can be posed quantitatively as well defined problems that require handling of constraints and determination of optimal trade-offs.

For the application of mathematical programming techniques, the following design methodology is followed. It is assumed that a representation of alternatives is developed for the choices of topologies and parameters. This representation, which is denoted as a superstructure, is then modelled with discrete and continuous variables. The former are used to model the selection of the topology, while the latter are used to model the selection of design parameters and the state variables in the system. The performance of the system and design specifications are represented through equations, and the criterion for the selection through an objective function. Finally, the design is obtained by solving the corresponding mathematical program, which in general corresponds to a mixed-integer nonlinear program (Grossmann, 1990). The MINLP may reduce to any of the following optimization problems depending on whether nonlinearities and/or discrete variables are absent in the formulation: linear programming (LP), mixed-integer linear programming (MILP) or nonlinear programming (NLP).

In order to apply the above design methodology, emphasis is first placed on the development of the superstructure of alternatives which may be represented in the form of a tree, or more generally, as a network. As there is no formal theory for developing these representations, they are illustrated through example problems. Basic concepts of optimization are then covered to first present NLP and LP techniques (in that order), in order to address design problems with only continuous variables. The techniques covered include successive quadratic programming, reduced gradient and the simplex algorithm. The modelling with 0-1 variables is introduced next together with special classes of Operations Research problems (e.g. assignment, plant location, knapsack, set covering) in order to address design problems that explicitly require discrete variables. The branch and bound method for MILP is described, as well as the Generalized Benders and Outer-Approximation algorithms for MINLP optimization. Finally, it is shown that inference problems for logic and heuristics can be modelled as MILP problems.

Throughout this part of the course students are given several homework assignments in which they have to model and solve small optimization problems in various domains. This is accomplished with the modelling system GAMS with which they can solve LP, NLP, MILP and MINLP problems. GAMS allows the students to automatically access the optimization codes MINOS for LP and NLP, ZOOM for MILP and DICOPT++ for MINLP. The latter is a code that was developed at the EDRC by Viswanathan and Grossmann (1990). Students also have access to the interactive computer code LINDO for LP and MILP problems. Although this code is not as powerful as GAMS, it has the advantage that it is easier to use.

In our experience students face several difficulties in this part of the course. Firstly, it is often the case that some of the students may have had none or very little previous exposure to optimization. We try to overcome this problem by not assuming previous knowledge, except for basic background in calculus and linear algebra. The second difficulty, which is perhaps the most serious one, is that students experience difficulty in formulating optimization problems. We try to overcome this problem by devoting some lectures to modelling, and presenting examples derived from the research work in the EDRC. Finally, although GAMS is a powerful platform for optimization, it requires some effort and time to learn the syntax. We try to overcome this problem by providing sample input files of GAMS, and offering the option to use the program LINDO which is far easier to use. However, LINDO is restricted to LP and MILP problems and has no capability of handling indexed equations and variables.

Knowledge based component. This component presents synthesis methodologies arising from the fields of Artificial Intelligence and knowledge-based expert systems (KBES). Due to the relative recency of the field, the applicable methodologies and approaches cannot be categorized and formalized to the same extent as the techniques of mathematical programming. The same general model of synthesis as in the first part of the course is used, with the following limitations:

1. synthesis tends to emphasize topology and may be less concerned with parameter value selection;

2. analysis may be non-existent or may only involve a weak quantitative model;

3. evaluation tends to emphasize *satisficing* rather than optimizing; both constraints and evaluation functions may be expressed in terms of heuristics (approximations, preferences, etc.)

Two methodologies are discussed in depth: rule-based knowledge-based expert systems and hierarchical decomposition. The rule-based formalism is used to decompose the design problem into a hierarchy of goals; sets of rules implement the identification, selection, evaluation or elimination of alternatives for each goal element. Hierarchical decomposition is a further formalization of this approach, where the knowledge base represents a hierarchy of system, subsystem or component alternatives available at each level of the artifact decomposition; problem-specific facts and heuristic constraints among alternatives at various levels represented in the knowledge base are used to systematically generate feasible alternate configurations. Other knowledge-based synthesis strategies, including heuristic search, case-based reasoning, analogical reasoning and model-based reasoning are treated in lesser detail and illustrated with engineering synthesis applications, whenever available.

In this part of the course, students are given two sets of homework assignments, in which they develop small expert systems using the tools provided. The first set uses VP-Expert, a PC-Based shell. The second set uses EDESYN, a domain-independent shell specifically geared to synthesis by hierarchical decomposition (Maher, 1988). A single application, understandable to students with varying technical backgrounds, is chosen for implementation; during the past semester, the problem was to design a screw for connecting two pieces of wood, subject to various constraints.

The students difficulties manifest themselves at two levels. First, students are unfamiliar with rule-based programming and attempt to program with strict procedural control structures. Second, the students have difficulty in collecting, organizing and coding domain knowledge, the traditional "bottleneck" of expert systems development. We try to overcome their problem by giving several lectures on knowledge acquisition, and directing students to handbooks on technical data. Some enterprising students have interviewed craftsmen and hardware store , salesmen to collect domain knowledge.

2. Project setting

Pedagogical setting of project. As stated above, weekly homework assignments are given throughout both portions of the course, illustrating and elaborating the lecture material on assigned problems. The problem sets are either generic (i.e., independent of any specific engineering domain) or drawn from specific disciplines (e.g., chemical, civil or mechanical engineering). In order to complement these assignments, which allow students to learn and master basic concepts in mathematical programming and AI, each student is required to complete a term project involving the two methodologies applied to a synthesis problem chosen by the student. In this context, the student projects serve two functions:

1. they are the only assignments that tie the two halves of the course together; and

2. they are the only opportunity the students have of applying the methodologies presented to a problem in their specific domain of specialization.

The first function may eventually be obviated as the two components of the course are more tightly integrated. The • second function will remain crucial. Students come into the course with a variety of engineering backgrounds, and as graduate students they intend to further specialize in their respective engineering disciplines. It is the theme of this and the other EDRC-sponsored courses that the best way to "open the student's eyes" to interdisciplinary methodologies, transcending their discipline-based education, is by providing a setting in which they can explore, apply and evaluate such methodologies in solving a "familiar" problem. Experience has shown that such an approach yields better understanding of the role of the methodologies presented than any set of instructor-assigned problems.

Project **organization.** Each student was given the choice of selecting a synthesis problem of his/her domain. The initial assignment involved: (1) the identification of the synthesis problem to be solved; (2) the identification of major elements constituting the qualitative and the quantitative aspects of their synthesis problem. This assignment was made after covering the mathematical programming part, and after the presentation of the rule-based KBES formalism but before the presentation of a framework for hierarchical decomposition. Below we briefly describe the initial feedback that was provided to the students.

Mathematical programming component. This component of the project requested: (1) a preliminary description of the superstructure of alternatives and the trade-offs to be examined; and (2) a preliminary formulation to identify the nature of the optimization problem.

The initial assignment was returned to the students with suggestions which in general tended to be of two types. One was for reducing the scope of the problem to simplify the formulations and to reduce the size of the optimization problems. The other was to redefine the design problem in order to expand the scope for optimization. In most cases the suggestions were of the first type. Also, the majority of the students elected to use GAMS for solving their optimization problems.

Knowledge-based component. This component of the project requested: (1) a preliminary identification of sources of knowledge to be incorporated (e.g., textbooks, handbooks, personal experience or interviews with experts); and (2) a preliminary design of the KBES, documented as a goal graph.

To assist the students, four possible models of the KBES and its integration to the mathematical programming component were presented:

1. *alternative synthesizer*, where the KBES is an alternative to the quantitative component, producing similar results based on qualitative considerations using satisficing;

2. *preprocessor* to the quantitative model, assisting in setting up the mathematical model by selecting discrete choices, initial parameters, etc. on the basis of qualitative, heuristic criteria;

3. *evaluator ox critic* of the quantitative model, evaluating its output in terms of qualitative constraints and other considerations not included in the quantitative model, and critiquing the quantitative solution; or

4. *redesign advisor* similar to an evaluator, but producing recommendations on how to reformulate the quantitative model so as to include recommendations identified by the qualitative evaluation.

The initial assignment was returned with comments and suggestions. The suggestions generally included recommendations to reduce the scope of the KBES to manageable size and suggestions for closer integration with the quantitative component. The majority of the students used the shell EDESYN for implementing the KBES.

Summary of projects. In the Fall of 1990, 18 students took the course, distributed as follows:

Mechanical Engineering	7
Chemical Engineering	6
Civil Engineering	4
Electrical and Computer Engineering	1

The projects dealt with a variety of different applications. These included design for assembly, two and three dimensional triangulation for finite element calculations, determination of trajectories for robotic spray gun, design of a gearbox, optimal project selection, synthesis of liquid-liquid extraction networks, synthesis of investment portfolios, design of truss structures and optimal wall positioning in high rise buildings. Below we briefly describe some of these projects.

3. Representative student projects

Optimum truss design and selection. In this project Susan Schrader, a first-year graduate student in Civil Engineering, investigated the optimum design of truss structures and the use of heuristics for selecting truss types and preliminary proportions. The mathematical optimization component formulated truss design as an LP problem subject to equilibrium and stress constraints. The geometry of the truss was fixed, and all potential truss members were included in the formulation. Under a single loading condition, the LP converged to a statically determinate configuration, as expected (i.e., only 6 of the 10 members had non-zero areas, corresponding to the 6 available equilibrium equations). This is a simple example of topology or configuration optimization. The knowledge-based component consisted of sets of rules for selecting the layout of the truss, the preliminary dimensions, and the preferred configuration from among six standard truss types. The heuristics were extracted from old (circa 1920) structural design textbooks; the student commented that these rules may not be realistic for modern bridge design practices. The student proposed two integration schemes: the KBES as a preprocessor supplying layout and dimensions to the LP program; or the KBES as a post-processor evaluating the optimal design in terms of suitability for its intended use.

Optimal project selection. Paul Peck, a first-year graduate student in Chemical Engineering, formulated a project selection scenario involving 7 projects. The projects are first pre-screened by a KBES which evaluates them on the basis of: the company's long range goals; the current economy; worker satisfaction; and public image concerns. Projects rated as acceptable are then optimized with a MILP formulation to maximize profit with constraints on project cost and available funds, engineering hours and land. The MILP contained 7 decision variables with an OR constraint on two of the projects. The integrated system was run on the following cases:

1. Base case: MILP selected projects 3, 6 and 7;

2. Stable economy, goal to increase production and improve public image: KBES rejected project 3 because it produces a harmful chemical, MILP

selected projects 2, 6 and 7; and

3. Unfavorable economy, good public image: KBES rejected projects 2 and 3 because of high startup costs, MILP selected projects 1, 6 and 7.

The student commented on the extreme sensitivity of MILP to the cost coefficients in the cost and profit functions, and on the limited knowledge base of the KBES implemented resulting only in accept/reject decisions rather than a desirability scaling factor.

Design of trajectory for robotic spraying. Rahul Bhargava, a graduate student in Mechanical Engineering working on an EDRC project, considered the problem on how to spray a uniform layer of metal on a 3-dimensional surface using a robotic spray gun. Aside from having to achieve good finish quality by uniform spraying, the objective is for the gun to move along a path that involves minimum changes in orientation while at the same time requiring a spraying sequence that requires minimum time. In order to formulate the optimization problem it was assumed that a set of N points was given which cover the entire surface. Each point is characterized by a 3-dimensional position and a direction which is normal to the surface at that point. To ensure good quality, the gun should be oriented along this vector. Given this information, the aim is to visit each of these points exactly once so as to minimize the distance traveled by the gun and the changes of orientation. This problem was formulated as a Traveling Salesman problem (TSP) in which the cost coefficients for moving from one point to the other include a combination of the distance and change of orientation. Although more efficient special purpose methods exist for the Traveling Salesman problem, the problem was solved with GAMS as an LP problem with a small subset of subtour elimination constraints. For problems with up to 10 points the LP solution yielded optimal closed tours. The KBES component for this project was used as a critic or evaluator for the optimal solution of the TSP to try to identify sections where uneven spray may occur due to the selection of points. Heuristics based on the experience of the student and various people working in the Rapid Tool Manufacturing project at the EDRC were coded in VP-Expert. The student noted that a limitation of the qualitative model was its local outlook. However, the student also felt that the expert system could be a useful advisory program to alert for potential faults, which in turn could suggest the redefinition of the spraying points.

Design **for** assembly. The goal of this project by Raju S. Mattikali, a graduate student in Mechanical Engineering working on an EDRC project, is to address assembly concerns during design synthesis. The objective is to determine the shape of components so as to facilitate assembly. The two parts of the student's project address this problem at two distinct levels of representation of a component. The quantitative portion represents the component as a flexible polygon with fixed lengths but changing angles. The interior angles are to be determined such that: the distance between the closest vertex of the polygon to a boundary edge is maximized; the total sum of distances is minimized; the polygon does not intersect any boundary edge; and the polygon remains ordered (i.e., no crossing edges) and convex. The resulting problem with non-linear objective functions and constraints was solved using GAMS. The KBES portion of the project assumes that the component's shape features (e.g., bosses, stubs, etc.) are identified and may be evaluated for handling and insertion in terms of the component's symmetry, size, weight, handling distance, type of fit, hold-down requirements, etc. The student commented that the two parts of the project could not be integrated, due to the absence of a mapping function between the low-level geometric representation used for the NLP problem and the high-level features used by the KBES.

Synthesis and modification of mortgage-based securities portfolios. In this project, Jonathan Knight, a first year student in Chemical Engineering, considered the selection and possible restructuring of a portfolio by quantitative and qualitative approaches, respectively. For the quantitative part, the objective in the project was to synthesize a portfolio of mortgage-backed securities from an alternative space of assets and hedges that would maximize future liquidation value. The portfolio was subject to cash flow constraints, maximum portfolio size constraints, asset purchasing constraints and portfolio management heuristics. MILP and MINLP models were developed and applied to a problem with 10 assets and 4 hedges. For the qualitative aspect, the objective was to design an expert system to modify or restructure the portfolio given uncertainties in future interest rates and prepayment rates. The major goal of the expert system was to avert possible financial disaster. The unique feature of this project was the fact that it used real world data that were supplied by an investment company. This in turn required considerable preprocessing of data to formulate the parameters in the quantitative and qualitative models. As for the integration of quantitative and qualitative knowledge, it is interesting to note that this was accomplished at two levels. Firstly, the mathematical models incorporated integer constraints that reflected management portfolio heuristics to eliminate alternatives that were unlikely to be financially attractive. Secondly, the qualitative component served as a redesigner of the quantitative solution in the event that the parameters used in the optimization would change significantly.

4. Generalizations

The experience in teaching the course on *Synthesis of Engineering Systems,* and particularly the feedback from student projects, clearly support our premise that engineering synthesis can significantly benefit from the creative integration of quantitative, mathematical optimization-based methods with qualitative, Artificial Intelligence-based methods including, but not limited to, knowledge-based expert systems methodologies. On the other hand, as the student projects amply demonstrate, the tools implementing these two methods presently form two unconnected "islands of automation" with no direct linkages between the two approaches. This latter observation suggests an agenda for considerable future work. This agenda is discussed below in the major categories of research, tool development, and education.

Research. A basic question that has emerged from the experience with the projects in this course, and in fact with other projects in the EDRC, is how to develop a common framework for combining or integrating the mathematical programming and Artificial Intelligence approaches for the synthesis of engineering systems. In general, it would seem that two major alternative paradigms are the following:

1. The quantitative and qualitative models act as two interfaced knowledge sources under a common coordination scheme.

2. The quantitative and qualitative models are tightly integrated under a common framework.

The first alternative would build on the emerging methodologies of distributed problem solving, cooperative problem solving and concurrent design (Cutkosky and Tenenbaum, 1990; Durfee et al., 1989; Lesser and Corkill, 1987). In such systems, the overall problem-solving task is distributed to a number of heterogeneous agents, each of which is capable of contributing some portion to the overall solution process. Communication, coordination and control of agents is exercised in a variety of ways, of which the blackboard architecture is most popular (Lesser and Corkill, 1987).

Within the context of engineering synthesis, agents in the distributed problem-solving system would consist of quantitative agents implementing various MILP and MINLP procedures and qualitative agents performing a variety of functions, including:

1. *Preprocessors* which may assist the user to formulate optimization problems, generate equations and constraints, select the superstructure and define the corresponding discrete variables, etc.

2. *Method selectors* which could examine the nature of the optimization problem and assist in selecting the appropriate mathematical programming agent(s) to be activated.

3. *Progress monitor* that wpuld monitor the progress of quantitative agents and recommend problem reformulation or alternate agent selection if the quantitative agents encounter difficulties (slow or no convergence, infeasibility, etc.).

4. *Evaluators, critics* or other *postprocessors* that may apply additional, qualitative criteria to the solutions generated by the quantitative agents.

5. *Redesign advisors* that may use the results of the postprocessors to recommend changes in the problem formulation and other design changes in the synthesis process.

Major research issues that need to be addressed include the development of symbolic problem representations that can support the full range of quantitative and qualitative processes and the development of domain-independent representations of the heuristics and problem superstructures involved.

It is to be understood that interfaced systems of the class sketched cannot aim at producing global optima over all the variables and constraints that may arise. However, they may lead to increasingly comprehensive optimization problems or, at least, provide optima over broader ranges of problems considerations than possible with direct, manual generations of quantitative optimization models.

The second alternative is the one where both qualitative and quantitative knowledge sources are activated simultaneously. This case would be particularly relevant for expediting the search in large combinatorial optimization problems. Here one possible integration framework is to convert the heuristics and logic of a knowledge base as an MILP model (see Post, 1987; Raman and Grossmann, 1991a), which can then be incorporated into a quantitative optimization model. The qualitative MILP model is obtained by stating the logic and heuristics of a knowledge base in conjunctive normal form which is then translated into the form of inequalities with 0-1 variables. Boolean variables are also assigned to the possible violation of heuristics whose weighted sum is minimized in the objective function. As discussed by Raman and Grossmann (1990), a quantitative integration can be accomplished at the formulation level in which the qualitative MILP model is added in the form of constraints to the quantitative model with a threshold value for the violation of the heuristics. This has the effect of limiting the search space. When heuristics are excluded, and only logic relations are retained, optimality can still be guaranteed, and the logic serves as cuts that reduce the combinatorial search of the problem. Alternatively, one may include the qualitative MILP constraints in one component of an algorithm, such as is the case of master problems for MINLP optimization. Both schemes for integration have shown to have the effect of often reducing the computational expense for the search in the quantitative model.

An important drawback, however, with a quantitative framework for integration is that the additional constraints of the qualitative MILP model increase the size of the optimization problem, and therefore computational savings are not always realized. To overcome this difficulty, an alternative is to perform the computations symbolically on the qualitative part. For instance, recently Raman and Grossmann (1991b) have shown how logical inference for selecting the branching in 0-1 variables and fixing subsets of them can be performed as part of a quantitative branch and bound search for MILP optimization. This integration scheme has proved to be much more efficient than the quantitative integration described above. However, this method is still limited to handling only logic information, not heuristics. Therefore, a major challenge that remains to be solved is how to integrate simultaneously symbolic computations of a knowledge base with quantitative computations for optimization.

Tool development. The last decade has witnessed the development of a number of modelling tools for optimization such as GAMS and LINDO, which represent a major improvement over the low level optimization routines which in the past required the development of FORTRAN routines and/or MPS files. Although the current modelling systems have greatly facilitated the use of optimization, the fact remains that they still require significant coding effort and learning for efficient use. Furthermore, except for few LP modelling systems, the transfer and preparation of data can be a major task. It would be clearly desirable to develop optimization tools that require little expertise by the users, which can handle symbolic information and be accessed in a variety of forms, and which can be easily interfaced to data bases or spreadsheets. Furthermore, nonlinear optimization tools have not achieved yet the reliability and robustness of linear programming codes. These are all areas which clearly deserve attention for further work. It should be noted that current work along these lines includes the development of the ANALYZE support system by Greenberg (1991) for LP optimization, the OSL routines for LP and MILP by IBM which have an open architecture, the What's Best and What-If Solver programs for LP and NLP optimization that can be accessed through spreadsheets, and the program REFORM for reformulation of NLP and MI(N)LP problems by Amarger et al. (1990).

In a similar vein, tools for knowledge-based synthesis are still rudimentary. The available KBES shells are largely geared to diagnostic problems. EDESYN (Maher 1988) is the first attempt to develop a domain-independent synthesis shell comparable in scope to the available diagnosis frameworks. Entirely new generations of knowledge-based synthesis frameworks suitable for broad classes of engineering synthesis problems are needed.

Education. Despite the fact that synthesis lies at the core of engineering design, progress has been slow in incorporating synthesis in the undergraduate and graduate engineering curriculum. Given the fact that decisions at the synthesis level have a major impact in the cost and quality of the designed artifacts, there is a clear need for synthesis to permeate engineering education. The experience with our course has shown that synthesis is teachable and that there are benefits from doing this in an interdisciplinary setting. This is of course not to say that discipline-based synthesis courses are not needed. These are certainly required to exploit the physical insight and knowledge of the particular discipline. On the other hand, an interdisciplinary course can emphasize basic design methodologies, as our course has done, and provide students with broader perspective.

Among the major challenges that we feel need to be addressed for effectively teaching synthesis as an interdisciplinary graduate course are the following. Firstly, how to present underlying synthesis concepts that emphasize strategies and formulations to prevent mathematical optimization from becoming a "black box^{11}? Secondly, how to to provide "synthetic" knowledge-bases to supplement students* missing or limited domain knowledge? Finally, how to effectively present the combination and relation of optimization and AI approaches to synthesis? Undoubtedly answers to these questions will require both basic research as well as additional experience in teaching these type of courses.

5. Summary and conclusions

This paper has reported the experience in teaching an interdisciplinary graduate course on synthesis at the Engineering Design Research Center at

Carnegie Mellon. The outline of this course has been described, emphasizing the' experience with projects where students attempted to combine optimization and AI in a variety of different domain applications. While these projects provided some interesting results and insights, the tools implementing these two methods presently form two unconnected "islands of automation" with no direct linkages between the two approaches. Based on this experience some possible directions for future research, tool development, and education have been pointed out.

6. References

Amarger, R., L.T. Biegler and I.E. Grossmann, "REFORM- An Intelligent Modelling System for Design Optimization", EDRC Report (1991).

Cutkosky, M.R. and J.M. Tenenbaum, "A Methodology and Computational Framework for Concurrent Product and Process Design, *Mechanism and Machine Theory,* 25:3 (1990).

Durfee, E.H., V.R. Lesser and D.D. Corkill, "Trends in Cooperative Distributed Problem Solving", *IEEE Transactions on Knowledge and Data Engineering,* KDE-1:1 (1989).

Glover, F., "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Optns. Res,,* 13, 533-549 (1986).

Greenberg, H.J., "Intelligent Analysis Support for Linear Programs", presented at National AIChE Meeting, Houston (1991).

Grossmann, I.E., "Mixed-Integer Nonlinear Programming Techniques for the Synthesis of Engineering Systems," *Res. Eng. Des.,* 1, 205-228-(1990).

Lesser, V.R. and D.D. Corkill, "Distributed Problem Solving" in S.C. Shapiro (editor), *Encyclopedia of Artificial Intelligence* 1, John Wiley, New York (1987).

Maher, M.L., "Engineering Design Synthesis: A Domain Independent Representation,Artt/icu'z/ *Intelligence for Engineering Design, Analysis and Manufacturing,* 1:3, 207-213 (1988).

Post, S., "Reasoning with Incomplete and Uncertain Knowledge as an Integer Linear Program", *Proc. Avignon 87: Expert Systems and their Application,* Avignon (1987).

Simon, H.A., "Two Heads Are Better Than One: The Collaboration Between AI and OR", *Interfaces,* 17, 8-15 (1987).