COMPLEXITY OF THE SYSTEM DESIGN PROBLEM

William L. Chapman
Hughes Aircraft Company
Tucson, AZ

Jerzy Rozenblit
A. Terry Bahill
University of Arizona
Tucson, Arizona

ABSTRACT

The system design problem describes the process used to translating the need or requirements for a system into an actual design. It requires selecting components from a given set and matching the interfaces between them. Those that can be connected to meet the top level system's input and output requirements are tested to see how well they meet the system's performance and cost goals. We will prove that this system design process is NP-complete by restricting the Knapsack problem, which is known to be NP-complete, to an instance of the system design process problem. The results indicate that designing optimal systems with deterministic, polynomial time procedures is not possible.

1 INTRODUCTION

System design is a process of translating or mapping the requirements of a system into a buildable design. The problem can be described in terms of set theory and formally described. Discovering the complexity of this problem statement is important to realize what approach for solution should be used.

2 NP-COMPLETE PROBLEMS

NP-complete is the name of a class of problems for which there is no known efficient algorithm for finding an optimal solution [1], [3]. As the problem size increases, the number of steps necessary to solve the problem increases exponentially. Efficient algorithms are those whose number of steps increase at the rate of a polynomial.

NP stands for nondeterministic polynomial. Mathematicians created a conceptual device called a nondeterministic machine to solve these problems. A nondeterministic machine has an infinite number of processors and two stages; a guessing stage and a checking stage. Each processor guesses an answer and the checker verifies that it is a good answer in polynomial time. Because an infinite number of processors exist all the guesses are done in parallel and the number of operations does not combinatorially explode. Of course, this is a fantasy machine , but it helps to illustrate the fact that a certain set of problems can only be solved in polynomial time if one of these machines is used. Any algorithm that can solve a problem in the class NP is polynomial if run on a nondeterministic machine. This is because although the amount of work required to solve the problem on one machine may increase to infinity, if the processors working on the problem increase to infinity instead, then the time to solve the problem will grow only at the rate of a polynomial for any single processor. There is also a set of problems called NP-hard that cannot be solved with a nondeterministic machine but are related to NP as shown in

0-7803-2531-1/95/$4.00

Figure 1. These problems are outside the scope of this paper.

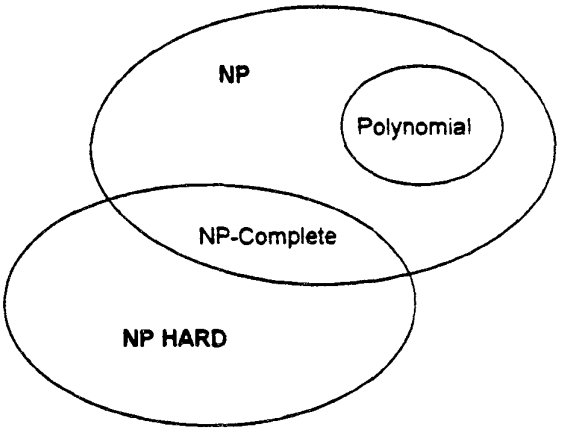

Figure 1 - How NP-complete is related to Polynomial and other algorithms.

Clearly, any polynomial algorithm could still use the nondeterministic machine because the algorithm could be restricted to one processor.

NP-complete is a class of problems that can be solved on a nondeterministic machine for which no known polynomial algorithm exists. It has never been proven that a polynomial algorithm does not exist - but no one has ever found one, and mathematicians do not think anyone ever will. One critical feature of all NP-complete problems is they can be mapped into an instance of each other by using a polynomial transformation. If even one of the problems in the class NP-complete can be shown to have a solution in polynomial time, then all of them have a solution. Yet none has been found in 30 years of searching by very talented people. Thus, it is generally agreed that if a problem is shown to be NP-complete, then no efficient algorithm for optimally solving the problem will ever be found.

3. THE SYSTEM DESIGN PROCESS PROBLEM

In terms of systems theory, system design can be described as stating a set of input, output and time restrictions for an overall desired system and a series of performance and cost measures [2], [5]. For a given set of components available to build the system, a possible system is configured that satisfies the system's input, output and time specifications. This system is then tested using some predefined test requirement to provide an overall system performance index (**PI**). This must exceed a customer provided acceptability limit. The cost of the system in terms of time, money or other resources is then computed into an overall cost index (**CI**). This **CI** must be less than some customer specified target value.

The systems approach to design can be characterized as follows:

Define a series of potential components $\mathbf{Z}_i$ that constitute the available technology to build the desired system. Each $\mathbf{Z}_i$ has a time index $\mathbf{T}_i$, an input port $\mathbf{I}_i$, and an output port $\mathbf{O}_i$. The ports provide the means of connecting the different components together to form a system. The components connect output ports, $\mathbf{O}_i$, to input ports, $\mathbf{I}_i$, using system coupling recipes, SCR, to form a potential system, $\mathbf{Z@}_j$. Define the overall input to the desired system as $\mathbf{I}_0$ and the overall output of the desired system as $\mathbf{O}_0$. For simplicity we examine only single input, single output systems that are made up of components that are single input, single output systems. (See Figure 2.) It seems reasonable that if this design is NP-complete then the more complex multi-input, multi-output design must also be NP-complete.

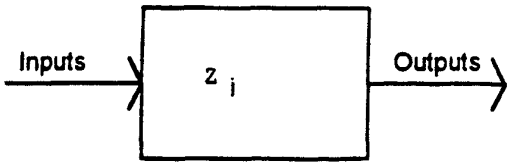

Figure 2 - A single input single output system.

The total set of potential systems, **Z@j**, that can be built from the components $\mathbf{Z_i}$ is shown in Figure 3.

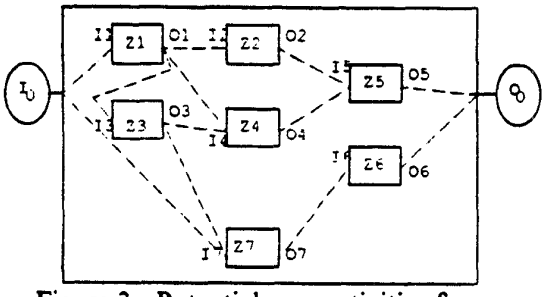

Figure 3 - Potential connectivities for a series of 7 components.

These connections can be expressed as a directed graph where the individual components are nodes and the possible connections between ports are the arcs. The initial source for the directed graph is the system input port, $\mathbf{I_0}$, and the initial target (or sink) for the directed graph is the output port, $\mathbf{O_0}$. Let the length of each arc represent the cost of connecting the two components (See Figure 4.) To find a potential system design we must find a path through a directed graph. Because we have restricted our attention to a single-input, single-output system components the connection between the system input and output is a path. If we had allowed multi-input, multi-output system components then it would be possible to obtain a subgraph instead of a path. Finding a subgraph within a network is also NP-complete, but we will not examine that problem in this paper.

Finding a path through a directed graph can be accomplished in polynomial time. Finding the shortest path through the graph can also be accomplished in polynomial time. The problem is that system design is not simply solving for one constraint such as the least cost. In addition a system must be found that has maximum performance. Having a minimum and a maximum to solve at the same time requires tradeoffs as a search for the best value is done. This requires much more searching especially as the number of options increases.

For a simple path the **CI** of the system is then the sum of the length of the arcs. The resultant path is equivalent to one system coupling recipe creating a system, **Z@j**. For the route in Figure 5 the system coupling recipe is

$$SCR=\{(I_0,I1Z3),(O1Z3,I1Z4),(O1Z4,I1Z5),(O1Z5,O_0)\}$$

This means that the system input, I_0, is connected to component Z3 input port 1. System component Z3 output port 1 is connected to system component Z4 input port 1, and so on. After the system is coupled it is tested per the requirements to obtain the **PI**.

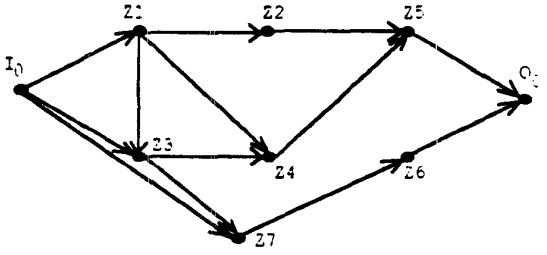

Figure 4 - A directed graph of the connectivities shown above.

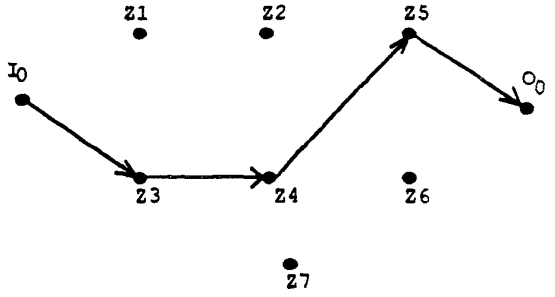

Figure 5 - One possible route through the directed graph above.

This process is how designs are created. An engineer finds components that satisfy the necessary input and output requirements and creates an interconnection of these parts to satisfy the performance and cost requirements. Several different systems (concepts, alternatives, models or prototypes) are often considered before a selection of the best possible is determined based on some tradeoff study. To guarantee a system is optimal would require testing all of the possible configurations.

4. THE SYSTEM DESIGN PROBLEM IS NP-COMPLETE

To illustrate NP-completeness we will now restrict the Knapsack problem to the systems design process problems described in the sections above.

It was proven by Karp that the Knapsack problem is NP-complete (Karp, 1972). It is formally described below.

Instance: A finite set U, a "size" $s(u) \in Z^+$ and a "value" $v(u) \in Z^+$ for each $u \in U$, a size constraint $B \in Z^+$, and a value goal $K \in Z^+$.

Question: Is there a subset $U' \subseteq U$ such that

$$\sum_{u \in U'} s(u) \le B \quad \text{and} \quad \sum_{u \in U'} v(u) \ge K$$

We have defined the system components as coupled by an SCR to create an overall system $\mathbf{Z@_j}$ which has associated measures $\mathbf{PI_j}$ and $\mathbf{CI_j}$. Let $\mathbf{K=PI_j}$ and $\mathbf{B=CI_j}$. Let **U=Z,** which is the set of possible components that can be connected per an SCR. Let $\mathbf{U'=Z@}$ be the subset of components selected from $\mathbf{Z_i}$ by means of the SCR to form **Z@**. Each component $\mathbf{u_K=Z_i}$ has an associated cost that contributes to the **CI**. Let **s(u)=cost(Z)**. Each component $\mathbf{u_K=Z_i}$ has an associated value that can be measured by the test requirement that contributes to the **PI**. Let **v(u)=value(Z)**. Based on the acceptability criteria, specified by the customer, the cost constraints are defined and so are the minimum acceptable performance criteria.

The values of each component, Zi, are combined into a composite performance measure, PIj. There are many ways the values can be obtained. The simplest is a linear combination. Any other continuous, monotonically increasing function would be harder to solve and thus require even more computer time. If we restrict the system design process problem to performance measures that combine linearly, and to cost measures that combine linearly, then

$$\sum_{Z_i \in Z@_j} \text{value}(Z_i) = PI_j \quad \text{and} \quad \sum_{Z_i \in Z@_j} \text{cost}(Z_i) = CI_j$$

then if we can find a $\mathbf{PI_j}$ and $\mathbf{CI_j}$ using a polynomial algorithm such that

$$\mathbf{PI_j > PI_0} \quad \text{and} \quad \mathbf{CI_j < CI_0}$$

then we have solved the system design problem. Hence, if we can solve the system design process problem then we can solve the Knapsack problem, but we know the

Knapsack problem to be NP-complete, therefore the system design process problem is NP-complete also. Figure 6 gives a summary of the mapping from the Knapsack Problem to the System Design Problem.

Showing that the Knapsack Problem can be reduced to the System Design Process Problem is sufficient for proving NP-Completeness because if a solution for the System Design Process problem was available we could use it to solve the Knapsack and hence all NP-Complete problems.

Knapsack		System Design Problem
a finite set U	→	**Z**
a subset U'	→	**Z@**
a "size" s(u)	→	**cost(Z)**
a "value" v(u)	→	**value(Z)**
a size constraint B	→	$\mathbf{CI}_j$.
a value goal K	→	$\mathbf{PI}_j$
$\sum_{u \in U'} s(u) \le B$	→	$\sum_{Z_i \in Z@_j} \text{cost}(Z_i) = CI_j$
$\sum_{u \in U'} v(u) \ge K$	→	$\sum_{Z_i \in Z@_j} \text{value}(Z_i) = PI_j$

Figure 6. A summary of the mapping.

5. IMPLICATIONS

The implication of the system design problem being NP-complete is that it is unlikely a computer will ever be created that can perform the design of a complex system better than a human. Subsets of the entire design process, such as routing and checking interfaces, are done better by computers now, however no computer algorithm exists to create even a simple automobile, factory or personal computer. The creation of a system is as much art as it is science, because the combinatorics involved require original solutions rather than fixed algorithms for solving the problem.

Once more complex issues of individual creativity and adjusting for perceived customer wants, rather than those that are accurately expressed, are considered, it becomes even more obvious that a totally automated design system is impossible. Research must focus on the human in the loop design solution as the only feasible approach to solving the System Design Process problem.

Most design problems are Continuous Improvement because most design is redesign. A feasible solution exists, but it is not optimal. Improvements are available and often not very hard to find. The performance of the system improves quickly at first, but optimality is hard to achieve. Eventually it is not worth the cost of the extra resources to improve the performance. See Figure 7.

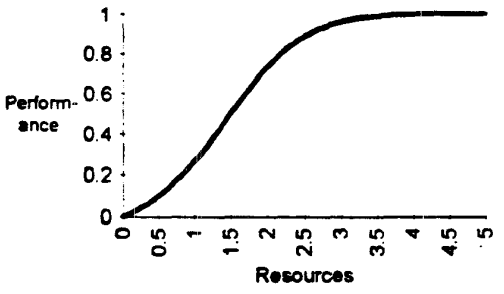

Figure 7- Continuous Improvement.

There are many design cases where a feasible solution does not already exist, but it is not a challenge to find one. We call these Original Innovative designs. It is original because no prior feasible solution exists that will be incrementally improved. This type of design requires innovation, because the initial feasible solution created will not be close to optimal without an innovative design that creates the feasible solution near the optimum. When the

designers are given a chance to start from scratch and design a new system then they are performing an original design.

The final type of design is called Breakthrough design. No known or easily obtainable feasible solution exists or has ever existed. Only a breakthrough in science or engineering will create a feasible solution. Resources are expended at an incredible rate with no improvement at all. If the project were stopped half way through the design effort there would be nothing to show for all the money spent. Finally a breakthrough occurs and performance improves rapidly. See Figure 8.

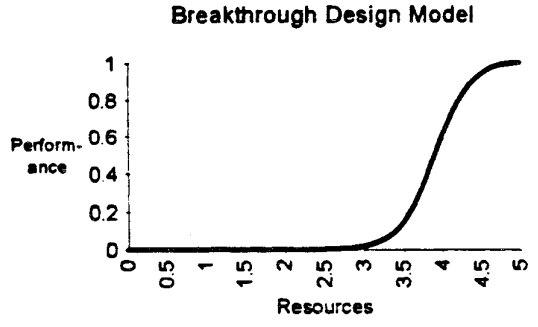

Figure 8 - The Breakthrough design.

The implications for the system design process are clear. Different approaches must be used when a feasible system is at hand versus when one is not easily obtainable. The Continuous Improvement problem will have a different system design process than that for Original Innovative design or for Breakthrough designs. None of the methods presented to approach these different design problems will guarantee optimality, but good solutions can be obtained for them all.

If it is so difficult to obtain optimality then one might ask why are there so many good systems? The answer lies within the solution techniques of NP-complete problems. Even the most difficult problems in this class have algorithms to obtain good solutions (that is, a solution within a few percent of a theoretical optimal when it is possible to compute) with relatively simple polynomial algorithms. The solution techniques applied to solve these problems (such as the Traveling Salesman Problem, Knapsack problem, maximum path through a network, minimum test collection, graph 3-colorability, etc.) to obtain good solutions can also be applied (and have been applied, knowingly or not) to the system design process. This provides a mechanism to analyze the tools and techniques of design.

We are now investigating the various methods of solving NP-complete problems and using these NP-complete algorithms to solve design case studies.

6. SUMMARY

This paper has demonstrated that the system design process is NP-complete by a mapping from the Knapsack problem. The implications are that achieving an optimal design for a complex system is not likely. It is possible to design forever without achieving an optimal solution. Therefore, limits must be set on the design early in the process. In addition creation of a computer based design system will be very difficult. The interesting aspect of NP-complete algorithms is that it is often quite easy to find near optimal solutions. Within the context of product design, optimal is not often an objective, but rather satisfaction of a problem statement. Therefore, a solution good enough to satisfy the customer may be within reach of a knowledge based design system. We plan on continuing research in this area by examining the relationship between algorithms that solve NP-complete problems and the system design methodology.

7. REFERENCES

[1] M. W. Bernand and R. L. Graham, "The Shortest Network Problem," *Scientific American*, January 1989, pp. 84-89.

[2] W.L. Chapman, A.T. Bahill and A.W. Wymore, *Engineering Modeling and Design*, CRC Press Inc., Boca Raton, 1992.

[3] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[4] R.M. Karp, "Reducibility Among Combinatorial Problems," *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher (eds), Plenum Press, New York, pp. 85-103, 1972.

[5] A. W. Wymore, *Model-based Systems Engineering*, Boca Raton: CRC Press, 1993.