An Architecture for Exploring Large Design Spaces

John R. Josephson[1], B. Chandrasekaran[1], Mark Carroll[1], Naresh Iyer[1], Bryon Wasacz[2, 3], Giorgio Rizzoni[2], Qingyuan Li[2], David A. Erb[4, 5]

The Ohio State University, Columbus Ohio, 43210 USA
[1] - Computer and Information Science Department
[2] - Mechanical Engineering Department
[3] Present address: Motorola SPS, Austin, TX, 78735 USA
[4] – Center for Automotive Research
[5] Present address: ERB Professional Services,
Upper Arlington, OH, 43221 USA

{jj|chandra|carroll|niyer}@cis.ohio-state.edu,
RA6734@email.sps.mot.com, rizzoni.1@osu.edu,
li-q@rclsgi6.eng.ohio-state.edu, dave_erb@mindspring.com

Abstract

We describe an architecture for exploring very large design spaces, for example, spaces that arise when design candidates are generated by combining components systematically from component libraries. A very large number of candidates are methodically considered and evaluated. This architecture is especially appropriate during the stage of conceptual design when high-level design decisions are under consideration, multiple evaluation criteria apply, and a designer seeks assurance that good design possibilities have not been overlooked. We present a filtering technique based on a dominance criterion that can be used to select, from millions of design candidates, a relatively small number of promising candidates for further analysis. The dominance criterion is lossless in that it insures that each candidate not selected is inferior to at least one of the selected candidates. We also describe an interactive interface in which the selected designs are presented to the designer for analysis of tradeoffs and further exploration. In our current implementation, the computational load is distributed among a large number of workstations in a client-server computing environment. We describe the results of experiments using the architecture to explore designs for hybrid electric vehicles. In a recent experiment more than two million distinct designs were evaluated.

Motivation

Design can be considered to start from a specification of properties and behavior that an artifact is intended to satisfy. It typically ends when the designer is able to describe a set of components and their interconnection, and a mode of use by a user (Chandrasekaran, 1990). If an instance of the artifact is constructed with the components in the specified inter-component relationships, and if a user interacts with it as described in the mode of use, then the artifact's properties and behavior are supposed to satisfy the given specifications.

Design, like all problem solving, can be formulated as a search in a problem space (Nilsson, 1971; Newell, 1980). Except in routine design tasks, the design process usually involves considering alternative configurations of components, alternative components, and various parameter values. The design candidates that arise during this process are usually evaluated using multiple criteria. Due to time and other resource limitations, designers usually consider only a narrow range of the possible combinations of components and configurations.

At times, the design problem may be formulated explicitly as a parameter optimization problem, and well-known optimization techniques may be applied (Wilde, 1978). Most commonly, these techniques are variations of hill climbing. Given a design candidate, the direction of change in which the gain in a performance measure is largest is first ascertained, a new design candidate is chosen in that direction, and this process is repeated until changes in any direction result in a decrease in the performance measure. However, this optimization technique is not always applicable. When design candidates are generated by changes of components and configurations, which may be unordered, there may be no adjacency relationship to exploit in the space of design candidates. Moreover, hill climbing techniques, by

requiring that a single evaluation function be defined, preclude explicit, local reasoning about tradeoffs among multiple performance criteria. In general, it is hard to use symbolic knowledge in numerical optimization schemes and thus they are not very good for early-stage design.

In this paper we describe a kind of design-space exploration that explicitly considers large numbers of design candidates, sampling widely from all regions of the design space. This exploration tells the designer how the design candidates in the space behave with respect to the various evaluation criteria, and helps to identify candidates or regions with interesting properties. Exhaustive exploration would be ideal, where all possible design candidates are examined. However, exhaustiveness is not essential for this kind of design exploration to be useful. What is needed is that all regions of the design space are sampled sufficiently to develop an understanding of the characteristics of the design space, which may well call for considering a very large number of design candidates. This kind of exploration may set the stage for a more detailed exploration of selected areas of the design space. Thus, this kind of exploration is especially appropriate for the conceptual design stage.

Because of the need to examine a very large number of candidates, this kind of design-space exploration may require substantial computing power. Fortunately, conceptual design is a relatively small part of the overall design process. Thus, allocation of substantial computing resources for this stage of design may be justified in view of its importance, especially as computing power is becoming more affordable. This kind of design space exploration has natural parallelism that can be exploited to distribute the computational burden; in our experiments we used a large collection of networked workstations to provide the needed computing power.

Overview of the Architecture

The overall architecture is that of an interactive decision-support architecture for design, as shown in Figure 1. A component/configuration library is available for generating design candidates. The user can specify constraints. The Good-Design Seeker generates design candidates by selecting components from the library and composing them, according to configuration templates (generic devices), to satisfy the given constraints. Several design critics evaluate each design candidate. Each critic assesses a candidate design from the point of view of a particular aspect of performance. One might focus on cost, another on convenience of use, yet a third on diagnosability, and so on. A critic might use an evaluation function based on the conclusions of other critics.

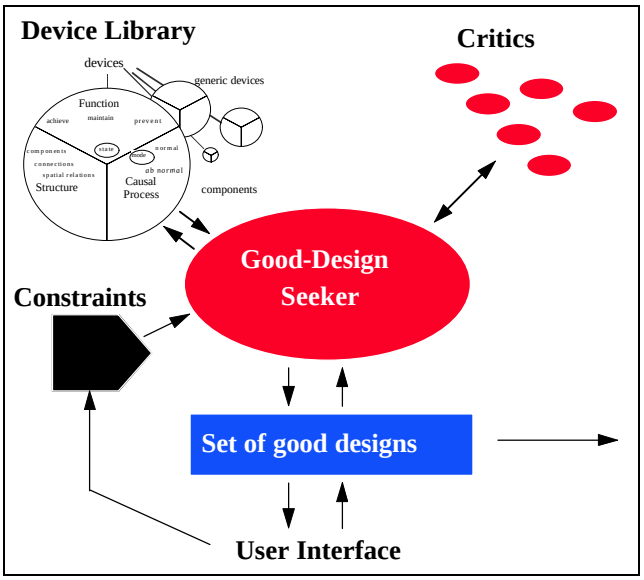

Figure 1. Decision-support architecture for design

The total number of design candidates might be quite large, so, for the designer to make effective use of the information from the critics, it would be desirable to have some form of filtering that selects a relatively small number of designs that are worth examining further. As it happens, we have discovered a lossless filtering criterion, i.e., one for which there are guarantees that there is no danger of excluding good designs. This will be described in a following section. In initial experiments, with real data, this filtering criterion has shown quite good performance, as will be described.

The designer is presented with the design candidates that survive the filtering process. Since a large number of candidates may yet remain, it is important to develop effective ways to aid the designer in investigating the properties of the surviving designs. Thus, the user interface is an important part of the architecture. We have discovered user-interface elements that are very useful. In particular, visualizing the set of surviving designs by way of interactive, connected, tradeoff diagrams enables the user to zoom in on subsets with desirable tradeoff characteristics, and so reduce the number of designs for further investigation to a manageable few. These will be described in a later section.

The scope of our project includes many issues in the design of component libraries and design critics that will not be discussed in this paper. In this paper, we will emphasize issues related to the exploration of large design spaces, using multiple criteria of evaluation. Besides a high-level conceptual description of the architecture, we will also describe our most recent implementation of it, which distributes the computation over a network of workstations, using otherwise idle machines, and has achieved explorations that consider more than two million candidates.

Good-Design Seeker

Design candidates are generated by systematically considering the members of a set of preset configurations or "generic devices." For each generic device, alternative component substitutions from a device library are systematically considered, and for components, various parameter values are considered chosen from a set of user-specified landmark values.

Design candidates are checked against any constraints that might rule them out. We have considered two sets of user-supplied constraints: one set is applied to partially specified designs and the other is applied to fully specified designs. Partial designs are created on the way to creating fully instantiated design candidates, and arise when not all of the components in a configuration have yet been chosen, or not all the parameter values. Sometimes partial designs may be checked against constraints with little computational cost. For example, if there is a constraint on total weight, a partially specified design may be rejected as soon its weight exceeds the limit, thus avoiding the substantial amount of work involved in generating and evaluating refinements of the design. Constraints that apply to fully specified designs are used to eliminate candidates. Such constraints might be used to eliminate implausible combinations or unacceptable performance.

Prior to constraint checking, designs are sent to various design critics, and each design is annotated with the conclusions from the critics. These conclusions can be used by the constraints, and are the basis for design filtering, which selects a subset of candidates to present to the designer. Dominance filtering is an important type of design filtering, and may be applied after generating and evaluating all candidates, or incrementally, as candidates are being generated and evaluated.

Dominance Filtering

We say that design candidate A **dominates** candidate B if A is superior or equal to B with respect to every criterion of evaluation and distinctly superior with respect to at least one criterion. Dominated designs need not be considered further - they may be filtered out. Among the designs that survive the dominance-filtering process, none is clearly superior to another. These designs are retained for further analysis.

To develop an intuition about what the dominance filter does, let us consider a geometric representation. Figure 2 illustrates the situation when there are only two evaluation criteria C_1 and C_2. Larger values of C_1 and C_2 are more desirable. Let M_1 be a design candidate. If another design candidate M_2 falls in region R_1, it can be eliminated, since M_1 would be clearly superior to M_2 on both criteria. If it falls in Region R_4, M_1 can be eliminated since M_2 would best M_1 on both criteria. If M_2 falls in Regions R_2 or R_3, both M_1 and M_2 need to be retained for the next stage since neither would be superior to the other with respect to all criteria of evaluation. This 2-dimensional description of dominance filtering generalizes to situations where there are more than two criteria; similar behavior will occur in the appropriate multidimensional space.

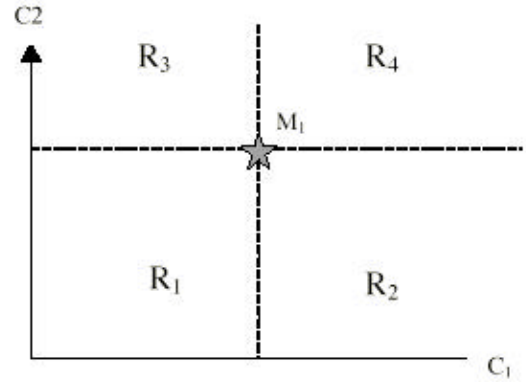

Figure 2. Illustration of dominance filtering.

Our algorithm for dominance filtering keeps a list of retained candidates, to which new candidates are compared. New candidates are compared serially to the items on the list. If the new candidate is dominated by some from the list, then there is no need to compare with the remaining elements in the list, and the new candidate is discarded, which contributes to efficiency. If the new candidate dominates a candidate in the retained list, the comparison still needs to continue with the other candidates in the list, since the new candidate may dominate some others as well. The final surviving set is independent of the order in which the candidates are generated and compared. Another probable source of efficiency in our implementation is that if a design on the list beats a new one, the dominator is moved to the front of the list. The idea is that this design is more likely to beat other new designs as well, in comparison with others on the list, and it will probably be efficient to consider it early when evaluating new candidates. We have not experimentally verified that moving designs up the list this way enhances efficiency, but it seems plausible and is implemented at little cost.

The Size of the Surviving Set. Let f be the fraction of the total design space that survives dominance pruning. If f is small, then the task of the designer is much eased. She only needs to consider a relatively small number of designs for further analysis while retaining assurance that the results speak about the entire design space. What kinds of values for f might one expect? This is an empirical question. We will describe experimental results which show that it is reasonable to expect small values of f in at least some real-world domains.

In a given domain, the value of f would, as a rule, increase as the number of evaluation criteria increases. That is, the probability that a design candidate is worse than another in all N dimensions of criticism will get smaller as N increases. However, f may become smaller as the size of the design space increases, say by considering parameter changes at finer resolutions. To see why, one can generalize from Figure 2 and consider each design to be a point in N-dimensional space, N being the number of criteria we are considering in dominance checking. The surviving set will be on the surface of the region of space wherein the designs lie. As more points (designs) are added within or near the region of designs in the N-

dimensional space, the total number of designs should increase at a faster rate than the number of designs on its surface. We will describe some experiments that explored how f tends to decrease as the number of designs increases.

Adding or Removing Evaluation Criteria. Suppose we have a surviving set of candidates after exploration and dominance filtering. Now suppose the problem specification changes, and one of the criteria is no longer relevant. Is there a simple way to construct the new surviving set from the previous one? If A dominates B in, say, N dimensions of criticism, of course A will still dominate B in N-1 dimensions. Thus, the elements of the surviving set for N dimensions will still dominate the previously pruned candidates in N-1 dimensions as well. Thus, the dominance algorithm simply needs to be re-run among the members of the N-dimension surviving set to compute the N-1 -dimension surviving set.

On the other hand, if the problem statement changes and a new criterion is added, the solution is not so simple. That A dominates B in N dimensions is no guarantee that A will still dominate B in N+1 dimensions. Maybe B happens to be better than A in the new dimension of criticism. This means that the new surviving set cannot be simply computed from the old surviving set. The dominance algorithm must again consider all the elements of the design space.

Independence of criteria. Dominance checking does not require evaluation criteria to be independent. In general, if the values according to two criteria are positively correlated, we can expect that use of these criteria will not increase the size of the surviving set as much as two independent criteria might. However, if values according to two criteria are negatively correlated (as they might well be when the designer is interested in investigating performance tradeoffs) we can expect the size of the surviving set to be larger than it would be if the criteria were independent.

Accuracy of Models. When we consider whether or not A is superior or equal to B with regard to every criterion, and superior in at least one, it is important to consider that the models upon which critics are based will have limited accuracy. The dominance filter can be adapted to take into account suspected model inaccuracy. A constant ε may be introduced for each critic such that A and B are considered to perform equally well with regard to the criterion of that critic if, as evaluated by the critic, the performance of the two differs by less than ε. Weakening the stringency of filtering in this way reduces the chances that a good design is mistakenly filtered out because of modeling inaccuracies. On the other hand, increasing ε for any critic will typically increase f, the surviving fraction. In choosing the value of ε for a particular critic, we express our estimate of the accuracy of the domain model used by that critic. Thus, choosing a value for ε gives a domain expert the opportunity to express a meta-knowledge judgment about the accuracy of the computer-based model.

Distributed Computing

Suppose that the design space has a million candidates, that we have five critics, and that each critic needs one tenth of a second for evaluating a candidate (all reasonable numbers). The total time on a serial machine for performing exhaustive evaluation will be about 6 days. Because of time requirements of this sort, we developed a novel computational architecture that employs Modula-3 (Cardelli, et. al., 1990) network objects (Birrell, et. al., 1994) to make use of the distributed computing environments that are presently commonly available in engineering institutions. A client-server architecture is employed that uses idle workstation time (in our case, often over 150 machines at a time) to allow the criticism of candidate designs to proceed in parallel.

The Exploration Interface

Even if dominance filtering is quite effective, one would expect hundreds if not thousands of designs to survive after exploring a design space consisting of, say, hundreds of thousands of design candidates. These would be designs where none would be clearly superior to another, based on the criteria that were used. What is a designer to make of so many designs, and how to help her narrow the choices? Clearly, additional knowledge is needed about choosing among designs, knowledge that was not incorporated during the search. If the additional knowledge were available in the form of new critics that were not used, the best way to proceed would be to do the search again using the new critics. (As we noted, the surviving set is quite sensitive to the addition of new critics.) Similarly, if the designer knew in advance how to weight the different criteria to form a composite evaluation, she might have used this composite evaluation to perform some form of optimization. However, we think that there is an important opportunity at the conceptual-design stage to develop a sense of the tradeoffs that arise from multiple criteria.

Our current visualization environment presents the designer with a set of tradeoff diagrams. Each diagram is a two-dimensional scatter plot, where the axes are a pair of design criteria, and all the surviving designs are plotted in that space. For example, if there are three critics with corresponding design criteria C_1, C_2, and C_3, up to three plots will be generated, one each for C_1-C_2, C_2-C_3, and C_3-C_1. Each design candidate will be represented in each of these plots. The designer can select any region in any of the plots and the design candidates that fall in the selected region in one diagram space will be highlighted in the other plots. The designer can thus easily observe how design candidates that look interesting in one of the diagrams fare with respect to other tradeoffs. Note that, in general, candidates that are contiguous in one of the plots will not necessarily be contiguous in the other plots.

Typically, a designer will explore the surviving designs by identifying regions or individual candidates that appear to have interesting properties in one diagram, selecting them, and examining their properties in the other diagrams. For example, the designer might note in one diagram that a small number of designs seem to have high evaluations in both dimensions. She might select this region. The interface would then highlight in the other diagrams the candidates in the selected region in the first

diagram. If their performance is satisfactory with respect to the other dimensions, she might mark that subset as worthy of further attention. A particularly important function of visual analysis is to see if there are regions in the diagrams where, for a relatively small sacrifice in performance in one dimension, a large gain is available in the other dimension. This sort of visual analysis gives the designer some understanding of the structure of the design space, and locates opportunities for favorable design tradeoffs.

One cannot anticipate all the forms of visual analyses that a designer might make using the tradeoff diagrams, nor imagine all of the kinds of insights that might be gained. In our experiments, we have found that the designers first note certain interesting properties in one region of one of the diagrams, think about what they noticed, and generate explanations for what struck them as interesting. At this point, they are able to make additional hypotheses about relationships in other diagrams and regions. In addition to identifying a small number of design candidates for further analysis, designers end up with a deeper understanding of the design space.

The number of possible tradeoff plots increases approximately with the square of the number of criteria. So, one must not demand that the designer consider all plots at once, but instead let the user choose which plots to see. Even if she never examines the results according to a particular criterion, it has still entered into dominance calculations. Dominance checking does not depend on the exploration environment. It is plausible that in many fields the number of criteria will not need to be very large.

While we are currently experimenting with tradeoff diagrams, the issue of how to effectively present exploration results to the designer is wide open. Certainly users should be able to select a subset of designs and throw them into a special Examination Set (ExamSet) for closer inspection. We have implemented this kind of selection for rectangular regions on tradeoff plots; users would like the ability to use the mouse to lasso regions of arbitrary shape. The user might want to create more than one such ExamSet and might want to union, or intersect, or pull a subset from any of them. The user should be able to look at any available ExamSet through multiple, cross-updating tradeoff plots, and other inspectors. We also envision providing the user with a set of abstraction agents (Abstractors) able to automatically form certain interesting generalizations about the designs in any chosen ExamSet. One such Abstractor should gather statistics on the design choices that are represented in the ExamSet, and if any choice is represented by more than 50% of the designs, it should produce a comment of the form:

> X% of the designs in <ExamSet> are designs where <design choice>.

For example, “88% of the designs in ExamSet-1 are designs where engine = Volkswagen #2.” or “All of the designs in ExamSet-2 are designs where configuration = parallel-hybrid.” This should be very useful to the designer in understanding and exploring the implications of the search results.

Ways of displaying and interacting with data in higher dimensions would be worth investigating. (See, for example, Tufte, 1983, 1990.) Automated clustering algorithms could identify interesting properties in the higher-dimensional spaces (Jain and Dubes, 1988). Higher-dimensional spatial analysis algorithms might be applied to identify outliers (Yip and F. Zhao, 1996). Outliers tend to have interesting properties, good or bad, or they may point to broken domain models or buggy software. Conversely, the absence of outliers should give the user some confidence in the fidelity of domain models and reassure the user that no especially good design possibilities have been overlooked.

Experimental Results

We will now describe a set of experiments that we have conducted in a real-world domain. At this point, no especially surprising designs have been discovered using the architecture. However, users have reacted favorably and remarked that designers can use tools such as this to explore the design space on their own and not merely go by the intuitions and gut-feelings of whatever strongly opinionated and highly vocal people they happen to be working with. The goal in reporting these experiments is to show how the proposed architecture works in practice, to describe some of what we have learned about the technology, to demonstrate candidate generation and filtering, and to show how the user interface works for interactive exploration. We also wish to draw attention to the significantly large design-space sizes the technology can already handle.

The Domain. The domain in which we performed our experiments was that of hybrid electric vehicles. Hybrid electric vehicles are automobiles that use both an electrical motor and an internal-combustion (IC) engine as power sources. (See Wouk, 1997, for an introduction to the problem area.) Electrical motors are attractive as power source partially because of the potential for using the motor as a generator during braking and thus recapturing the kinetic energy of the vehicle. They also have good emission properties, making them attractive for city use. On the other hand, they have limited range because of the limitations of current battery technology. Hybrid vehicles use an IC engine to extend range, either to move the vehicle when battery power is low, or to recharge the batteries. An interesting issue in the design of hybrid vehicles is the control policy, which is the formula that decides when to use which source of power for movement and when to activate the IC to charge the batteries.

The domain has a number of attractive features for exercising our architecture. The underlying design space is not simply generated by parametric variations; for example, there are four distinct vehicle configurations. There are also several criteria for evaluating performance, and no simple way to combine them a priori into a weighted composite objective function. Moreover, there are well-defined mathematical models for the various design criticisms of interest. With several alternative components to be considered for each component choice point within a configuration alternative, the design space

grows combinatorially and rapidly. When we include consideration of alternative control-policy parameters, the combinatorial explosion of design alternatives is even more dramatic. Modeling of hybrid-electric vehicles is described in Baumann, et. al. (1988); details of the domain models used in our experiments are available in Wasacz (1997) and Li (1998).

The most basic components are the four vehicle configurations, or types of vehicles: IC engine only; Electric motor only; Parallel hybrid; Serial hybrid. Once a commitment has been made to a particular configuration, there is a further need to specify the set of components required by that configuration, and parameter values associated with the various components. This set is different for each of the four types, although some of the components are common to each type. First, we describe components and parameters common to all of the vehicle types.

Transmission, of which the available types are: Automatic, Manual or CVT (continuously variable). For manual transmissions there is a choice of the number of gears (3, 4, or 5-speed) and the corresponding gear ratios. Choice of Shift Speeds for manual transmissions determines various performance characteristics. Also: Downshift Speed, which is the vehicle speed at which a lower gear ratio is chosen for manual and automatic transmissions.

The following components and parameters must be specified for some vehicle types but not others:

Electric motors in different sizes (small, medium and large) and their respective weights,

IC engines varied by their sizes (small, medium or large) and their corresponding weights, torque-speed characteristics and fuel-efficiency maps,

Batteries, which include the number of cells in the pack with their respective weights,

Speed reduction ratio, which refers to the gear ratio between the electric motor and the axle and,

Control policy, which is applicable only to hybrid vehicles and is varied in terms of four parameters, the high and low values of vehicle velocities and high and low values of the state of charge at which the switch is made from primary reliance on one engine to primary reliance on another.

The user supplies a set of plausible choices for each of these components, and the Good-Design Seeker explores them all. Candidates are generated by first selecting a vehicle configuration from the four choices, then systematically going through all relevant combinations of component choices and parameter-value choices. Constraints are applied at this stage, as described earlier. The design space size can be made to grow very large by stepping component changes through very small increments, e.g., small changes in cutoff speeds or gear ratios. The reader will note that we were careful in our experiments not to artificially inflate the sizes of the spaces to be explored by using unreasonably small steps of variation. Once interesting regions of space are explored during a first round of processing, finer distinctions may be made in selected areas of the space and additional explorations, similar to the first but at greater resolution, may be undertaken.

Design Critics. The following design critics were employed in the experiments we report here: Maximum acceleration, top speed, city-driving efficiency, and highway-driving efficiency. Maximum acceleration was calculated as the time to reach 60 MPH. We also included experimental city-driving and highway-driving range critics, but did not use them in dominance checking. The details of how the vehicles were modeled for computing these various performance characteristics are beyond the scope of this paper. Briefly, dynamical models of vehicle performance in the form of differential equations were constructed using well-understood vehicle-modeling methods. The equations were simulated temporally, i.e., the values of the operating parameters were determined as a function of time using appropriately chosen time increments. City and highway driving models imposed contours of desired velocities that were incorporated into the simulations. The models were simulated using Matlab and Simulink, widely available application packages, especially appropriate for simulating complex systems governed by differential equations. Of course, city and highway efficiency are correlated, and these are somewhat inversely correlated with both top speed and maximum acceleration.

Dominance Filtering. So far, we have run a series of experiments, progressively exploring larger spaces. In Table 1, we present results for our largest run to date (made in March 1998), and "fake" smaller runs constructed from random sampling of the results of this run. This insures that our investigation of dominance filtering is not skewed by recent improvements in the simulation models used by the critics. This experiment used the four critics mentioned in the previous section.

Experiment	Designs Considered	Survivors	Percentage survivors
A	1,798	71	3.949
B	17,711	173	0.977
C	179,874	556	0.309
D	1,796,025	1,078	0.060

TABLE 1. Effectiveness of dominance filtering as the size of the space increases.

Note that dominance filtering appears to be quite effective in eliminating a large fraction of the design space. In this case, a small design space was cut by well more than an order of magnitude, and the largest was cut by more than three orders of magnitude. If these results are representative of what can be expected, dominance filtering will be a practical way to help reduce the complexity of explicitly comparing very large numbers of design alternatives.

In order to investigate how the surviving fraction behaves with respect to the number of dimensions of comparative criticism, the dominance algorithm was run on a set of 17,711 candidates corresponding to Experiment

B of Table 1 using subsets of two and three criteria each. Table 2 summarizes the results for two criteria each, while Table 3 does the same for three criteria. The entries in these tables represent the numbers of surviving designs. In Table 2, the rows and columns indicate the two criteria. (Since the situation is symmetric, only the top half of the table is filled.) In Table 3, the column heading indicates which one of the four criteria is not used.

	MPG (City)	Max Accel.	Top Speed
MPG (Highway)	2	24	18
MPG (City)		36	18
Max Accel.			10

TABLE 2. Effectiveness of dominance filtering for two criteria. Number of surviving designs is shown

Not used:	MPG (Highway)	MPG (City)	Max Accel.	Top Speed
	103	83	35	76

TABLE 3. Effectiveness of dominance filtering for three criteria. Number of surviving designs is shown.

The data in both tables, and the fact that 173 designs survived with four criteria, support the hypothesis that, as the number of criteria increases, the tendency is for the effectiveness of dominance filtering to decrease. That is, a larger percentage of the candidates survive. However, we have observed that sometimes the size of the surviving set decreases as criteria are added. This presumably occurs because designs that were regarded as equally valued in the original set are now regarded as differently valued because the extra criterion allows some to dominate others.

Scale of computation. In our largest experiment (Experiment D in Table 1), 2,152,698 designs were evaluated, of which 1,796,025 were fully specified. Fully specified designs were evaluated according to four performance criteria, which required multiple simulations of each design. Dominance filtering reduced the number to 1,078 best designs for human analysis. 207 workstations were used as clients during the experiment; from zero to 159 were running at any one time. The experiment used 164 hr. 41 min. of wall-clock time, 14 hr. 54 min. of CPU time on the server, for generation and evaluation, and approximately 4.5 hr. of wall-clock time for dominance filtering performed as serial post processing. We note that our experiment used idle workstation in a university computer science department during the last week of classes of the term. Presumably, the wall-clock time would have been less at any other time. Approximately half of the computation was accomplished over the weekend.

Exploration Interface. Figure 3 shows one of the six tradeoff diagrams from Experiment D obtained by considering the four criteria two at a time. In each diagram, each point that is plotted corresponds to a design that survived the strict dominance test. The user can use the pointing device to select subsets for further examination, save and load such subsets and further narrow them.

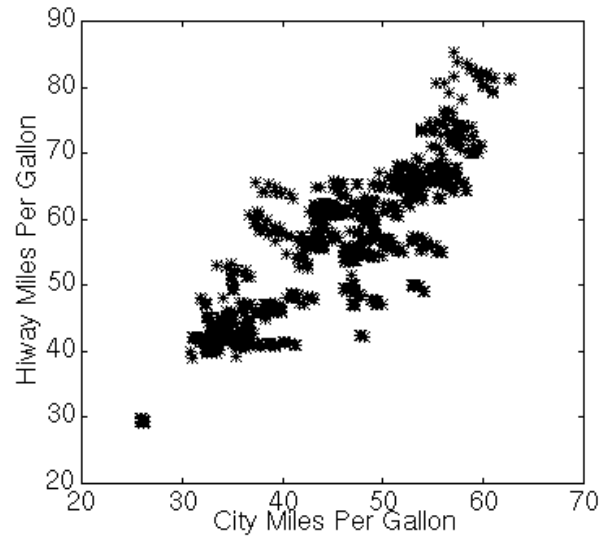

Figure 3. One of the tradeoff diagrams from Experiment D

Through the series of experiments, a striking phenomenon was the usefulness of the visualization interface for debugging domain models and ensuring sufficient realism. The diagrams quickly revealed designs with implausible performance characteristics; for example, at one stage of model development we saw vehicles with unreasonably high top speeds. On one occasion, it was noticed that for several designs, the values of city driving efficiency were higher than those for highway driving efficiency, which is quite unreasonable. On another occasion, stratification of points in tradeoff diagrams alerted us that time steps for the acceleration simulations had been set too coarsely. It is clear that visual exploration of this sort helps to ensure model accuracy. The tradeoff diagrams show the results of exercising the models over an extremely broad range of combinations of inputs. Model inaccuracies are given every opportunity to betray their presence through anomalies that appear in visualizations of the results.

Discussion

"Exploration" is a term that has been often used in the design literature, but with no single meaning. For example, for Navinchandra (1991), exploration consists of deviating "... from the beaten path to generate unconventional solutions to problems." Sometimes exploration is contrasted with search: exploration is supposed to be examining a space without any specific goal, while search is supposed to be examining a space for a specific object. We use the term "exploration" in the sense of examining a region thoroughly to develop an understanding of it.

In this paper, we have suggested that large-scale exploration can make a significant contribution at the

stage of conceptual design. We have presented a software architecture for large-scale exploration. The architecture is comprised of: a Good-Design Seeker, which systematically generates and evaluates large numbers of candidate designs using multiple criteria, filters that pass only superior designs, and a visualization environment that enables designers to investigate tradeoffs among the superior designs and select subsets for further analysis. The Good-Design Seeker may use component substitution and exhaustive methods similar in spirit to drug discovery by "combinatorial chemistry" (Economist 1998). Filtering based on dominance is practical to implement and promises to reduce the number of alternatives to be considered from vast to manageable; it has been remarkably effective in a small number of relatively realistic experiments in one domain. The visualization environment presents the user with the results of multi-criterial evaluation, without forcing the evaluation to a single criterion. Tradeoffs are displayed and the user is able to bring human evaluation and judgment to bear in choosing designs for further investigation.

Additionally, we have demonstrated at least one way to make the demanding computations practical: distribute the criticism of designs over a network of workstations (potentially as large a pool as there are friendly hosts on the Internet). We conjecture that very large-scale exploration of the space of design alternatives has been so computationally demanding that even the possibility of it has remained outside the thought processes of design practice, even within the computer-supported design community. We believe that computing technology today has advanced to where such large-scale explorations are practical for many design domains. We have presented experimental results that lend strong early encouragement to claims of practicality.

Acknowledgments

This material is based upon work supported by The Office of Naval Research under Grant No. N00014-96-1-0701. The support of ONR and the DARPA RaDEO program are gratefully acknowledged. Standard disclaimers apply.

References

Baumann, B., Rizzoni, G., Washington, G.: 1998, Society of Automotive Engineers Technical paper 981061, SAE Special Publication 1356, Electronic Engine Controls 1998: Sensors, Actuators, and Development Tools.

Birrell, A., Nelson, G., Owicki, S., and Wobber, E.: 1994, Network Objects, DEC Systems Research Center.

Cardelli, L., Donahue, J., Glassman, L., Jordan, M., Kalsow, B., and Nelson, G.: 1990, Modula-3 Report (revised), Nov 89, including a final update insert, Twelve Changes to Modula-3 dated Dec 1990, DEC Systems Research Center.

Chandrasekaran, B.: 1990, Design Problem Solving: A task analysis, *AI Magazine*, **11**(4), 59-71.

Economist 1998, Combinatorial Chemistry, *The Economist*, March 14th 1998.

Jain, A., Dubes, R.: 1988, *Algorithms for clustering data*, Prentice Hall, Englewood Cliffs, NJ.

Li, Qingyuan: (1998) "Development And Refinement of a Hybrid Electric Vehicle Simulator and Its Application in Design Space Exploration," M.S. Thesis, Department of Mechanical Engineering, the Ohio State University.

Navinchandra, D.: 1991 *Exploration and Innovation in Design: Towards a Computational Model*, Springer-Verlag, New York.

Newell, A.: 1980, Reasoning, Problem Solving and Decision Process: The problem space as a fundamental category, in *Attention and Performance VIII*, R. Nickerson (ed.), Lawrence Erlbaum, Hillsdale, NJ

Nilsson, N.: 1971, *Problem- Solving Methods in Artificial Intelligence*, McGraw Hill, New York.

Tufte, E.: 1983, *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Conn.

Tufte, E.: 1990, *Envisioning Information*, Graphics Press, Cheshire, Conn.

Wasacz, B.: 1997, Development and Application of A Hybrid Electric Vehicle Simulator, MS thesis, Department of Mechanical Engineering, The Ohio State University

Wilde, D. J.: 1978, *Globally Optimal Design*, Wiley, New York.

Wouk, V.: 1997, Hybrid Electric Vehicles, *Scientific American*, October 1997, 70-74.

Yip, K. and Zhao, F.: 1996, Spatial aggregation: theory and applications, *Journal of Artificial Intelligence Research*, 5:1-26.