Abstraction and Decomposition in Hillclimbing Design Optimization

Thomas Ellman
Mark Schwabacher
Department of Computer Science
Hill Center for Mathematical Sciences
Rutgers University
New Brunswick, New Jersey 08903
(908) 932 - 4184
{ellman,schwabac}@cs.rutgers.edu

CAP-TR-14

Abstract

The performance of hillclimbing design optimization can be improved by abstraction and decomposition of the design space. Methods for automatically finding and exploiting such abstractions and decompositions are presented in this paper. A technique called "Operator Importance Analysis" finds useful abstractions. It does so by determining which of a given set of operators are the most important for a given class of design problems. Hillclimbing search runs faster when performed using this this smaller set of operators. A technique called "Operator Interaction Analysis" finds useful decompositions. It does so by measuring the pairwise interaction between operators. It uses such measurements to form an ordered partition of the operator set. This partition can then be used in a "hierarchic" hillclimbing algorithm which runs faster than ordinary hillclimbing with an unstructured operator set. We have implemented both techniques and tested them in the domain of racing yacht hull design. Our experimental results show that these two methods can produce substantial speedups with little or no loss in quality of the resulting designs.

Content Area: Automated Reasoning, Search, Reasoning about physical systems, Knowledge Acquisition, Machine Learning.

1 Introduction

Parameter design problems can be difficult to solve when the design space includes a large number of variables. Design problems become particularly difficult when design goals depend formally on all parameters, and when the nature of the dependence is complex. For example, in the design of ship hulls, the hydrodynamic performance of the ship depends, in principle, on all aspects of the hull geometry. Furthermore, the dependency is difficult to analyze by any method short of numerically solving the relevant hydrodynamic fluid equations. In design problems such as this, there is no obvious way to decide which parameters are most important. Nor is there an obvious way to decompose the design problem into independent subproblems. Computational costs of design optimization can therefore become quite expensive.

Techniques for automatically simplifying such problems are presented in this paper. The techniques use computational experiments to find useful abstractions and decompositions of parameter design problems. A technique called "Operator Importance Analysis" is used to find abstractions, by determining which parameters are most important to the design problem at hand. A technique called "Operator Interaction Analysis" is used to find decompositions, by determining which parameters interact with each other to a significant degree. Both techniques produce information that can be used to speed up the performance of a hillclimbing design optimization algorithm, with little or no penalty in the quality of the resulting artifact. Our operator analysis techniques have been developed as part of the "Design Associate", a system assisting human experts in the design of complex physical engineering structures [Ellman *et al.*, 1992].

2 Why is this AI?

Artificial Intelligence has been defined as the automation of tasks that are considered to require "intelligence", when performed by humans [Winston, 1984, Simon, 1981]. Abstraction and decomposition are techniques that are often used by human designers. For example, human yacht designer know which features of a yacht's hull geometry are most important to optimize, and which should be "abstracted away", i.e., ignored, and given default values. Likewise human designers know which geometric features have the potential to interact with each other, and must therefore be optimized jointly, and which can be "decomposed", i.e., optimized separately. Thus, to the extent that the Operator Importance and Interaction Analyses do successfully perform this type of abstraction and decomposition, they may be said to automate a type of intelligent human behavior. These methods may also be seen as attacking a standard machine learning problem: Acquiring knowledge that can improve the performance of a problem solver. In particular, Operator Analysis acquires abstraction and decomposition knowledge that speeds up the performance of hillclimbing search algorithms. It thus automates a task that is analogous to tasks performed by some well known methods of speedup learning, such as problem-reformulation [Amarel, 1968], and explanation-based learning, [Mitchell *et al.*, 1986, Laird *et al.*, 1987].

3 Related Work

Several investigators have previously applied AI techniques to parameter design optimization: Methods of intelligently adapting the parameters of hillclimbing search during a single

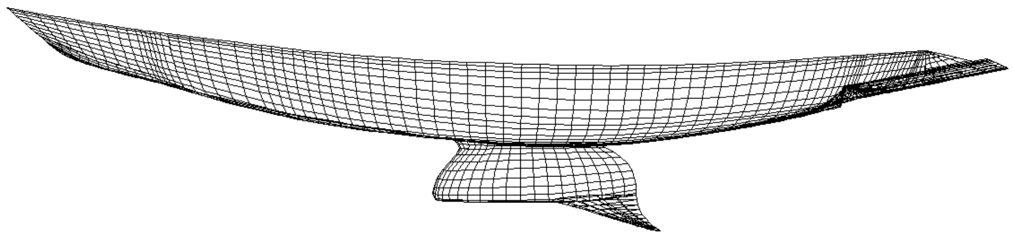

Figure 1: The Stars and Stripes '87

problem solving session are reported in [Orelup *et al.*, 1988]. In contrast to this, our techniques find abstractions and decompositions in advance of problem solving. Furthermore, the abstractions and decompositions are applied to many problem solving sessions. Methods of decomposition that are analogous to, but technically different from ours are discussed in [Sobieszczanski-Sobiesky, 1982] and [Rogers, 1989]. Engineous, a parameter design system discussed in [Tong, 1988], has a Parameter Knowledge Base which contains information on parameter interactions provided by human experts. We present methods to obtain similar interaction data automatically through numerical experiments. A survey of AI methods in Mechanical Engineering Design, which includes a section on feature extraction methods, is found in [Finger and Dixon, 1989]. Our "Operator Importance" analysis can be seen as a feature extraction method.

4 Yacht Hull Design: A Testbed Domain

Our abstraction and decomposition methods are being developed and tested in the domain of 12-meter racing yachts, a class of yachts that once raced in the America's Cup competition. An example of a 12-meter yacht, the Stars and Stripes '87, is shown in Figure 1. [1] Racing yachts can be designed to meet a variety of objectives. Possible yacht design goals include: *Course Time Goals*, *Rating Goals* and *Cost Goals*. In our research we have chosen to focus on a course time goal, i.e., minimizing the time it takes for a yacht to traverse a given race course under given wind conditions. Thus $Course - Time$ is conceptually a function of three things: (1) The yacht geometry, represented as a set of B-Spline Surfaces; (2) The race course, represented as a set of $(distance, heading)$ pairs; and the wind speed, represented as a scalar number, in knots. Our system evaluates $Course - Time$ using a "Velocity Prediction Program", called "VPP" [Letcher, 1991].

A yacht geometry is represented by a set of B-Spline surfaces [Rogers and Adams, 1990]. Yacht designs are modified by operators that manipulate the B-Spline surfaces. Each surface is represented by a matrix of control point 3-vectors. Shape modification operators implement transformations that include rotation, translation and scaling. Each operator may transform an entire matrix (i.e., an entire surface) or only a sub-matrix (i.e., a portion of the surface). Weighting functions are used to smoothly vary the effect of an operator over the surface of a yacht hull. Finally, each operator takes a *magnitude parameter* that controls the strength of the operator. A parameter value of zero has no effect. Values of α and $-\alpha$ have opposite effects.

[1]This yacht figures prominently in the history of America's Cup racing. In particular, it won the America's cup back from Australia in 1987 [Letcher *et al.*, 1987].

- Global Scaling Operators: *Scale-X*, *Scale-Y* and *Scale-Z* change the overall dimensions of a racing yacht, by uniformly scaling all surfaces.
- Prismatic Coefficient Operators: *Prism-X*, *Prism-Y* and *Prism-Z* make a yacht's canoe-body more or less streamlined, when viewed along the X, Y and Z axies respectively.
- Keel Operators: *Scale-Keel* and *Invert-Keel* change the depth and taper ratio of the keel respectively.

Figure 2: Yacht Modification Operators

1. Let $\{S_1, ..., S_n\}$ be a family of operator sets.
2. Let $\{e_1, ..., e_m\}$ be a range of step sizes.
3. For $(i = 1 \ldots n)$ do: For $(j = 1 \ldots m)$ do *Hillclimbing-Search*(S_i, e_i).

Figure 3: Hierarchic Hillclimbing

A parameterized search space is thus specified by providing an initial prototype geometry and a set of operators for modifying that prototype. Some shape modification operators implemented in the Design Associate are described in Figure 2. Our current set of shape modification operators was obtained by asking human domain experts for an exhaustive list of all features of a yacht's shape that might be relevant to the racing performance of a yacht. The operators in Figure 2 modify a subset of the relevant features, which we have adopted for the purpose of testing our abstraction and decomposition methods.

5 Hierarchic Hillclimbing for Design Optimization

Hillclimbing search is useful for attacking design optimization when the number of parameters is so large that exhaustive search methods are not practical. Our system uses steepest-descent as our basic hillclimbing method [Press *et al.*, 1986]. The steepest-descent algorithm operates by repeatedly computing the gradient of the evaluation function. (In the yacht domain, this requires computing the partial derivatives of $Course - Time$ with respect to each operator parameter.) The algorithm then takes a step in the direction of the gradient, and evaluates the resulting point. If the new point is better than the old one, the new point becomes the current one, and the algorithm iterates. The algorithm terminates if the gradient is zero, or if a step in the direction of the gradient fails to improve the design. We have modified this basic algorithm in two principal respects: First, we arrange for the algorithm to use a range of different step sizes, starting with large steps and proceeding to small step sizes when large ones fail to improve the design. Second, we partition the operators into an ordered collection of subsets. Each subset of operators is used separately in a steepest-descent hillclimbing search. Hillclimbing runs faster when the operators are so partitioned. Although search paths tend to be longer, the algorithm saves considerable time in computing gradients, which makes up for the cost of the longer path. The "Hierarchic Hillclimbing" algorithm is shown in Figure 3.

A number of additional enhancements to the hillclimbing algorithm have been adopted to deal with some practical difficulties arising in the yacht design domain. The program we use to compute $course - time$ (VPP) is a commercial software product. Nevertheless, it suffers from a number of deficiencies that make hillclimbing difficult. For example, it may return a spurious root of the balance of force equations that it solves. It may also exhibit discontinuities, due to numerical round-off error, or due to discretization of the (theoretically) continuous yacht hull surface. These deficiencies can produce "noise" in the evaluation function surface over which the hillclimbing algorithm is moving. The algorithm can easily get stuck at a point that appears to be a local optimum, but is nevertheless not locally optimal in terms of the true physics of the yacht design space. To overcome these difficulties, we have endowed the hillclimbing algorithm with the ability to climb over hills of limited height and limited width. The resulting algorithm is more robust than the original algorithm; however, it still does not always reach a true local optimum.

6 Abstraction and Decomposition

Abstraction and decomposition techniques have a long history in Artificial Intelligence. Abstraction techniques for planning and theorem proving go back as far as [Sacerdoti, 1974]. The importance of decomposition in problem solving was discussed at length in Simon's classic work on AI and Design, [Simon, 1981]. Methods of automatically finding good abstractions and decompositions have more recently received considerable attention, as manifested in a series of workshops on the subject [Ellman, 1990], [Ellman, 1992]. Nevertheless, until now, little attention has been given to these issues in the context of parameter design tasks.

Methods of automatically finding abstractions and decompositions for parameter design problems are discussed in this section. The abstraction and decomposition problems can be understood in terms of the data needed to instantiate the "Hierarchic Hillclimbing Algorithm" described in Figure 3. To begin with, the system must decide which shape modification operators are worth using at all for the current design problem. "Operator Importance" analysis is used to select a good subset of operators. The selected operator subset may be said to induce an abstraction of the design space. The abstract space includes those shape features manipulated by the selected operators, and omits the shape features manipulated only by unselected operators. In addition, the system must partition the operators into subsets, and find a suitable ordering of the subsets. "Operator Interaction Analysis" is used to find such an ordered partition of the selected operators.

6.1 Abstraction via Operator Importance Analysis

Operator Importance Analysis proceeds from the following observation about the factors that determine when a particular operator will be useful for a given design problem: In the Design Associate, a shape modification operator is used to modify an initial prototype P, which was previously designed to satisfy some goal G, into a new design P' that satisfies a new goal G'. We conjecture that an operator will be more or less useful for this purpose depending on the manner in which the goals G and G' differ from each other. For example, suppose that G and G' are $CourseTime$ goals that refer to the same courses, but with different expected wind speeds. Human experts believe that large yachts are suited for high wind speeds while small yachts are suited to low wind speeds. We therefore expect that global scaling operators such as $Scale\text{-}X$, $Scale\text{-}Y$ and $Scale\text{-}Z$ will be especially important in modifying a design for a new wind speed. In contrast, suppose that G and G' involve the same wind speeds, but different courses. In this case, we would expect a different set of operators to be most important. In

For each operator OP and goal dimension D do:

1. Start with P_0 optimized for goal G.
2. Perturb goal G along dimension D.
3. Optimize P_0 to P_1 using operator OP.
4. Measure percentage improvement from P_0 to P_1 .
5. Record $Importance(OP, D)$.

Figure 4: Operator Importance Algorithm

general, we expect the importance of operators to depend on the difference between the goal G for which the prototype P was designed, and the new design goal G'.

The Operator Importance Algorithm is outlined in Figure 4. This procedure assumes that the set of all possible goals is parameterized by some set of "goal dimensions" along which two goals can differ from each other. For example, in the yacht design domain, goals can be parameterized by quantities such as wind-speed as well as D_{up}/D_{across} and D_{down}/D_{across} (the portions of the course that go up wind and down wind), among others. For each operator OP and each goal dimension D the operator importance algorithm builds a table $Importance(OP, D)$. This table records the degree to which operator OP will be useful if the initial prototype was designed for goal G, the new design goal is G', and the two goals G and G' differ along along dimension D.

6.2 Decomposition via Operator Interaction Analysis

Operator Interaction Analysis proceeds from the following idea about the manner in which operators interact with each other. Two operators O_1 and O_2 are said to interact with each other if the optimal parameter-value of one depends on the parameter-value of the other. For example, suppose the optimal value of O_2 depends on the value of O_1. In this case, the operator O_1 should not appear later than O_2 in the ordered partition of operator sets used in Hierarchic Hillclimbing. Likewise, if the optimal value of O_1 depends on the value of O_2, then the operator O_2 should not appear later than O_1 in the ordered partition. If each optimal value depends on the other, the operators belong in the same operator subset, so that they will be used together during Hierarchic Hillclimbing.

The Operator Interaction Algorithm is described in Figure 5. This procedure builds a table $Interaction(OP_i, OP_j)$ that measures the degree of interaction between each pair of operators. Notice that interaction is not necessarily a symmetric relationship. Thus $Interaction(OP_i, OP_j)$ is not necessarily equal to $Interaction(OP_j, OP_i)$. Once this table is constructed, a numerical threshold is used to determine which interactions are considered significant. Application of the threshold converts the numerical interaction matrix into a boolean matrix representing a directed graph of interactions. A partition of the operators is obtained by finding the strongly connected components of the graph, and forming an operator subset for each component. The partition is then partially ordered in the following way: Operator subset S_i precedes subset S_j whenever there is a path from an operator in S_i to an operator in S_j. The resulting ordered partition can then be used in the Hierarchic Hillclimbing algorithm described in Figure 3.

For each pair of operators OP_i and OP_j do:

1. Let P_0 be an initial prototype.
2. Optimize P_0 to P_1 using operator OP_i.
3. Optimize P_1 to P_2 using operator OP_j.
4. Optimize P_2 to P_3 using operator OP_i.
5. Measure percent improvement from P_2 to P_3.
6. Record $Interaction(OP_i, OP_j)$.

Figure 5: Operator Interaction Algorithm

- T_0 = Course time with no optimization.
- T = Course time using ordinary hillclimbing.
- T' = Course time using abstraction or decomposition.
- N = Number of VPP runs using ordinary hillclimbing.
- N' = Number of VPP runs using abstraction or decomposition.
- $Quality = \frac{T_0 - T'}{T_0 - T}$
- $Cost = \frac{N'}{N}$

Figure 6: Quality and Cost Definitions

7 Experimental Results

Abstraction and decomposition are expected to implement a tradeoff between the computational *cost* of designing an artifact, and the *quality* of the artifact. In order to measure this tradeoff, we compare the performance of Hierarchic Hillclimbing (using a reduced set of operators partitioned into several subsets) to the performance of an ordinary hillclimbing algorithm (using all the available operators in a single operator set). We expect ordinary hillclimbing to obtain higher quality designs than result from Hierarchic Hillclimbing; however, we also expect ordinary hillclimbing to incur higher computational costs. Precise definitions of these quality and cost measures are shown in Figure 6.

7.1 Experimental Results of Operator Importance

Measurements of operator importance levels were obtained for each of the eight operators described in Figure 2. All the measurements involved starting with a variant of the "Stars and Stripes '87" yacht that was optimized for the "America's Cup" race course and a wind speed of 10 knots. One set of importance levels was obtained by changing the target wind speed (from 10 to 16 knots) and re-optimizing. Another set of importance levels was obtained by changing the race course (from America's Cup to a mostly upwind course) and re-optimizing. The resulting importance tables are shown in Figure 7.

We then ran a series of experiments to demonstrate the tradeoff between the cost and quality that may be attained by exploiting the operator importance data. In particular, we measured the quality and cost levels obtained using the "best" 2, 4 and 6 operators, as predicted by the Operator Importance Studies. The quality and cost levels were measured using the same test problems as were used to collect the importance data. The resulting

Change Windspeed	
Operator	Importance
Scale-Z	2.4020%
Scale-Keel	1.9788%
Scale-X	1.7181%
Scale-Y	1.6189%
Prism-Y	1.1853%
Prism-Z	0.7560%
Invert-Keel	0.1633%
Prism-X	0.0000%

Change Racecourse	
Operator	Importance
Scale-Keel	3.2057%
Scale-Z	2.7546%
Scale-X	0.6342%
Scale-Y	0.4290%
Prism-Y	0.1917%
Prism-Z	0.0117%
Prism-X	0.0001%
Invert-Keel	0.0000%

Figure 7: Operator Importance Tables

tradeoffs are illustrated in Figure 8. On the whole, the graphs show a monotonic increase in both quality and cost, as more and more operators are included; however, in some cases, the increase is non-monotonic. Circles on the graph indicate cases in which we believe the hillclimber got stuck at a false local optimum. If we exclude these cases, the curves are monotonic, and we see the tradeoff that we hypothesized. [2]

We also ran experiments to determine how the operator subsets chosen by the Operator Importance Studies compare with other subsets of operators. In particular, we ran the hill-climber using every subset of 2, 4, and 6 operators, on each of the two test problems for which we collected the importance data. We measured the performance of the average sets of 2, 4 and 6 operators. We also noted the performance of the "true" best 2, 4, and 6 operators. Figure 9 compares the performance of the operators that were predicted to be best (by Operator Importance Analysis) with the true best operators, the average case operators, and the predicted worst operators. Numbers in italics in this table (and all subsequent tables) indicate that we have evidence that the hillclimber got stuck at a false local optimum. Notice that the predicted best operators were not the true best in any of the cases; however, the predicted best were always better than average (except in the one case where the hillclimber got stuck).

We conjecture the following explanation of why Operator Importance Studies did not find the true best sets of operators. Operator Importance measures operators one at time. This ignores the possibility of synergistic interactions between operators, i.e., when two operators enhance each other's value, and redundancies, i.e., when two operators have nearly the same effect. For example, we believe there is a synergy between Invert-Keel and Scale-Keel. Invert-Keel had an importance of almost zero when used by itself, but Scale-Keel and Invert-Keel together turned out to be the most important pair of operators. On the other hand, we believe Scale-Z and Scale-Keel are somewhat redundant, since they both change the depth of the keel.

We also ran experiments to determine how well Operator Importance data transfers to test problems other than those for which we collected the importance data. In particular, we tested the generality of the operator importance data shown in the left side of Figure 7, i.e., the importance levels for changes in wind speed. This data was collected for a change in wind speed from 10 knots to 16 knots. We tested whether it would apply well to problems in which the target wind speed changes from 10 knots to values other than 16 knots. The performance of the "predicted best" and "predicted worst" 2, 4, and 6 operators on these

[2]Notice that in some cases, there are quality numbers greater than 100%. These indicate that a subset of the eight operators produced a higher quality boat than all eight operators. This suggests the hillclimber got stuck at a false local optimum when using all 8 operators.

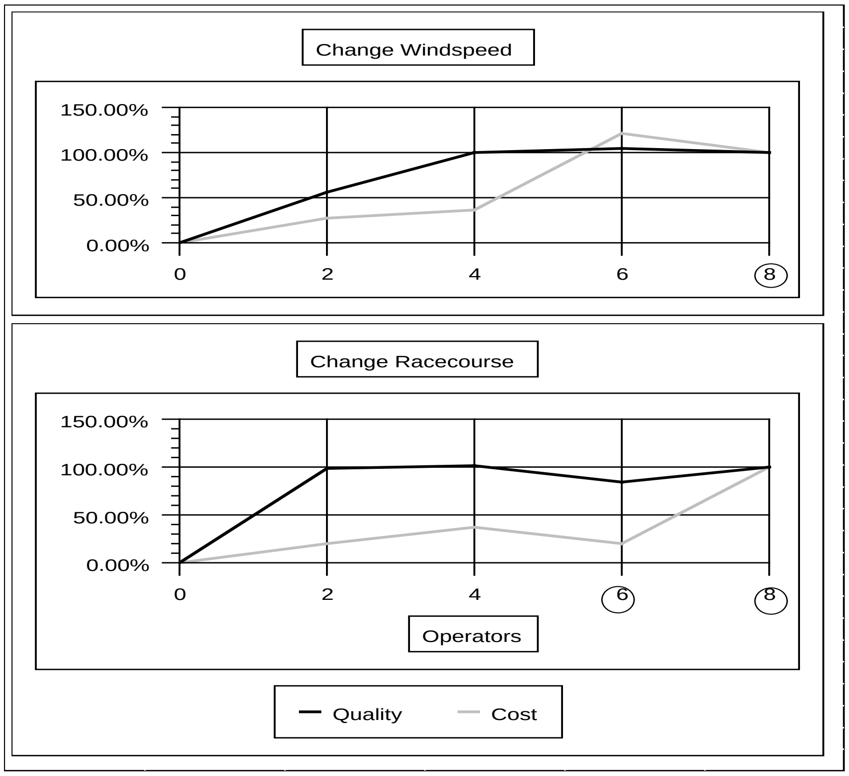

Figure 8: The Cost-Quality Tradeoff

additional problems is summarized in Figure 10. The word “Yes” indicates that the predicted best operator set achieved better quality designs than the predicted worst. The word “No” means that it did not. Notice that importance data generalizes well for target wind speeds close initial test problem, (near 16 knots), and less well for more distant wind speeds.

7.2 Experimental Results of Operator Interaction

Measurements of operator interaction levels were obtained for all pairs of the eight operators described in Figure 2. All the measurements involved starting with the “Stars and Stripes ’87” and optimizing it for a 10 knot wind and the “America’s Cup” race course. The resulting

		Change Windspeed	Change Racecourse
2 Operators	Predicted Worst	6.01%	0.00%
	Average	48.06%	50.89%
	Predicted Best	56.21%	98.82%
	True Best	84.14%	104.33%
4 Operators	Predicted Worst	5.91%	27.95%
	Average	78.08%	74.02%
	Predicted Best	99.61%	101.28%
	True Best	118.90%	105.60%
6 Operators	Predicted Worst	80.13%	29.53%
	Average	89.12%	86.46%
	Predicted Best	104.11%	*84.65%*
	True Best	105.56%	101.97%

Figure 9: Performance of Operators Selected by Importance Studies

		Wind Speed				
		16	14	12	8	6
Operators	2	Yes	Yes	Yes	No	No
	4	Yes	Yes	Yes	No	Yes
	6	Yes	Yes	Yes	*Yes*	No

Figure 10: Performance of operator importance study using other environments.

	SX	SY	SZ	PX	PY	PZ	SK	IK
Scale-X		0.03%		0.06%	0.04%	0.04%	0.06%	0.02%
Scale-Y				0.42%	0.16%	0.15%	0.28%	0.03%
Scale-Z								
Prism-X								
Prism-Y				0.70%				
Prism-Z		0.01%		0.73%	0.09%		0.22%	0.06%
Scale-keel				0.70%	0.30%			
Invert-keel				0.27%				

Figure 11: Operator Interaction Table

interactions are shown in Figure 11. A threshold of 0.01% was used to distinguish between significant interactions, and insignificant ones. (All insignificant entries were left blank.) The associated interaction graph is shown in Figure 12. The graph has the following 7 strongly connected components: $\{Prism - X\}$, $\{Prism - Y\}$, $\{Scale - Keel\}$, $\{Invert - Keel\}$, $\{Prism - Z, Scale - Y\}$, $\{Scale - Z\}$, and $\{Scale - X\}$. The order listed above is one of many total orderings of the operator subsets that is consistent with the interaction graph. It was chosen on the basis of weaker interactions (represented by dotted lines in the figure) which were used to constrain the location within the ordering of the operator $\{Scale - Z\}$. This ordering is the predicted "best" decomposition, according to the operator interaction data. In contrast, the data predicts that the "worst" decomposition would be obtained by reversing the order of these operator subsets, and also splitting the one doubleton set into two singleton subsets.

In order to test the value of Operator Interaction Studies, we compared performance of the predicted best decomposition and the predicted worst decomposition on several test problems. We first ran the hillclimber using the predicted best and worst decompositions, using a goal defined by the America's Cup Course (which is the course for which the data was collected), and averaged the results over five different target wind speeds: 6, 8, 10, 12, and 14 knots. We also ran the hillclimber using the same ("best" and "worst") decompositions, using a goal defined by a ten knot wind speed (which is the wind speed for which the data was collected), and averaged the results over two different race courses: the America's Cup Course, and an entirely upwind course. The results are shown in Figure 13. Thus the top row tests generality of the interaction data for changes in wind speed. The bottom row tests generality of the interaction data for changes in race course.

Notice that the predicted best decomposition produced a better quality boat, on average, than did the predicted worst decomposition. Further, since the numbers in this table are averaged over several design goals, one may conclude that interaction data generalizes to environments other than the one for which the data was collected. Notice also that the predicted "best" decomposition produces approximately the same quality as using no decomposition, at less than half the cost. It is also interesting to note that the predicted worst decomposition

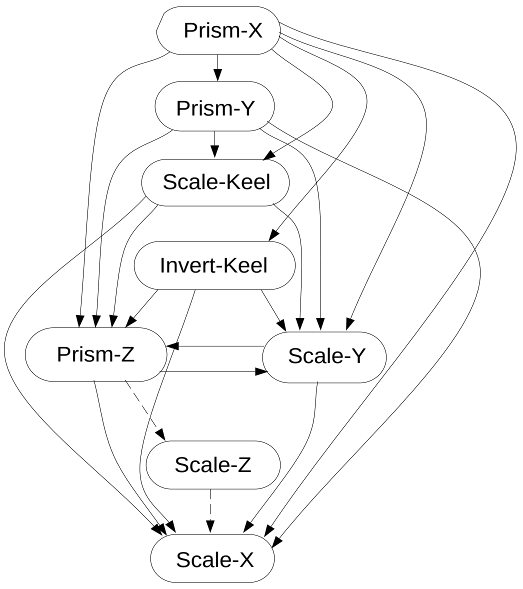

Figure 12: Operator Interaction Graph

	Decomposition	Average Quality	Average Cost
Change Wind Speed	Worst Decomp	**89.29%**	**9.23%**
	Best Decomp	**101.67%**	**46.71%**
Change Race Course	Worst Decomp	**92.63%**	**6.76%**
	Best Decomp	**102.22%**	**44.07%**

Figure 13: Performance of Decomposition

produces a boat with about 90% of the quality at less than 10% of the cost. The low cost of the predicted worst decomposition is explained by the fact that it is totally decomposed, i.e., there is only one operator on each operator subset.

7.3 Generalization of Operator Analysis Data

Data produced by Operator Importance and Interaction Analyses will be most useful if it can be applied to multiple design tasks. This raises the following generalization issue: The operator analyses are carried out using a set of benchmark design problems, i.e., a list of prototypes and a revised goal for each prototype. When the system encounters a new design problem, it must determine whether the new problem is sufficiently similar to the benchmark problems so that the operator analysis data can be safely applied. One approach would define a similarity metric on the parameter space of yacht shapes. Operator analysis data would be applied to any yacht geometry within some fixed distance of all or most of the benchmark prototypes. If this simple approach does not work, more sophisticated Machine Learning techniques may be necessary to properly generalize the results of Operator Importance and Operator Interaction analyses in order to apply them to a range of problems.

8 Evaluation

Our experimental results demonstrate the value of the abstractions and decompositions produced by Operator Importance and Interaction Studies. The abstractions and decompositions significantly reduced the cost of of hillclimbing, with little or no reduction in the quality of the resulting designs. Nevertheless, in order to be truly useful, the savings must justify the costs of obtaining the importance and interaction data themselves. Consider that an Operator Importance study typically requires about one third as many evaluations as an entire hillclimbing run using all the operators. Thus if an importance study can save one third the cost of hillclimbing, it will pay for itself. On the other hand, an Operator Interaction study typically requires as many evaluations as four hillclimbing runs using all the operators. Thus an interaction study is unlikely to be cost effective, unless its results are general enough to be amortized over multiple design problems. Our experiments suggest that the decompositions produced by the interaction study can indeed be so generalized.

9 Future Work

Our experimental results are limited in part by difficulties in getting the hillclimber to reach a true local optimum, in the presence of noise in the evaluation function. One approach is to apply or adapt our abstraction and decomposition techniques to work with other optimization methods that are possibly more robust than steepest-descent, e.g., downhill simplex [Press *et al.*, 1986]. Preliminary tests with the downhill simplex method suggest that it gets stuck just as easily as steepest-descent; however, we have some ideas for extending downhill simplex to make it still more robust. Another approach would be to test our methods in a domain that does not exhibit the problem of a noisy evaluation function.

Operator Importance studies might be improved by developing methods of detecting synergies and redundancies among operators. Unfortunately, a systematic study of all subsets of operators would be too computationally expensive to carry out in practice. Nevertheless, a greedy approach could be used to find useful sets of operators. This method would start by choosing the most useful single operator. At each iteration, it would measure the importance of each remaining operator, taken in combination with previously selected operators, adding the best to the previously selected set. The algorithm would terminate when additional operators do not significantly improve the results.

We used the result of the interaction study to partition the operators into subsets that can be optimized sequentially. The information provided by the interaction study also could be used to partition the operators into subsets that could be optimized in parallel on several processors. After all of the parallel optimizations were complete, the Design Associate would sum the parameters from the individual optimizations to produce a boat that would be approximately globally optimal.

The computational costs of finding good abstractions and decompositions might be diminished by developing methods that are more analytic in character, than our current set of purely empirical methods. For example, one might construct a simplified or qualitative model of the physics of the design domain. The model would be designed to predict some or all operator importance or operator interaction levels a priori, without recourse to expensive calls to an exact evaluation function. Such an analytic approach, or a combined analytic/empirical approach, might lead to the most cost effective means of finding good abstractions and decompositions.

10 Acknowledgments

The research reported in this paper has benefited from numerous discussions with all members of the Rutgers CAP project, including Saul Amarel, Andrew Gelsey, Haym Hirsh, John Keane, Chun Liew, Ringo Ling, Gerry Richter, Louis Steinberg, Chris Tong, Tim Weinrich and Ke-Thia Yao. This research was supported by the Defense Advanced Research Projects Agency (DARPA) and the National Aeronautics and Space Administration (NASA), under DARPA-funded NASA grant No. NAG 2-645.

References

[Amarel, 1968] S. Amarel. On representations of problems of reasoning about actions. In D. Michie, editor, *Machine Intelligence 3*, pages 131 – 171. American Elsevier, New York, NY, 1968.

[Ellman *et al.*, 1992] T. Ellman, J. Keane, and M. Schwabacher. The rutgers cap project design associate. Technical Report CAP-TR-7, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1992.

[Ellman, 1990] T. P. Ellman, editor. *Working Notes of the AAAI Workshop on Automatic Generation of Approximations and Abstractions.* Department of Computer Science, Rutgers University, New Brunswick, New Jersey, 1990.

[Ellman, 1992] T. P. Ellman, editor. *Working Notes of the AAAI Workshop on Approximation and Abstraction of Computational Theories.* Department of Computer Science, Rutgers University, New Brunswick, New Jersey, 1992.

[Finger and Dixon, 1989] S. Finger and J. R. Dixon. A review of research in mechanical engineering design. part ii: Representation, analysis, and design for the life cycle. *Reseach in Engineering Design*, 1(1):121 – 137, 1989.

[Laird *et al.*, 1987] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: Architecture for general intelligence. *Artificial Intelligence*, 33:1 – 64, 1987.

[Letcher *et al.*, 1987] J. S. Letcher, J. K. Marshall, J. C. Oliver, and N. Salvesen. Stars and stripes. *Scientific American*, 257(2), August 1987.

[Letcher, 1991] J. S. Letcher. *The Aero/Hydro VPP Manual.* Aerh/Hydro, Incorporated, Southwest Harbor, Maine, 1991.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based learning: A unifying view. *Machine Learning*, 1(1):47 – 80, 1986.

[Orelup *et al.*, 1988] M. F. Orelup, J. R. Dixon, P. R. Cohen, and M. K. Simmons. Dominic ii: Meta-level control in iterative redesign. In *Proceedings of the National Conference on Artificial Intelligence*, pages 25–30, St. Paul, MN, 1988. Morgan Kaufmann.

[Press *et al.*, 1986] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes.* Cambridge University Press, New York, New York, 1986.

[Rogers and Adams, 1990] David F. Rogers and J. Alan Adams. *Mathematical elements for computer graphics.* McGraw-Hill, 2nd edition, 1990.

[Rogers, 1989] J. L. Rogers. A knowledge-based tool for multilevel decomposition of a complex design problem. Technical Report NASA Technical Paper 2903, Langley Research Center, National Aeronautics and Space Administration, Hampton, VA, 1989.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115 – 135, 1974.

[Simon, 1981] H. A. Simon. *The Sciences of the Artificial.* MIT Press, Cambridge, MA, 1981.

[Sobieszczanski-Sobiesky, 1982] J. Sobieszczanski-Sobiesky. A linear decomposition method for large optimization problems—blueprint for development. Technical Report NASA TM-83248, National Aeronautics and Space Administration, 1982.

[Tong, 1988] S. S. Tong. Coupling symbolic manipulation and numerical simulation for complex engineering designs. In *International Association of Mathematics and Computers in Simulation Conference on Expert Systems for Numerical Computing*, Purdue University, 1988.

[Winston, 1984] P.H. Winston. *Artificial Intelligence.* Addison-Wesley, Massachusetts, 1984.