N93-17507

# Approximation, Abstraction and Decomposition in Search and Optimization

S8-61
136882
p.8

Thomas Ellman
Department of Computer Science
Rutgers University
ellman@cs.rutgers.edu

## 1. Synthesis of Search Control Heuristics

One portion of my research has focused on automatic synthesis of search control heuristics for constraint satisfaction problems (CSPs). I have developed techniques for automatically synthesizing two types of heuristics for CSPs: Filtering functions are used to remove portions of a search space from consideration. Evaluation functions are used to order the remaining choices. My techniques operate by first constructing exactly correct filters and evaluators. These operate by exhaustively searching an entire CSP problem space. Abstracting and decomposing transformations are then applied in order to make the filters and evaluators easier to compute. An abstracting transformation replaces the original CSP problem space with a smaller abstraction space. A decomposing transformation splits a single CSP problem space into two or more subspaces, ignoring any interactions between them. Both types of transformation potentially introduce errors into the initially exact filters and evaluators. The transformations thus implement a tradeoff between the cost of using filters and evaluators, and the accuracy of the heuristic advice they provide. I have shown these techniques to be capable of synthesizing useful heuristics in domains such as floor-planning and job-scheduling, among others. (See [Ellman, 1992].)

## 2. Synthesis of Hierarchic Problem Solving Algorithms

Another portion of my research is focused on automatic synthesis of hierarchic algorithms for solving constraint satisfaction problems (CSPs). I have developed a technique for constructing hierarchic problem solvers based on numeric interval algebra. My system takes as inputs a candidate solution space $S$ and a constraint $C$ on candidate solutions. The solution space $S$ is assumed to be a cartesian product $R^n$ where $R$ is a set of integers. The constraint $C$ is assumed to be represented in terms of arithmetic, relational and boolean operations. From these inputs the system constructs an abstract solution space $S_a$ as a cartesian product $R_a^n$ where $R_a$ is a set of disjoint intervals that covers $R$. The system also constructs an abstract constraint $C_a$ on abstract solutions. The abstract constraint $C_a$ is obtained from the original constraint $C$ by replacing ordinary arithmetic operations with interval algebra operations and replacing boolean operations with boolean set operations. The abstract space $S_a$ and abstract constraint $C_a$ are then used to build a hierarchic problem solver that operates in two stages. The first stage finds an abstract solution in the space $S_a$ of intervals. The second stage refines the abstract solution into a concrete solution in the original search space $S$. I have shown this approach to be capable of synthesizing efficient problem solvers in domains such as floor-planning and job-scheduling, among others. (See [Ellman, 1992].)

## 3. Decomposition in Design Optimization

Another portion of my research is focused on automatic decomposition of design optimization problems. We are using the design of racing yacht hulls as a testbed domain for this research. Decomposition is especially important in the design of complex physical shapes such as yacht hulls. Exhaustive optimization is impossible because hull shapes are specified by a large number of parameters. Decomposition diminishes optimization costs by partitioning the shape parameters into non-interacting or weakly-interacting sets. We have developed a combination of empirical and knowledge-based techniques for finding useful decompositions. The knowledge-based method examines a declarative description of the function to be optimized in order to identify parameters that potentially interact with each other. The empirical method runs computational experiments in order to determine which potential interactions actually do occur in practice. We expect this approach to find decompositions that will result in faster optimization, with a minimal sacrifice in the quality of the resulting design. Implementation and testing of this approach are currently in progress. (I am pursuing this research in collaboration with Mark Schwabacher.) (See [Ellman *et al.*, 1992].)

## 4. Model Selection in Design Optimization

Another portion of my research is focused on intelligent model selection in design optimization. The model selection problem results from the difficulty of using exact models to analyze the performance of candidate designs. For example, in the domain of racing yacht design, an exact analysis of a yacht's performance would require a computationally expensive solution of the Navier-Stokes equations. Approximate models are therefore needed in order diminish the costs of analyzing and evaluating candidate designs. In many situations, more than one approximate model is available. For example, in the yacht design domain, the induced resistance of a yacht can be predicted by solving La Place's equation - an approximation of Navier-Stokes - or by using a simple algebraic formula. The two approximations differ widely in both the costs of computation and the accuracy of the results. Intelligent model selection techniques are therefore needed to determine which approximation is appropriate during a given phase of the design process.

We have attacked the model selection problem in the context of hillclimbing optimization. We have developed a technique which we call "gradient magnitude based model selection". This technique is based on the observation that a highly approximate model will often suffice when climbing a steep slope, because the correct direction of change is easy to determine. On the other hand, a more accurate model will often be required when climbing a gradual incline, because the correct direction of change is harder to determine. Our technique operates by comparing the estimated error of an approximation to the magnitude of the local gradient of the function to be optimized. An approximation is considered acceptable as long as the gradient is large enough, or the error is small enough, so that each proposed hillclimbing step is guaranteed to improve the value of the goal function. Implementation and testing of this approach are currently in progress. I am pursuing this research in collaboration with John Keane. (See [Ellman *et al.*, 1992].)

## References

T. Ellman, J. Keane, and M. Schwabacher. The rutgers cap project design associate. Technical Report CAP-TR-6, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1992.

T. Ellman. Idealization-based methods for constraint satisfaction problems. Working Notes of the AAAI Workshop on Approximation and Abstraction of Computational Theories (Forthcoming), July 1992.