Model-based Support for Mutable Parametric Design Optimization

Ravi Kapadia and **Gautam Biswas**
Computer Science Department
Vanderbilt University
Nashville, TN 37235
email: ravi, biswas@vuse.vanderbilt.edu

Abstract

Traditional methods for parametric design optimization assume that the relations between performance criteria and design variables are known algebraic functions with fixed coefficients. However, the relations may be *mutable*, i.e., the functions and/or coefficients may not be known explicitly because they depend on input parameters and vary in different parts of the design space. We present a model-based reasoning methodology to support parametric, mutable, design optimization. First, we derive *event models* to represent the effects of the system's parameters on the material that flows through it. Next, we use these models to discover mutable relations between the system's design variables and its optimization criteria. We then present an algorithm that searches for "optimal" designs by employing sensitivity analysis techniques on the derived relations.

Introduction

Traditional methods for parametric multicriteria design optimization in Operations Research, e.g., (Steuer 1986) and Artificial Intelligence (see examples in (Tong & Sriram 1992)) assume that the relations between performance criteria and design variables are known algebraic functions with fixed coefficients. However, in many real world applications, the relations may be *mutable*, i.e., the functions and/or coefficients may not be known explicitly because they depend on input parameters and vary in different parts of the design space.

We employ *model-based reasoning* techniques to develop a framework for parametric, mutable design optimization. Given a configuration of the system, we are interested in tuning its design variables to optimize desired objectives while meeting specified constraints. We describe techniques that start from a valid design solution and generate better solutions for the required artifact. Our methodology first develops *event models* that capture the effects of the system's parameters on the material that flows through it. Next, we use these models to discover mutable relations between the system's design variables and its optimization objectives. Then, we employ sensitivity analysis based heuristic techniques to determine directions and magnitudes of change of design variables to generate better design solutions.

[1]Copyright ©1999, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Mutable design optimization

A parametric design optimization problem contains the following elements. *Optimization objectives*, $y_1, \ldots, y_m$, represent behavior and performance parameters that must be optimized, e.g., speed of execution and manufacturing cost. *Design variables*, $v_1, \ldots, v_n$, represent system parameters that can be changed by the designer to affect system performance and attain optimization objectives. *Design constraints*, $C_1, \ldots, C_p$, on design variables and system behavior arise from domain principles and serve to define valid designs. *Optimization relations* describe how changes in design variables cause changes in optimization criteria. In general, the relation between each optimization objective y_i and the design variables $v_1, \ldots, v_n$, for a given input task I is defined as: $y_i \bowtie F_i(I, v_1, \ldots, v_n)$, where the relation operator $\bowtie \in \{=, \leq, \geq, <, >, \propto\}$.

The relation between y_i and the design variables $v_1, \ldots, v_n$, is *invariant* if $F_i(I, v_1, \ldots, v_n) = \sum_{j=1}^{n} c_j * v_j$, and each c_j is a constant. We refer to y_i as an *invariant* optimization objective (Steuer 1986; Tong & Sriram 1992). When $F_i(I, v_1, ..., v_n) = \sum_{j=1}^{n} f_{ij}(I, v_1, \ldots, v_n) * v_j$, the relation between y_i and $v_1, \ldots, v_n$, is *mutable* since the coefficient of v_j is dependent on I and $v_1, \ldots, v_n$. We refer to y_i as a *mutable* optimization objective.

In this paper, we consider real world design problems which combine both invariant and mutable optimization objectives. In our applications, $f_{ij}(I, v_1, \ldots, v_n)$ is not a continuous function over the design space, i.e., it varies at different points in the design space. Due to the complex nature of the system, it is not practical to express the function in the form of an analytic expression. Typically, it is possible to determine the function's value at a particular point in the design space through simulation. We use model-based reasoning techniques to determine each $f_{ij}(I, v_1, \ldots, v_n)$ for a specified input task I and a given design solution.

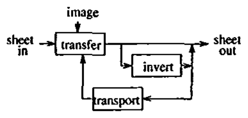


Figure 1: Reprographic machine

Reprographic machine

As a test bed for our design optimization methodology, we use a digital reprographic machine (e.g., printer, photocopier), which is a computer-controlled electromechanical system that produces documents by manipulating images and sheets of paper. Given a configuration for this system, we are interested in tuning its parameters so that the designed machine optimizes job completion times and manufacturing cost, while meeting specified design constraints.

In the simplified form of Fig. 1, the system prints simplex (one-sided) and duplex (two-sided) sheets. A sheet enters the machine through an input port. An image is transferred (or printed) to the sheet as it passes through the *transfer* component. A simplex sheet passes through without inversion (i.e., it bypasses inversion) on its way to the output port. A duplex sheet is inverted, routed to the *duplex transport* along the duplex loop, an image is transferred on the back side of the sheet (*transfer*), and the sheet is inverted again (*inverter*), before it is sent to the output port. Parameters that affect desired optimization criteria include the transit times of the components, and the capacity of buffers at the input and output ports (Kapadia & Fromherz 1997). To generate a document or job (i.e., an ordered sequence of simplex and duplex sheets), the transportation and printing of sheets must satisfy behavior constraints, for example, sheets must be manipulated such that they are available at the output in the specified order, and they must not collide with each other anywhere in the paper path.

State-of-the-art reprographic machines are equipped with control software that determines optimal times to schedule its operations (Kapadia & Fromherz 1997). Table 1 shows an optimal schedule for the document consisting of one simplex sheet followed by two duplex sheets, and then a simplex sheet (i.e., s_1, d_2, d_3, s_4) which is completed in 12 time units. While generating this schedule, we assumed the following parameter values: printing an image on to a sheet requires 1 unit of time, inverting a sheet takes 2 units, transporting a sheet along the duplex loop requires 3 units, and bypassing inversion takes 1 unit. The job completion time is a function of the system parameters and the schedule for the job. For example, the arrival time of s_4 at the output in Table 1 is a function of component transit times and the sequence of events that preceded it. There is no predefined optimization function that applies to all jobs; the function may be different for each new job. Furthermore, with a different set of design parameters, different schedules are possible for the same job description. For example, if inverting a sheet takes 1 time unit, and bypassing inversion takes 3 units, the optimal schedule for s_1, d_2, d_3, s_4 introduces sheet d_2 at time 0, d_3 at time 1, s_1 at time 2, and s_4 at time 7, which generates the complete job in 11 time units. Thus, optimizing the behavior of the reprographic machine system comprises a mutable optimization problem.

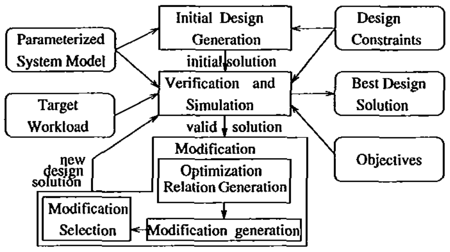


Figure 2: Design optimization methodology

Design optimization methodology

Fig. 2 shows the architecture of our design optimization methodology. Using *design constraints*, *initial design generation* produces a valid, though usually not optimal, initial solution (typically, one that has worked in similar cases in the past). *Simulation* uses the *system model* to generate system behavior for a given design solution and specified input tasks chosen from the *target workload*. *Verification* checks that the solution and its behavior satisfy specified *design constraints*. *Modification* encompasses multiple tasks. *Optimization relation generation* maps each mutable *optimization objective* to system design variables for a particular design solution. *Modification generation* identifies a set of modifications to a design solution that improve desired objectives. *Modification selection* compares different modifications and picks the most promising one to generate the next design solution. Our methodology iterates through these tasks until it finds the *best design solution*.

Parameterized system model. Component parameters model attributes of the system components that affect the behaviors of interest. Consider a representative set of parameters for the reprographic machine system shown in Fig. 1. k_1 is the transit time of the transfer component, k_2 is the transit time for inversion of a duplex sheet, k_3 is the transit time for bypassing inversion, and k_4 is the transit time for transporting a duplex sheet for a second printing. Our test bed is best modeled as a discrete event system with discrete valued parameters. In (Kapadia, Biswas, & Fromherz 1997) we adapted the Environment Relationship (ER) net framework of (Ghezzi *et al.* 1991) (an extension of basic Petri nets, that incorporates time modeling) to represent our system model. Tokens represent material (e.g., sheets) and their attributes capture the material's properties. A component is a collection of places and tran-

Component	Place in ER net	Time												
		0	1	2	3	4	5	6	7	8	9	10	11	12
Input	p_1	d_2		d_3		s_1		d_2		d_3		s_4		
Transfer	p_2		d_2		d_3		s_1		d_2		d_3		s_4	
Inverter	p_3				d_2		d_3	s_1			d_2		d_3	s_4
Transport	p_4				d_2		d_3							
Output	p_5							s_1			d_2		d_3	s_4

Table 1: Optimal printing sequence for document s_1, d_2, d_3, s_4

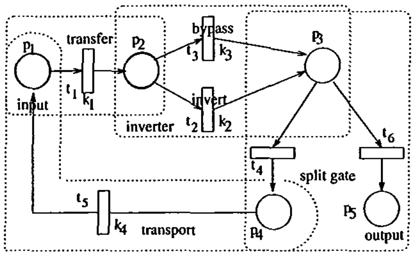


Figure 3: Reprographic machine model with ER nets

sitions. Places correspond to input and output ports. Transitions correspond to functions that the component performs and associated actions describe its behavior, i.e., how the function transforms the material flowing through the component. Component parameters are properties associated with transitions and places in an ER Net. For example, in Fig. 3 which shows a system level model for the reprographic machine of Fig. 1, the inverter component is modeled by transitions t_2 and t_3, places p_2 and p_3, and k_2 and k_3 represent the transit times for the invert and bypass operations, respectively. The ER net model is used to simulate the behavior of the system as shown in Table 1 and to develop parameterized optimization relations between design variables and optimization objectives.

Design constraints. In this paper, we present two classes of design constraints. *Domain constraints* define valid designs by constraining the domains of design variables. *Cost constraints* capture our intuition that associated with each decision is a price that must be paid to implement that decision. Other classes of constraints, including *behavior* and *performance* constraints, are covered in (Kapadia 1999). We present examples of domain and cost constraints later in the paper.

Objectives. A system should be designed and evaluated with respect to its expected use. For instance, a printer used mainly for books and magazines will require a different set of parameters from another used for printing short reports and office memos. We assume that a *target workload*, J $= \{J_1, J_2, \ldots, J_j\}$, which is composed of a finite set of the most likely jobs in the user's workplace, captures the intended user's workload characteristics. An objective is to develop a design solution that optimizes the performance of the system with respect to this workload, where the performance of a design solution on a job is measured in terms of the job completion time (jct). In the examples that follow, we assume $J = \{\{s_1, d_2, d_3, s_4\}, \{s_1, d_2, d_3, s_4, d_5, d_6\}\}$[1], and our objective is to *minimize job completion time* for each job in J. Simultaneously, the designer may choose to *minimize manufacturing cost* ($Cost$).

Design solution. A *design solution* is an assignment of values to design variables. A *valid* design solution is one that meets design constraints, while an *optimal* design solution is a valid solution which optimizes desired objectives. However, a design solution that optimizes one set of jobs may not produce good solutions for others, and the objectives of minimizing cost and maximizing performance are often in conflict with each other. Thus, there may be no single design solution that optimizes all objectives. A *non-inferior* design solution is a valid solution with the property that there is no other known valid solution that will yield an improvement in one criterion without causing a degradation in at least one other criterion. Our methodology develops non-inferior design solutions which are accepted or rejected based on the designer's preferences. We allow designers to articulate their preferences either prior to the generation of design solutions, after the generation of design solutions, or progressively, i.e., during the optimization process (similar to (Mollaghasemi & Evans 1994)).

Design optimization algorithm. Our design optimization methodology is implemented as a hill climbing algorithm as follows.

```
1. Generate mutable optimization relations for a given
     design solution from the ER net model and input
     task descriptions. This may require simulation
     of system behavior one or more times, e.g.,
     for each job in the target workload:
       (a) simulate its behavior;
       (b) generate a relation between the job's compl-
           etion time and the system's design variables.
2. Generate a set of modifications that lead to better
     performance on the objectives.
3. If there is no valid modification, stop.
4. Otherwise, select the "best" modification and
     apply it to the existing solution.
```

[1]A simplifying assumption in this paper is that each job in the target workload has equal priority. (Kapadia & Fromherz 1997), describe techniques for dealing with jobs with different priorities.

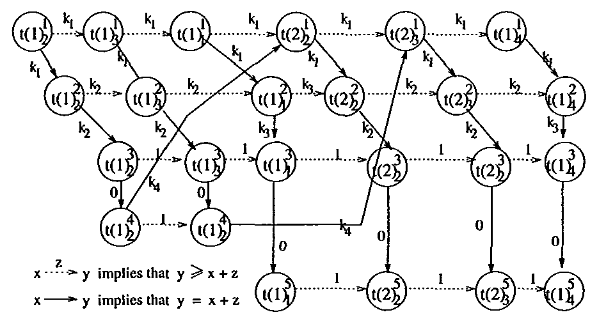

x ··z··> y implies that y ⩾ x + z
x ——> y implies that y = x + z

Figure 4: Event model for Table 1

5. Modifying the parameters of a design solution changes the mutable optimization relations. Determining a new relation may require simulation, which is computationally expensive and we want to avoid it unless it is necessary. Thus, repeat steps 1 to 4 with the following changes.
   (a) Before step 1(a) check if simulation is required to generate the new relation. If yes, perform 1(a), otherwise skip it for that objective.
   (b) Before step 2, check if the new design solution is inferior to the *best known solution*. If yes, stop. Otherwise, make it the best known solution.

In this paper, we focus on steps 1 and 2 of this algorithm.

Generating optimization relations

The relationship between a mutable optimization objective y_i, the input task I, and the design variables $v_1, \ldots, v_n$, is: $y_i \bowtie \sum_{j=1}^{n} f_{ij}(I, v_1, \ldots, v_n) * v_j$. Generating a mutable optimization relation requires building an *event model* (that reflects the mutability of f_{ij}) from an ER net behavior simulation for a given design solution and document description (as shown in Table 1). An event model is a directed acyclic graph where each vertex represents an event (informally, an event is the arrival of a token in a place in the ER net system model) and a directed edge between two vertices represents a precedence relation between the corresponding events.

Fig. 4 shows the event model for the events depicted in Table 1. Formally, $t(r)_i^j$ denotes the event that the i^{th} token (i.e., sheet) arrives in place j on its r^{th} pass through the machine. Edges in the event model correspond to temporal constraints among events in the ER net model of Fig. 3 as explained below (see (Kapadia 1999) for details).

• There is an edge for every firing of a transition in the generation of the system's behavior. The transit time associated with the transition is represented by the weight along the corresponding edge in the event model, e.g., Fig. 4 shows an edge $t(1)_2^1 \rightarrow t(1)_2^2$ with weight k_1 which represents transition t_1 which prints an image to a sheet (where $t(1)_2^1$ is the event that enables t_1 and $t(1)_2^2$ is the result of firing t_1).

• Sheets arrive at the output (p_5) in the specified order, e.g., s_1 must arrive before d_2 at the output. Thus, we have the following *order* constraints: $\forall i = 1, \ldots, n, r = 1, 2 : \quad t(r)_i^5 < t(r)_{i+1}^5$. Each order constraint is represented by a dotted edge with a weight of 1 (since there must be a separation of at least one unit of time between the arrival of two successive sheets at the output).

• There must be no collisions in any place in the machine, so two or more sheets cannot be present in the same place at the same time. Thus, we have the following *resource allocation* constraints: $\forall x, y = 1, \ldots, n, p, q = 1, 2 : \;$ if $x \neq y$, $t(p)_x^j \neq t(q)_y^j$. In a known, valid sequence of events, $t(p)_i^j > (<) t(q)_k^j$. The dotted edge $t(p)_i^j \rightarrow t(q)_k^j$ indicates that event $t(p)_i^j$ precedes $t(q)_k^j$. The weight of the edge is the largest transit time (w) of all the transitions for which place j is an input, since tokens must arrive at least w units apart in j to avoid a collision after the firing of any transition enabled by the tokens in j.

We get relations between design variables and optimization criteria as follows. First, we substitute parameter values for the edges in the event model. Then, we use a topological sort to get the *critical path* which establishes a lower bound on the time that must be spent to execute the job. For the example of Fig. 4, we get: $t(1)_2^1 \rightarrow t(1)_2^2 \rightarrow t(1)_2^3 \rightarrow t(1)_2^4 \rightarrow t(2)_2^1 \rightarrow t(2)_2^2 \rightarrow t(2)_2^3 \rightarrow t(2)_3^3 \rightarrow t(2)_3^5 \rightarrow t(1)_4^5$. Finally, we find the *lower bound* on the job completion time by symbolically summing the parameters along the critical path, e.g., job completion time for J_1, $jct_1 \geq 2 * k_1 + 3 * k_2 + k_4$. To extract better performance from the system, we must direct our efforts at "minimizing" this relation.

Each job's event model may have a different structure and, therefore, it may have a different critical path leading to a different optimization relation. For example, job $J_2 = s_1, d_2, d_3, s_4, d_5, d_6$, which is completed in 21 time units has the optimization relation: $jct_2 \geq 3 * k_1 + 6 * k_2 + 2 * k_4$.

When the designer modifies the parameters of a design solution, event models (and their optimization relations) may change, as stated in step 5 of our design optimization algorithm. If the structure of an event model remains the same (i.e., the set of vertices and edges is identical, but only some weights on the edges are changed) it requires updating the weights of the edges and then performing a topological sort to discover a new optimization relation. Otherwise, it is necessary to derive a new event model by simulating system behavior. (Kapadia 1999) describes the use of strong domain knowledge to predict when the structure of an event model may change with alterations to a design solution necessitating the simulation of system behavior.

Modification generation

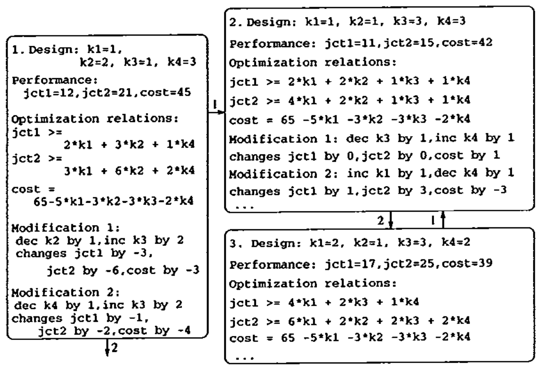
Given a design solution, modification generation identifies a set of variables that must be changed and the direction and magnitude of change for each variable in this set to bring about desired improvement in the optimization objectives. We use the sensitivity of an optimization objective's lower bound to changes in design variables (e.g., incrementing k_4 by 1 increases the lower bound for jct_2 by 1, but incrementing k_2 by 1 increases it by 6) to determine suitable parameter modifications.

The following example illustrates our approach. We assume that the initial design solution is $k_1 = 1$, $k_2 = 2$, $k_3 = 1$, and $k_4 = 3$. We have the following mutable optimization relations: $jct_1 \geq 2 * k_1 + 3 * k_2 + k_4$, and $jct_2 \geq 3 * k_1 + 6 * k_2 + 2 * k_4$. The following invariant relation: $Cost = 65 - 5 * k_1 - 3 * k_2 - 3 * k_3 - 2 * k_4$, indicates that lowering transit times increases manufacturing cost and that the cost is more sensitive to changes in some design variables than others. The constant 65 represents fixed manufacturing costs. We assume the following domain constraints: $k_1 \in \{1, 2\}$, $k_2 \in \{1, 2, 3\}$, $k_3 \in \{1, 2, 3\}$, and $k_4 \in \{2, 3, 4, 5\}$. (See (Kapadia 1999) for details on cost and domain constraints.)

We extend the sensitivity analysis method of (Biswas, Kapadia, & Yu 1997) for qualitative, steady state diagnosis to identify parameter modifications for design optimization. Let $Y = \{y_1, y_2, \ldots, y_m\}$ be the set of optimization criteria and $V = \{v_1, v_2, \ldots, v_n\}$ be the set of design variables. We have optimization relations between V and Y. A high level description for our modification generation algorithm is presented below.
(1) Establish a qualitative $(+, -, 0)$ relationship between each optimization criterion (y_i) and each design variable (v_j) by determining the sensitivity of y_i with respect to v_j. This is obtained by taking the partial derivative $\partial y_i / \partial v_j, \forall i, j$, e.g., $\partial jct_1 / \partial k_2 = 3$. The relationship is + (−) if an increase in v_j causes an increase (decrease) in y_i and 0 if v_j has no effect on y_i.
(2) Choose an appropriate direction of change for each y_i. If y_i is to be maximized (minimized), increase y_i, i.e., y_i+ (decrease y_i, i.e., y_i-).
(3) Express the relation between y_i and V as a Conjunctive Normal Form (CNF) expression, e.g., (a) $Cost- \leftarrow (k_1+) \vee (k_2+) \vee (k_3+) \vee (k_4+)$, (b) $jct_1- \leftarrow (k_1-) \vee (k_2-) \vee (k_4-)$, and (c) $jct_2- \leftarrow (k_1-) \vee (k_2-) \vee (k_4-)$. The logical formulas represent cause-effect relations between optimization criteria and design variables, e.g., from relation (a) we see that decreasing k_1, k_2, or k_4 reduces jct_1.
(4) Find a satisfying assignment to the CNF formula $y_1 \wedge y_2 \wedge \ldots \wedge y_m$. While this problem is inherently exponential, CNF satisfiability algorithms like GSAT (Russell & Norvig, 1995) have been reported to solve large problems in reasonable times. In our example, the following satisfying assignments form a set of qualitative modifications: $(k_1+) \wedge (k_2-), (k_1+) \wedge (k_4-), (k_1-) \wedge (k_2+), (k_1-) \wedge (k_3+), (k_1-) \wedge (k_4+), (k_2+) \wedge (k_4-), (k_2-) \wedge (k_3+), (k_2-) \wedge (k_4+), (k_3+) \wedge (k_4-)$.
(5) Assign magnitudes to the design variables in the set of qualitative modifications that satisfy the following difference relations:
$\Delta y_1 = \Delta v_1 * \frac{\partial y_1}{\partial v_1} + \Delta v_2 * \frac{\partial y_1}{\partial v_2} + \ldots \Delta v_n * \frac{\partial y_1}{\partial v_n}$,
$\Delta y_2 = \Delta v_1 * \frac{\partial y_2}{\partial v_1} + \Delta v_2 * \frac{\partial y_2}{\partial v_2} + \ldots \Delta v_n * \frac{\partial y_2}{\partial v_n}, \ldots,$
$\Delta y_m = \Delta v_1 * \frac{\partial y_m}{\partial v_1} + \Delta v_2 * \frac{\partial y_m}{\partial v_2} + \ldots \Delta v_n * \frac{\partial y_m}{\partial v_n}$.
In our example, we have the following relations:
$\Delta Cost = -5 * \Delta k_1 - 3 * \Delta k_2 - 3 * \Delta k_3 - 2 * \Delta k_4$,
$\Delta jct_1 = \Delta k_1 + 3 * \Delta k_2 + 1 * \Delta k_4$,
$\Delta jct_2 = 3 * \Delta k_1 + 6 * \Delta k_2 + 2 * \Delta k_4$.
In general, we cannot directly solve these equations to generate magnitudes for design variable modifications because we do not know values of Δy_i $(\forall i = 1, \ldots, m)$ and there may be no closed-form solution to this system of equations. Instead, we analyze each candidate to determine the magnitudes of modification for its variables as follows. A variable in a modification is labeled *free* if there is no optimization criterion which is adversely affected by its direction of change, otherwise it is *bound*. Then, we use the following heuristics to determine optimal parameter magnitude changes. To each free element, greedily assign the largest magnitude of change consistent with domain constraints to maximize its beneficial effects on optimization criteria. To each bound element, assign the smallest incremental change consistent with its constraints. This minimizes its adverse effects on the optimization objectives. For example, in modification candidate $(k_2-) \wedge (k_3+)$, k_2 is bound (since k_2- increases *Cost*) so we assign it the smallest increment, i.e., $\Delta k_2 = -1$, while k_3 is free, so we assign it the largest increment, i.e., $\Delta k_3 = 2$.
(6) If the modification pushes a variable outside its range as defined by its domain constraint, the resulting design solution is invalid. Eliminate any modifications that result in invalid solutions by checking domain constraints.
(7) Finally, eliminate inferior modifications. For example, $(k_3+) \wedge (k_4-)$ with $\Delta k_3 = 2, \Delta k_4 = -1$ has the following effect on optimization objectives: $\Delta Cost = -4, \Delta jct_1 = -1, \Delta jct_2 = -2$ while $(k_2+) \wedge (k_4-)$ with $\Delta k_2 = 1, \Delta k_4 = -1$ affects the optimization criteria as follows: $\Delta Cost = -1, \Delta jct_1 = 1, \Delta jct_2 = 4$. Since $(k_3+) \wedge (k_4-)$ is better than $(k_2+) \wedge (k_4-)$ for *every* criterion, the latter is an inferior modification and can be eliminated. We are developing strategies that incorporate designer's preferences to discriminate among multiple non-inferior modifications.

Interactive design navigation

We have built a modeling and design optimization prototype tool in the language `CLP(fd)` (a constraint logic programming language over finite domains) to model systems using ER Nets, and generate and optimize their designs. The designer and our tool may collaborate to exploit their different strengths. The user provides the system with an initial design solution which is valid but non-optimal. The design tool simulates system behavior for this solution, discovers mutable optimization relations, and applies sensitivity analysis techniques to identify suitable modifications. The user may guide the tool to explore the more promising regions of the design solution space by choosing which modifications



```
1. Design: k1=1,
        k2=2, k3=1, k4=3
Performance:
  jct1=12,jct2=21,cost=45

Optimization relations:
jct1 >=
        2*k1 + 3*k2 + 1*k4
jct2 >=
        3*k1 + 6*k2 + 2*k4
cost =
   65-5*k1-3*k2-3*k3-2*k4

Modification 1:
 dec k2 by 1,inc k3 by 2
 changes jct1 by -3,
     jct2 by -6,cost by -3
Modification 2:
dec k4 by 1,inc k3 by 2
changes jct1 by -1,
    jct2 by -2,cost by -4
```

```
2. Design: k1=1, k2=1, k3=3, k4=3
Performance: jct1=11,jct2=15,cost=42
Optimization relations:
jct1 >= 2*k1 + 2*k2 + 1*k3 + 1*k4
jct2 >= 4*k1 + 2*k2 + 1*k3 + 1*k4
cost = 65 -5*k1 -3*k2 -3*k3 -2*k4
Modification 1: dec k3 by 1,inc k4 by 1
changes jct1 by 0,jct2 by 0,cost by 1
Modification 2: inc k1 by 1,dec k4 by 1
changes jct1 by 1,jct2 by 3,cost by -3
 ...
```

```
3. Design: k1=2, k2=1, k3=3, k4=2
Performance: jct1=17,jct2=25,cost=39
Optimization relations:
jct1 >= 4*k1 + 2*k3 + 1*k4
jct2 >= 6*k1 + 2*k2 + 2*k3 + 2*k4
cost = 65 -5*k1 -3*k2 -3*k3 -2*k4
 ...
```

Figure 5: Interactively navigating the design space

to apply and which performance criteria to optimize. A trace of the tool's navigation of the design space is shown in Fig. 5. In the initial solution state, the user chooses modification 1 to generate the second solution. Next, he chooses modification 2 to generate the third solution. Having examined three states the user may choose solution 2, i.e., $k_1 = 1, k_2 = 1, k_3 = 3, k_4 = 3$ with performance $jct_1 = 11, jct_2 = 15, cost = 42$ which is non-inferior in the space of known design solutions (but not necessarily non-inferior in the context of all solutions).

Discussion

This paper presents a mutable optimization problem and describes model-based reasoning techniques to address different aspects of this problem. We present a model-based reasoning approach to discover mutable optimization relations and a sensitivity analysis algorithm that uses the mutable relations to optimize multiple conflicting performance objectives. We apply our methods to optimize a real system's dynamic performance parameters. The techniques described in this paper are part of an overall methodology to address system-level design optimization where the components of the system may belong to different domains (Kapadia 1999).

(Kapadia & Fromherz 1997) address dynamic design optimization using constraint satisfaction techniques but their search process is only slightly better than exhaustive search. Our heuristic modification generation algorithm explores fewer solutions in the design space. (Goel & Chandrasekaran 1989), have also used model-based reasoning to effectively navigate the design space. They represent interactions in the form of directed, acyclic causal graphs that are created a priori by system designers. Our event models, which are generated dynamically, are more similar to the work of (Williams 1990), who develops a network of interactions by envisioning the behavior of the system from a model of its components and their interconnections.

Representing an event model as a graph structure will allow us to apply a number of well known algorithms for analyzing system behavior (e.g., maximum flow, shortest path) which may support other forms of optimization (e.g., capacity optimization). Our design optimization algorithm is sensitive to the initial solution: if the initial solution is in a good region of the design space, optimal or near optimal solutions will be found; otherwise, it is likely to get trapped in a local optimum. We must formalize our techniques for detecting changes to optimization relations with alterations in design solutions without resorting to simulation. Preliminary results for our methodology are promising but extensive testing and empirical evaluation is required. We will study scalability issues by considering a larger and more diverse target workload and a larger set of component parameters. We anticipate that some form of problem decomposition may be required to overcome the increase in time complexity as the number of optimization criteria rise for a larger workload.

Acknowledgements. We thank Markus Fromherz for helping us refine the ideas presented in this paper.

References

Biswas, G.; Kapadia, R.; and Yu, X. 1997. Combined qualitative-quantitative diagnosis of continuous valued systems. *IEEE Trans. on Systems, Man, and Cybernetics* 167–185.

Ghezzi, C.; Mandrioli, D.; Morasca, S.; and Pezze, M. 1991. A unified high-level petri net formalism for time critical systems. *IEEE Trans. on Software Engg.* 160–172.

Goel, A., and Chandrasekaran, B. 1989. Functional representation of designs and redesign problem solving. In *Proc. of AAAI-89*, 1388–1394.

Kapadia, R., and Fromherz, M. 1997. Design optimization with uncertain application knowledge. In *Proc. of the Tenth Intl. Conf. IEA/AIE-97*, 421–430.

Kapadia, R.; Biswas, G.; and Fromherz, M. 1997. Hybrid modeling for smart system design. In *Proc. Tenth Intl. FLAIRS*, 111–115.

Kapadia, R. 1999. Model-based support for system-level mutable parametric design optimization. Technical Report TR-99-02, CS Dept. Vanderbilt University, Nashville TN 37235.

Mollaghasemi, M., and Evans, G. 1994. A unified high-level petri net formalism for time critical systems. *IEEE Trans. on Systems, Man, and Cybernetics.* 1407–1411.

Steuer, R. 1986. *Multiple Criteria Optimization: Theory, Computation, and Application.* John Wiley & Sons.

Tong, C., and Sriram, D. 1992. *Artificial Intelligence in Engg. Design. Vol. I.* Academic Press Inc.

Williams, B. 1990. Interaction-based invention: Designing novel devices from first principles. In *Proc. AAAI-90*, 349–356.