

INTERACTIVE DESIGN OPTIMIZATION OF ARCHITECTURAL LAYOUTS

JEREMY J. MICHALEK* and PANOS Y. PAPALAMBROS

*Optimal Design Laboratory, Department of Mechanical Engineering,
University of Michigan, Ann Arbor, Michigan 48109-2125, USA*

(Received 28 August 2001; In final form 26 February 2002)

Many areas of design involve both quantifiable and subjective goals, preferences, and constraints. Aesthetic and other subjective aspects of design are typically ignored in optimization models because they are difficult to model with mathematics; however, they are extremely important in areas such as product design and architectural design. This article presents an interactive method for integrating mathematical optimization with human decision-making during conceptual design of architectural floorplan layouts. The optimization models and algorithms were presented in a previous article. Here, an object-oriented representation allows the designer to interact with physically relevant building objects during optimization. The designer's interaction causes the program to dynamically change the optimization representation on-the-fly by adding, deleting, and modifying objectives, constraints, and structural units. This work presents mathematical optimization as a tool to assist the designer in refining ill-defined design problems during the early conceptual design phase. The designer can quickly explore design alternatives visually and computationally by taking advantage of computational algorithms to maintain feasibility and compute efficient solutions.

Keywords: Interactive optimization; Conceptual design; Floorplan; Architectural design; Layout

1 INTRODUCTION

Automated design generation tools are not adequate for many problems in architectural design for several reasons. Architectural design involves subjective decisions about aesthetics, human traffic patterns, and other preferences that are difficult to model or quantify mathematically. For design qualities that can be well defined, it is often difficult to foresee all issues that may affect the optimization model before observing some results.

In addition to modeling issues, automated tools face computational difficulties. Currently, methods that can compute or verify the global optimum of a function involve systematic exhaustive search with some kind of tree-pruning or stochastic methods. For large problems, these methods fail to perform adequately in a reasonable amount of time. Allowing the designer to use experience and intuition to guide the search can improve search time relative to well-defined objectives and constraints as well as to take into account unmodeled objectives and constraints. The designer can also assist gradient algorithms that may have

* Corresponding author. E-mail: michalek@umich.edu

computational difficulties related to non-smooth objective and constraint functions by guiding the design away from first-order discontinuities.

An implementation of the tool presented in this paper can be downloaded at <http://ode.engin.umich.edu>.

2 PREVIOUS WORK

This work is relevant to research in areas of *interactive optimization* as well as *exploratory CAD tools* designed to assist the designer during conceptual design. A summary of previous work in these two areas is given below.

2.1 Interactive Optimization

Most work on interactive optimization focuses on multi-objective optimization. This is because preferences are difficult to express before some idea of the trade-off is gained from comparison of results. Multi-objective interactive techniques use some variation of the Interactive Weighted Tchebycheff (IWT) approach [15]. In this approach the multi-objective problem is converted into a single objective problem by minimizing the weighted sum of distances to an ideal point:

$$f(\mathbf{x}) = \sum_{i=1}^n w_i f_i(\mathbf{x}) \quad (1)$$

where n is the total number of objective functions.

Figure 1 shows f_1 and f_2 as two competing objective functions. An optimization algorithm will generate different points on the Pareto curve depending on how the objectives are weighted. The designer alters the weights of each objective to tune trade-off preferences and move along the Pareto set. The process of choosing weights can be difficult and

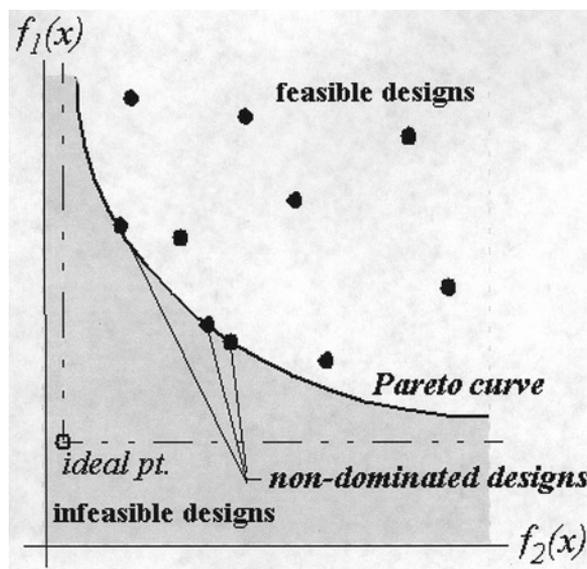


FIGURE 1 Pareto set in a multi-objective optimization problem.

non-intuitive. Several researchers have proposed methods to improve this interactive procedure [12–15]. For this application, however, the simple IWT approach with linear weights is used because the objective functions are generally not competing. The approach here builds on the well-known concept that the designer must generally see some physical results in order to understand design trade-offs.

Researchers have also looked at how designers can interact with optimization models during the optimization process. *OptdesX*, a commercial optimization tool [17, 18], allows the designer to monitor changes in the design variables during optimization; however, visualization is limited to numerical information, and the designer cannot change the design problem itself without reprogramming and recompiling. Tidd *et al.* [20] developed a system for a designer to interact with design variables and resultant design behaviors graphically, allowing the designer to interact with the design problem in an intuitive way.

Optimization tools are still needed that allow the designer to interactively and intuitively define and refine the optimization problem statement and guide the search algorithms into areas of interest and away from computational traps. The tool presented in this article addresses such needs for architectural layout design.

2.2 Exploratory Design CAD Tools

Most CAD systems are very good at producing precise, accurate renderings of well defined designs. Many CAD systems also include detailed analysis packages for obtaining simulation feedback on the performance of a design. However, very few CAD packages address the needs of a designer during the initial conceptual exploratory phase of design when the problem is ill-defined, and the solution has not yet been decided.

“The conceptual phase of design (or ideation phase) is the initial phase where the internal ideas of the designer are externalized or explored interactively and represented tentatively in some form using any medium, . . . These are usually early sketches, rough mock-up models and concept renderings. Normally during the conceptual phase, designers quickly represent as many as possible different solutions in a short time. These are evaluated visually before exploring further possibilities. There are two important characteristics during this phase. First, the quick and intuitive representation of concepts. Secondly, the generation (in a very short time) of many different solutions and variations. Sketching is the most widely used imagery aid in evolving new ideas during the conceptual phase.” (Stuyver and Hennessey [22])

Most architectural CAD tools are either drafting or analysis tools – neither focusing on the rapid generation of design alternatives. However, some researchers have attempted to create tools to provide more assistance during conceptual design exploration. Liggett and Mitchell [10] used a probability model to create an interactive design tool for building spatial allocation. Their tool provides graphic feedback to the designer on the probable ‘goodness’ of design moves at each step. The designer can then use this feedback, along with intuition and other considerations that are not represented in the model, to guide space allocation selection. Arvin and House [19] created a physics-based system specifically for architectural layout in which rooms are connected with simulated springs and dampers, and a dynamic simulation is run to push and pull the rooms into position. Kharrufa *et al.* [11] developed an architectural CAD tool allowing the designer to interact with intuitive objects, such as spaces, instead of drawing lines.

The vast majority of design problems are ill-defined – especially during the conceptual design phase. In an ill-defined problem the constraints on the problem are not fully formulated. An interactive tool for early design stage conceptualization can help the designer to interactively refine the ill-defined design problem, provide an intuitive representation to interact with, provide a simple interface that enables rapid design and exploration of alternatives, and provide computational feedback on the performance of designs. The design tool presented in this work addresses these issues for architectural layout design. The interactive

design tool provides assistance that allows the designer to refine the problem definition during exploration, and the use of optimization allows the designer to quickly generate high-quality layouts and to receive both visual and computational feedback.

3 THE MATHEMATICAL MODEL

The mathematical representation used to optimize the layout geometry is described in detail in a previous article [2]. The model poses the architectural floorplan layout problem as a process of finding the best location and size of a group of interrelated rectangular *Units*. Using this representation, rooms, hallways, doorways (*Accessways*), and boundaries are all represented as combinations of orthogonal rectangular *Units*. The algorithm's job is to position the rectangular *Units* in a meaningful, feasible way that meets all design specifications and optimizes the layout.

Figure 2 shows a simple layout example. In this example, the living room, bedroom, and bath function as living space, the hallway functions as a pathway, and the *Units* labeled "A" function as *Accessways* by allocating a space for a doorway or opening between two *Units*. The model reported in the companion article [2] describes a toolbox of design constraints that can be used to maintain geometric relationships between the *Units*. The constraints can prohibit intersection of two *Units*, force intersection of two *Units*, force one *Unit* to the edge of another, bound the size of a *Unit*, constrain the build cost of the structure, maintain feasible window sizes, and maintain minimum natural lighting levels in each room. The program automatically chooses and adds the appropriate default constraints. For instance, if the designer adds a new room to the layout, the program will automatically add constraints to prohibit intersection with all other rooms in the layout. These default constraints can be changed manually by the designer; however, in most cases the designer need not worry about the underlying mathematical structure because the *Units* are set up to behave as expected by default. Additionally, the designer can select from several objectives that minimize heating, cooling and lighting costs, minimize wasted space, and minimize hall or accessway size.

The geometry optimization approach uses CFSQP [4–6] a version of Sequential Quadratic Programming that maintains feasible design iterates once a feasible design has been found. This means that each iteration of the algorithm yields a feasible design alternative, and the

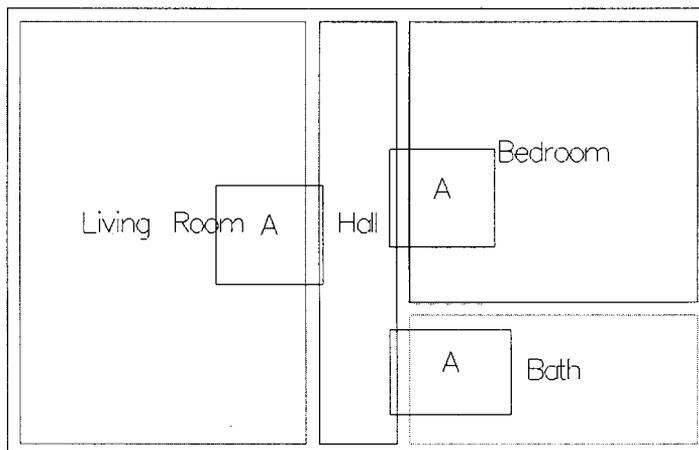


FIGURE 2 An example layout showing four different types of Units.

progression of the algorithm moves toward improved design alternatives. Allowing the designer to see this progression of designs visually introduces opportunities for making design and modeling decisions based on the progression of the algorithm.

Additionally, the topology optimization formulation presented in the companion article [2] can be used to automatically generate a feasible design that can be used to begin interactive exploration, although the topology search itself is not interactive.

4 INTERACTION WITH THE MODEL

A building geometry optimization design tool was created to allow the designer to interact with the layout problem during the optimization process (see Fig. 3). This interactive tool allows the designer to interact with the optimization problem in three ways: *defining the problem*, *guiding the search*, and *exploring the design space*. These three types of real-time interaction are described below.

4.1 Interactive Problem Definition

The interactive design tool allows the designer to add, delete, and modify objectives, constraints, and Units during optimization to refine the problem definition. The designer can set up the initial problem and start optimization. At each iteration, the current design is displayed. The designer can watch how the design is changing and use that information to change the problem definition at any point during the optimization. This is useful because design is an iterative process for the designer as well as for the algorithm. When the designer

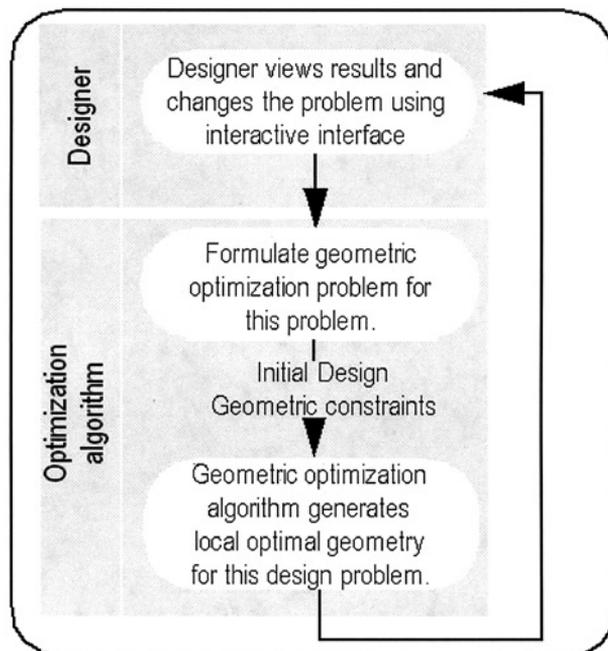


FIGURE 3 Interactive building layout optimization method. The designer uses feedback from the algorithm to refine or change the problem definition and guide search.

has visual feedback, s/he can realize new preferences or forgotten constraints, and can explore how changes in the problem definition affect the design solutions.

Refining the problem during optimization is accomplished using an object-oriented representation of the building. Each time an optimization is performed, the program translates the object-oriented representation into a set of mathematical design variables, objectives and constraints. This automatic translation allows a new mathematical design space to be formulated automatically with a different dimensionality or different objective and constraint functions. Computationally, the problem is actually being regenerated, and a completely new mathematical optimization is started in the new design space at a point analogous to where the designer left off in the old design space. However, the mathematical reformulation is hidden from the designer, so it appears as though the optimization is progressing naturally with the new design change. This allows a new way to think about modeling changes during optimization because the designer sees modeling changes as simple design moves that can be easily added and experimented with during conceptualization.

4.1.1 Addition, Deletion, and Modification of Objectives

As the designer receives feedback of the optimization progression, s/he may want to change the definition of the design objective. A designer watching the design progression will be able to notice many types of layout deficiencies visually, and s/he can add a new objective to enforce a preference away from the deficiency. For example, if layout solutions are lacking in use of space for living areas, the designer can add a *Minimize Wasted Space* objective. After seeing the optimization progression, the designer may decide that some objectives are unimportant, or the designer may wish to remove an objective to simplify the problem or observe how the design progression reacts without the objective. If all objectives are removed, then the algorithm terminates when it finds a feasible design.

If the designer chooses more than one objective function, the individual objectives are combined into a single objective function using a weighted sum (Eq. (1)). The weights, or relative importance, of each individual objective can be changed to reflect preference in competing objectives. Defining appropriate weights for a set of objectives is nontrivial; however, using the interactive tool, the designer can change objective weights as s/he observes how the design is progressing during optimization. Defining appropriate weight values is easier once the designer can see how the designs react to a particular set of weights. Furthermore, a good set of weights in one area of the design space may be poor in another area, so it is important to have the flexibility to change them during optimization. It is common practice to revise objective weights *after* finding an optimum; however, this method allows the designer to interject *during* the process if s/he sees the design migrating toward an undesirable area of the design space. In the future, other methods of describing objective preference, such as using aspiration points [12,13], could be implemented to make the process more intuitive for the designer.

4.1.2 Addition, Deletion, and Modification of Constraints

As the designer receives feedback of the optimization progression, s/he may want to change the definition of the design constraints. If the design progresses into an undesirable area of the design space, the designer can dynamically add new constraints to prevent search in that area. After seeing the optimization progression, the designer may decide to remove certain constraints in order to achieve a better solution. Often decisions about which constraints should be ignored cannot be made until a designer has seen some physical designs.

Also, some constraints can be relaxed by modifying a numerical bound, such as a minimum area constraint. Once the designer has seen some feasible design alternatives, s/he may choose to relax certain numerical bounds in order to achieve a better solution.

4.1.3 Addition, Deletion, and Modification of Units

As the designer receives feedback of the optimization progression, s/he may want to change the layout elements themselves. Extra *Units* may be added to change the problem (*i.e.* add an extra bedroom or closet) or to enforce connectivity (*i.e.* add an extra accessway). Units may also be deleted if they become extraneous in a particular layout. Rooms may be deleted (explore a two bedroom instead of a three bedroom apartment) or forced connections may be relaxed (*i.e.* remove an accessway and allow two Rooms to separate). Units can also be stretched or moved during optimization to force the search into a different area of the design space. Modification can be used to guide search into an area of interest.

4.1.4 Change of Variable Formulation

The geometric layout design problem can be formulated with several alternative variable representations, each having its own representation used by CFSQP. The object-oriented representation, however, can calculate constraints and objectives regardless of which design parameters are used as optimization variables. It is easy to switch between representations during optimization if one representation produced better optimization behavior in certain areas of the design space.

4.2 Guiding Search

With the ability to modify design variables during search, the designer can guide the optimization process. Because the design variables are geometric in nature, the designer can interact with the variables in an intuitive way. If the designer sees the design moving into an undesirable area of the design space, s/he can intervene and force search into a new area of the space by manipulating Units. This method uses the designer's experience and intuition to guide global search along with the efficiency and accuracy of gradient algorithms to direct local search.

In addition, the designer can help the optimization algorithm to avoid computational traps. Gradient algorithms assume that all functions of the design variables are continuous and have continuous derivatives. If the problem representation violates these assumptions, the algorithm may have unpredictable behavior. In particular, the geometric building layout formulation presented in the previous article [2] has non-smooth gradient constraint functions in some areas of the design space. If the algorithm has computational difficulties near one of these areas, usually it will appear to be stuck, and the designer can nudge the Units slightly away from the non-smooth area to resume normal optimization. This ability is important for problems that have some irregularities.

4.3 Interactive Design Exploration

The interactive layout optimization tool can be viewed as an interactive sketchpad for exploring design alternatives. As a typical procedure, the designer would

1. Define Rooms and Halls: Define which Rooms will be included in the building (kitchen, bedroom, etc.) and what are acceptable sizes for each Room (length, sq. ft.).
2. Move Rooms Into Rough Location: Rough dimensions also can be set by stretching.

3. Define Connections: Add Accessways to define which Rooms will be connected.
4. Choose an Objective: Choose an objective to optimize for.
5. Add Additional Constraints: Add any special constraints besides those added by default.
6. Optimize: The optimization algorithm will compact the geometry into a locally optimum layout.
7. Examine Results: Check results visually and check estimated performance values calculated by the objective function.
8. Iterate: Use information to refine the problem definition or guide search into a new area.

Using default room sizes and constraints, the problem setup (steps 1–5) can be completed in less than one minute for a typical two-bedroom apartment layout. Optimization for the same problem usually terminates within a few seconds. At this point, making changes to the design, such as relocating a room, is just a matter of dragging rooms into alternative positions and re-optimizing. This process is extremely fast, and many design alternatives can be easily examined both visually and computationally. The speed and simplicity offer a lot of potential as an exploratory tool. Examining alternative configurations is often faster than sketching, and possibly more intuitive because the designer manipulates objects (rooms, halls, etc.) instead of lines.

In addition to receiving quick visual feedback about various configurations, the design tool also provides computational feedback about design performance with respect to the design objectives. The designer can immediately see quantitative data about design alternatives, including

1. Performance Cost: An estimate of the annual heating, cooling, and lighting cost.
2. Build Cost: An estimate of the cost of materials (glass and walls) to build.
3. Natural Lighting: An estimate of the lighting levels in each room for a given environment.
4. Living Space: An estimate of the building area that is used for living space in comparison to area used for passageways or wasted space.

In the current implementation, all of these estimates are rough estimates, which is appropriate for a conceptual exploratory tool; however, more accurate models could be used if necessary. One drawback to using gradient-based optimization techniques is that the functions must all be smooth: complex non-smooth simulation functions can be smoothed using surrogate modeling techniques to fit into this model [8]. The use of rough models to provide computational feedback to the designer during conceptualization offers the potential for consideration of important computational objectives early in the design process as well as the opportunity to explore how design changes affect building performance.

5 DEMONSTRATION STUDY

A simple apartment design study will show how the optimization tool can be used to help the designer sketch design concepts, formalize the design problem objectives and constraints, use visual and computational feedback to refine the problem definition, and quickly explore design alternatives and trade-offs. The study shows how the computational efficiency of optimization algorithms can be combined with subjective, intuitive decision-making of human designers during design exploration.

First, the initial layout concept is sketched out (Fig. 4). This design consists of two bedrooms, a full bath, kitchen, dining room and living area with a hallway structure consisting of two main paths. Connections are defined as shown in the picture by adding accessways between Units. Each Unit is initialized with its own default constraints for minimum area, length, and

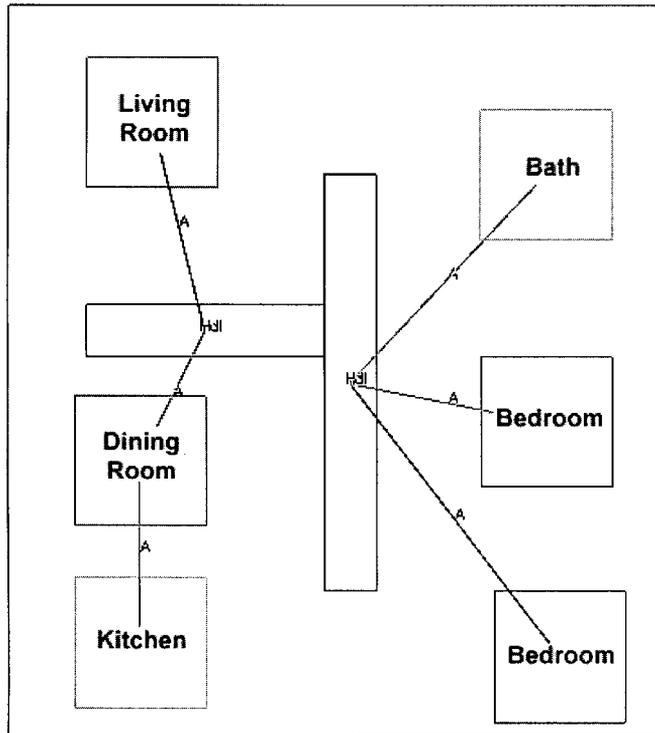


FIGURE 4 Initial layout.

width. Default constraints are automatically added to prohibit intersection between all Room–Room and Room–Hallway combinations. Default constraints are also added to force all Units inside the building bounds. Each Accessway is added by the designer to guarantee connected access between two Units by doorway or opening (accessways are displayed here as lines between two Units). Default constraints are added that force intersection between the Accessway and each of the connected Units such that the intersection overlap is large enough for a door or opening. Using the defaults, this entire process takes about one minute.

Next, design objectives are specified. This design will minimize annual heating cost and wasted space. The optimization process takes a few seconds, and the solution is shown in Figure 5 (accessway connectivity is represented as a line between two units, and the actual accessway position is shown as a rectangular Unit marked 'A').

Suppose that after viewing the results, the designer decides that it will be more economical if the bath and kitchen are close together so that piping can be clustered. The bath is dragged down below the kitchen, and the design is re-optimized. The optimization algorithm repacks the Units (Fig. 6).

Results show that the designer forgot an important constraint. The Hallway between the living room and dining room was intended to be an entryway, but the designer failed to specify that the Hallway must connect to an external wall. The designer now adds a new constraint to force the hall against the west wall and optimizes the design again (Fig. 7).

Results now match the designer's intentions. The layout is saved as a design alternative, and the designer continues to explore other alternatives. The designer now considers moving the bath to the east side of the apartment to separate the public space from the private space better, and also to allow more room for the private rooms while shrinking the bedrooms.

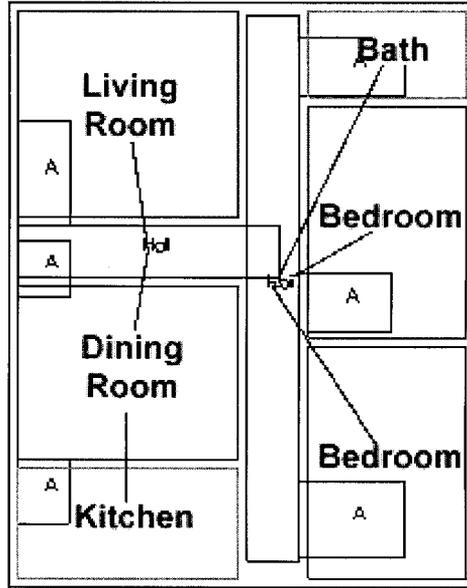


FIGURE 5 Initial solution.

Another design alternative is produced (Fig. 8). The ability to make design changes and quickly examine results, consequences, and trade-offs is very useful. The effects of design changes can be seen quickly – often faster than sketching with pencil and paper.

The designer notices that the doorway (accessway) to the north bedroom could be moved south, and the living room could take up the space currently occupied by a hallway. The living room is moved north of the hallway and enlarged so that it uses the space. The apartment is re-optimized (Fig. 9).

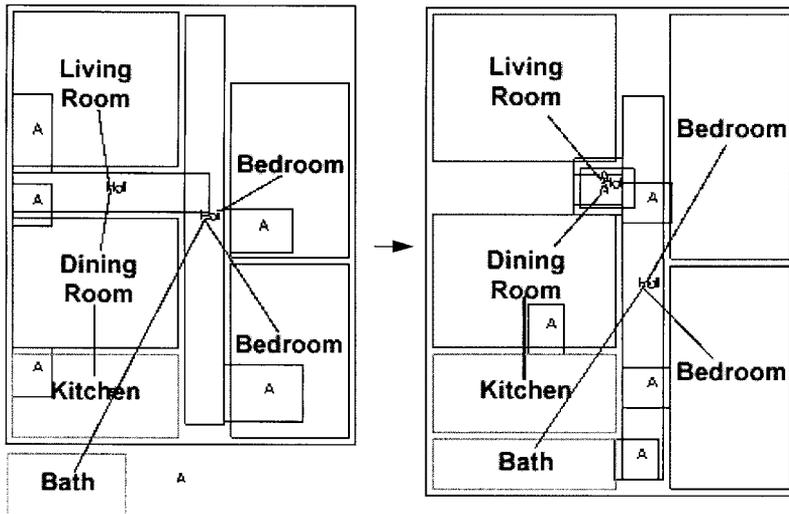


FIGURE 6 Bath is relocated.

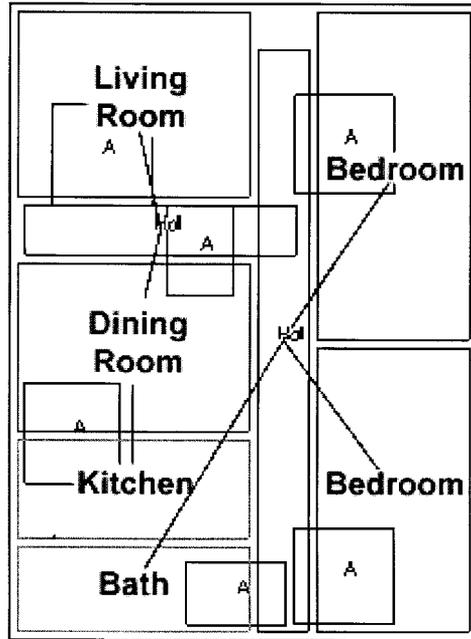


FIGURE 7 A forgotten constraint is added to the model.

The designer examines the results and decides that the default minimum area for the living room (120 sq. ft.) was too small. The minimum area is increased to 150 sq. ft. and the design is re-optimized (Fig. 10).

This ability to change constraint parameters allows designers to perform parametric studies intuitively and examine trade-offs. Many constraints, such as minimum allowable room area

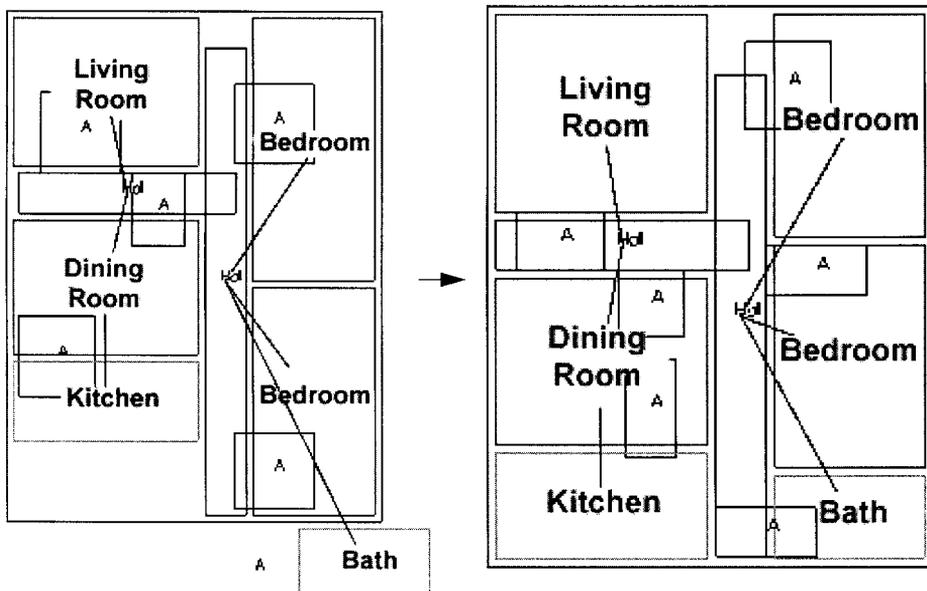


FIGURE 8 A more compact arrangement is found.

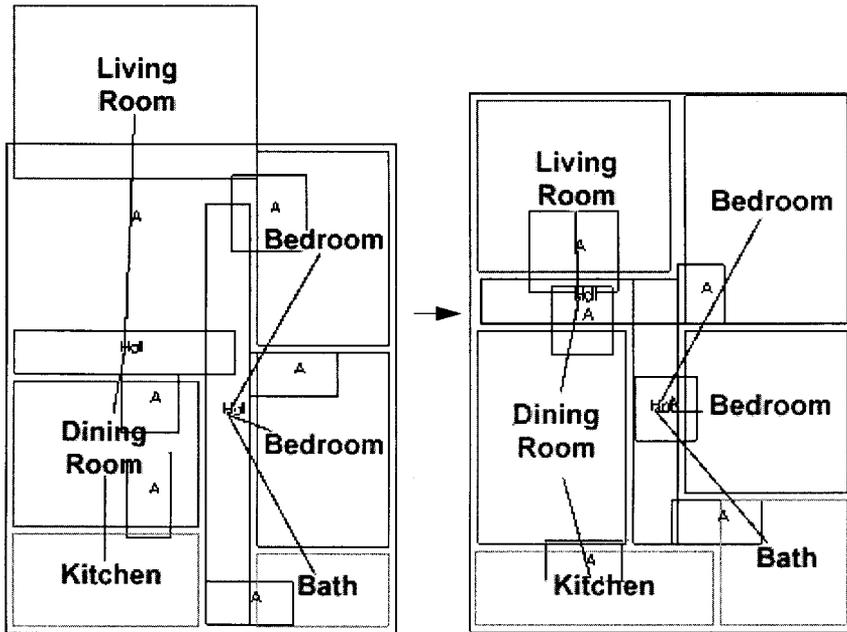


FIGURE 9 A hallway is reduced.

constraints, are flexible because these constraints can be relaxed if doing so provides a significant gain somewhere else. A designer using this tool can often see visually where relaxing constraints can improve the design, and it is quick and easy to explore constraint relaxation. In this case, the constraint is increased to provide more space, and the algorithm

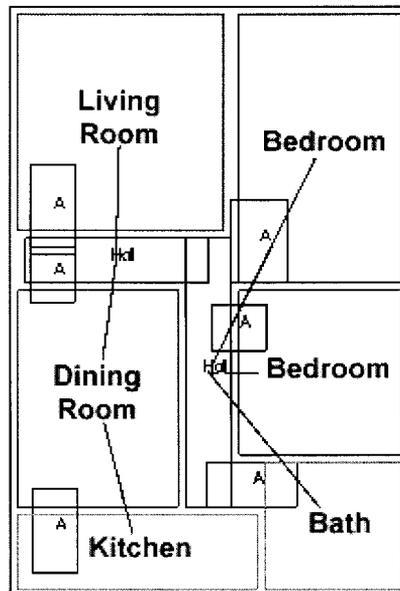


FIGURE 10 Living room size is increased.

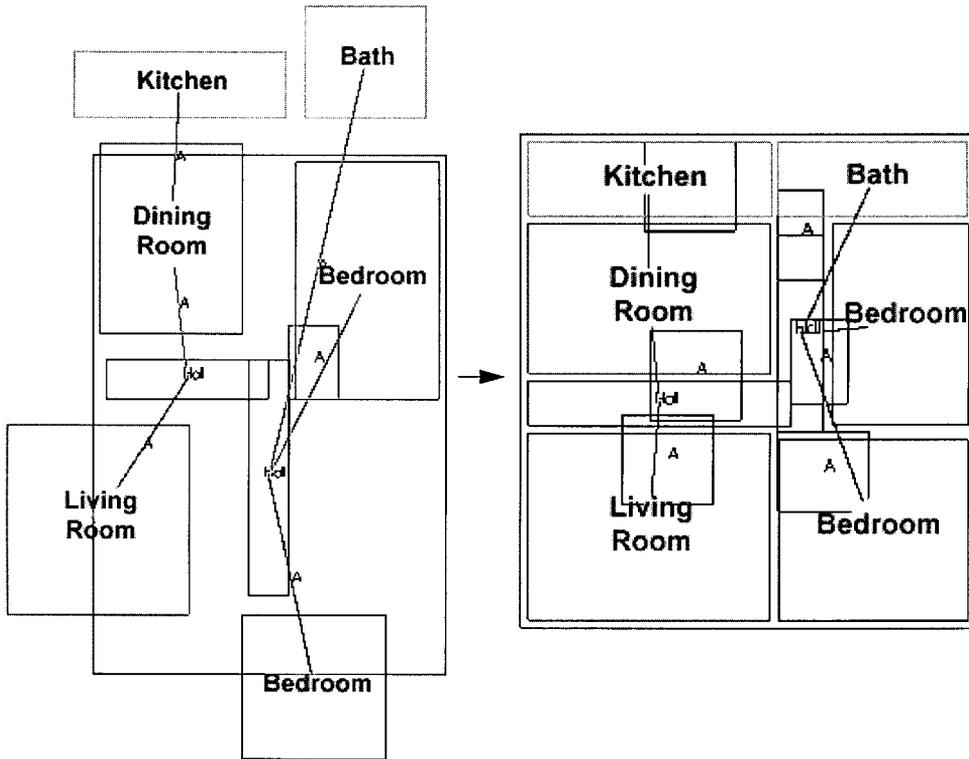


FIGURE 11 A more compact arrangement is found.

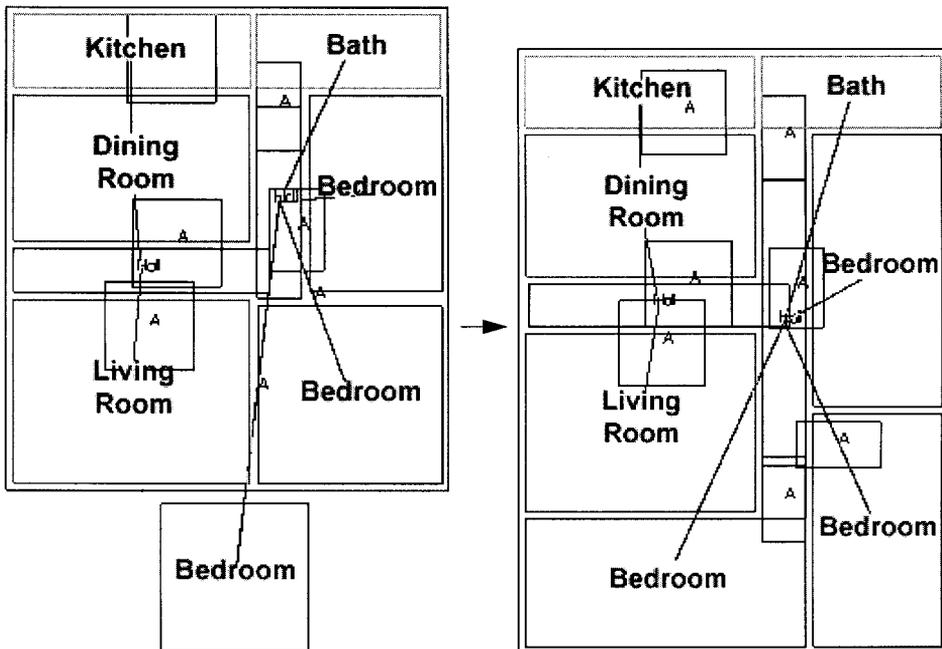


FIGURE 12 A third bedroom is added.

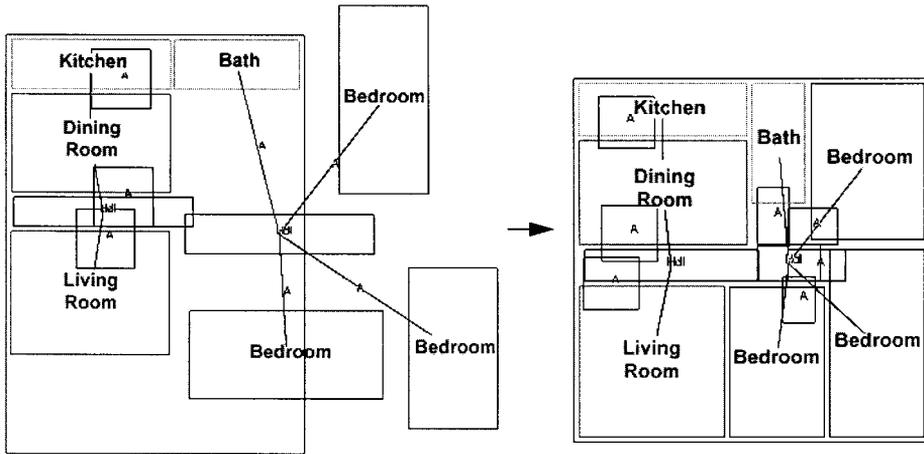


FIGURE 13 The private zone is moved.

enlarges the living room to meet the new constraints. A new design alternative has been created. The designer saves this design and moves on to explore.

Now the designer wants to reduce the size of the apartment. The last design measured 27×40 (1080 sq. ft.). A new design arrangement is considered by moving the kitchen and bath to the north end, and moving the living room to the south. The design is re-optimized (Fig. 11). The resulting design measures 30×33 (990 sq. ft.) – a reduction of 90 sq. ft. The design alternative is saved.

The designer decides to consider adding a third bedroom. The designer places the third bedroom and adds an Accessway. The new design is optimized (Fig. 12). The new three-bedroom layout can now be compared to the two bedroom layout. Adding an extra bedroom increases the apartment size from 30×33 (990 sq. ft.) to 30×42.5 (1275 sq. ft.), and the heating cost will rise an estimated 15%.

The new three-bedroom layout is saved.

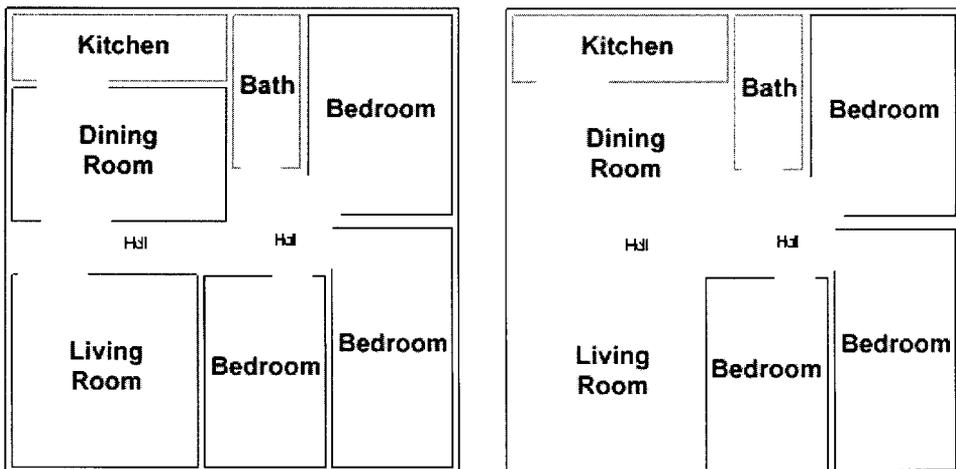


FIGURE 14 Visualization options.

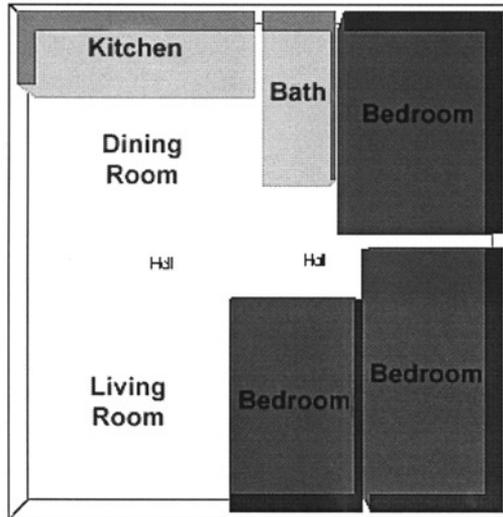


FIGURE 15 Visualization in three dimensions.

After examining the results, the designer decides that the private area should be more distinctly separate from the public area. The three bedrooms are moved to the west side of the apartment, and the new design is optimized (Fig. 13).

The result is accepted and saved as an alternative three bedroom arrangement.

The design can be viewed in several different ways to get a better feeling for the layout. Accessways can be viewed as openings or doorways, and the living room and dining room can be viewed without physical walls (Fig. 14).

The layout can also be viewed in three dimensions to give a feel for the space. Three dimensional views can be used to help visualize the look and feel of the interior space (Figs. 15 and 16).

This example demonstrated how the interactive optimization design tool can be used to quickly generate and compare design alternatives visually and computationally. The designer

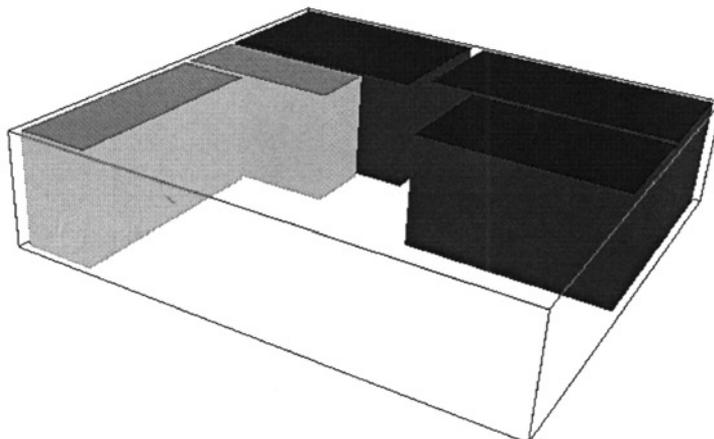


FIGURE 16 Visualization in three dimensions.

uses the optimization tool during conceptualization to help refine the problem goals, understand design tradeoffs, and explore design options. The entire process of generating, visualizing, examining computational feedback, and using that information to explore new designs in the above example was completed in about ten minutes. This tool offers speed for design concept sketching with the power of computational feedback and optimization results.

6 CONCLUSIONS

The interactive design optimization tool described here shows the significant potential for computational optimization algorithms to be used in the early conceptual design stages. Design conceptualization is an extremely important part of the design process, and it is one that computer tools typically ignore because of the poorly understood nature of creativity and subjective judgement. The tool assists the designer with design generation and evaluation, rather than attempting to automate these processes completely. This approach allows the designer to maintain control, to use quantitative and subjective judgements where appropriate, to pursue creative exploration. Extending such capabilities to more generalized domains will fill an important gap in the present array of computer design support tools.

References

- [1] Michalek, J. (2001). Interactive layout design optimization. *MS Thesis*, University of Michigan.
- [2] Michalek, J., Choudhary, R. and Papalambros, P. Y. (2002). Architectural layout design optimization. *Engineering Optimization*, (this issue).
- [3] Papalambros, P. Y. and Wilde, D. J. (2000). *Principles of Optimal Design – Modeling and Computation*, 2nd ed. Cambridge University Press, Cambridge, England.
- [4] Zhou, J. L. and Tits, A. L. (1995). An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. *SIAM Journal of Optimization*, **6**, 461–487.
- [5] Lawrence, C. T., Zhou, J. L. and Tits, A. (1995). User's guide for CFSQP Version 2.3: A C code for solving (large scale) constrained nonlinear (minimax) optimization problem, generating iterates satisfying all inequality constraints, *Technical Report TR-94-16r1*, Institute for Systems Research, University of Maryland.
- [6] Zhou, J. L. and Tits, A. (1993). Nonmonotone line search for minmax problems. *Journal of Optimization Theory and Applications*, **76**(3), 455–476.
- [7] Mayne, D. Q. and Polak, E. (1976). Feasible directions algorithms for optimization problems with equality and inequality Constraints. *Mathematical Programming*, **11**, 67–80.
- [8] Barton, R. R. (1994). Metamodeling: A state of the art review. *Proceedings of the 1994 IEEE Winter Simulation Conference*, Lake Buena Vista, FL 237–244.
- [9] Schwarz, A., Berry, D. M. and Shaviv, E. (1994). On the use of the automated building design system. *Computer-Aided Design*, **26**(10), 747–761.
- [10] Liggett, R. S. and Mitchell, W. J. (1981). Interactive graphic floor plan layout method. *Computer-Aided Design*, **13**(5), 289–298.
- [11] Kharrufa, S., Saffo, A., Aldabbagh, H. and Mahmood, W. (1988). Developing CAD techniques for preliminary architectural design. *Computer-Aided Design*, **20**(10), 581–588.
- [12] Tappeta, R. and Renaud, J. E. (1999). Interactive multiobjective optimization design strategy for decision based design. *Proceedings of the 1999 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DETC99/DAC-8581, 1–15.
- [13] Diaz, A. (1987). Interactive solution to multiobjective optimization problems. *International Journal for Numerical Methods in Engineering*, **24**, 1865–1877.
- [14] Miettinen, K. and Makela, M. M. (2000). Interactive multiobjective optimization system WWW-NIMBUS on the Internet. *Computers and Operations Research*, **27**, 709–723.
- [15] Palli, N., Azarm, S., McCluskey, P. and Sundararajan, R. (1998). An interactive multistage epsilon-inequality constraint method for multiple objectives decision making. *ASME Journal of Mechanical Design*, **120**, 678–686.
- [16] Wieghardt, K., Hartmann, D. and Leimbach, K. R. (1997). Interactive shape optimization of continuum structures. *Engineering Structures*, **19**(4), 325–331.
- [17] Balling, R., Parkinson, A. and Free, J. (1984). Interactive optimization in engineering design. *Proceedings of the ASCE 3rd Conference on Computing in Civil Engineering*, San Diego, CA.

- [18] Parkinson, A. R., Balling, R. J. and Free, J. C. (1984). Optdes. BYU: A software system for optimal engineering design. *ASME Computers in Mechanical Engineering Conference*, Las Vegas, NV.
- [19] Arvin, S. A. and House, D. H. (1999). Modeling architectural design objectives in physically based space planning. *ACADIA*, 212–225.
- [20] Tidd, W. F., Rinderle, J. R. and Witkin, A. (1992). Design refinement via interactive manipulations of design parameters and behaviours. *Design Theory and Methodology – DTM '92*. American Society of Mechanical Engineers, New York, NY.
- [21] Parmee, C. and Bonham, C. R. (2000). Towards the support of innovative conceptual design through interactive designer/evolutionary computing strategies. *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, **14**, 3–16.
- [22] Stuyver, R. and Hennessey, J. (1997). A support tool for the conceptual phase of design. http://www.io.tudelft.nl/research/IDEATE/papers/stuy_hci.htm.
- [23] Hennessey, J. (1997). The IDEATE Project: Exploring computer enhancements for conceptualizing. http://www.io.tudelft.nl/research/IDEATE/papers/henn_els/henn_els.htm.
- [24] Duffy, A. H. B., Persidis, A. and MacCallum, K. J. (1996). NODES: A numerical and object based modeling system for conceptual engineering design. *Knowledge-Based Systems*, **9**, 183–206.
- [25] Scriabin, M. and Vergin, R. (1975). Comparison of computer algorithms and visual based methods for plant layout. *Management Science*, **22**(2), 172–181.
- [26] Simon, H. A. (1973). The structure of Ill-structured problems. *Artificial Intelligence*, **4**, 181–201.