Creating Personalized Documents: An Optimization Approach

Lisa Purvis
Steven Harrington
Xerox Corporation
800 Phillips Road, 128-27E
Webster, NY 14580
{lpurvis | sharrington}@crt.xerox.com

Barry O'Sullivan
Eugene C. Freuder
Cork Constraint Computation Centre
Department of Computer Science
University College Cork, Ireland
{b.osullivan | e.freuder}@cs.ucc.ie

ABSTRACT

The digital networked world is enabling and requiring a new emphasis on personalized document creation. The new, more dynamic digital environment demands tools that can reproduce both the contents and the layout automatically, tailored to personal needs and transformed for the presentation device, and can enable novices to easily create such documents. In order to achieve such automated document assembly and transformation, we have formalized custom document creation as a multiobjective optimization problem, and use a genetic algorithm to assemble and transform compound personalized documents. While we have found that such an automated process for document creation opens new possibilities and new workflows, we have also found several areas where further research would enable the approach to be more broadly and practically applied. This paper reviews the current system and outlines several areas where future research will broaden its current capabilities.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *office automation;* I.7.2 [**Document and Text Processing**]: Document Preparation – *Desktop publishing;* I.7.4 [**Document and Text Processing**]: Electronic Publishing*;* F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems – *Sorting and searching, Routing and layout;*

General Terms

Algorithms, Performance, Design, Experimentation.

Keywords

Genetic algorithm, constraint-based reasoning, multiobjective optimization, constrained optimization, automated layout, document design.

1. INTRODUCTION

The digital networked world is a sea of information. Individuals need this information in different forms, at different times, on different devices. While there is a lot of information, only a portion of it is relevant to each individual at a particular time, and the information needs of an individual change over time. Businesses are also finding that personalized information is more effective in keeping customers, both in e-commerce and traditional settings. The new, more dynamic digital environment demands tools that can automatically create documents, tailored to personal needs and transformed for the presentation device.

We have formalized the creation of personalized documents as a multiobjective optimization problem, and use a genetic algorithm to automatically assemble such documents. We begin in Section 2 by providing a motivation for our work, and background on other approaches to custom document creation. In Section 3, we describe our formulation of the problem as multiobjective optimization solved with a genetic algorithm, along with empirical results that illustrate the complexities of the multiobjective search space, and the techniques we have applied to address those complexities. In Section 4 we describe those areas that remain on our future research agenda, and then conclude in Section 5 with a summary.

2. MOTIVATION / RELATED WORK

2.1 Variable Information Documents

In the printing world, personalized documents are often called "variable information" documents. Such documents contain areas that are common across a set of documents, as well as areas that are "variable". Simple examples of such documents are bills and statements, which contain different information for each individual (e.g. account number, address, name), as well as information that is the same for everyone (e.g. company logo). Variable documents are also beginning to become more complex documents such as brochures and catalogs, which include variable images and color.

In traditional variable information (VI) document creation applications, the creation is accomplished by an expert in graphic arts, databases, layout, document design, etc. This expert document creator develops an overall layout for the document that includes slots for the variable data. The creator also finds or creates appropriate content pieces, and specifies rules for how to fill in the custom slots with this content, or places the content in a database and then links the slots to particular fields in the database. The VI application then creates a document for each customer by inserting the data for the customer into its linked slot. These types of templates are typically called "lick and stick", because the template has "art holes" which are defined by the document creator, and then the variable data is placed into those art holes to form different instances of the document.

The resulting set of documents is typically quite similar: each custom slot has one piece of content of about the same size, and the general layout is the same for all instances, regardless of the available content pieces. Thus, the traditional process not only requires extensive time and expertise from the document creator, but it also does not respond dynamically to varying amounts or types of content pieces, or to different output devices. Furthermore, the template creator is responsible for ensuring that the final document will adhere to good design principles, and is therefore aesthetically pleasing.

It is with these limitations in mind that we began our research to automate the process of creating a personalized document. With such an automated approach, we can open up the space of personalized document workflows to the novice, as well as enable the documents themselves to be more dynamic and transform themselves for different individuals and different devices.

For example, consider a document such as the catalog page shown in Figure 1A. In personalizing the document for a different individual, we may wish to remove some of the elements because they are not relevant to that individual. When we remove the elements as shown in Figure 1B, the original design of the document no longer works well. Our system can fix the bad design to create the new layout shown in Figure 1C.

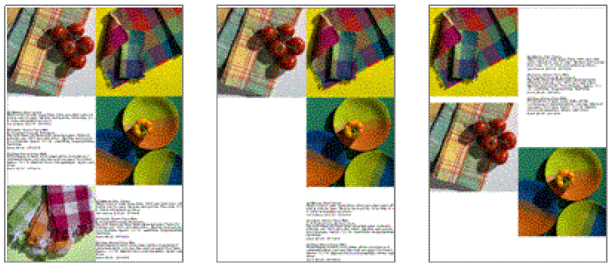


Figure 1A, 1B, 1C. Original catalog page, changed to remove some content, and then redesigned to be aesthetically pleasing again.

2.2 Related Constraint-Based Document Creation Research

There has been work in the creation of documents using constraint-based approaches. Dengler et al use a constraint-based approach to generate diagrams [6]. Another related work is in the adaptation of synchronized multimedia presentations [18], where extensions to HTML are explored that enable adapting web documents to different conditions and user needs. Graf has developed a constraint-based layout framework called LayLab [11], and applied it to various applications such as flowchart layout and pagination of yellow pages. Kroener has explored the combination of several AI techniques (constraint processing included) to achieve dynamic document layout [12]. The constraints research group at the University of Washington has explored constraint-based document layout for the web [3].

The main difference between these efforts and the one described here is that the multiobjective optimization methodology attempts to optimize the document layout according to a set of design constraints, in addition to finding a layout that satisfies certain spatial and physical style constraints. This in turn enables the approach to accommodate varying design preferences.

The constraints necessary to represent aesthetic design qualities are non-linear in nature, as well as "soft", and thus an approach using the typical simplex algorithm is not possible here. Being a multiobjective problem, this also means that a single optimum such as computed by the simplex algorithm isn't practical. We have chosen to use a genetic algorithm to solve our multiobjective optimization problem since genetic algorithms work with populations of solutions, and thus multiple pareto-optimal solutions can be found in a genetic population in a single run. Furthermore, genetic algorithms have been shown to be particularly successful in problems that are difficult to formulate mathematically (e.g. non-linear or poorly understood objectives such as "design" objectives), and on problems that exhibit multimodality, discontinuity, randomness, and noise [7].

3. DOCUMENT CREATION AS MULTIOBJECTIVE OPTIMIZATION

Over the last few years, much research has been devoted to applying genetic algorithms in multiobjective optimization problems [4]. Real-world problems typically involve simultaneous consideration of multiple performance criteria. These objectives are often in conflict with one another, such that advancement in one objective causes deterioration in another. The genetic algorithm has been found to be robust in the face of such ill-defined problem landscapes.

The document creation problem is a good example of such a difficult, multiobjective optimization problem. In document design there are multiple design objectives that are typically desired. The tradeoffs and interactions between these design objectives create a large and complex search space. The large problem space and real-life efficiency requirements preclude a complete algorithm from being practical. Thus we opted for the genetic algorithm approach that has promise in terms of flexibility and efficiency.

3.1 Modeling a document as a genome

Our methodology specifies the document, its content components, its layout requirements, and its desired aesthetic criteria as elements of a multiobjective optimization problem. Solving this problem results in an automated document layout for the set of content components that satisfies not only certain primitive content and layout constraints, but also advantageously fulfills desired design properties that provide a way to ensure that the generated document is well designed.

As such, we model each of the document parameters that can be modified as genes in the genome. As an example, consider the document template shown in Figure 2. This document template specifies that the positions and sizes of both areaA and areaB can be changed.

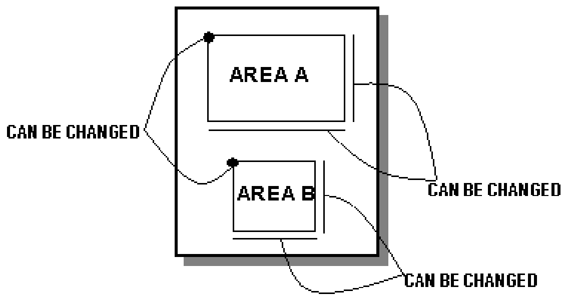


Figure 2. Example Document Template

Thus, the genes in this example are: areaA-topLeftX, areaA-topLeftY, areaB-topLeftX, areaB-topLeftY, areaA-width, areaA-height, areaB-width, areaB-height. The resulting genome contains these eight genes.

We also model each of the desired design qualities as objectives to maximize. If more than one design quality is desired (as is typically the case), the problem becomes a multiobjective optimization problem. In this case, we sum the individual objective function scores to produce the overall optimization score for a particular solution. We can furthermore weight each of the desired qualities with a priority, so that the overall optimization score becomes a weighted sum of the individual objective function scores.

The genetic algorithm allows us to explore the search space of possible documents, evolving towards the one that best meets the desired design objectives.

3.2 Design Qualities as Objectives

In our initial work, we attempted to use a complete systematic search algorithm to solve the document layout problem. Thus, we didn't include design objectives, but rather only had simple lower-level layout constraints that were required to be satisfied, such as left-of, right-of, left-align, top-align, width < 350, etc. We quickly learned that while such a system could find layouts that satisfied the constraints, the solutions weren't necessarily well designed, and thus typically weren't acceptable.

As an example, consider the document shown in Figure 3. We had specified simple content and layout constraints: we wanted a title, a number of images about food, and an advertisement. Furthermore, we wanted the title to be above the images, and the images above the advertisement, we wanted the three content areas to be the same width, and none of the content items could overlap one another. The result adheres to the constraints, but there are problems in that the images are all sandwiched together, the document isn't balanced, etc.

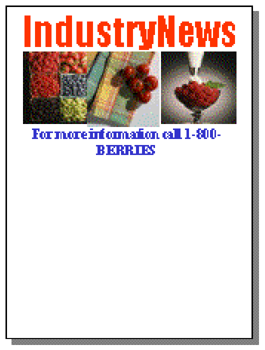


Figure 3. Solution Satisfying Basic Layout Constraints

What we needed was a means to find a solution that not only satisfied the basic content and layout constraints, but that was the "best" solution according to some aesthetic design criteria.

We began by encoding some of the basic design criteria that one might find in any book on basic design principles [19]. We encoded such principles as alignment, balance, legibility, compactness, text and image balance, etc. Each design criterion measures one aspect of the document. For instance, the alignment criterion provides scores that indicate how well the edges of the elements within a document align to one another. Each criterion ranges from 0 to 1, with 1 being a perfect, maximized score for that particular design quality. For instance, the documents shown in Figure 4 range in alignment value from close to 0 for the most unaligned, to close to 1 for the most aligned.

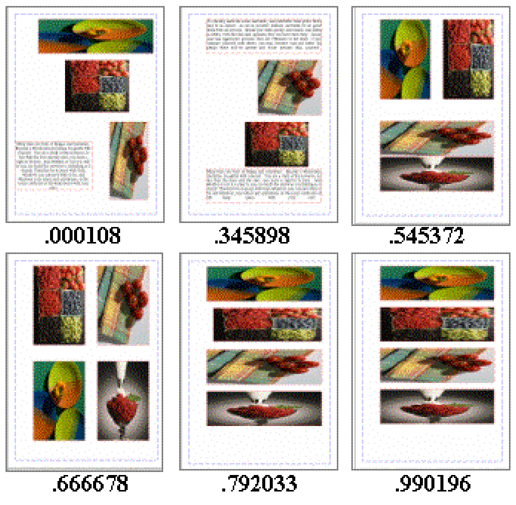


Figure 4. Examples of Alignment Scores

3.3 Optimizing Design Qualities

We then combine the individual scores for each design quality into an overall measure of how "optimal" the document is for all of the design measures combined. We currently use a simple weighted sum to compute the overall score. For example, if we are optimizing the document layout based on the qualities of balance and alignment, and each are weighted the same, we obtain the scores as shown in Figure 5, with the document on the right side considered best because it has the higher total score.

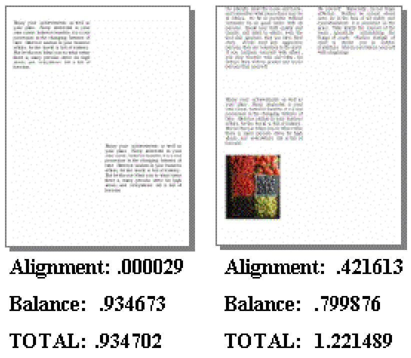


Figure 5. Example of Two Equally Weighted Objectives

If, however, we weight Balance 5 times higher than Alignment, the document on the left is considered best (score of 4.6733 versus 4.4209).

We also have "required" constraints in our document layout problem, which describe properties that must be satisfied in the resulting solution. Such required constraints specify, for example, that document elements cannot overlap, and must stay within the page margins.

The interaction between these required and desired properties results in a complex search space. Furthermore, as the number of competing objectives increases and less well-behaved objectives are considered, the problem rapidly becomes increasingly complex [8]. In our empirical testing, we have verified that our application domain indeed exhibits many of the difficult behaviors of a typical multiobjective optimization problem. This has enabled us to take advantage of several existing techniques to improve the behavior of our approach. It has also made clear those areas that are in need of further research for more improvement. The next section details how these typical difficult behaviors are exemplified in our automated document creation application domain, and how approaches from existing literature have helped mitigate those behaviors.

3.4 Methodologies for Solving Constrained Optimization Problems

The tradeoffs and intricacies of measuring which solutions are to be considered the "best" are common characteristics of the multiobjective optimization problem that makes its solving difficult.

In most optimization problems, there is a significant distinction between the search space and the feasible search space [14]. In solving constrained optimization problems, we search for a feasible optimum. During the search process, we must deal with various feasible and infeasible individuals. Much work has been done on addressing the issue of how to handle infeasible individuals within evolutionary computation techniques. Many different paradigms have emerged. For example, maintaining the feasibility of individuals in the population by means of specialized operators, imposing restrictions such that any feasible solution is "better" than any infeasible solution, considering constraints one at a time in a particular linear order, repairing infeasible solutions, and penalizing individuals that fail to meet the constraints.

3.4.1 Death Penalty Approach

The simplest of all strategies is the "death penalty" method, which rejects infeasible individuals. This method works reasonably well when the feasible search space F is convex and it constitutes a reasonable part of the whole search space S. Otherwise, such an approach has been found to have serious limitations [14]. For example, in problems where the ratio between the sizes of F and S is small and an initial population consists of infeasible individuals only, it might be essential to improve them (as opposed to 'reject' them). Moreover, quite often the system can reach the optimum solution easier if it is possible to "cross" an infeasible region.

We have found all of the above to be true in our empirical testing within our automated document creation domain. To illustrate the point, consider our three "hard" constraints: that document elements must be within the margins (C1), elements cannot overlap (C2), and elements must fit into their designated region (C3) (e.g. text cannot overflow a region). Examples of each of these constraints being violated are shown in Figure 6.

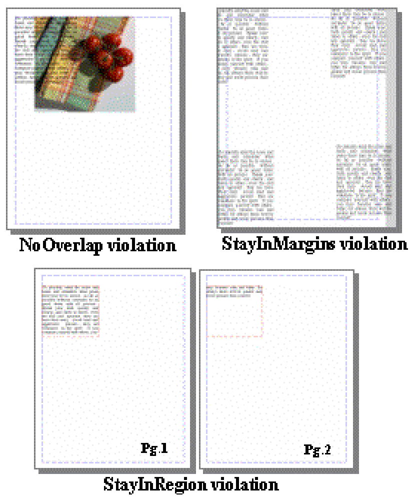


Figure 6. Examples of Hard Constraint Violations

The first example shows the text and the image overlapping, thereby violating constraint C1. The second image shows the text flowing outside the margin boundaries (shown by a dashed line), thereby violating constraint C2. The third and fourth images show two pages of the document, where the text does not fit into the region, and thus overflows onto the second page, thereby violating constraint C3.

When we used a death penalty method to reject documents that do not meet all of these hard constraints, our performance suffered. In fact, on all but the simplest of documents, the genetic algorithm failed to do anything at all – no initial solutions were generated (since none satisfied all of the constraints), and our termination criterion of stopping after 50 generations of no change was achieved without finding any solutions.

If we repaired some individuals in the initial population in order to have something feasible to start with, we typically got only the repaired individuals as solutions, with the genetic algorithm only able to make slight, insignificant improvements on these artificially repaired individuals, as shown in Figure 7. The graph shows several example runs where the initial repaired individual's score was improved only once during the course of the run, and then remained stationary for the remainder of the run.

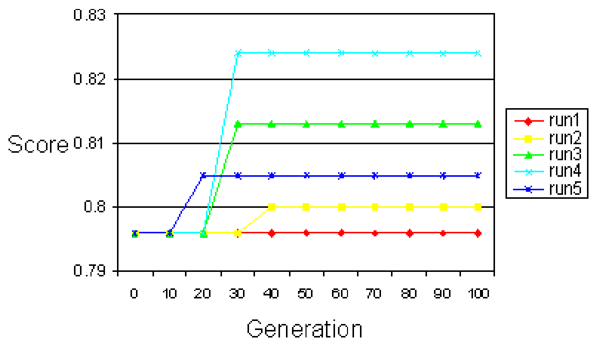


Figure 7. Behavior of GA Using Initially Repaired Individuals.

This caused the solutions to always be very much like the repair algorithm initially produced them – meeting the constraints, but otherwise not very good. That is, the remaining "soft" constraints were not optimized effectively. Figure 8 shows some of the results of such runs, showing that while the hard constraints are satisfied, the soft constraints such as appropriate image sizes and good balance are not.

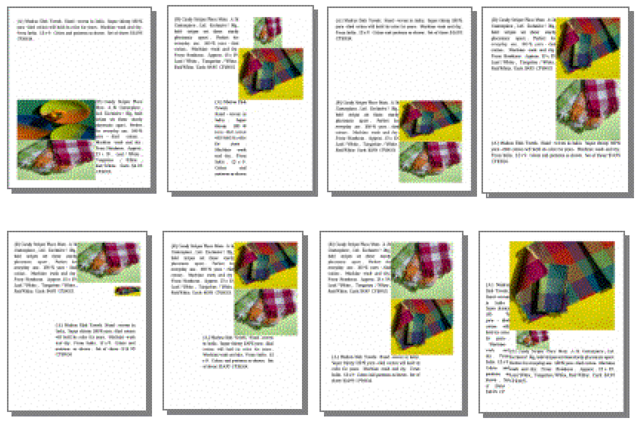


Figure 8. Examples of Artificially Repaired Individuals

Our empirical results, therefore, support the hypothesis that limiting the search to the feasible part of the search space does not always enhance search. Furthermore, our experiments also support the correlating hypothesis that GAs optimize by combining partial information from all of the population, and that thereby allowing infeasible individuals to provide information rather than just throwing them away should improve performance.

3.4.2 Penalty / MultiObjective Optimization Approach

One way to allow infeasible individuals to contribute is to penalize them rather than throw them away. We adopted a multiobjective optimization method (equivalent to penalty approaches) [9], where the objective functions o_j and constraint violation measures c_k constitute a (j+k)-dimensional vector ν:

ν: $(o_1, o_2, o_3, \ldots o_j, c_{j+1}, c_{j+2}, \ldots c_k)$.

Our problem attempts to maximize the components of ν, and thus intuitively we are rewarding individuals that satisfy the constraints and penalizing those that do not.

We analyzed the performance of the GA when moving just one, two, or all three of our "hard" constraints into the vector ν as constraint violation measures c_k, which measure how close we are to satisfying the constraint. When moving just one of the hard constraints into the vector, we already found that the genetic algorithm's productivity was improved. By productivity, we mean on average, how many improved solutions the genetic algorithm is finding per generation. With just one hard constraint moved into the vector, we got an average of .11 solutions per generation versus .07 when all were hard constraints, indicating that indeed we were improving the diversity and productivity of the search. Of course, not all of these solutions were feasible, averaging at only 8.5% of those solutions encountered during the run as being feasible, and only 50% of the end result of the run being a feasible solution.

When we moved two of the hard constraints into the vector, the situation further improved. We were getting even more productivity from the genetic algorithm (average of .16 solutions per generation), and 12.5% of those solutions encountered during the run were feasible, and 68.7% of the runs ended on a feasible solution.

Finally, moving all three of the hard constraints into the vector gave the best results, with an average productivity of .21 solutions per generation, and 39% of the solutions encountered being feasible, and 100% of the runs ending on a feasible solution. Furthermore, when all three hard constraints were considered as objective measures (weighted higher than the rest of the non-constraint objectives), the first feasible solution was found much earlier in the run than in the other two scenarios: the feasible solution was found on average with 76.4% of the run left, versus 23.8% for 1 hard constraint, and 22.2% for 2 hard constraints.

Another benefit of using the objective approach was that the scores obtained were higher – when all three hard constraints were considered as objectives, we got a highest average score of .956, versus a score of .774 when two constraints were considered as objectives, and a score of .678 when one constraint was considered an objective. These results are summarized in Figure 9 Note that all of our test runs were done with a termination criterion of 50 generations without change, a population size of 100, crossover rate of 1.0, mutation rate of 0.1, and replacement rate of 0.7.

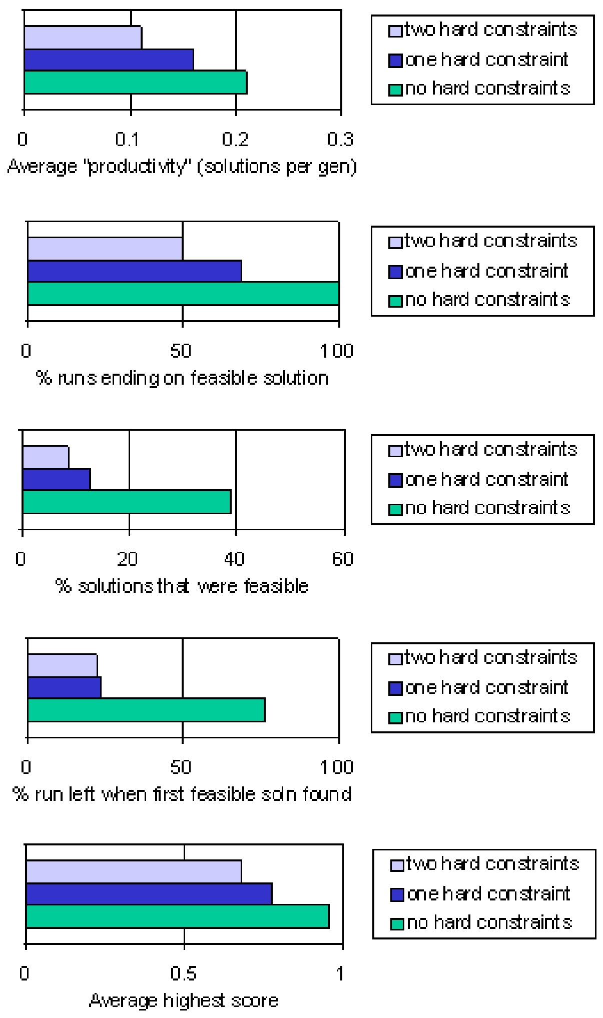


Figure 9. Results from soft/hard Constraint Tests

Our empirical results support the fact that in highly constrained problem situations, the death penalty approach provides poor results. In using the multiobjective optimization method where all of the constraints and objectives are part of the overall objective vector, a feasible solution is found faster, and the runs result in higher scores.

3.4.3 Individual Constraint / Objective Influence on Behavior of the Genetic Algorithm

During our empirical testing of the death penalty versus multiobjective/penalty approach, we also noticed that including different constraints / objectives could make the genetic algorithm behave very differently.

While examining the behavior of the genetic algorithm with each 'hard' constraint enabled in turn, we got a sense for which of our three constraints made it most difficult for the genetic algorithm to find a solution.

In our tests, the NoOverlap constraint appeared to be the most difficult to satisfy for the genetic algorithm. It had the highest average number of generations to convergence out of our 3 constraints, as well as the lowest average score, the lowest percentage of final solutions that ended up as feasible solutions, and the lowest "productivity" (i.e. average number of solutions per generation). The next hardest was the StayInMargins constraint, with the StayInRegion faring slightly better. These results are summarized in Figure 10.

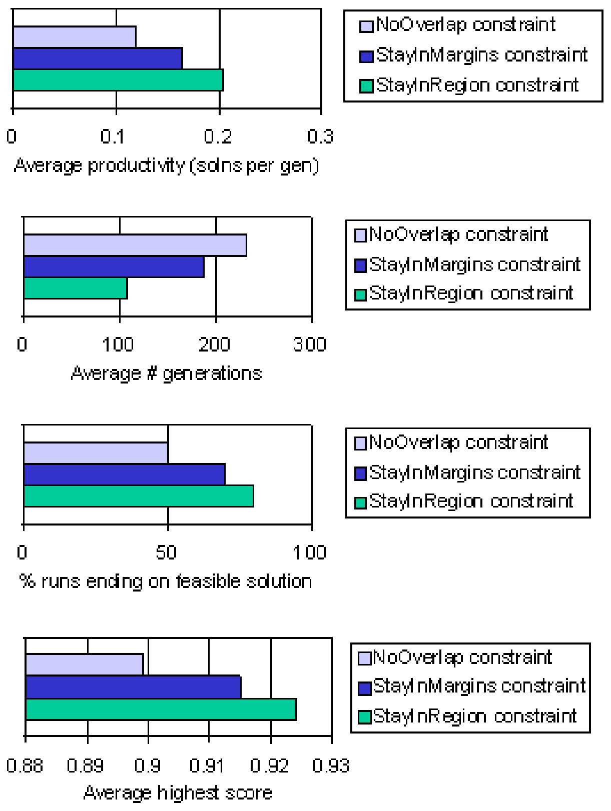


Figure 10. Comparison of Hard Constraints

Due to these differences in behaviors between the constraints, we therefore hypothesize that determining the appropriate set of constraints (minimal set, that allows the algorithm to be the most productive) is an important area for future research, which we will discuss further in Section 4.

3.4.4 GA Behavior After Finding First Feasible Solution

Another behavior that became apparent during our empirical tests using the multiobjective approach (i.e. no hard constraints) was the desirable behavior of quickly finding a feasible solution, and then spending the rest of the time optimizing the other (non-constraint) objective measures. This behavior can be seen in the examples graphed in Figure 11. Note that this behavior is in contrast to the behavior noted with the death penalty approach, where the GA was not able to improve the artificially created feasible individuals.

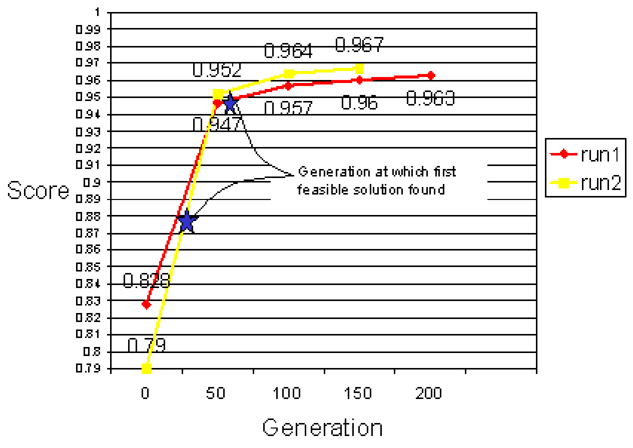


Figure 11. GA Continually Improves Early Feasible Individuals

We furthermore noted that the genetic algorithm could be made to explore very different regions of the search space while looking for the first feasible solution. The genetic algorithm was influenced in its search by the weightings on the other non-constraint objectives. It typically started out with a first feasible solution that also favored the most important non-constraint objective. This then influenced the remainder of its search, resulting in potentially very different results depending on the weightings of the objectives.

Consider the example initial iterations shown in Figure 12, where the genetic algorithm was set up with a high weight on the objective for filling up the page. We can see that while attempting to satisfy the highly-weighted "constraint" objectives (noOverlap, stayInMargins, stayInRegion), the solutions it is finding are those which also best address the next most-highly weighted criterion, in this case how much of the page is filled.

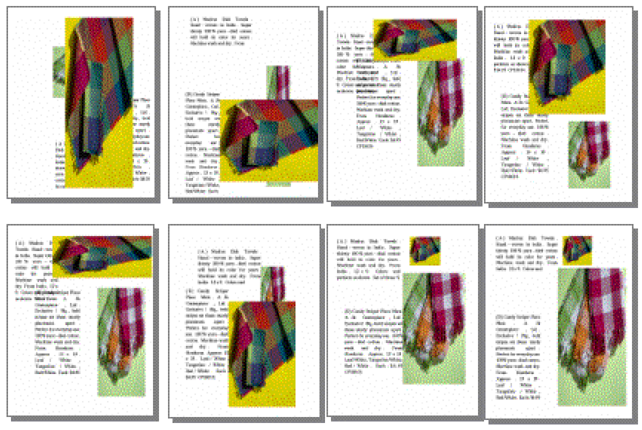


Figure 12. Sample Run With High Weight on Filling the Page

In Figure 13 we show the initial intermediate results of the genetic algorithm when set up with a high weight on maintaining the appropriate image sizes. As you can see, the results are quite different.

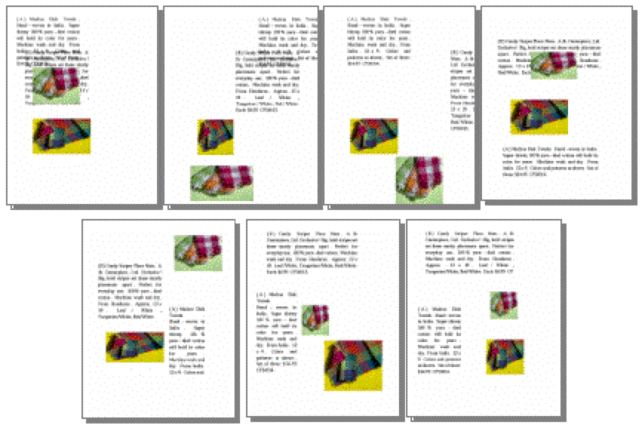


Figure 13. Sample Run with High Weight on Natural Image Size

The run shown in Figure 12 clearly favors those solutions that are better at filling the page, at a sacrifice of other objectives such as balance and appropriate image sizes, while the run shown in Figure 13 favors those solutions that keep the images the proper size, rather than fill up the page. If allowed to run long enough, the genetic algorithm does attempt to find a solution that optimizes both objectives. However, in the short run its focus is on the objective(s) with the highest weights.

In a real-life, online problem solving environment, the complexity of the multiobjective problem often prevents one from waiting until the genetic algorithm completes its run. Rather, an answer might be desired as soon as is practically possible. Thus, the first focus of the algorithm after achieving a feasible result is important, since the genetic algorithm will likely be stopped for an answer soon after that initial feasible result is found. The genetic algorithm will get potentially vastly differing results depending on which objective it focuses on after its initial successful find of a feasible solution. The weights on each individual objective also make a large difference in which objective gets the next focus.

The preferred weighting of the objectives is therefore very important, especially if we intend to ask for an early result from the GA, before it has had time to complete its run and optimize all of the objectives completely.

It is, however, a large challenge to model the utility function that describes the relative importance of each objective. It has been stated that incorporating designer preferences within a multiobjective evolutionary algorithm (MOEA) approach is a crucial area for further research [7]. The commonly used weighting scheme, which ranks the multiple objectives in terms of their importance to one another, requires in-depth information concerning various tradeoffs and valuations of each solution. This data is not commonly fully available in practice. Furthermore, in dynamic and personalized systems, the preferences might change over time, from designer to designer, and even from solution to solution.

Thus, another of our areas for future research is in gathering preference information from the designer during the optimization process. This will enable the system to respond to dynamically changing preferences, and alleviates the designer from having to specify all preferences a priori. It furthermore will enable a more directed search to useful portions of the tradeoff surface of the solution space.

4. FUTURE RESEARCH AGENDA

As we described in our empirical results, we found that how the constraints are specified (hard versus soft, which ones are considered as "hard" constraints, etc.) makes a large difference to the performance of the genetic algorithm. Thus, one area for future research is to acquire the problem constraints accurately from the user using automated constraint acquisition techniques, as will be described in section 4.1.

We also found that using the constraints as highly weighted objectives rather than "hard" constraints results in better performance. It allows the GA to focus itself on improving the other objectives after finding an initial feasible result. However, given that the initial intermediate results can be vastly different according to which other objectives are weighted highest, in real-time scenarios it is important that the weights accurately reflect the desires of the user. Modeling user desires within multiobjective evolutionary algorithms is an emerging area of research. We describe in Section 4.2 our goal of addressing the expression of preferences using a tradeoff generation methodology.

Lastly, the way in which the appropriate tradeoffs and constraints can be generated or acquired may be made simpler by allowing the user to specify which of several examples best exemplifies his desires. Thus, a third area for future research is in applying case-based reasoning techniques to ease the task of understanding user desires as well as generating appropriate tradeoffs. We provide some background on the applicability of case-based reasoning in this context in section 4.3.

4.1 Constraint Acquisition

Variable information document creation requires considerable communication amongst experts representing many diverse functions in the design of a document. For example, graphic artists may wish to define constraints on acceptable color combinations, while marketing experts may specify constraints on the style of presentation of material for a particular market segment. Furthermore, the input of end-users and customers is also required. This can give rise to significant communication difficulties since experts are often not able to "speak the same language". This has consequences for each expert's ability to articulate their requirements.

To overcome these difficulties, humans often make use of examples. In particular, humans can present and recognize examples of a requirement being satisfied or violated; however, they cannot appropriately articulate the requirement itself. For example, a customer may be trying to specify a constraint to a graphic artist without being able to use the correct technical terms for the relevant concepts. By pointing out to the artist examples of what is acceptable, the customer can be assisted in finding the right term for what he is trying to articulate. This can be regarded as an instance of interactive constraint acquisition. The customer cannot articulate his constraint but, by interacting with the artist, his examples of what he would accept or reject help to define his constraint. However, the problem with using examples as a way of articulating requirements is that we rely on our communication partner's ability to generalize correctly from them.

A number of the authors have already begun studying interactive constraint acquisition using techniques from the field of Machine Learning [16]. This work has also been motivated from experiences from the field of design and product development [17]. The approach that has been adopted for acquiring constraints from examples is based on version space learning [15]. Critical to version space learning is the notion of an hypothesis space of constraints over which a general-to-specific ordering is known. Figure 14 presents an hypothesis space of constraints that can be used to acquire simple geometric constraints.

Informally, the hypothesis space in Figure 14 has a general-specific ordering, from the top to the bottom, over a set of constraints. Each node of the hypothesis space is a conjunction of the nodes that are below it and connected to it. For example, Vertical is a conjunction of Top and Bottom. This means that if a user accepts both Top and Bottom, Vertical is also acceptable. If the user accepts both Right and Top, Orthogonal is acceptable. As a consequence of Orthogonal being acceptable, Right, Left, Top and Bottom are also, by definition, acceptable. As examples are given to an acquisition system, the hypothesis space can be used to generalize the examples in order to converge on the target concept by posing queries to the user.

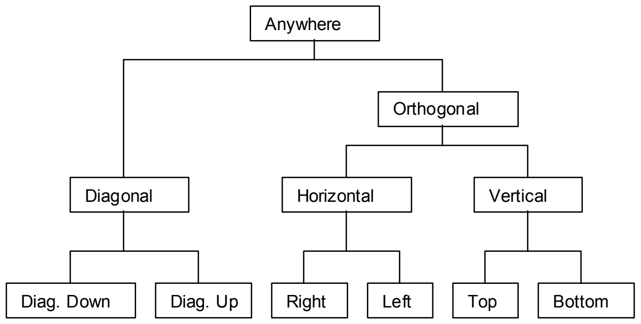


Figure 14. An example version space for placement constraints.

Our current work in this area is focused on acquiring sets of constraints from a minimum number of interactions (examples and queries).

4.2 Automated Tradeoff Generation

Typically in multiobjective optimization, one is searching for the set of nondominated, equally efficient solutions, known as the Pareto-optimal set [9]. In order to select a "most-appropriate" solution from all of the non-inferior alternatives, a decision process is necessary, along with some representation of the user's desires. Three broad classes of preference methods within multiobjective problems exist: a priori articulation of preferences, a posteriori articulation of preferences, and progressive articulation of preferences.

In a priori articulation, the user expresses preferences in terms of an aggregating function, which combines individual objective values into a single utility value. This is the typical weighted-sum approach that we have been using up until now. In a posteriori articulation of preferences, the decision-maker is presented with a set of candidate non-inferior solutions, and then chooses from that set. In progressive articulation of preferences, decision-making and optimization occur at interleaved steps. At each step, partial preference information is supplied by the decision-maker to the optimizer, which in turn generates better alternatives according to the information received [9].

We hypothesize that in a real-time, online setting, the progressive articulation approach is the most attractive, as it would enable one to quickly guide the GA towards the most preferable areas of the solution space, as well as incorporate preferences that change over time. There is of yet very little work in the area of explicitly handling preferences within evolutionary multiobjective problems [5].

Many real systems, however, involve reasoning about preferences. In such systems, a situation can be reached where all the desires/objectives cannot be achieved. At this point we could consider "tradeoffs" between the preferences and objectives in the problem. For example, when designing a document, we might reach a point where the desired balance cannot be achieved for a given alignment score. However, the user or system could accept a tradeoff: *"it is more important to have a well aligned document than one which is balanced"*. Ideally, we would like the system to suggest appropriate tradeoffs to the user. Some of the authors have begun studying the issues associated with modeling and generating tradeoffs in interactive systems [10]. In that work tradeoffs were modeled as additional crisp constraints. More recent work has extended the approach to tradeoff generation to include soft constraints and preferences [2].

In the context of variable information document creation we envisage using tradeoff generation to relax the notion of linear combination of design qualities in the objective function to a more customized objective. One consequence of applying tradeoff generation in this context is that we modify the basis upon which our GA evaluates fitness. This modifies the search space that the algorithm explores. In addition, the effect of hard constraints can be relaxed in a principled manner. Overall, we have a more differentiating tool for finding more satisfactory documents that meet the most important criteria to the maximal degree and the others to a satisfactory degree.

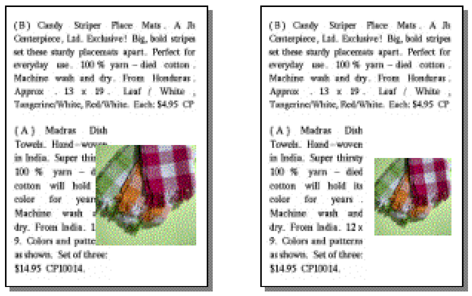
4.3 Case-Based Reasoning for an Easier Specification of User Desires

The reliance on past experience that is such an integral part of human problem solving has motivated the use of case-based reasoning (CBR) techniques. A CBR system stores its past problem solving episodes as cases that later can be retrieved and used to help solve a new problem. Case-Based Reasoning is based on two observations about the nature of the world: that the world is regular, and therefore similar problems have similar solutions, and that the types of problems encountered tend to recur [13]. When these two observations hold true, it is worthwhile to solve new problems by reusing prior reasoning.

The process by which a case-based reasoner operates has been described by Aamodt & Plaza [1] as a cyclical process comprised of the four Rs: RETRIEVE the most similar case(s), REUSE the case(s) to solve the problem, REVISE the proposed solution if necessary, and RETAIN the new solution as a new case. We can abstract the CBR process as one of recalling an old similar problem, and adapting that problem/solution to fit the new situation requirements.

We believe that a CBR process holds promise for helping the user to express his preferences and/or problem constraints, and will be synergistic with the notions of constraint acquisition and tradeoff generation described in the previous sections. For instance, CBR methodologies exist for determining which examples are relevant in which problem-solving contexts, for retrieving the appropriate examples, for representing the examples, for maintaining a consistent and coherent set of examples, and for learning new examples. As we incorporate the notions of using examples to acquire constraints and generate tradeoffs between constraints, CBR will provide building blocks upon which to do so.

During tradeoff generation we may want to present a number of examples to elicit the appropriate relationship between maintaining natural image size versus allowing overlap. A case base might contain numerous existing examples, indexed by the constraints that are violated and/or the objective weightings. For example, the documents pictured in Figure 15 could be existing cases known to work well in discriminating between maintaining natural image size and allowing overlap. Receiving information back about which one is more preferred would help indicate the relative importance of maintaining image size versus allowing overlap.


(B) Candy Striper Place Mats. A Jh Centerpiece, Ltd. Exclusive! Big, bold stripes set these sturdy placemats apart. Perfect for everyday use. 100 % yarn – died cotton. Machine wash and dry. From Honduras. Approx. 13 x 19. Leaf / White, Tangerine/White, Red/White. Each: $4.95 CP

(A) Madras Dish Towels. Hand – woven in India. Super thi 100 % yarn – d cotton will hold color for year Machine wash dry. From India. 1 9. Colors and patte as shown. Set of three: $14.95 CP10014.

(B) Candy Striper Place Mats. A Jh Centerpiece, Ltd. Exclusive! Big, bold stripes set these sturdy placemats apart. Perfect for everyday use. 100 % yarn – died cotton. Machine wash and dry. From Honduras. Approx. 13 x 19. Leaf / White, Tangerine/White, Red/White. Each: $4.95 CP

(A) Madras Dish Towels. Hand – woven in India. Super thirsty 100 % yarn – died cotton will hold its color for years. Machine wash and dry. From India. 12 x 9. Colors and patterns as shown. Set of three: $14.95 CP10014.

Figure 15. Two cases that could elicit preferences for NoOverlap versus Natural Image Size.

In our future research, we will explore how such case-based examples, along with the associated CBR techniques for indexing, retrieval, and adaptation might be used to build up the appropriate set of constraints and to model user preferences within a multiobjective optimization methodology.

5. SUMMARY

Personalized documents are becoming more and more important, especially as the amount of available information continues to explode in the digitally networked world. People need effective ways to assemble and make sense of the vast sea of information available at their fingertips.

We are working on a methodology to enable an automated and easy approach to creating customized documents. We model the problem of assembling a document from a set of components as a multiobjective optimization problem. As such, we have encountered the many typical challenges in applying an optimization approach. At the same time, the various techniques that have already been devised to address these challenges have also been of use in our application domain.

We found that the simple death-penalty approach for handling constrained optimization problems is not effective enough in our tightly constrained domain. The multiobjective approach, equivalent to penalty approaches, provides us with much better performance. We furthermore found that the objectives and constraints themselves can exert a great influence over the behavior of the genetic algorithm, and thus it is important (yet difficult) to find the right set and the right weightings for each objective.

It is these remaining challenges that have prompted us to look towards new techniques in constraint acquisition, tradeoff generation, and case-based reasoning in order to broaden the applicability of our approach to personalized document creation. We hope that addressing these challenges will allow multiobjective optimization to be more broadly and practically applied, both within personalized document creation, as well as in other application domains.

6. ACKNOWLEDGEMENTS

The collaboration between the Xerox Corporation and the Cork Constraint Computation Centre is sponsored with a grant from the Xerox University Affairs Committee. Barry O'Sullivan is also supported by Enterprise Ireland under Grant SC/02/289. Eugene Freuder is also supported by Science Foundation Ireland under Grant 00/PI.1/C075.

7. REFERENCES

[1] Aamodt A., Plaza E. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, AI Communications 7(i), pp. 39-59.

[2] Bistarelli S., O'Sullivan B., Modelling Tradeoffs Using Soft Constraints, Proceedings of the ERCIM/CologNet International Workshop on Constraint Solving and Constraint Logic Programming, Budapest, Hungary, July 2003.

[3] Borning A., Lin R., Marriott K. Constraint-Based Document Layout for the Web. Multimedia Systems, Vol. 8 No. 3, pp. 177-189, 2000.

[4] Coello C. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. Knowledge and Information Systems Journal, Volume 1(3), 129-156, 1999.

[5] Coello C. Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. Proceedings of the 2000 Congress on Evolutionary Computation, 2000.

[6] Dengler E., Friedell M., Marks J. Constraint-Driven Diagram Layout. Proceedings of the 1993 IEEE Symposium on Visual Languages, 330-335, Bergen, Norway, 1993.

[7] Fleming P.J., Purshouse R.C. Evolutionary Algorithms in Control Systems Engineering: A Survey. Control Engineering Practice, 10, pp. 1223-1241, 2002.

[8] Fonseca, C.M., Fleming P.J. An Overview of Evolutionary Algorithms in Multiobjective Optimization. Evolutionary Computation 3(1): 116, 1994.

[9] Fonseca C.M., Fleming P.J. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 1998.

[10] Freuder E.C., O'Sullivan B. Generating Tradeoffs for Interactive Constraint-Based Configuration, Seventh International Conference on Principles and Practice of Constraint Programming - CP 2001, Short paper, pp 590-594, Paphos, Cyprus, November 2001.

[11] Graf W.H. The Constraint-Based Layout Framework LayLab and its Applications. Electronic Proceedings of the ACM Workshop on Effective Abstractions in Multimedia, 1995.

[12] Kroener A. The DesignComposer: Context-Based Automated Layout for the Internet. Proceedings of the AAAI Fall Symposium Series: Using Layout for the Generation, Understanding, or Retrieval of Documents, 1999.

[13] Leake, D. Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press, 1996.

[14] Michalewicz, Z. A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. Proceedings of the 6th International Conference on Evolutionary Programming. MIT Press, Cambridge, MA, 1995, pp. 135-155.

[15] Mitchell, Tom. Generalization as search. Artificial Intelligence, 18(2):203–226, 1982.

[16] O'Connell S., O'Sullivan B., Freuder E.C. A Study of Query Generation Strategies for Interactive Constraint Acquisition, Applications and Science in Soft Computing, Advances in Soft Computing Series, Springer Verlag, 2003.

[17] O'Sullivan B., O'Connell S., Freuder E.C. Interactive Constraint Acquisition for Concurrent Engineering, Proceedings of the 9th International Conference on Concurrent Enterprising - ICE-2003, June 2003.

[18] Rousseau F., Garcia-Macias A., Valdeni de Lima J., Duda A. User Adaptable Multimedia Presentations for the WWW. Electronic Proceedings from the 8th International World Wide Web Conference, 1999.

[19] Williams, R. The Non-Designers Design Book. Peachpit Press, Berkeley, CA, 1994.