

# A Design Space Approach to Analysis of Information Retrieval Adaptive Filtering Systems

Dmitriy Fradkin  
DIMACS  
Piscataway, NJ

dfradkin@paul.rutgers.edu

Paul Kantor  
DIMACS  
Piscataway, NJ

kantor@scils.rutgers.edu

## ABSTRACT

In this paper we suggest a new approach to analysis and design of IR systems. We argue for design space exploration in constructing IR systems and in analyzing the effects of individual modules and parameters. We present results of experiments with parametric interpolation, or “homotopy”, between two systems, and show, incidentally, that the best results are not achieved at the endpoints, and may lie outside the bounding hypercube defined by our choice of parameterization. Three distinct classes of interpolation are introduced to deal with the complexities of the specific example.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; D.2.2 [Software Engineering]: Design Tools and Techniques

## General Terms

Design, Experimentation

## Keywords

Homotopy, Rocchio, Adaptive Filtering

## 1. INTRODUCTION

Constructing and tuning a good Information Retrieval (IR) system for a specific topic or corpus can be viewed as a design problem. The value of such a system can be evaluated on the basis of its performance (such as precision and recall, or other measures based on detector theory and utility) on some evaluation set, considered in relation to the time and computational resources it requires to accomplish its various tasks.

In many other technologies, such as HVAC (heating, ventilation and air-conditioning) layout design [14], VLSI chip design [3] and aerospace vehicle design ([12], [6], [18], [13]) ap-

propriate methods for automatically exploring design space and seeking optimal designs have been developed.

Historically, in the design of IR systems, however, ad hoc approaches prevail. Typically, some “system” X described in the literature is taken as a basis, numerous discrete choices regarding term representation and matching are made and a few, model-specific, parameters are tuned. Many of the details are not spelled out in any report. The system may be simply referred to as a “modification” of X, thus obscuring the changes made. This widespread approach, on the one hand tends to obscure connections between models, and on the other hand hides significant differences between systems bearing the same name.

We cannot draw an exact parallel to the modeling of physical systems, as described above. In particular, it might be argued that, while an airplane or chip design *evaluation* is based on scientific physical principles, the experimental evaluation of an IR system depends on the particularities of the topics, judges, and corpus examined. Thus an experiment is not as precise an indicator of its performance on another task. However, it is reasonable to assume that there are some broad categories of corpora with which the system should work well. Indeed, without such an assumption, it would be hard to justify any of the large projects such as TREC (<http://trec.nist.gov/>), CLEF [4] or NTCIR [9]. For example, if a system is evaluated on a news corpus, using business questions, it is reasonable to believe that for many other news corpora and similar questions, its performance would be comparable. Such a system would not necessarily perform well on a corpus of scientific abstracts, and the reverse also holds. Systems for those tasks may require different tunings, or even different designs. But, analogously, the design of a fighter aircraft is quite different from that of a passenger airliner.

We propose that the task of designing an IR/AF (Adaptive Filtering) system should be approached in the same principled way as the task of designing a chip or an airplane. Instead of viewing different models and representations in isolation from each other, we should try to situate them as points in a *continuous and connected* parameterized design space, to which existing design space exploration and optimization methods should be applicable.

The method of homotopic optimization [17] was originally developed to solve difficult problems by smoothly distorting them into easy problems. Ultimately we would hope to do that for the IR/AF problem. The present note may be regarded as a first step in this program. The idea of homotopy provides a useful way of connecting the isolated points ex-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'04, November 8–13, 2004, Washington, DC, USA.  
Copyright 2004 ACM 1-58113-874-1/04/0011 ...\$5.00.

amined by specific teams, so that the larger space may be explored. When procedures for exploration have been defined, it will be possible to consider important questions such as the convexity of the objective function in the design space. We note that the choice of the parameterization will affect the convexity, as a monotone transformation may convert a convex function into one that is not, and vice versa.

We illustrate our approach by “connecting” two systems which, as it happens, share the same “common name”. However, as will be seen in the following, this common name obscures a set of deep differences which are, we believe, representative of the differences between methods having different “common names”. We find satisfactory ways to interpolate the methods with regard to all of the design choices. As a byproduct, we examine the objective function along the principal diagonal (a kind of “linear bridge” between two islands in design space) and find somewhat improved performance at a point along that bridge. This is, of course, not surprising, since a line in the design space is unlikely to have one of its endpoints closer to the optimum than any of its interior points. We also show that in a vicinity of this “good diagonal point” further improvement is possible. But we caution that this exploration of the TREC2002 Adaptive Filtering task is a specific instance of the problem to be solved, and do not lay too much emphasis on the specific design point that is best for this instance. We examine the stability of our results in two important ways, and find that the fundamental conclusions are unaltered by (a) a random change in the order in which information is presented to the adaptive system and (b) a change to problems from a substantially different corpus (the OHSUMED collection [8]).

The model developed here will also make it possible to study the relation between *method* and *problem*. Specifically, for each of many instances of the problem, one may find the optimizing set of parameters. One may then study that set of parameters, which are now points in the space representing the systems. If these points cluster into a single region, we may say that it defines the “solution to the problem” in the given design space. If the solutions form two distinct clusters then we should conclude that the instances represent two meaningfully different types of problems, and should look for the characteristics of that difference. This “solution driven” approach is complementary to, and more scientifically executable than the large literature on “natural types of problems” which seeks to divide problems according to the such diverse factors as the “the purpose of the inquiry” and “the age of the user”. When the NRRC RIA [7] data become generally available they will provide an excellent corpus on which to conduct these explorations.

## 2. OVERVIEW

In this text capital case letters denote sets, lower case letters denote scalars and vectors. We use letter  $t$  to denote a term,  $d$  - a document,  $q$  - query. Bold face brackets  $(\cdot)$  are used to denote inner product.

### 2.1 Types of Parameterization

Suppose that we have two real-valued functions  $f, g$  defined on some space  $X$ . Then for  $\lambda \in [0, 1]$ , the function  $f_\lambda = \lambda f + (1 - \lambda)g$  is clearly an interpolation between  $f$  and  $g$ . However, when a function seems to be defined in terms of a “natural” parameter, we may wish to interpo-

late that parameter. An example is the use of so-called “term weights”, which represent a diagonal metric in “term space”. In some formulations, the proposed weight appears linearly. In others, it appears raised to the second power, with a gloss suggesting that “both query terms and document terms are being weighted”. From the mathematical point of view, there is nothing special about using either the first or second power of some number  $\omega(t)$ , and a natural interpolation is  $\omega(t)^{1+\lambda}$ . Even more complex transformations suggest themselves when the function in question is defined in a piecewise fashion, or by an iterative process.

When any such interpolation is defined, it is natural to ask whether the endpoints are in any sense “privileged”. We believe that there are three broad cases. In one case, the endpoints have been chosen quite arbitrarily and it is easy to “move” the endpoints so that the original interval becomes a sub-interval of the new one. In this case it may or may not be possible to extrapolate the results by extending a parameter outside of the unit interval. In the second case, the endpoints are in some sense “natural”, and it is hard to imagine extending a system beyond them. Of course, when we believe that the endpoints are not such natural points, we should consider whether we can identify natural endpoints at the same time that we do the analysis. In the third case, it may seem sensible to extend the parameter outside the unit interval, with no limitation.

### 2.2 Organization of this paper

The paper is structured in the following way. In Section 3 we discuss in some detail the TREC Adaptive Filtering task, data and utilities. Section 4 is the core of this paper: it contains the description of homotopy between two very different “Rocchio” systems, involving a total of 10 parameters. In Section 5 we analyze the performance of the “intermediate” systems and the effects of interpolation parameters. Finally, in Section 6 we present our conclusions.

## 3. TREC-11

A detailed description of TREC-11 adaptive filtering task and of results can be found in [10]. Essentially, systems are given a small number (3) of examples of relevant documents for a topic, and a larger body of materials similar to the ones that will be used to test the system. Thereafter, documents are presented in a pre-assigned sequence, and the system must decide whether to send them for “judgment”. Only by sending a document for judgment can the system learn whether it is relevant. A document in the test set could relate in one of the three ways to a given topic. It could be positive for a topic, negative for a topic or unjudged. During the testing stage, or the adaptive filtering itself, the classifier may be modified, based on its performance, to better respond to future documents. The utility function gives 2 units for a relevant document and charges 1 unit if it is not.

The two systems that we chose to interpolate between are DIMACS Rocchio [2] (TREC run dimacs11aAPQ) and the Rocchio system constructed by Chinese Academy of Science (CAS) [16] (TREC run ICTAdaFT11Ua). A Rocchio model uses the original query text, the known relevant examples, and information about the centroids of the vectors representing the set of judged relevant documents and the set of judged but not relevant documents. Despite the fact that both systems are versions of the “same” method (Rocchio [11]), there was a significant difference in the performance.

The CAS system had the best results in adaptive filtering, while DIMACS's Rocchio performance was no more than mediocre.

### 3.1 Data

The dataset used in TREC-11 adaptive filtering is the well-known RCV1 corpus provided by Reuters for research purposes [1]. It consists of approximately 800,000 news stories covering a time period of a year in 1996-1997. The first 6 weeks' items, 20 August through 30 September were used as a training set. The size of the training set was approximately 23,000 documents. The rest of the documents formed the test set.

There were 100 topics. The first 50 of these were constructed by assessors at NIST and were known as "assessor" topics. The remaining 50 topics, known as "intersection" topics, were build as intersections of pairs of Reuters categories.

Each topic had exactly three labeled positive (relevant) documents in the training set. All other documents in the training set were unlabeled with respect to that topic. However, the large size of the training set allowed evaluation of term frequency statistics.

### 3.2 Utilities

Performance of a system on each topic was evaluated according to a truncated version of a linear utility measure.

We need some terminology. Let  $T$  be the test set. Let  $T^+$  be the set of all positive documents for a topic in the test set. Let  $D$  be a set of all submitted documents, and let  $D^+$  be the set of positive documents submitted,  $D^-$  the set of negative documents submitted and  $D^u$  as the set of unlabeled documents submitted. As usual,  $|S|$  denotes the number of elements in a set  $S$ .

Then, the T11SU measure is defined as:

$$\begin{aligned} \text{T11SU} &= \frac{\max(\text{T11NU}, -0.5) + 0.5}{1.5} \\ \text{T11NU} &= \frac{2|D^+| - (|D^-| + |D^u|)}{2|T^+|}. \end{aligned}$$

## 4. ROCCHIO CLASSIFIER FOR ADAPTIVE FILTERING

This Section presents the central part of our work. We discuss the choices involved in constructing a Rocchio classifier. We pay particular attention to design decisions distinguishing the DIMACS method and the CAS method. The key innovation is a method to interpolate between two systems which, sometimes, make use of very different constructs. It is our belief that the homotopic approach described here can be useful in analyzing IR system design choices and in constructing better systems.

The interpolation is controlled by 10 parameters  $\lambda_*$ . Each of these parameters takes values in the interval from 0 to 1, with value 0 corresponding to the DIMACS choice of a particular component, and the value 1 corresponding to the CAS choice.

### 4.1 Term Representation

Both DIMACS and CAS use the standard bag-of-words representation for documents. That is, each document is represented by a real-valued vector, labeled by terms. The entry for each term is derived from the number of times that

term occurs in this document. This is "term frequency" of a term with respect to a document or a query is denoted by  $f'(t, d)$ .

Both DIMACS and CAS represent a term by the same transform of the term frequency:

$$f(t, d) = \begin{cases} 1 + \log(f'(t, d)), & f'(t, d) > 0 \\ 0, & f'(t, d) = 0 \end{cases}$$

### 4.2 Term Weighting

Based on the training corpus it is possible to derive a measure called the document frequency,  $i'(t)$  - number of documents that contain  $t$ . The so-called "term frequency" is usually weighted by some function of the term's document frequency. The intuition behind this approach is that the most frequent terms are not particularly useful in classification and should be weighted down, while rarer terms should have more weight.

DIMACS uses the following formula:

$$i_D(t) = \log\left(\frac{1 + |T|}{1 + i'(t)}\right)$$

CAS uses formula:

$$i_C(t) = \begin{cases} \log\left(1 + \frac{|T|}{i'(t)+1.0}\right), & i'(t) \geq 6 \\ 0, & i'(t) < 6 \end{cases}$$

The interpolating system weights the terms as follows:

$$i(t, \lambda_i) = i_D(t)(1 - \lambda_i) + i_C(t)\lambda_i.$$

The interpolating parameter is called  $\lambda_i$  where the letter "i" refers to the inverse document frequency. While it is possible in principle to update the idf weight of a term during adaptive filtering, neither the DIMACS nor the CAS method implemented this option.

### 4.3 Computing Scores

In Rocchio methods, a query is represented by a vector  $q$ , much as a document is. We will describe how the vector  $q$  is constructed in Section 4.8. In the current section we only discuss how score of a document, given a query vector, is computed.

DIMACS computes the score of a document in relation to the query as an inner product of their representative vectors:  $s_D(d, q) = (d, Wq)$ .

CAS computes the score as the cosine of the angle between these vectors:  $s_C(d, q) = (d, Wq) / |d||Wq|$ , where  $|d|$  denotes Euclidean norm of vector  $d$ , i.e.  $|d|^2 = \sum_{t \in d} f^2(t, d)$ .

In both cases  $W$  is a diagonal matrix whose entries are interpreted as the weights of the components. However, along the interpolation in  $\lambda_i$  the DIMACS component-wise weights are  $i(t, \lambda_i)$  for term  $t$ , while the CAS weights are  $i^2(t, \lambda_i)$ .

The interpolation method computes the component-wise weights as follows:

$$w(t, \lambda_w) = (i(t, \lambda_i))^{1+\lambda_w}$$

Therefore, it is more accurate to write  $s_D(d, \lambda_w)$  and  $s_C(d, \lambda_w)$ . Interpolation between  $s_D(d, \lambda_w)$  and  $s_C(d, \lambda_w)$  will be described in Section 4.5.

Obviously other ways can be chosen to interpolate between the first and second powers of a number, but this seems to us the most "natural".

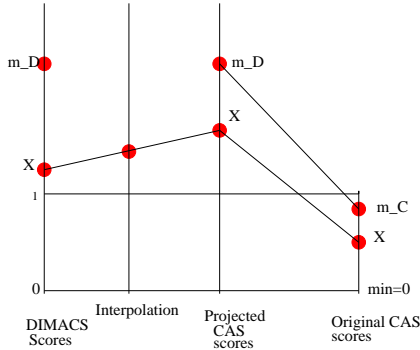


Figure 1: Interpolation of Scores and Thresholds

#### 4.4 Setting the Classifier Threshold

A classifier “decides” whether a document is relevant based on comparison of document score with a topic-specific threshold. DIMACS and CAS have different methods for computing this threshold on the basis of training documents and for updating it on the basis of the test documents.

The DIMACS method recomputes a threshold  $\tau_D(q, i)$  after submitting and receiving a relevance judgment for an  $i$ -th document. The scores for all previously seen labeled and pseudolabelled documents are recomputed and an optimal threshold (with respect to utility) is chosen.

CAS starts with fixed initial threshold values (0.19 for assessor topics and 0.27 for intersection topics; note that the use of a cosine scale keeps all scores in the interval  $[0, 1]$ ). They set  $\delta = 0.005$  and update as follows:

$$\tau_C(q, i) = \begin{cases} \tau_C(q, i-1) + \delta, & \text{if T11U}_q < 0 \text{ after a submission} \\ \tau_C(q, i-1) - \delta, & \text{if } z_{i-1} = 60000 \\ \tau_C(q, i-1), & \text{otherwise} \end{cases}$$

where  $z_i$  is a counter, keeping track of the number of documents seen since the last submission:

$$z_i = \begin{cases} z_{i-1} + 1, & \text{if } d_{i-1} \text{ is not submitted} \\ 0, & \text{if } z_{i-1} = 60000 \text{ or } d_{i-1} \text{ is submitted} \end{cases}$$

Since for each moment in time there is only one value of  $\tau_C$  and  $\tau_D$  for a given query, we drop  $i$  from the notation so the CAS threshold is denoted by  $\tau_C(q)$ , and the DIMACS threshold is  $\tau_D(q)$ .

Interpolation between  $\tau_C(q)$  and  $\tau_D(q)$  is described Section 4.5.

#### 4.5 Interpolation of Scores and Thresholds

Interpolation of thresholds and scores proved to be somewhat tricky. The difficulty originates with the decision about normalization. DIMACS does not normalize scores while CAS does. Furthermore, the methods for adaptively adjusting thresholds (and for setting initial thresholds) are very different. DIMACS estimates initial thresholds from training data and readjusts them after every document judgment, while CAS uses fixed initial thresholds and changes them only when the method starts performing badly, either by

submitting a lot of negative documents or by not submitting any.

We interpolate for each topic separately. For each topic, we compute the scores of documents according to the DIMACS and the CAS approaches (with all the parameters being the same, the difference is in the normalization). We thus can obtain the largest DIMACS and CAS scores for documents for this topic ( $m_D$  and  $m_C$  respectively). We know that the smallest possible score is 0. This enables us to define a linear map from CAS score space to DIMACS score space. The result of the mapping serves as one endpoint of interpolation, while DIMACS score serves as the other. This idea is illustrated in Figure 1.

Therefore, given a CAS score  $s_C(d, \lambda_w)$ , we consider its DIMACS equivalent to be:

$$\psi(s_C(d, \lambda_w)) = \frac{m_D}{m_C} s_C(d, \lambda_w). \quad (1)$$

The mapping of a CAS threshold is done using the same function  $\psi$ . Since the mapping is linear, it does not affect the results of filtering. Therefore, a classifier that simply performs this mapping of CAS scores would have the same performance as without this mapping. It remains only to interpolate between the equivalent image of the CAS threshold and the DIMACS threshold.

The interpolation between CAS and DIMACS scores or thresholds  $s_C$  and  $s_D$  is done as follows:

$$s(s_C, s_D, \lambda_S) = \psi(s_C)\lambda_S + s_D(1 - \lambda_S)$$

where  $\lambda_S \in [0, 1]$  is the interpolation parameter.

#### 4.6 Set Representation

DIMACS and CAS create vectors to represent the sets (of positive documents, negative documents, etc.) in different ways. DIMACS represents these sets by the averages of the vectors in these sets. CAS uses the sums of all the vectors in these sets.

In the interpolating system, vectors representing each set are given by:

$$v(S, \lambda_r) = \frac{1}{1 + (1 - \lambda_r)(|S| - 1)} \sum_{a \in S} a.$$

$v(S)$  is a vector, and  $\lambda_r$  is an interpolation parameter.

#### 4.7 Pseudo-labeled Documents in Training

As mentioned above, the training set contains only 3 positive examples for each topic. It also contains a large number of documents for which labels are not available.

This suggests the idea of trying to find examples in the training set that are “sufficiently similar” or “sufficiently dissimilar” to the positive examples. We shall refer to these examples as “pseudo-positive” and “pseudo-negative”. This allows us to extend the set of documents that serve as basis for the classifier.

CAS does not make use of pseudo-labeled documents during the training stage.

DIMACS obtains pseudo-labeled documents from the unlabeled training data. This approach relies on parameters called “density” ( $d_+$  and  $d_-$ ) and “proportion” ( $p_+$  and  $p_-$ ). Essentially, among the top  $p_+|D|$  and bottom  $p_-|D|$  unlabeled documents sorted by similarity to  $q^{terms}$  and  $v(D^{i+})$ ,

pseudo-positives and pseudo-negatives are chosen uniformly at random with probabilities of  $d_+$  and  $d_-$  respectively.

In our systems all these parameters ( $d_+$ ,  $d_-$ ,  $p_+$  and  $p_-$ ) are interpolated to 0 using  $\lambda_p$ . Note that according to our conventions the endpoint  $\lambda_p = 1$  corresponds to the CAS choice where each parameter has the value 0.

This same idea must also be applied to unjudged documents during the filtering stage. We discuss this in greater detail in Section 4.9.

## 4.8 Query Initialization

The Query is initially represented as a vector of terms describing it:  $q^{terms}$ . Initialization in both the DIMACS and CAS models follows the generalized formula:

$$q^{init} = \alpha' q^{terms} + \beta' v(D^{i+}) + \gamma' v(D^{i-}) + x' v(D^{i+}_p) - y' v(D^{i-}_p)$$

where  $D^{i+}$  is the set of positive documents,  $D^{i-}$  is the set of negative documents,  $D^{i+}_p$  is the set of pseudo-negative documents and  $D^{i-}_p$  is the set of pseudo-positive documents at the initialization stage. All these documents are from the training data only.

The process of constructing  $q^{init}$  thus consists of two stages. In the first stage, only query terms and positive documents are used. In the second stage, unlabeled documents are sorted by their score with respect to  $q^{init}$ . Pseudo-labeled documents are chosen and incorporated into the initial query, according to the above formula. Finally, the threshold is chosen to maximize utility, as described in Section 4.4.

Since no negative training documents were given for training,  $\gamma' = 0$  for both the DIMACS and CAS methods.

In the CAS model,  $\alpha' = 3$ ,  $\beta' = 1$ ,  $x' = 0$  and  $y' = 0$ . In other words, no pseudo-labeled documents are chosen at initialization, and only the first stage of constructing  $q^{init}$  is needed.

In the DIMACS model,  $\alpha' = 1$  and  $\beta' = 1$ . The situation is somewhat more complicated for  $x$  and  $y$ , which are defined as:  $x' = 2/|D^{i+}_p|$  and  $y' = 5/|D^{i-}_p|$ . (The DIMACS method for finding documents in  $D^{i+}_p$  and  $D^{i-}_p$  is discussed in Section 4.7).

We interpolate between the DIMACS and CAS values for  $\alpha'$ :  $\alpha' = 3\lambda_\alpha + 1(1 - \lambda_\alpha)$ . The parameter  $\beta'$  doesn't need to be interpolated. The parameters controlling  $x'$  and  $y'$  are all interpolated from their respective values to 0 using the interpolation parameter  $\lambda_p$  (from Section 4.7).

$$q^{init}(\lambda_\alpha, \lambda_p) = (3\lambda_\alpha + (1 - \lambda_\alpha))q^{terms} + v(D^{i+}) + (1 - \lambda_p)x'v(D^{i+}_p) - (1 - \lambda_p)y'v(D^{i-}_p)$$

## 4.9 Unjudged Documents In Test

A document that is submitted by a system but for which no label is available is called "unjudged". Unjudged documents can in principle be also assigned a pseudo-positive or pseudo-negative label by the system.

The DIMACS system ignores unjudged documents. The CAS system however labels unjudged documents as pseudo-negative if their score is less than 0.6.

In the combination system we interpolate linearly between 0 and 0.6:

$$\tau_u(\lambda_u) = 0.6\lambda_u + (1 - \lambda_u)0 = 0.6\lambda_u$$

## 4.10 Query Update

The system can adjust its behavior during filtering based on the labels it receives for submitted documents. The update for both the CAS and DIMACS system consists of changing the query vector and adjusting thresholds. The methods for adjusting thresholds used by both systems were discussed in Section 4.4.

The rule for constructing the query vector on the basis of test documents can be stated as follows:

$$q = \alpha q^{init} + \beta v(D^+) - \gamma v(D^-) + x v(D^+_p) - y v(D^-_p)$$

For CAS, the parameter values are:  $\alpha = 1$ ,  $\beta = 1$ ,  $\gamma = 1.8$ ,  $x = 0$ ,  $y = 1.3$ . For DIMACS, the parameter values are:  $\alpha = 1$ ,  $\beta = 1$ ,  $\gamma = .125$ ,  $x = 0$ ,  $y = .0125$ .

The parameter  $x$  can be ignored since neither system uses "pseudo-positives". The parameters  $\alpha$  and  $\beta$  have the same values in DIMACS and CAS and are not interpolated. The parameters  $\gamma$  and  $y$  are interpolated using  $\lambda_\gamma$  and  $\lambda_y$  in a straightforward way.

Therefore the final formula is as follows:

$$q(\lambda_\gamma, \lambda_y) = q^{init} + v(D^+) - (1.8\lambda_\gamma + 0.125(1 - \lambda_\gamma))v(D^-) - (1.3\lambda_y + 0.0125(1 - \lambda_y))v(D^-_p)$$

## 4.11 Quitting Strategy

The DIMACS method uses the following quitting strategy: if, after submitting 50 documents, the utility is negative, no more documents will be submitted for this topic. (The value of 50 was chosen after experiments with an independent training corpus). The CAS method never stops submitting documents, i.e. its "give-up" point is at infinity.

The quitting strategy can be described as function of a parameter  $\epsilon$ : a system quits if, after submitting  $\tau_q = 1/\epsilon$  documents, the utility is negative. For DIMACS,  $\epsilon_D = 0.02$ , and for CAS  $\epsilon = 0$ .

This allows us to interpolate between these values as follows:

$$\tau_q(\lambda_q) = \frac{1}{0.02(1 - \lambda_q)}$$

The actual values of  $\tau_q$  for  $\lambda_q$  in  $\{0, 0.2, 0.4, 0.6, 0.8\}$  are  $\{50, 62.5, 83.3, 125, 250\}$ . When  $\lambda_q = 1$ , the quitting strategy is not applied.

## 4.12 Limits of Interpolation Parameters

As noted in Section 2.1, interpolation parameters can, in some cases, be extended beyond the interval  $[0, 1]$  without difficulty. For a number of the parameters in our representation, that extension eventually encounters limits because the corresponding parameters of the training procedure or the Rocchio model cannot be negative. Some parameters also have additional constraints. The resulting intervals are rather complex, and are shown in Table 1.

We would like to note that different regions of Rocchio parameter space can be explored using our method by moving the endpoints (instead of extending the interpolation variables outside of  $[0, 1]$ ). If the new endpoints are well-defined, so will be all the points between them.

$\lambda$	0.0	0.2	0.4	0.6	0.8	1.0	CAS
Average T11SU, $\lambda_q = 1$	0.033	0.103	0.260	0.364	<b>0.404</b>	0.394	0.405
Average T11SU	0.113	0.139	0.263	0.364	<b>0.404</b>	0.394	0.405

**Table 2: Results of diagonal interpolation between DIMACS and CAS Rocchio.** Results in the first row were obtained without quitting strategy. Results in the last row were obtained with interpolation of  $\lambda_q$  (Section 4.11). The entry in column 0.0 in the last row corresponds to the actual DIMACS TREC 11 submission (run dimacs11aAPQ). The last column contains the result of CAS method. Our  $\lambda = 1$  results do not match it - please refer to text for a discussion of this.

$\lambda$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Average T11SU, $\lambda_q = 1$	0.005	0.008	0.051	0.184	0.361	0.445	0.467	0.474	0.463	0.464	0.482
Average T11SU	0.138	0.131	0.132	0.164	0.319	0.440	0.467	0.474	0.463	0.464	0.482

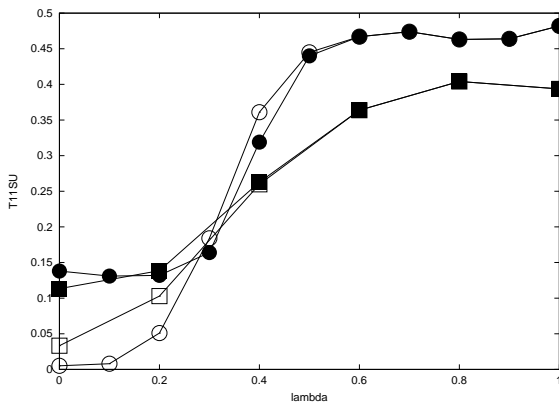
**Table 3: Results of diagonal interpolation between DIMACS and CAS Rocchio on OHSUMED dataset.** Results in the first row were obtained without quitting strategy. Results in the last row were obtained with interpolation of  $\lambda_q$  (Section 4.11).

Parameter	Limits
$\lambda_\alpha$	$[-0.5, \infty)$
$\lambda_\gamma$	$[-\frac{5}{67}, \infty)$
$\lambda_p$	$(-\infty, \min(\frac{1}{x^r}, \frac{1}{y^r})]$
$\lambda_y$	$[-\frac{1}{103}, \infty)$
$\lambda_u$	$[0, \frac{5}{3}]$
$\lambda_i$	$[-\min_{t \in T} \frac{i_D(t)}{i_D(t) + i_C(t)}, \infty)$
$\lambda_w$	$(-\infty, \infty)$
$\lambda_S$	$[0, 1]$
$\lambda_r$	$[0, \infty)$
$\lambda_q$	$[0, 1]$

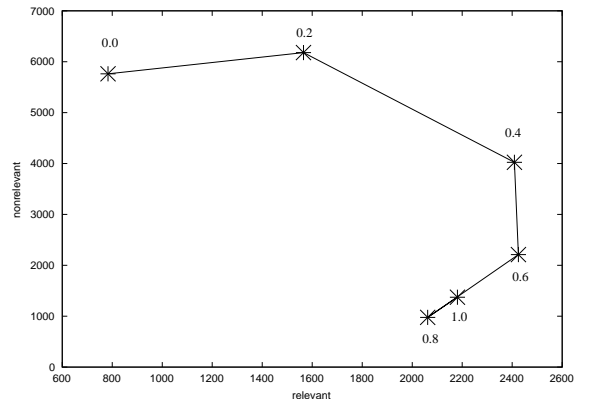
**Table 1: Limits on interpolation variables**

Parameter	Parameter Values				
	0.6	0.7	0.8	0.9	1.0
$\lambda_\alpha$	0.405	0.407	...	0.404	<b>0.407</b>
$\lambda_\gamma$	0.409	<b>0.412</b>	...	0.404	0.400
$\lambda_p$	0.381	0.392	...	0.407	<b>0.407</b>
$\lambda_y$	0.404	0.405	...	0.402	0.403
$\lambda_u$	0.401	0.401	...	0.405	0.405
$\lambda_i$	0.404	0.404	0.404	0.404	0.404
$\lambda_w$	0.403	0.403	...	<b>0.409</b>	0.405
$\lambda_S$	<b>0.410</b>	0.408	...	0.406	0.397
$\lambda_r$	0.401	0.405	...	<b>0.409</b>	0.395
$\lambda_q$	0.404	0.404	...	0.404	0.404

**Table 4: contains results of exploration of region around  $\lambda = 0.8$ .** Cell  $ij$  contains the average T11SU score for the system described by  $\lambda$  having value 0.8 and parameter in row  $i$  having the value in the second row of column  $j$ . (Therefore all values in column 0.8 are the same and are given only once.)



**Figure 2: Plot of Tables 2 and 3.** The lines with squares show results on TREC-11, while the lines with circles show results on OHSUMED data. The lines with filled figures correspond to the second rows (interpolating  $\lambda_q$ ). The lines with the empty figures correspond to the first rows ( $\lambda_q = 1$  - no quitting strategy).



**Figure 3: Plot of the number of relevant and non-relevant documents (x-axis and y-axis respectively), as  $\lambda$  varies from 0.0 to 1.0.** The points are labeled with the corresponding value of  $\lambda$ .

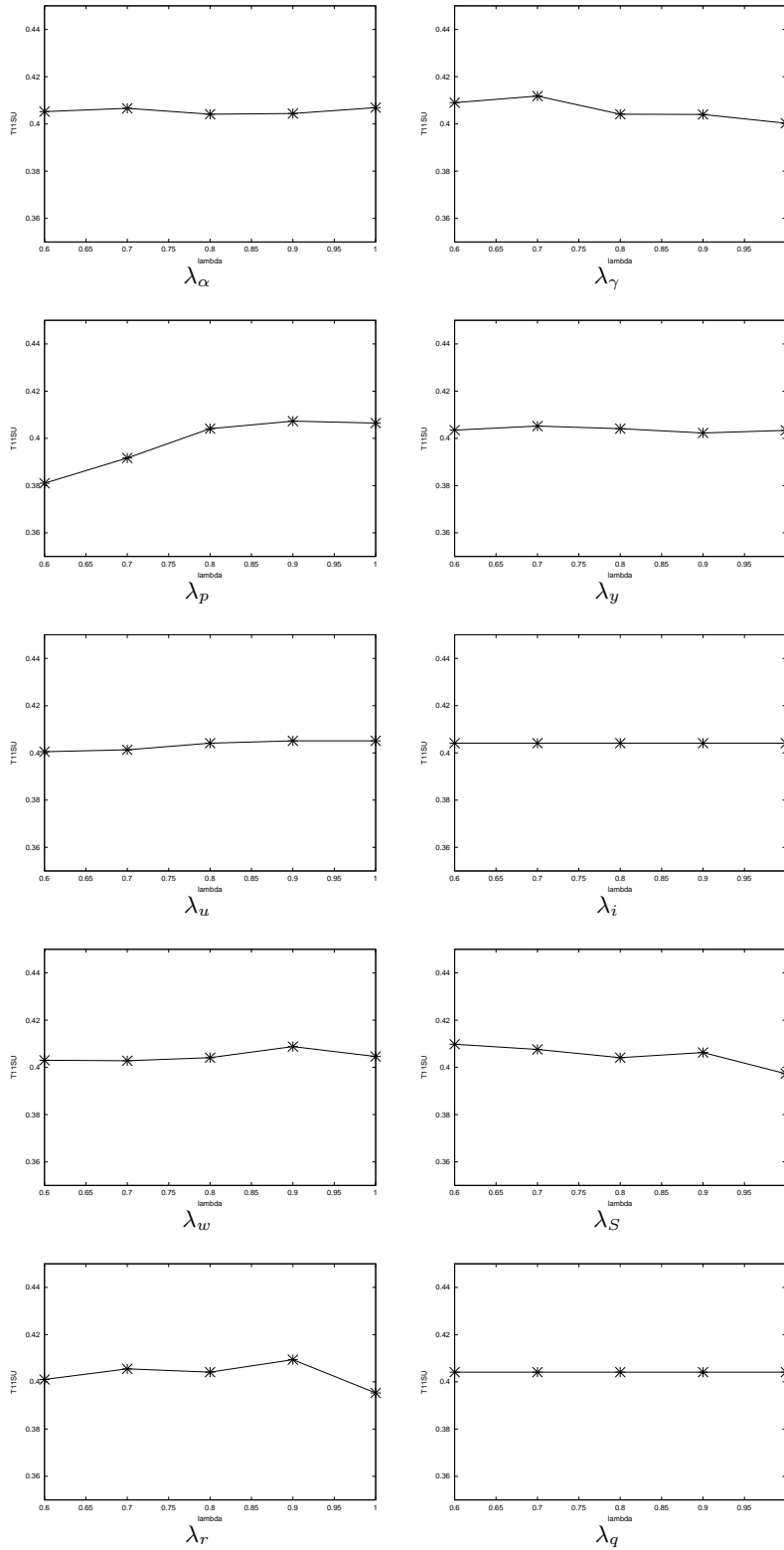


Table 5: Graphs of T11SU measure as a function of individual interpolation parameters, while all other parameters are fixed at 0.8. The x-axis interval shown on each plot is from 0.6 to 1.0; y-axis interval is from 0.35 to 0.45.

## 5. RESULTS

Numerous experiments were run using a set of relevance labels released for TREC-11 in summer 2002, and evaluated with an updated set of labels, released after the TREC-11. Therefore the scores given here should be directly comparable with the scores in the TREC-11 reports.

We examined a direct path between the DIMACS and CAS models (a line in design space connecting DIMACS and CAS endpoints). This was done using a single global interpolation parameter  $\lambda$  to control all individual interpolation parameters. In other words, unless otherwise specified all interpolation parameters were set equal to  $\lambda$ . We refer to the path between the two systems described by this global parameter  $\lambda$  as the “diagonal.”

Results along the direct diagonal path are given in Table 2 and plotted in Figure 2. We note that our endpoint does not reach the actual CAS result at TREC. It is likely that this discrepancy in results is due to the use of a different parser, stemmer and/or a stoplist. Such a difference would lead to a somewhat different vocabulary and have a small but measurable effect on the performance. It is also possible that some aspect of the CAS system have been misunderstood by us in our communications with the CAS researchers [15].

The first row of the Table 2 contains results obtained without use of quitting strategy (that is,  $\lambda_q = 1$ ). The second row contains results obtained with interpolation of  $\lambda_q$ , i.e.  $\lambda_q = \lambda$ . The differences between the two rows suggest that the main effect of quitting strategy is to protect the system from doing extremely poorly, and that when a system performs well on its own ( $\lambda = 0.6, 0.8, 1.0$ ), the quitting strategy with  $\tau_q \geq 125$  does not affect the score. This is easily visible on Figure 2.

The maximum score along the diagonal is obtained at  $\lambda = 0.8$ . We explored the neighborhood of this point by varying each interpolation parameter separately around the value 0.8. The results are given in Tables 4 and 5. The best result in each column is presented in bold text. A number of the results in Table 4 are better than the best diagonal result of 0.404 and, indeed, better than the CAS result of 0.405 (for example, results with  $\lambda_\gamma = 0.6, 0.7$  or with  $\lambda_r = 0.9$ ). This suggests that we might obtain even better results by going further off the diagonal and exploring the design space more widely. There may, of course, be complicated interactions between parameters, so that the change of one parameter may affect the importance of another. We also notice that the utility does not behave linearly in interpolation parameters (for example in  $\lambda_r$ ), as can be easily seen from the Table 5, which contains plots from each row of Table 4.

To complement the picture given by these utility measures, we also considered the number of positive and negative documents submitted by a system. Figure 3 shows the parametric variation of these parameters on the diagonal run (including varying the  $\lambda_q = \lambda$  parameter). It is interesting to observe that the initial improvement is due to submitting drastically greater numbers of relevant documents ( $\lambda = 0.0$  to 0.2 to 0.4). Then, as the number of relevant documents increases, the submission of not relevant documents decreases (0.4 to 0.6). At the next step, nonrelevant submissions continue to decrease, but the relevant submissions also go down (0.6 to 0.8 - the lowest point on the graph). Finally, on step (0.8 to 1.0) the graph makes a step in the opposite direction. The exact set of parameters giving the best result according to a particular measure depends on that measure. In our

case, the measure is an average of truncated linear utilities, which explains the complexity of relation between the numbers of relevant and nonrelevant documents retrieved and the utility.

Table 6 presents the same information as Figure 3, but as a fraction of the total number of relevant and nonrelevant documents that are retrieved along the diagonal path. Here the “step back” from 0.8 to 1.0 is easier to see than in the graph. We also note that while there is significant overlap between the results along the diagonal path, the methods in fact achieve very similar scores by retrieving a number of *different* relevant documents.

Table 7 (which is structured similarly to Table 4) explores results in the region around  $\lambda = 0.8$ . Once again, we cannot obtain a clear conclusion about effects of parameters from a table alone. The effect of changing a parameter at a given point ( $\lambda = 0.8$ ) can be different for relevant and nonrelevant documents, and may also depend on the direction of the change. For example, increasing  $\lambda_r$  leads to increased submission of both kinds of documents, while increasing  $\lambda_\gamma$  or  $\lambda_y$  leads to decrease in number of submitted documents. Parameters  $\lambda_\alpha$  or  $\lambda_w$  have a nonlinear effect on number of relevant or nonrelevant submissions, which may mean that they have a local optima with respect to these kinds of documents around 0.8.

While the plots of relevant and non relevant documents submitted (that is, passing the filter) help us to see into the complexity of the problem, they are not a good basis for optimization. Therefore we propose that a specific utility measure be used in optimization studies.

We did preliminary evaluation of the differences in performance of CAS and 0.8-point system on topics of different types. Our results (Table 8) show that CAS performed somewhat better on intersection topics, while 0.8-point system had higher utility on assessor topics. However, a detailed study is needed before conclusions can be made about the relations between individual parameters and topics (or groups of topics).

In order to determine whether our results are due to some peculiarity of the TREC corpus, or of the particular ordering of documents, we conducted two more sets of experiments.

In the first group of experiments, we randomly reordered the documents in the TREC test set and ran our interpolation system with parameter  $\lambda$  set to 0, 0.8 and 1.0. This was repeated 5 times. The results are in Table 9, and they are consistent with the results using the standard ordering (Table 2).

In the second group of experiments, we applied our interpolation system to the OHSUMED dataset [8]. Documents in the years 1987-1988 were used for training, while those from the years 1989-1991 were used as a test set. We used 77 queries, specified only by relevant documents in the training set (i.e. there was no  $q^{terms}$  vector). Another significant difference from the TREC-11 AF task was a relatively large number of positive examples in the training set. The results of homotopy on this data are given in Table 3 and Figure 2 (due to a smaller size of OHSUMED data we were able to explore the diagonal more fully). The overall shape of the utility curves is the same as on the TREC dataset. The rise at the 1.0 endpoint suggests that the optimal performance on this dataset may be achieved with the interpolation parameters outside  $[0, 1]$  region.

$\lambda$	0.0	0.2	0.4	0.6	0.8	1.0
relevant	0.243	0.483	0.746	0.751	0.639	0.676
nonrelevant	0.426	0.457	0.299	0.164	0.072	0.102

**Table 6:** Fraction (of the total over the diagonal path), for each value of  $\lambda$ , of the relevant and not relevant documents submitted by the system. The totals are 3228 and 13510 respectively. Note that the best performance is not obtained when the largest number of relevant documents is submitted. The graph of actual values is given in Figure 3.

	assessor		intersection	
	# topics	avg. T11SU difference	# topics	avg. T11SU difference
CAS better than 0.8	18	-0.047	25	-0.037
0.8 better than CAS	25	0.062	13	0.011
CAS and 0.8 equal	7	0	12	0
Total	50	0.014	50	-0.015

**Table 8:** Comparison of CAS and 0.8 homotopy point on assessor and intersection topics. Positive difference indicates 0.8 performed better than CAS.

Parameter	Parameter Values					
	0.7		0.8		0.9	
	r	n	r	n	r	n
$\lambda_\alpha$	2086	975	...	...	2089	1010
$\lambda_\gamma$	2273	1233	...	...	1923	830
$\lambda_p$	2043	1129	...	...	2014	939
$\lambda_y$	2106	1005	...	...	2029	948
$\lambda_u$	2065	1037	...	...	2071	977
$\lambda_i$	2062	977	2062	977	2062	977
$\lambda_w$	2055	1000	...	...	2119	1007
$\lambda_s$	2153	1021	...	...	2123	1031
$\lambda_r$	2037	993	...	...	2149	1044
$\lambda_q$	2062	977	...	...	2062	977

**Table 7:** Cell  $ij$  contains the number of relevant and nonrelevant documents submitted by the system described by  $\lambda$  having value 0.8 and parameter in row  $i$  having the value in the second row of column  $j$ . (Therefore all values in column 0.8 are the same and are given only once.) The total number of known relevant documents is 9050, while that of known nonrelevant is 71264.

$\lambda$	0.0	0.8	1.0
Average T11SU	0.108	0.406	0.391
Standard Deviation	0.002	0.002	0.004

**Table 9:** Results of experiments with re-ordered TREC test set (averages across 5 runs).

## 6. DISCUSSION

Researchers in IR have long known that “the devil is in the details”, and that changes in the implementation of a specific method can produce significant changes in the measured performance. This has led to confusion over what ought to be meant by the words “the XX system”. Other than a claim of ownership, such a text often does not tell us enough to permit replication of the experiment, and does not contribute to broader scientific understanding of the role of specific design choices in improving IR performance. The present note illustrates that problem, using two approaches that could well be described as “the Rocchio classifier”. Yet the performance of one is nearly four times that of the other! A newcomer to the field might well be puzzled as to whether the “classifier” in question is a good one or a bad one.

We hope that this note, by providing specific models for interpolation between pairs of design choices, even quite complex ones, will aid other researchers to explore their own systems, and the regions “between systems” in a systematic fashion. While the “texts” that describe systems for learning and classification are often radically different, the underlying computational mechanisms are usually “connectable” in a reasonable design space, with a bit of ingenuity.

Costruction of homotopy between two linear classifiers, such as Rocchio and linear SVM, should prove to be rather similar to what is described here. (SVM decision hyperplane can be seen as a sum of weighted positive and negative support vectors, analogous to Rocchio query vector  $q$ .) We believe that the well-known “kernel trick” can be used to interpolate between different non-linear SVMs, or between a non-linear SVM and a linear classifier in the same fashion.

One significant possible implication of this work is to incorporate ideas of homotopy and connectability of systems into a framework for designing new systems. Two systems may be connected in a fashion described here and the path between them can be explored for a system with better performance. It should also be possible to connect more than two systems in such a fashion. In this case it would be more accurate to talk of exploring a region in design space generated by the systems.

A more abstract approach would involve combination of

systems at a more abstract level. For example, we can broadly talk about systems having a representation module, a score computation module and a thresholding module. The interaction between these modules defines the behavior of a system. Two or more very different systems having only this kind of structure in common may be combined at each of these levels, producing a new system. The resulting system is quite different from classical combination of classifiers, where the systems are combined by adding an extra level of post-processing “on top”. A potential advantage over the approach discussed in this paper is much greater simplicity of constructing the combinations: the few high-level modules can be considered “black boxes” and their outputs combined using a small number of parameters and then fed to the next level.

Finally, we note that in moving to design space we do not avoid the two fundamental problems. First, the range of parameters cannot be explored exhaustively, and since the performance measures are not smooth functions, techniques for non-smooth optimization [5] will be needed. For problem areas such as text filtering where ground truth requires human classification labelled data will always be in short supply and overfitting will always be a potential problem.

## 7. ACKNOWLEDGMENTS

The authors thank the KD-D group for its support through National Science Foundation grant number EIA-0087022 to Rutgers University. The views expressed in this article are those of the authors, and do not necessarily represent the views of the sponsoring agency.

The authors also thank Andrei Anghelescu and Vladimir Menkov for providing the source code of the DIMACS system, Jamie Callan - for the LEMUR Toolkit, all members of the DIMACS Working Group on Monitoring Message Streams (<http://www.stat.rutgers.edu/madigan/mms/>) for helpful discussions and suggestions, the researchers at the Chinese Academy of Sciences for sharing details of their method, Ian Soboroff at NIST for providing some of the submitted results of DIMACS and CAS, and the anonymous reviewers for their insightful comments.

## 8. REFERENCES

- [1] Reuters corpus volume 1. <http://about.reuters.com/researchandstandards/corpus/>.
- [2] A. Anghelescu, E. Boros, D. Lewis, V. Menkov, D. Neu, and P. Kantor. Rutgers filtering work at trec 2002: Adaptive and batch. In *The Eleventh Text REtrieval Conference (TREC-11)*. NIST, 2002.
- [3] A. Bahuman, K. Rasheed, and B. Bishop. An evolutionary approach for vlsi standard cell design. In *The Congress on Evolutionary Computation (CEC'2002)*, 2002.
- [4] M. Breschuaer and C. Peters. Cross-language evaluation forum: Objectives, results, achievements. *Information Retrieval*, 7:7–32, 2004.
- [5] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley Inter-science, New York, New York, 1983.
- [6] A. Gelsey, M. Schwabacher, and D. Smith. Using modeling knowledge to guide design space search. In *Artificial Intelligence in Design*, Stanford, CA, June 1996.
- [7] D. Harman. Towards more reliable information retrieval technology. 2003 ARDA Workshop
- [8] W. R. Hersh, C. Buckley, T. J. Leone, and D. H. Hickam. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (Special Issue of the SIGIR Forum)*, pages 192–201, 1994.
- [9] N. Kando. Evaluation of information access technologies at ntcir workshop. In *Results of the CLEF 2003 Cross-Language System Evaluation Campaign*, pages 445–454, August 2003.
- [10] S. Robertson and I. Soboroff. The TREC 2002 filtering track report. In *The Eleventh Text REtrieval Conference (TREC-11)*. NIST, 2002.
- [11] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1971.
- [12] M. Schwabacher and A. Gelsey. Intelligent gradientbased search of incompletely defined design spaces. Technical Report HPCD-TR-38, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1996.
- [13] L. Steinberg and K. Rasheed. Optimization by searching a tree of populations. In *Genetic and Evolutionary Computation Conference (GECCO'99)*, 1999.
- [14] S. Szykman. Optimization as a driver for design space exploration. In *The Optimization in Industry Conference*, Palm Coast, Florida, March 1997.
- [15] B. Wang. Personal communication, January 2003.
- [16] H. Xu, Z. Yang, B. Wang, B. Liu, J. Cheng, Y. Liu, Z. Yang, X. Cheng, and S. Bai. TREC-11 experiments at CAS-ICT: Filtering and web. In *The Eleventh Text REtrieval Conference (TREC-11)*. NIST, 2002.
- [17] W. I. Zangwill and C. B. Garcia. *Pathways to Solutions, Fixed Points, and Equilibria*. Prentice Hall, 1981.
- [18] G.-C. Zha, D. Smith, M. Schwabacher, K. Rasheed, A. Gelsey, D. Knight, and M. Haas. High performance supersonic missile inlet design using automated optimization. In *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1996.