Local Search: A Guide for the Information Retrieval Practitioner

Andrew MacFarlane[1] and Andrew Tuson[2]

Department of Information Science[1]
Department of Computing[2]
School of Informatics
City University London
{andym,andrewt}@soi.city.ac.uk

Abstract

There are a number of combinatorial optimisation problems in information retrieval in which the use of local search methods are worthwhile. The purpose of this paper is to show how local search can be used to solve some well known tasks in Information Retrieval (IR), how previous research in the field is piecemeal, bereft of a structure and methodologically flawed, and to suggest more rigorous ways of applying local search methods to solve IR problems. We provide a Query based taxonomy for analysing the use of local search in IR tasks and an overview of issues such as fitness functions, statistical significance and test collections when conducting experiments on combinatorial optimisation problems. The paper gives a guide on the pitfalls and problems for IR practitioners who wish to use local search to solve their research issues, and gives practical advice on the use of such methods. The Query based taxonomy is a novel structure which can be used by the IR practitioner in order to examine the use of local search in IR.

Keywords: combinatorial optimisation, information retrieval, local search, evaluation

1. Introduction

There are various tasks in information retrieval that are computationally difficult if not impossible to solve using normal computational methods. These problems can be NP hard and require the application of combinatorial optimisation methods, particularly the utilisation of local search. There has been a great deal of work in the area, and readers interested in reviews should refer to Chen (1995) and Sebastiani (2002). Having read the literature, we have identified a number of areas which need to be tackled in order to improve the quality of work in applying local search to IR problems:

- Local search techniques are used piecemeal without any thought about what might be the best method for a particular IR problem.
- There is no overall overview or structure of how local search can be applied to IR, understanding the available techniques and which local search method is best suited to what task.
- Much of the research is methodologically flawed in terms of evaluation techniques, in particular the application and choice of test collections.

In this paper we define a number of IR tasks, and show how local search either has, or could be, used to tackle combinatorial optimisation problems. We propose the following solutions to the problems in applying local search to IR specified above:

- We discuss a technique of knowledge based optimizer design through which we can understand local search techniques in general. This gives us a methodology with which we can apply to the problem.
- Using the current practice in knowledge based optimizer design we describe how knowledge from the IR domain can be used to inform local search techniques through the application of a Query based taxonomy to various problems in IR.
- We examine how a methodologically sound experiment can be used to improve the quality of research in the area.

Our overall purpose is to give an outline of the area and provide practitioners in IR who have combinatorial optimisation problems a method with which they can use to conduct their experiments. The evidence used to build this advice comes from current practice in both areas from the literature, theory in the area of optimizer design and the experience of applying techniques to various problems by both authors. However limitations of these techniques lie mainly in their effective design e.g. Papadimitriou & Steiglitz (1982) note that “the design of effective local search algorithms has been, and remains, very much an art”.

The paper is structured as follows. We describe local search using an example from IR in section 2, from research done so far with hillclimbers. Extensions to hillclimbers for IR are then described in section 3, providing an overview of the various neighbourhood search techniques and giving references for further reading - though the reader is directed to other introductions (Osman, 1995; Osman and Laporte, 1996; Osman and Kelly, 1996; Reeves 1993). The concept of knowledge based optimizer design is outlined in section 4. In section 5 we then outline how local search techniques can be used in IR

using a Query based taxonomy, and then outline ideas for good practice when applying local search methods to IR problems in section 6 by discussing the issue of evaluation. A conclusion is given at the end and we discuss the implications of the research.

2. What is Local Search?

Local search is a method which can be applied to computationally complex problems (which therefore cannot be completed in a reasonable length of time). In this class of problems there is no specific algorithm or heuristic which can be applied to solve the problem in its entirety. An example of the application of local search are hillclimber techniques used by the City University Okapi team on the Routing track at the Text Retrieval (TREC) conference run by NIST (Voorhees and Harman, 2000). This is the process of using relevant documents from a training set to produce a set of terms to be applied to a test set. When choosing terms from a training set, the reader can imagine that normal relevance feedback techniques (Robertson et al, 1995) can be used to select a set of terms, using the kind of techniques suggested by Robertson (1990) for limited query expansion. In limited query expansion the number of terms would normally be restricted to the 15-20 range. However, a further process can be applied by increasing the initial set of terms to say 200 and try to find the best possible combination of terms from this term set. This cannot realistically be done as the computational requirement is of order $O(2^{200})$, therefore optimisation techniques must be used in order to select a good set of terms from this second stage process – it is known as a Combinatorial Optimisation Problem or COP. Using this example we can give a more precise definition of local search in IR terms.

Consider a Combinatorial Optimisation Problem (COP) which has the form:

- A space of possible solutions, *S*, with possibly constraints on valid solutions;
- A `quality' for each solution, quality(s) where $s \in S$;

where the objective is to find a solution *s* such that quality(s) is maximised and the constraints satisfied. In the context of the Okapi experiments described briefly above the space of all possible solutions is the power set of terms, and an example of the quality of the solution is average precision. Other evaluation measures have been tried in these experiments, but it has been empirically found that average precision is a good predictor of other measures, but other measures are not good predictors for average precision (Robertson et al, 1996) – although there is conflicting evidence on this, see further discussion of this issue below in section 6.2.

Such problems are usually NP-hard (Garey and Johnson, 1979) and therefore, for a problem of large enough size, the time taken to find an optimal solution by exact methods will become prohibitive. Fortunately, in practice a `good enough' answer in the time available is all that is required, which opens up the possibility of using heuristic methods that do not guarantee optimality, but work well in practice. A good enough answer in our example would be to produce a term set which would produce better results on the test set than just using the initial set of terms from relevance feedback.

One heuristic approach would be to employ some form of hillclimbing that requires the following decisions to be made:

- Find a suitable encoding scheme for the candidate solutions in *S*, such as a list/matrix of average precision scores;
- Choose a quality metric for each solution, *quality(s)*, such as an increase in average precision;
- Utilise a method of modifying an encoded solution to give another solution (a move operator) *e.g.* add/delete terms from term set, or adjust weight for term/document pairs (reduce weight, increase weight).

There is usually more than one possible solution (valid or invalid) that can be produced by applying the move operator on a given solution, therefore we define a neighbourhood, *N(s,m)*, as the set of solutions in *S* that are produced by applying the move operator, *m*, on a solution *s*. Once the above have been defined, the general algorithm is quite straightforward and is described below:

- Generate an initial solution (in our example the top three ranked terms from relevance feedback) and evaluate it;
- Apply and evaluate a move in the neighbourhood of the current solution and apply an acceptance criterion to decide whether to use the new solution e.g. add a new term and evaluate its usefulness using average precision;
- Go back to step 2 until a termination criterion is reached.

The termination condition is merely when the user would like to stop the search. Examples would include when a certain amount of CPU time has elapsed, or when a solution of a certain quality has been found. In the Okapi experiments the termination conditions included a limit of 100 iterations for term selection and when no increase in average precision was found in a single iteration (Robertson et al, 1995). The acceptance criterion determines whether a new solution generated by a move operator replaces the current solution, and introduces a bias towards better solutions in the search. Acceptance criteria, and thus hillclimbers, can be broadly classified as follows:

- Any-Ascent hillclimbers (AHC) accept moves that give (new) solutions, which have better or equal quality than the current solution, curr (i.e. accept when $quality(s_{new}) \geq quality(s_{curr})$). A common variant is stochastic hillclimbing (SHC) where moves are tried in a random order. Not used on Okapi experiments.
- First-ascent (FAHC) hillclimbers operate similarly, but take the first improvement in quality found (i.e. accept if $quality(s_{new}) > quality(s_{curr})$). Okapi algorithm: Choose First Positive (Robertson et al, 1995).
- Steepest-ascent (SAHC) hillclimbers, systematically evaluate the entire neighbourhood of the current solution and accept the most improving move (i.e. accept if $quality(s\_best) > quality(s\_curr)$ and $quality(s_{best}) > quality(s_{any} \mid s_{any} \in N(s_{curr}, m))$). Okapi algorithms: Find Best and Choose All Positive (Robertson et al, 1995).

The bias introduced by the acceptance criterion, though necessary, can lead to problems. This happens when an implicit assumption is made about the landscape (Tuson, 2000) - which is a graph induced by the move operator and quality function that connects solutions accessible to each other by a move operator - is correlated in such a way as to lead the hillclimber into a region of the landscape with high-quality solutions. Deviations from this ideal can lead to the hillclimber being deceived (the correlations lead the hillclimber away from the optimum), or stuck in local optima. This can force the hillclimber to choose a set of terms which may not do so well on the test set, causing overfitting. Overfitting of terms on the training set can be a significant problem unless care is taken on the use of the training set. Fortunately, these potential difficulties do not necessarily arise in practice, as this assumption is not too far from the reality if a suitable move operator is used e.g. an add/remove terms strategy in Okapi hillclimbing (Robertson et al, 1996).

3. Extending the Hillclimber

The most common type of meta-heuristic, commonly known as neighbourhood (or local) search, extends hillclimbing in some fashion, usually by relaxing the acceptance criterion, so to escape local optima. Thus they can be placed in a common implementation framework such as described by the pseudo-code below (Rayward-Smith, 1994).

```
neighbourhood_search()

        initialise(P);      // generate starting solution(s)
        while(!finished(P))
                Q = select_solution(P); // choose solution(s) to perturb

                R = create_moves(Q);    // apply move operator(s)

                P = merge_sets(P,Q,R);  // merge to obtain a new set P

        END while

END neighbourhood_search
```

where *P*, *Q*, and *R* are sets of solutions in *S*. Although it is usual to have only one solution in a set, Evolutionary Algorithms, described later, are the exception to this rule. With the problem encoding, move operator(s), quality measure specified, and by defining the above functions appropriately, any of the versions of neighbourhood search described in the remainder of this paper can be implemented. In the following sub-sections, we show how the hillclimbers can be extended to tackle the term selection problem below, using the example of the Okapi experiments discussed above.

3.1 Iterated Hillclimbers and GRASP

One way out of the problem of local optima is to restart the hillclimber with a different initial solution when a local optimum has, or is suspected to have, been found. This is known as iterated hillclimbing. A common criterion for restart is when a certain user-defined number of evaluations have been made without an improvement in solution quality. The search will then resume in a different part of the search space, with a different, and possibly better quality, local optimum.

A variant of this, GRASP, Greedy Randomised Adaptive Search Procedure (Feo, Bard and Claflin, 1991) - a trade mark of Optimization Alternatives, Austin, Texas, incorporates a construction phase where the new starting point is generated using a greedy algorithm which is then followed by a local search improvement phase. The intelligent initialisation procedure used in GRASP attempts to start the search in the vicinity of good solutions. More emphasis is thus placed on the initialisation procedure than the other components of neighbourhood search, to the extent that its design is highly problem-specific and it often is adaptive in nature, making use of past experience in the search. This contrasts with the role of the improvement phase which is merely to locate a local optimum. Applications of GRASP include vehicle routing (Hjorring, 1995), and single machine scheduling (Feo, Venkatraman and Bard, 1991). A review of other applications, such as flight scheduling for airlines, as well as directions for further reading is given in (Feo, Bard and Claflin, 1991). This could be applied to the Okapi term selection problem by restarting the selection process when the top

average precision 'peak' is reached and continually reusing evidence from a number of term selection runs to improve the final result.

3.2 Simulated Annealing

Simulated Annealing (SA) is based upon an analogy between optimisation and the annealing process in physics (Kirkpatrick et al, 1983). In terms of neighbourhood search, it can be thought of as a form of stochastic/any-accept hillclimbing with a modified acceptance criterion that accepts lower quality solutions with a probability (the Metropolis criterion) given by the equation below:

Equation 1

where T_k is the 'temperature' at time-step k. In plain English, the temperature controls how likely it is for a lower quality solution to be accepted, and thus allows SA to escape local optima; i.e. low quality differences/high temperatures indicate a higher probability of acceptance of a lower quality solution. At $T = 0$, SA is equivalent to SHC.

The temperature varies with time according to a cooling schedule where, usually, the temperature is reduced as the optimisation progresses. This allows exploration of the search space at the start of the search, followed by exploitation of a promising region later on. The technique-specific choices of initial and final temperatures and the form of the cooling schedule are important in order to obtain a balance between exploration and exploitation (also termed intensification/diversification). However, there is no reason why the cooling schedule should be of any particular form, or even monotonically decreasing - the choice is problem-dependent. That said, two common cooling schedules that generally work well (Lundy and Rees, 1986), are given below:

Equation 2

Work on SA has looked at the theory (Hajek, 1988), cooling schedules (Lundy and Mees, 1986), and applications of SA such as sequencing (Osman and Potts, 1989; Ogbu and Smith, 1990), timetabling (Abramson, 1991) and the Steiner problem in graphs (Downsland, 1991). Further information on SA can be found in a variety of sources (Collins et al, 1988; Van Laarhoven and Aarts, 1989; Aarts and Korst, 1989).

This technique has been used in Okapi routing experiments, with disappointing results (Walker et al, 1998). A very simple simulated annealing process was tried, reducing the 'temperature' at a number of stages for term re-weighting, until a zero or 'quench' temperature was reached. Unlike the hillclimbers used in previous experiments on Okapi, terms which actually decreased the average precision were kept, but with a probability which decreases with the current temperature. This was done so that the term selection process could escape from a local maximum (which the hillclimbers were unable to do). The results on the training set were encouraging when using the re-weighting operation, but when the term sets were applied to the test set the results on the queries varied significantly, and it appears that the annealing process was overfitting terms on the training set. A deterministic re-weighting process was combined with a mild simulated annealing mechanism, which gave a noticeable increase on the training set, but the authors were unable to demonstrate an improvement on the test set. It was noted that a significant number of tuning parameters would need to be investigated such as the temperature reduction function, the re-weighting process (which terms to choose, how to vary the re-weighting), and rules for both ending a stage in the procedure, and the simulated annealing process itself. This method was not investigated further with respect to the term selection process.

3.3 Threshold Methods

Threshold methods are additional extensions of stochastic/any-accept hillclimbing, that use the idea of a threshold which sets a level below which new solutions will not be accepted (i.e. acceptance is deterministic). For similar reasons to SA, the threshold, L_k is time varying. Threshold methods including the following;

- Threshold Accepting (TA) accepts a new solution if its quality is not below a set threshold relative to the current solution (Dueck and Scheuer, 1990) (i.e. accept if $quality(s_{new}) \geq quality(s_{curr}) - L_k$).
- Record-To-Record Travel (RTRT) accepts a new solution if its quality is not below a certain threshold relative to the best solution or record found during the search so far (Dueck, 1990) (i.e. *if* $quality(s_{new}) \geq quality(s_{best}) - L_k$).
- The Great Deluge Algorithm (GDA) accepts a new solution if its quality is not below an absolute quality threshold - the current water-level (Dueck, 1990) (i.e. if $quality(s_{new}) \geq L_k$).

These techniques differ in the way the threshold is used. For all of these techniques it is usual to vary L_k according to the schedule $L_{k+1} = L_k + \alpha$, though there is no reason why others cannot be used. For further details and applications, the reader is referred to the papers (Althofer and Koschnick, 1991; Sinclair, 1993).

This scheme was not used in Okapi term selection experiments but it is easy to see how two of these schemes could be used to augment the optimisation process. For example RTRT could be utilised by using something like a 0.05 limit on increase in average precision, below which evaluations are rejected. The GDA would not be a useful method for term selection as there are no absolute values for average precision, which can vary significantly with the set of topics, and from topic to topic.

3.4 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are based upon the theory of evolution by natural selection (Darwin, 1859). A population of solutions is maintained, and allowed to 'evolve'. Three main styles of EA exist: Genetic Algorithms (GAs) (Holland, 1975), Evolutionary Programming (EP) (Fogel et al, 1966), and Evolution Strategies (ESs) (Rechenburg, 1973), but the basic idea behind them is the same and the differences can be considered historical. As an example of the concept, the following describes a simple EA with steady-state reproduction:

1. Generate an initial population of solutions;
2. Select two parent solutions from the population according to their quality;
3. Apply move operator(s) to generate a new solution, s_{new};
4. If s_{new} is superior to the worst member of the population, s_{worst}, then s_{new} replaces s_{worst};
5. Go back to step 2 until the termination criterion is reached.

There are many flavours of EAs, as almost all of the stages have a wide choice of alternatives; however all have population-based acceptance criteria as a set of solutions are maintained and a new set of solutions are produced by applying the available move operator(s) and then somehow merging the two sets. Two types of moves are commonly used: mutation (roughly analogous to asexual reproduction), which is equivalent to the conventional move operator, and a binary move operator, crossover (roughly analogous to sexual reproduction) which selects two candidate solutions and (probabilistically) swaps information between them. An example is `two-point' crossover which randomly picks two points along the strings and swaps the contents of the string between those points, to produce a child as shown in Figure 1.

Figure 1: An Example of Two-Point Crossover

The usual rationale for why crossover is a useful search operator is that the recombinative process can bring together portions of separate strings associated with high fitness to produce even fitter children. Both the population-based nature of the search, and the crossover operator help to avoid the search being trapped in local optima.

Unfortunately, some practitioners still equate EAs with encoding solutions as binary strings - due to a misunderstanding of early theoretical work on the schema theorem (Holland, 1975). This is simply not true (Radcliffe, 1992) and successful EA applications use whichever encoding is appropriate to the problem. Such applications are extremely varied, covering fields as diverse as chemistry (Cartwright and Harris, 1993), machine learning (Goldberg, 1989), and Operational Research (OR). Example applications in OR include: sequencing problems (Reeves, 1995), vehicle routing (Thangiah, 1995), and timetabling (Corne et al, 1993). A variety of textbooks are available though Michalewicz (1992) is a good starting point for those interested in how to apply EAs, whereas the more recent `Handbook of Evolutionary Computation' is recommended for its in-depth coverage of the field (Baeck et al, 1997).

An example of the way this would work with term selection would be to generate some strings of the term set size, with each element of a string representing either the absence or presence of a term. Average precision scores for these strings would then be generated, applying move operators and replacing the worse average precision scores until a termination condition was reached. This scheme could be extended for term reweighing by assigning a positive integer for each element of the string to indicate not only the presence of a term, but the level at which it would be re-weighted.

3.5 Tabu Search

Tabu Search (TS) (Glover, 1990) is based on steepest-ascent or first-ascent hillclimbing, and avoids local optima in a deterministic way, based on an analogy with memory, which centres on the use of a tabu list of moves/solutions (or their features) which have been made/visited in the recent past (tabu tenure) of the search. Applications of TS include: vehicle routing (Semet and Taillard, 1993), graph colouring (Hertz and de Werra, 1987), and path assignment (Oliveria and Stroud, 1989). In TS, the acceptance criterion of the hillclimber is altered slightly so that if no improving move can be found after the neighbourhood has been fully examined, then the move that gives the least drop in quality is taken. Then a basic form

of memory, recency, is used, which is short-term in nature. In essence, any move/solution that is on the tabu list cannot be made/revisited; this prevents the search cycling in an already explored area of the search space.

In addition, aspiration criteria can be used to override this mechanism under certain circumstances (e.g. when making a tabu move would lead to a higher quality solution). As neighbourhoods can be quite large and thus expensive to search fully, candidate list strategies are often used to choose subsets of the neighbourhood to search. An alternative approach is to use some form of cheap, but approximate, evaluation procedure.

A form of long term memory, frequency, can be used to direct the search by adjusting the quality function (e.g. solutions closer to a frequently visited area of the search space are penalised more than distant ones). Other forms of memory are: quality, which refers to solutions of high quality, and is often used to intensify search in the region of good solutions; and influence, which is a measure of change in solution structure, and often used as part of an aspiration criterion. Of course, all of these memory structures have to be defined for the problem being solved.

Examples of research in this area include dynamic rules for tabu tenure (Battiti and Tecchiolli, 1994), hashing functions to identify tabu solutions (Hasan and Osman, 1995), and hybrids with other methods (Osman and Christofides, 1994). However, a recency-based approach with a simple neighbourhood structure and a restricted candidate list strategy will often produce good results (Reeves, 1993). For more information, the reader is directed to Glover and Laguna (1997).

In the context of Okapi experiments, this would require a term removal from the current working set, and the term that yields the least reduction in average precision is chosen for removal. This term is recorded in the tabu list to prevent any move to add it to the term set until the recency criteria is met. An alternative is to choose a set of terms that yields the least reduction in average precision. Generic aspiration criteria could include making a term non-tabu if using it leads to a better solution than the current best. One way to use long term memory is to keep sets of terms already examined and to reduce average precision by say 0.1 if the optimizer attempts to use that particular set of terms again.

3.6 Related Methods

Another technique that fit into the framework of neighbourhood search is Iterative Repair (Zweben et al, 1993), which views moves as `repairs' to flaws in the solutions and has been successfully applied to coordinate space shuttle ground processing, and Tabu Thresholding (Glover, 1995), which is a variant of TS that replaces some of the memory structures in TS with a form of randomisation. Needless to say, hybrids of all of the techniques described here are also possible.

Meta-heuristic techniques also exist which are not local search-based, though they are less commonly used. However two are worthy of mention. Ant Systems are based on the use of pheromone trails by ants (Dorigo and Gambardella, 1997) - a population of `ants' makes a path through the solution construction process, where the choice at each stage is balanced between following the trails of other ants, and what a greedy heuristic would decide. Finally, Hopfield Nets arose from neural network research on associative memories and can also be used for optimisation (Hertz et al, 1991).

3.7 Genetic Programming

One sub-area of evolutionary computation/local search that is worth mentioning separately is that of Genetic Programming (GP) (Koza, 1992). Though the original idea was to evolve computer programs, for our purposes GP can be best thought of as the use of an EA to optimise mathematical functions represented as trees. The leaf/terminal nodes indicate variables and constants that are successively operated on by function nodes on their way to the root node, giving the final result.

The choice of the terminal and function nodes is user (and also problem) dependant, and GP can potentially be used for any application that a neural network can, e.g. regression and classification. However GP does have one distinct advantage. If the function nodes are chosen well, the trees evolved by GP can be readily interpretable. One notable example was the use of GP, with a set of function nodes corresponding to Monod kinetics, in the identification of the kinetics of a fermentation process (Pohlheim and Marenbach, 1996). The results provided valuable insights into the process for the chemists and biologists working on the process.

In addition the tree structure of GP allows the tree to be of whatever complexity is needed to solve the problem, whereas the complexity of a neural network is constrained by the user's chosen network configuration. However, as this could affect the chances of GP `overfitting' the data, it is unclear whether it is an advantage. In fact, users of GP are advised to read a good neural networks text such as (Masters, 1993) because issues such as overfitting, data preparation, and validation, are equally applicable in GP.

4. Knowledge based Optimizer Design

Local search optimizers are considered weak (need little domain knowledge) and heuristic (not guaranteed to find the optimal solution); therefore they are not suitable for all problems and should not be considered a panacea. Practitioners should first ask, can the problem be solved exactly by complete search? Also, it should be asked whether human/current performance is "good enough" (or all that can be achieved)? Although local search optimizers are relatively straightforward to understand, care does need to be taken in their design and evaluation. Practice in both the IR and optimisation literature provides much useful guidance. In any case, it is worth noting that simplicity is often effective: a simple hillclimber can often produce the desired performance in practice.

In general, the guiding principle today seems to be "mould the algorithm to the problem, not the problem to the algorithm" (Luke, 1996; Fogel and Angeline, 1997). There are at least two reasons for this. Firstly, as local search optimizers have been applied to a wider range of problems, it has been increasingly found that many do not map well into a binary string- or vector-based representation. Thus, workers have been driven to experiment with more natural representations and operators and have often found success by doing so. Secondly, and more fundamentally, the "No Free Lunch" theorems of Wolpert and Macready (1995) state broadly that, for any algorithm, any elevated performance on one class of problems is exactly countered by poorer performance on another class. Thus, an optimizer tailored for a specific application by the incorporation of problem-specific knowledge in the encoding and operators is likely to outperform a canonical, "black-box" implementation. However, it is not enough just to say that domain knowledge is important, guidelines are required on how to approach a problem, extract the salient features, and to map these onto the optimizer. Therefore the design of local search optimizers remains very much an art, and a `methodological gap' remains between the literature, textbooks, and the application of these techniques that needs to be addressed (Tuson, 2000).

Despite the above discussion and the record for IR applications and local search, the question of how to design an effective optimizer is non-trivial and remains an ad hoc process, partly (but not wholly) due the lack of an effective theory. In this context the need for a knowledge based design approach for local search is required. One recent approach (Tuson, 2000) is based on the idea of viewing optimizers as Knowledge Based Systems (KBS): "a computer system that represents and uses knowledge to carry out a task" (Stefik, 1995). Furthermore the AI community has, for many years, been addressing the problems of how to design a Knowledge Based System (KBS), and therefore if we could place these optimizers into a KBS framework, then the reuse of the extensive research into KBS design may be possible.

One such useful concept from the KBS literature is the distinction made between the knowledge level (the description of what domain knowledge is present in the system), and the symbol level (the description of how the knowledge is represented in the system - the data structures used and their operations on them) (Newell, 1982). The design of optimizers is, at present, conducted primarily at the symbol level, and therefore the knowledge/domain assumptions made by the optimizer designer are implicit, and may well not be properly utilised. A knowledge level analysis would make these assumptions explicit and ensure that they are fully considered in the optimizer design. To this end, (Tuson, 2000) proposes that domain knowledge used by an optimizer can be split into three roles:

- Problem Solving Knowledge - this assists the search by providing problem-specific knowledge that structure and guides the search.
- Problem Specification (Goal) Knowledge - specifying what are desirable solution(s) (i.e. the evaluation function).
- Search Control Knowledge - given a defined search space, how do we go about searching it? Our knowledge of the search process is represented here.

Given that we would be interested in exploiting knowledge of a user's information needs it would appear that problem-solving knowledge role is most relevant. However, exactly what is meant by this? A working definition is that it is all of the knowledge that can be directly related to the problem itself, and which is not involved with specifying the quality of a solution. This allows for a clean separation from the technique-specific aspects of the search algorithm. Once these roles have been defined, then there are various types of knowledge sources available that need to be identified – some example optimizer problem solving knowledge sources are outlined below:

- Problem features that correlate with solution quality (i.e. the decision variables);
- How these features interact (as strongly interacting decision variables could be usefully considered as a single unit);
- Areas of the search space which could be excluded from the search (to allow reductions of the size of the search space);
- Areas of the search space where good solutions lie (as an initialisation strategy could be used to start the search in a potentially productive region).

Some reflection should suggest to the reader that the above sources could be framed as questions that an IR practitioner would be able to answer in his role as a domain expert without necessarily having any expertise in optimisation. In addition, (Tuson, 2000) shows that the above sources can be formalised in such a way that suitable Evolutionary Algorithms (EA) crossover and mutation operators (or equivalent local search operators) can be derived. Therefore it appears possible that, in the future, optimizer design will centre on the modelling and acquisition of expert knowledge, rather than algorithm design in much the same way that the KBS community has been advocating (Motta, 1997). Furthermore, KBS methods such as those used for knowledge acquisition should allow the designer to more effectively take out the required domain knowledge. Such knowledge based design approaches should help ensure the scalability of IR applications in the future.

5. How Can Local Search be used in IR?

The discussion so far indicates that there are three classes of design choices that are made in using local search methods in practice:

- How solution quality is measured using a fitness function;
- What encoding and operators are used, and;
- What local search method is used and how it is configured.

The first issue is a specification of what the IR practitioner considers to be a good solution. Often this is straightforward, only where there are multiple objectives or a constraint upon solution feasibility does the user need to think carefully (these are situations where the standard texts can provide guidance). The examples presented here should also be helpful.

The second point refers to the use of domain knowledge in the design of the local search optimizer as described earlier in the discussion on knowledge based design. It is here that the insights of the IR practitioner can be brought to bear most effectively.

The third class of design choice would appear at first glance the most important, after all, we are addressing local search. However it should be considered secondary. The reason is that it is impossible to say in any meaningfully general way that a particular local method is more suited to a given task than another. This is borne out by practical experience as well as theoretical results such as the no free lunch theorems (see section 4 above). In any case, they are based on similar assumptions as all are based on hillclimbing in some manner. It is the fit of the overall optimizer, in terms of its domain knowledge, to the problem that is the key to its success. In fact, practitioners would be well advised to start with a simple hillclimber in the first instance and focus on domain knowledge before considering more complex local search methods.

So the fit between the overall optimizer design and the task to be solved is key. Therefore, it is useful to give the reader a taxonomy within the context of the IR tasks that local search may be applied to. This allows the focus to be rightly put on issues that relate to the problem domain such as the suitability of the problem representation to the task. We put forward a Query based taxonomy, showing elements of a query to which optimisation can be applied. For each of the components of this taxonomy we state what can be done via local search and describe some current work in the area for each. Some of these can be done in conjunction with each other e.g. term selection with raw weight optimisation (e.g. okapi) or in isolation. This should help the IR practitioner to relate their problems to local search optimizer decisions. The purpose of the taxonomy is therefore to provide the domain information needed for the knowledge based optimizer design process.

We describe a number of tasks to which local search can be applied, and we then discuss the kinds of uses that local search techniques can be applied to in terms of both matching and term operations. Examples of work done in the area are provided.

Figure 2: Query based taxonomy for local search in IR

5,1 Tasks in IR

We define an IR task as a process which has its own functionality, but which may on occasion share some processes. This is related to the type of search the user needs and is intimately related to their particular information need. Examples of tasks are as follows:

- **Ad-hoc and interactive search:** A user types in a query to resolve a current information need and uses query modification techniques such as relevance feedback to refine their query. Local search could be useful in matching queries to documents, optimising relevance feedback and providing adaptive user interfaces. Mathematical logic in search models is important here. It is difficult to use such techniques for real time search, but prior or post processes to search such as parameter optimisation for term weighting schemes are valid applications.
- **Information filtering:** A user has a long term information need and poses a query. Documents are 'pushed' through this query and filtered to the user. Optimisation techniques are very useful for this task (more details are given above). The techniques can be extended to social groups (collaborative filtering) as well as individuals. A 'profile' query is created by this task.
- **Text classification:** A 'queryless' task at least initially. Organises text into groups or classes either by learning on the text itself (unsupervised) or by using evidence from pre-defined categories or taxonomies (supervised). A lot of clustering work has been done on this task. It can be thought of as another type of indexing process.
- **Topic detection and tracking:** Another 'queryless' task, akin to text classification & filtering, but more refined in that it is split up into two main subtasks of topic detection (a classification task) and tracking (a filtering sub-task). A profile query can be created by this task.

We will use these tasks in our discussion below using examples from the literature.

5.2 The Role of the fit between technique and task

As noted above, we cannot match the technique and task without consideration of the particular problem at hand, with the relevant domain knowledge. A task, having its own particular functionality, will place requirements that will be different for that unique functionality, but will share some requirements with other tasks where functionality is shared. For example, most tasks will require some kind of ranking mechanism (apart from the case where Exact Match operators are required), while lexical analysis of terms is done at indexing. Techniques can also be applied to more than one element of the taxonomy, as in the Okapi experiments where both terms selection and term weights were optimised. There are some cases where local search techniques could be applied to any given task for a particular part of the taxonomy (see the discussion on stemming below).

The practitioner must consider the issue of how a query (or profile in the case of some 'queryless' tasks) from a given task is represented (Chen, 1995). For example, weights may be best represented as a vector of floating-point numbers, and a term selection task as a binary string, one bit for each term to be included/excluded. In GAs, a query can be represented by a chromosome, where each query term is represented by a Gene. Each Gene can either represent the presence or absence of a given term, or a more complex entity which represents a term together with its weight. Operators on chromosomes such as crossover (Vrajitoru, 1998) can be investigated, in order to ascertain which is best for a particular task. In Neural Networks, associations between terms or between terms/documents can be established with the links assigned some weight (a floating-point number). Methods such as back propagation can then be investigated over the networks created by these associations (Chen Shankaranarayanan and She, 1998).

5.3 Matching operations

The first major class of operations to be considered is matching operations i.e. taking a query and applying an operation to retrieve a set of documents which 'match' that query. This usually means taking sets retrieved from inverted lists (Harman et al, 1992) and merging them together to create a final set for presentation. Two major types of operations in IR are defined: Exact Match (Wartik, 1992) and Best Match (Harman, 1992b).

5.3.1 Exact Match

Local search is applied to the information filtering task, where the requirement is to build a Boolean query user profile. Exact match operators such as Boolean (AND, OR, NOT) and Adjacency (phrases, same sentence, same paragraph) can be optimised in conjunction with the query terms they are applied to. Genetic Programming (GP) has been applied to this problem. The query is treated like a program, and different forms of the query are 'evolved' and evaluated against the fitness function. Cordon et al (2006) use a Multi-objective pareto based evolutionary algorithm to optimize Boolean queries. Smith and Smith (1997) use a GP method which creates an initial 1,000 'Boolean' organisms using the terms set and three Boolean operators (AND, OR, NOT), 'breeding' the organisations until some optimum is reached. Fernandez-Villacanas Martin and Shackleton (2003) compare a GP with a GA using a number of fixed bits for genes to represent function types and negation. Both of these methods are based on decision trees. Cordon et al (2002) also use GP on fuzzy sets, where a fuzzy set weight is generated and optimised using a simulated annealing process – this method is a hybrid Best/Exact match scheme.

5.3.2 Best Match

Best match operators, unlike exact match ranking documents given some order, usually based on the motive of retrieving more relevant documents higher up the rank. In order to do this, term weighting schemes assign a weight using some function that utilises collection statistics such as inverse document frequency (IDF) and term frequency (TF). There are two main way of optimising this information; applying a technique to a pre-defined weight or learning a function from a set of given parameters. Optimisation on this aspect is applicable to virtually all tasks in IR, although for the ad-hoc task this will mean offline processing as real time support for optimisation operations would be very hard to support.

With respect to learning on a pre-defined weight, the usual method is to use some matching function model such as BM25 and optimise the weight by varying it. A scenario could be envisaged where no matching function was used, and a random weight generated to be optimised on. In IR terms this would be theoretically very suspect, as matching functions such as BM25 would do far better than random weights, and matching on pre-defined weights produced by the former would provide a better initialisation in the search space. Some examples of work done in this area include Martin-Bautista et al (1999), who learn weights on fuzzy set genes and Horng & Yeh (2000) who use a GA to adapt pre-assigned weights produced by the vector space model. Other types of evidence can also be used to adapt weights. Lam et al (2003) use link information to optimise weights for documents on an information filtering task. Belew (1989) used a NN to adapt weights on links between authors, documents and keywords. A disadvantage with this method is there is no trail of evidence as to why a particular weight has been produced.

However optimising on functions does produce evidence – a list of parameters which have been produced during training. These parameters can either be variables such as term frequency or tuning constants such as K1 and B used in the BM25 weighting function (Robertson et al, 1995). As with Best Match operators, weighting functions can be treated as

programs, and optimised using Genetic Programming techniques. Fan et al (2004a; 2004b; 2004c; 2005) learn ranking functions per query on variables only (no constants are used), the argument being that there is no best match function for a query or query set. Alternatively Burges et al (2005) learn ranking functions using NN gradient descent methods utilizing a function based on a preferable ranking for documents – the ranking from each instantiated parameter value is compared with this preferred value, and the NN learns accordingly. Taylor et al (2006) compare this method with line search, and conclude that gradient descent appears to be a very useful method for such a task and that gradient descent and line search track each other quite closely with respect to final model effectiveness. There is also considerable interest in optimising tuning parameters for weighting functions. For example the BM25 weighting function will require values for each K1, B constant pairs when undertaking XML retrieval – the number of values depending directly on the element set size (Wei et al, 2006). In structured document retrieval element parameters for weighting functions can also be optimised: Trotman (2005) uses a GA to learn field parameters for each node of the XML tree using a range of 0 to 1 with each parameter – each node in the tree being a gene, the chromosomes representing the flattened tree. Choosing values for constants can therefore become a significant issue, and can quickly become unmanageable for any manual selection method. In such cases local search can provide a reasonable solution.

5.4 Term Operations

The second major class of query operations is on the terms which constitute the 'query' and are the source for matching operations (see above). Term operations are divided up into two main types, term selection using relevance feedback (Harman, 1992a) or operations individual words such as stemming (Frakes, 1992b).

5.4.1 Term Selection

Local search is useful for tasks which need to optimise a set of terms e.g. information filtering, text classification and topic detection and tracking. For the most part this optimisation process works on the presence or absence of terms in a query, as described in the Okapi examples above. Other examples include Chen Shankaranarayanan and She (1998) who use a GA in conjunction with an SA in order to optimise term selection. Queries can be treated as fixed size or variable size entities; the Okapi experiments used the former, whereas Lopez-Pujalte et al (2003a) used a variable size chromosome together with a binary representation for a gene. Wong et al (1993) describe the process of learning a term association matrix using NN's. i.e. relationships between terms. The strength of relationships are fed in to the NN; relationships are therefore learned, not the strength of them (there is no reason why optimisation could not be applied to each relationship). Terms for a given query can then be selected on the back of the relationships identified in the learning process. Tamine et al (2003) and Boughanem et al (2002) describe a technique that evaluates multiple queries using population niches and merging the results. Niches are sections of the population which are allowed to evolve independently. Mock and Rao Vermuri (1997) and Mock (1996) describe a global hill climbing method to select terms that uses collaborative evidence from a set of queries rather than evidence from a single queries task, a method targeted at information filtering. This method may be useful for some collaborative tasks, but its use is primarily for tasks that require optimisation for an individual user on the basis of their particular information need. This is a good example of when to use local as opposed to global search. Global search could usefully be applied to indexing schemes such as stop wording, particularly words which are reasonably frequent, but may be poor search terms in some circumstances.

5.4.2 Word operations

In this context we refer to operations on terms themselves, i.e. the constituent part of the queries being optimised. Word operations include stemming, truncation, lexical analysis and synonym detection. Most tasks use these operations, and therefore local search can be generically applied to word operations in them. Some operations such as stemming and lexical analysis are done at indexing time for the most part and would therefore be candidates for global search techniques (as with stop wording). Query time operations such as truncation would be a more appropriate target for local search. Note that there are some similarities between stemming and truncation, but the former is based on a series of rules while the latter is more free form. We could consider the use of GP for stemming, treating a set of rules as elements of a program and optimising to find the best set for a stemmer. The authors are not aware of any work done using GP, but Mladenić (2002) describes a local search algorithm based on k-Opt, a neighbourhood operator used for the travelling salesman problem, for learning stemmer rules in text classification. Operations on truncation could be used in conjunction with term selection to find good query stems at search time (no work has been identified in this area). Synonym detection is another phrase for dimension reduction; this can either be done at indexing or search time. An example of using local search for dimension reduction is the work by Kuflik et al (2006), which uses a term reduction process similar to term selection, where the number of dimensions is reduced at every iteration until some limit has been reached. The method uses a version of a NN called the Guteman-Boger algorithm.

6. Evaluation methods for Local Search in IR

The most appropriate method for evaluation when applying local search to IR is the laboratory style evaluation method. This requires the use of test collections (see below), rather than users in an operational style of evaluation. It is entirely possible for users to feed relevance assessments into a local search process, but this is out of the scope of the paper and we do not discuss it here. The emphasis here is on applying laboratory style of evaluations in a rigorous way when evaluating local search, by looking at the role of test collections and fitness functions. They are used to drive the process of local search and have an impact on the outcomes. We point out how some research in the area has been methodologically flawed, and how these issues can be resolved by looking at benchmarking and test collections as well as fitness functions used in the optimization process.

6.1 Benchmarking and Test Collections

The Cranfield paradigm has had a significant impact on the evaluation of IR systems over the years, particularly after the advent of TREC. Test collections for benchmarking have been developed including the somewhat small Cranfield set (containing 1400 records) and the GOV2 set which is 426GB in size, and contains 25 million web documents. A test collection does not only contain documents however. Just as important are the topics or information needs which must be serviced over the document set, and relevance judgements which indicate both relevant and irrelevant documents for that information need on the given document set. Test collections provide the basic framework for benchmarking and evaluation for local search experiments. There are some very important issues about each element of a test collection which must be considered when choosing such a set for evaluation purposes.

Test collections come in two types, those with special purpose and general purpose document datasets. Examples of both are Cranfield-2 which is a set of abstracts for the area of metallurgy (Cleverdon, 1967), and the TREC collection (Harman, 1995) which contains a range of documents including government reports and newspaper articles etc. It is better to use more general datasets as any statements made about the applicability of a method are more likely to be more generally useful; conclusions on an experiment on a special collection may not be so applicable to other types (Frakes, 1992a). Experiments on specialised collections should be done in the full knowledge of the limitations of those datasets, in particular an understanding of what they where designed for (Sparck Jones and van Rijsbergen, 1976). The size of a collection to be used for experiments is a very important issue. Many practitioners in the area still use Cranfield test collections (Lopez-Pujalte, 2003a; Cordon et al, 2006; Vrajitoru, 1998), and other use even small datasets e.g. a set of 359 abstracts by Cordon et al (2002), and a set of 200 'cases' by Fernandez-Villacanas Martin and Shackleton (2003). Whilst it is difficult to assert what the best size is for using test collections in experimentation, these datasets are almost certainly too small as the collection statistics (such as IDF, TF) will be very different from larger datasets which are of more interest to users. Early in the field of test collection research Sparck Jones and van Rijsbergen (1976) asserted that small collections "are unsatisfactory on purely statistical grounds, and the increasing scale of operational systems makes the results obtained with them appear irrelevant to real systems". Given the size of the web, this view is strongly reinforced. Hawking and Robertson (2003) provide some theoretical and empirical evidence on the relationship between collection size and effectiveness, in particular that precision at N documents retrieved (P@N) declines with a sample collection. Cranfield, being a set of abstracts for a special area (Cleverdon, 1967) thus has a particular set of problems in the context of experimentation and evaluation (Sparck Jones and van Rijsbergen, 1976). Local search practitioners have used reasonably sized and general datasets for experiments e.g. Tamine et al (2003) who use standard TREC datasets and Fan et al (2004a) who use the WT10gb collection (which is 10GB in size and contains 1.69 million records). Once a collection has been chosen for experimentation, it needs to be separated into training, validation and test sets to avoid overfitting (Sebastiani, 2002). Overfitting in the context of IR means that a term set produced from the training set yields poor results on the test set. Fan et al (2004a) use a scheme where 49% of the test collection is assigned for training, 21% for validation and 30% for test. Standard datasets used at TREC by Okapi for the Routing/Filtering tracks use around 2/3 of the collection for training and 1/3 for testing. The Okapi strategy of splitting the training set into extract and select sets is much the same as a split between training and validation sets. This is known as the holdout method. This can be extended by using the K fold cross validation method, where the holdout method is repeated K times. There is very little work on the issue of formally modelling datasets. One example is Cormack and Lynam (2006), who treat datasets as 'source populations' in order to examine the issue of random errors in them.

The second strand in a test collection is the set of information needs and topics, which are used for forming the query to be serviced over the collection. The issues of information need type and query set size need to be considered carefully. It should be noted that the issue of query size is not one to be overly concerned about for many tasks, most of which use relevance feedback – a process that produces plenty of terms. With respect to the type of information need, what we mean is how hard it is to formulate a successful query in order to retrieve relevant documents. If possible, it is best to have a mix of hard and easy topics, as this is more likely to be a reflection of users' information needs (Sparck Jones and van Rijsbergen, 1976). The number of information needs used in the experiment needs to be of a reasonable size; it is unlikely that any conclusions drawn from a small query set is transferable to other sets (Sparck Jones and van Rijsbergen, 1976). For example, Lam et al (2003) use 7 queries in one experiment for optimising search and 13 in another optimising relevance feedback. The Cranfield collection does have one advantage, it has 225 information needs associated with it. It should be noted however, that the number of records returned per query should also be considered together with the query

set size. For example, Buckley and Voorhees (2000) argue that when using web collections at TREC which evaluate only on the top 10 retrieved documents, many more than 50 queries will be needed. They also recommend the use of at least 25 topics, with 50 being preferable.

The third strand of the test collection, relevance judgements, are subjective assessments made by humans on information needs formulated into queries and applied to document sets. They can be negative or positive. Typically the relevance judgements are binary (0 or 1), but there is increasing interest in using graded relevance (Kekäläinen and Järvelin, 2002). The distribution of relevant documents differs significantly between collections (Hawking and Robertson, 2003), a further aspect any practitioner should think about when choosing a test collection. With respect to size, there is no rule about how many positive relevance judgements are needed for a particular topic. This number will naturally vary between topics where queries can either be hard or easy to formulate. However, it is a moot point as to how many relevance judgements are needed to make the best use of local search, or, to put it another way, when have enough relevant documents been accumulated for local search to become useful. For the hillclimbers used for Okapi experiments, around 20/30 is approximately the number of positive relevance judgements required to make such a scheme worth while i.e. very little will be gained by using the Okapi methods on 1 or 2 relevant documents. The number of positive relevance judgements has a direct effect on the fitness function; a topic with few relevant documents and optimised using average precision could quickly finish with a perfect score of 1 (MacFarlane, 2000). It is unclear how this evidence is likely to behave on the test set.

There are a couple of quite contentious issues in IR evaluation generally which impact on any practitioner interested in using local search for IR. The first of these is the issue of statistical significance testing in order to find any real difference between two outputs e.g. a comparison between an Okapi relevance feedback and hillclimbing procedures. Kuflik et al (2006) used the t-test to evaluate significance on their experiments, while Burges et al (2005) use Wilcoxon test. The t-test assumes a normal distribution, while the Wilcoxon is a non-parametric test. Sanderson and Zobel (2005) assert that the t-test is highly reliable under certain conditions (with respect to size of topic set and hit lists per topic), and is more useful than the Sign or Wilcoxon tests. This issue is somewhat contentious as the underlying distribution of relevant documents in topics is difficult to establish; relevance assessment may differ significantly on the relevant documents for a particular topic, where there can be no agreement at all between the assessors (Voorhees, 1998). Care therefore needs to be used when using statistical testing.

The second of these contentious issues relates to treatment of test collections for benchmarking purposes. Sebastiani (2002) asserts that comparisons for classifiers can only be performed when the test collection is the same (identical documents, topics and relevance judgements), the same split between training and test set is used, and finally the same evaluation measure must be applied to the test set. Unfortunately this is not the case from the literature in local search in IR, for example Lopez–Pujalte et al (2002) only use 33 of the 225 Cranfield queries for their experiments. Drawing any conclusions from the literature on which is the best local search technique for any given task is therefore very difficult if not impossible (Sparck Jones and van Rijsbergen, 1976). Similar statistical and experimental design issues arise in evaluating local search optimizers, which, given their stochastic nature, require evaluations to be made over a number of runs, e.g. (Christiensen, 2007).

6.2 Fitness functions

There is a bewildering array of fitness functions for local search in IR, but there is one common element for them: use of relevance judgements as the primary source of information (see above). Functions can either use positive relevance judgements, or use both positive or negative relevance judgements in conjunction with each other. For the most part this means binary relevance assessments, but non binary user assigned scores are used e.g. a score between 0-1 is assigned by a user in (Martin-Bautista et al, 1999). Graded relevance assessments as put forward by Kekäläinen and Järvelin (2002) are also be potentially used in fitness functions e.g. using an ideal ranked set to compare the difference between two outputs when using Gradient Descent methods (Burges et al, 2005; Taylor et al, 2006). We divide fitness functions into two types, set-based and rank-based.

Set based functions are used in binary classification tasks (Lewis, 1997), which separate documents into accepted and rejected elements, accepted elements being in the set, rejected being outside. The sets that this type of fitness function evaluates do not imply order. The two main forms of set based measures are Precision and Recall, used since the days of the early Cranfield experiments. In the context of local search for IR, set based functions are largely used on Exact Match operations, such as (Smith and Smith, 1997; Fernandez-Villacanas Martin and Shackleton, 2003; Cordon et al, 2006) who use functions which utilise a weighted combination of precision and recall. Recall has also been used as a fitness measure (Cordon et al, 2002). Other methods for set based fitness functions include one used by Chen, Yi-Ming et al (1998) and Yang et al (2000), which utilises the difference between two retrieved sets using a Jacquard scoring mechanism.

Rank based functions do imply order and are used to evaluate the fitness of Best Match operators. It has been suggested by Fan et al (2004a) from Lopez-Pujalte et al (2003a) that order based functions yield better results than set based methods, and this is not inconsistent with evidence found in Okapi experiments. The most popular fitness function appears to be average precision; a recall-based precision measure used in standard IR evaluations such as TREC. This was found to be the best predictive measure in Okapi experiments (Robertson et al, 1995). Buckley and Voorhees (2000) assert that

average precision is a "reasonably stable and discriminating choice" for general purpose retrieval. Others who have used this measure include Fan et al (2006) and Horng & Yeh (2000). Lopez-Pujalte et al (2002) and Lopez-Pujalte et al (2003b) provide an extensive survey on 12 fitness functions, and found that the choice of fitness function was essential when guiding GA's through the search space. Other examples include use of the Guttman model, a statistical measure of rank correlation which has many of the same properties of average precision according to Tamine et al (2003). All rank based measures evaluate on the top N documents, for example the standard top 1000 retrieved used for TREC style evaluations. However not all fitness functions work in this way, for example Utility functions. These are based on the preference of a user for a given document, which can either be based on rank preferences (Fan et al, 2004a) or based on rewarding ranking mechanisms which retrieve relevant documents, whilst penalising them for retrieving irrelevant documents. Utility functions are tuned to preferences by some parameters e.g.

Equation 3

where A and B are relevant and non-relevant documents respectively and u^1, u^2 are the utilities for retrieving relevant and non-relevant. Fan et al (2004a) show their rank based utility functions provide better results than using average precision – this contradicts results from the Okapi experiments (see above).

What fitness function is best for a particular task, or element of the taxonomy? For the most part the task itself will determine the fitness function as different tasks imply different evaluation measures e.g. the topic detection and tracking task is evaluated by miss rate (new events not detected and assigned to existing events) and false alarm rate (events already identified incorrectly detected as new events). In some tasks therefore, fitness functions based on average precision are not appropriate. Slight modifications to average precision may be useful however. If the user is more interested in harder topics, geometric average precision (GMAP) may be a more useful measure to use (Robertson, 2006). This measure is more sensitive to observations at the lower end of the ranking scale than mean average precision, potentially allowing the local search method to emphasize those elements (e.g. terms) which yield better results at the bottom end.

The stability of the function used is another very important consideration. Stability in this context means the ability to use a fitness function in order to predict a good outcome for another. In recent years the Binary Preference or BPREF function (Buckley and Voorhees, 2004) has been put forward for IR evaluation. There has been some debate on the merits of using average precision for evaluation (Buckley and Voorhees, 2004) as these measures do not explicitly use identified non-relevant documents, and do not differentiate between identified and non identified non-relevant documents. When you use measures such as average precision, some important evidence is discarded which could be used to support optimisation. With BPREF the information recorded is pairs of non-relevant documents ranked higher than relevant documents, so preference of non-relevant over relevant documents is explicitly recorded in the function. Buckley and Voorhees (2004) demonstrated that the measure correlated with the original ranking, while mean average precision did not. With small numbers of relevance assessments (say 1 or 2) it is difficult to use BPREF measure as there are too few pairs to usefully distinguish between objects leading to severe overfitting. There is a question on the stability of the BPREF function at low sampling rates – correlations between systems rankings tend to deteriorate at this level (Aslam et al, 2006). However, it is not clear that using little evidence is much use for optimisation problems in IR (see section 6.1 above).

Should the same function be applied to the training set as to the test set? Taylor et al (2006) assert that ranked based methods are not efficient in terms of run time for tuning ranking function parameters. They suggest that gradient descent methods using cost functions are far more efficient than fitness functions that depend on ranking of documents (relevant or irrelevant). However rank based methods are more in line with what the user wants from an IR system, and attempts to use methods outside of this could pose serious problems.

7. Conclusion and Discussion

This paper has concerned itself with providing an introduction to the neighbourhood search paradigm and the main techniques in use today for IR. We have introduced a Query based taxonomy for examining local search techniques, which show when optimisation can be applied, and which techniques are appropriate for each element of the taxonomy or a particular task. The paper also examined methodological issues and good practices from the IR and optimisation literature. A number of conclusions can be drawn from this paper.

- Though local search optimizers show clear promise and wide applicability, it is hard to say definitely if many of the proposed methods have achieved real improvements over conventional IR approaches, a notable exception being (Robertson et al, 1995).
- Because of this, comparisons between the different optimizer techniques are hard to make. This is for two reasons; from an IR perspective, inadequate and differing test collections are used in evaluation; from a local search perspective insufficient use of samples and hypothesis testing is made (Hooker, 1995).
- To take the use of local search in IR forward, frameworks are needed to help IR practitioners relate their experience to the appropriate design choices for local search. As a starting point this paper has proposed a Query based taxonomy to

assist in decisions regarding what is optimised, and reviewed an approach due to (Tuson, 2000) that guides the use of domain knowledge in optimizer design.

The next stage in this area of research is to take the taxonomy and theory of optimizer design and build a framework with which to better inform the practitioner of choices in deploying local search methods. The framework would take account of the known limitations of optimizer design (Papadimitriou & Steiglitz, 1982), but would take the taxonomy derived from the literature in this paper to build concepts on which methods can be chosen to solve given problems.

Finally the metaphors used should not be taken too literally. For example, the evolutionary metaphor has expressed itself in a belief that such natural processes inherently possess some additional computational power; examples of this view can be readily found in the evolutionary computation literature. However, the No Free Lunch (NFL) results discussed earlier and the effectiveness of other approaches such as Tabu Search do speak against adhering to this view too strongly. Furthermore (Glover and Laguna, 1997) argue that though such metaphors do have a place, to help suggest ideas from which to launch an investigation, but that problems begin when the metaphor is taken too far and is allowed to define the actual research agenda.

Acknowledgements

Many thanks to Marcus Andrews for his positive feedback on the paper, and to the reviewers of an earlier draft for their constructive comments.

References

Abramson, D. (1991). Constructing school timetables using simulated annealing: sequential and parallel algorithms, Management Science, Vol. 37, No. 1, pp. 98-113.

Althofer I. and Koschnick, K.U. (1991). On the convergence of `threshold accepting', Applied Mathematics and Optimisation, Vol. 24, No. 1, pp. 183-195.

Aslam, J.A., Pavlu, V. and Yilmaz, E. (2006). A statistical method for systems evaluation using incomplete judgements. In: Belkin, N. and van Rijsbergen, K. (eds). 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: SIGIR 2006, pp. 541-548.

Battiti R. and Tecchiolli, G. (1994). The reactive tabu search. ORSA Journal on Computing, Vol. 6, No. 2, pp. 126-140.

Baeck, T., Fogel, D.B. and Michalewicz. Z. (1997). Handbook of Evolutionary Computation, Oxford University Press, Oxford.

Belew, R.K. (1989). Adaptive Information Retrieval: using a connectionist representation to retrieval and learn about documents. In: Belkin, N. and van Rijsbergen, K. (eds). 12th International Conference on Research and Development in Information Retrieval: SIGIR'89, pp. 11-20.

Boughanem, M., Chrisment, C. and Tamine, L. (2002). On using genetic algorithms for Multimodel Relevance Optimisation in Information Retrieval, Journal of the American Society for Information Science and Technology, Vol. 53, No. 11, pp. 934-942.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender G. (2005). Learning to rank using Gradient Descent. In: De Raedt, L., Wrobel, S. Proceedings of the 22nd International Conference on Machine Learning, pp. 89–96.

Buckley, C. and Voorhees, E.M. (2000). Evaluating evaluation measure stability. In: Belkin, N.J., Ingerwersen, P. and Leong, M. (eds). Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: SIGIR 2000, pp. 33-40.

Buckley, C. and Voorhees, E.M. (2004). Retrieval evaluation with Incomplete information. In: Jarvelin, K., Allen, J., Bruza, P. and Sanderson, M. Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: SIGIR 2004, pp. 25-32.

Cartwright H.M. and Harris, S.P. (1993). Analysis of the distribution of airborne pollution using genetic algorithms. Atmospheric Environment, Part A, General Topics 27:1212, pp. 1783-1791.

Chen, H. (1995). Machine learning for information retrieval: neural networks, symbolic learning, and genetic algorithms. . Journal of the American Society for Information Science and Technology, Vol. 46, No. 3, pp. 194-216.

Chen, H, Shankaranarayanan, G., She, L. and Iyer, A. (1998). A machine learning approach to inductive query by examples: an experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing. Journal of the American Society for Information Science and Technology, Vol. 49, No. 8, pp. 693-705.

Chen, H., Yi-Ming, C, Ramsey, M and Yang, C. (1998). An intelligent personal spider (agent) for dynamic Internet/Intranet searching. Decision Support Systems, Vol. 23, No. 1, pp. 41-58.

Christiensen, S. (2007). Tutorial Notes: Using Appropriate Statistics. Presented at the Genetic and Evolutionary Computation Conference (GECCO 2003).

Cleverdon, C. W. (1967). The Cranfield test on index language devices, ASLIB Proceeding, 19. In: Spark Jones, K and Willet, P. (eds), Readings in Information Retrieval, Morgan Kaufmann, San Francisco, pp. 47- 59.

Collins, N.E., Eglese, R.W. and Golden, B.L. (1988). Simulated annealing - an annotated bibliography, American Journal of Mathematical and Management Sciences, Vol. 8, pp. 209-307.

Cordon, O. Herrera-Viedma, E. and Luque, M. (2006). Improving the learning of Boolean queries by means of an multiobjective IQBE evolutionary algorithm. Information Processing and Management, Vol. 42, No. 3, pp. 615-632.

Cordon, O., Moya, F. and Zarco, C. (2002). A new evolutionary algorithm combining simulated annealing and genetic programming for relevance feedback in fuzzy information retrieval systems. Soft Computing, Vol. 6, No. 5, pp. 308-319.

Corne, D. Fang, H.-L. and Mellish, C. (1993). Solving the Module Exam Scheduling Problem with Genetic Algorithms. In: Chung, P.W.H., Lovegrove, G. and Ali, M. (eds), Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Gordon and Breach Science Publishers, pp. 370-373.

Cormack, G.V. and Lynam, T.R. (2006). Statistical precision of Information Retrieval evaluation. In: Belkin, N. and van Rijsbergen, K. (eds). 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: SIGIR 2006, pp 533-540,

Darwin, C. (1859). On the Origin of Species, John Murray, London.

Dorigo, M. and Gambardella, L.M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 53-66.

Downsland, K.A. (1991). Hill-climbing, simulated annealing, and the Stenier problem in graphs. Engineering Optimization, Vol. 17, No. 1, pp. 91-107.

Dueck, G. and Scheuer. T. (1990). Threshold Accepting: A General Purpose Optimisation Algorithm Superior to Simulated Annealing, Journal of Computation Physics, Vol. 90, No. 1, pp. 161-175.

Dueck, G. (1990). New Optimisation Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel, Technical report, IBM Germany, Heidelburg Scientific Center, Heidelburg.

Fan, W. Fox, E.A. Pathak, P. and Wu, H. (2004a). The effects of fitness functions on Genetic Programming-based ranking discovery for web search. Journal of the American Society for Information Science and Technology, Vol. 55, No. 7, pp. 628-636.

Fan, W. Gordon, M.D. and Pathak, P. (2004b). Discovery of context-specific ranking functions for effective information retrieval. IEEE Transaction on Knowledge and Data Engineering, Vol. 16, No. 4, pp. 523-527.

Fan, W. Gordon, M.D. and Pathak, P. (2004c). A generic ranking function discovery framework by genetic programming for information retrieval, Information Processing and Management, Vol. 40, No. 4, pp. 587-602.

Fan, W. Gordon, M.D. and Pathak, P. (2006). An integrated two-stage model for intelligent information routing. Decision Support Systems, Vol. 42, No. 6, pp. 362-374.

Feo, T.A. Bard, J.F. and Claflin, R.F. (1991). An overview of GRASP methodology and applications, Technical report, University of Texas at Austin, Austin.

Feo, T.A. Venkatraman, K. and Bard, J.F. (1991). A GRASP for a Difficult Single Machine Scheduling Problem. Computers & Operations Research, Vol. 17, No. 8, pp. 635-643.

Fernandez-Villacanas Martin, J.L. and Shackleton, M. (2003). Investigation of the importance of the genotype-phenotype mapping in information retrieval, Future Generation Computer Systems, Vol. 19, No. 1, pp. 55-68.

Frakes, W. (1992a), Introduction to information storage and retrieval systems, In: Frakes, W.B. and Baeza-Yates (eds), Information Retrieval: Data Structures and Algorithms, Prentice Hall, New Jersey, pp 1-12.

Frakes, W. (1992b). Stemming algorithms. In: Frakes, W.B. and Baeza-Yates (eds), Information Retrieval: Data Structures and Algorithms, Prentice Hall, New Jersey, pp 131-160.

Fogel, D. B. and Angeline, P. J.. (1997). Guidelines for a Suitable Encoding. In Bäck, T... Fogel, D. B , and Michalewicz, Z. (Eds.), Handbook of Evolutionary Computation, Section C1.7, IOP Publishing and Oxford University Press, Bristol/New York, 1997.

Fogel, L.J., Owens, A.J. and Walsh, M.J. (1966). Artificial Intelligence Through Simulated Evolution, John Wiley and Sons.

Garey, M.R. and Johnson, D.S. (1979). Computers and Interactability: a Guide to the theory of NP-Completeness, Freeman, San Francisco.

Glover. F.W. (1990). Tabu Search: A Tutorial, Interfaces, Vol. 20, No. 4, pp. 445-460.

Glover, F.W. (1995). Tabu Thresholding: Improved Search by Nonmonotonic Trajectories. ORSA Journal on Computing, Vol. 7, No. 4, pp. 426-442.

Glover, F.W. and Laguna, M. (1997). Tabu Search, Kluwer Academic Publishers, Boston.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization & Machine Learning. Addison Wesley, Reading.

Hajek. B. (1988). Cooling schedules for optimal annealing, Mathematical of Operational Research, Vol. 13, No. 2, pp. 311-329.

Harman, D. (1992a). Relevance feedback and other query modification techniques. In: Frakes, W.B. and Baeza-Yates (eds), Information Retrieval: Data Structures and Algorithms, Prentice Hall, New Jersey, pp 241-263.

Harman, D. (1992b). Ranking algorithms. In: Frakes, W.B. and Baeza-Yates (eds), Information Retrieval: Data Structures and Algorithms, Prentice Hall, New Jersey, pp 363-392.

Harman, D. (1995). The TREC Conferences, Procedings of HIM'95, In: Spark Jones, K and Willet, P. (eds), Readings in Information Retrieval, Morgan Kaufmann, San Francisco, pp. 247-256.

Harman, D, Fox, E., Baeza-Yates, R, and Lee, W. Inverted files. (1992). In: Frakes, W.B. and Baeza-Yates (eds), Information Retrieval: Data Structures and Algorithms, Prentice Hall, New Jersey, pp. 28-43.

Hawking, D. and Robertson, S.E. (2003). On collection size and retrieval effectiveness. Information Retrieval, Vol. 6, No. 1, pp 99-150.

Hasan M. and Osman, I.H. (1995). Local search algorithms for the maximal planar layout problem. International Transactions in Operations Research, Vol. 2, No. 1, pp. 89-106.

Hertz, A. and deWerra, D. (1987). Using tabu search techniques for graph coloring, Computing, Vol. 39, No. 4, pp. 345-351.

Hertz, J., Krogh, K. and Palmer, R.G. (1991). Introduction to the Theory of Neural Computation, Addison-Wesley, Reading.

Hjorring, C.A. (1995). The Vehicle Routing Problem and Local Search Metaheuristics. PhD thesis, The University of Auckland, New Zealand.

Holland, J.H. (1975). Adaptation in Natural and Artificial Systems,The University of Michigan Press, Ann Arbor.

Hooker, J.N. (1995). Testing heuristics: We have it all wrong. Journal of Heuristics, Vol. 1, pp 33-42.

Horng, J.T. and Yeh, C.C. (2000). Applying genetic algorithms to query optimization in document retrieval, Information Processing and Management, Vol. 36, No. 5, pp. 737-759.

Kekäläinen, J and Järvelin, K. (2002). Using Graded Relevance assessments in IR evaluation. Journal of the American Society for Information Science and Technology, Vol. 53, No. 13, pp. 1120-1129.

Kirkpatrick, S., Gelatt, Jr., C.D. and Vecchi, M.P. (1983). Optimization by Simulated Annealing, Science, Vol. 220, No. 4598, pp. 671-680.

Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge.

Kuflik, T., Boger, Z. and Shoval, P. (2006). Filtering search results using an optimal set of terms identified by an artificial neural network, Information Processing and Management, Vol. 42, No. 2, pp. 469-483.

Lam, W., Wang, W. and Yue, C. (2003). Web discovery and filtering based on textual relevance feedback learning. Computational Intelligence, Vol. 19, No. 2, 136-163.

Lewis, D.D. (1997). The TREC-5 Filtering Track, In: Vooheers, E.M. and Harman, D.K., Proceedings of the Fifth Text Retrieval Conference, Gaithsburg, Maryland, November 20-22, 1996, NIST Special Publication 500-238, pp. 75-96.

Lopez-Pujalte, C., Guerrero Bote, V.P. and de Moya Anegon, F. (2002). A test of genetic algorithms in relevance feedback, Information Processing and Management, Vol. 38, No. 6, pp. 793-805.

Lopez-Pujalte, C., Guerrero-Bote, V.P. and de Moya-Anegon, F. (2003a). Order-based fitness functions for genetic algorithms applied to relevance feedback. Journal of the American Society for Information Science and Technology, Vol. 54, No. 2, pp. 152-160.

Lopez-Pujalte, C., Guerrero-Bote, V.P. and de Moya-Anegon, F. (2003b). Genetic algorithms in relevance feedback: a second test and new contributions, Information Processing and Management, Vol. 39, No. 5, pp. 669-687.

Luke, B. T. (1996). An Overview of Genetic Methods. In Devillers, J. (Ed.), Genetic Algorithms in Molecular Modelling, Academic Press, London, pp. 35-66

Lundy, M. and Mees. A. (1986). Convergence of an annealing algorithm. Mathematical Programming, Vol. 34, No. 1, pp. 111-124.

Martin-Bautisata, M.J., Vila, M. and Larsen, H.L. (1999). A Fuzzy genetic algorithm approach to an adaptive information retrieval agent. Journal of the American Society for Information Science and Technology, Vol. 50 No. 9, pp. 760-771.

Masters, T. (1993). Practical Neural Network Recipes in C++, Academic Press, London.

MacFarlane, A. (2000). Distributed inverted files and performance: a study of parallelism and data distribution methods in IR, PhD Thesis, City Univeristy.

Motta, E. (1997). Reusable Components in Knowledge Modelling, PhD Thesis, Open University,U.K.

Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs, Artificial Intelligence. Springer-Verlag, New York.

Mladenić, D. (2002). Automatic word lemmatization. In: Erjavec, T, and Gros, J. (eds), Proceedings of ISJT'02, Information Society Language Technologies, pp. 153-159. [Available on: http://nl.ijs.si/isjt02/zbornik/sdjt02-26mladenic.pdf -Visited 15th September 2008]

Mock, K.J. (1996). Hybrid Hill-Climbing and Knowledge-Based techniques for Intelligent News Filtering, In: Clancy, W.J, and Weld, D. (eds), Proceedings of the Thirteenth National Conference on Artificial Intelligence, 2-8. Menlo Park, Calif..: AAAI Press, Vol. 1, pp. 48-53.

Mock, K.J. and Rao Vemuri, V. (1997). Information filtering via Hill Climbing, Wordnet, and index patterns, Information Processing and Management, Vol. 33 No. 5, pp. 633-644.

Newell, A. (1982). The Knowledge Level, Artificial Intelligence, Vol. 18 No. 1, pp. 87-127.

Ogbu F.A. and Smith, D.K. (1990). The application of the simulated annealing algorithm to the solution of the n/m/P/C_{max} flowshop problem, Computers and Operational Research, Vol. 17 No. 3, pp. 243-253.

Oliveira, S. and Stroud, G. (1989). A parallel version of tabu search and the path assignment problem, Heuristics for Combinatorial Optimisation, Vol. 4, No. 1, pp. 1-24.

Osman. I.H. and Potts, C.N. (1989). Simulated annealing for permutation flow-shop scheduling, OMEGA, Vol. 17, No. 6, pp. 551-557.

Osman, I.H. and Christofides, N. (1994). Capacitated clustering problems by hybrid simulated annealing and tabu search, International Transactions in Operations Research, Vol. 1 No. 3, pp. 317-336.

Osman. I.H. (1995). An introduction to Meta-Heuristics. In Lawrence M. and Wilsdon, C. (Eds), Operational Research Tutorial Papers, Operational Research Society, pp. 92-122.

Osman. I.H. and Kelly, J.P. (1996). Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, Boston.

Osman. I.H. and Laporte, G. (1996). Metaheuristics for combinatorial optimisation problems: An annotated bibliography, Annals of Operational Research, Vol. 63 No. 5, pp. 513-623.

Papadimitriou, H., and Steiglitz, K. (1982). Combinatorial Optimisation: Algorithms and Complexity. Prentice-Hall, New Jersey.

Pohlheim, H. and Marenbach, P. (1996). Generation of Structured Process Models Using Genetic Programming, In T. C. Fogarty (ed.), Evolutionary Computing: AISB Workshop (LNCS 1143*)*, Springer Verlag, Berlin, pp. 102-109.

Radcliffe, N.J. (1992). Non-Linear Genetic Representations. Technical Report, Edinburgh Parallel Computing Centre.

Rayward-Smith, V. J (1994). A Unified Approach To Tabu Search, Simulated Annealing and Genetic Algorithms. In: Rayward-Smith, V.J. (ed) The Proceedings of the UNICOM Seminar on Adaptive Computing and Information Processing, Vol. 1, pp. 55-78.

Rechenburg, I. (1973). Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzberg.

Reeves, C.R. (1993). Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, Oxford.

Reeves, C.R. (1995). A genetic algorithm for flowshop sequencing, Computers & Operational Research, Vol. 22, No. 1, pp. 5-13.

Robertson, S. (1990). On term selection for query expansion. Journal of Documentation, Vol. 46, No. 4, pp. 359-364.

Robertson, S. (2006). On GMAP – and other transformations. In: Yu, P.S., Tsotras, V.J., Fox, E.A, and Liu, B. (eds) Proceedings of ACM Fifteenth Conference on Information and Knowledge Management: CIKM 2006, pp. 78-83.

Robertson, S, Walker, S., Jones, S., Hancock- Beaulieu, M., and Gatford, M. (1995), Okapi at TREC-3, In: Harman, D (ed). Proceedings of the Third Text Retrieval Conference, Gaithersburg, November 1994, NIST SP 500-226, pp 109-126.

Robertson, S, Walker, S., Jones, S., Hancock- Beaulieu, M., Gatford, M. and Payne, A. (1996), Okapi at TREC-4, In: Harman, D (ed). Proceedings of the Fourth Text Retrieval Conference, Gaithersburg, November 1995, NIST SP 500-236, pp. 73-96.

Sanderson, M. and Zobel, J. (2005). Information retrieval systems evaluation: effort, sensitivity, and reliability. In: Marchionini, G., Moffat, A., Tait, J. Baeza-Yates, Ziviani, N. (eds). Proccedings of the 28th Annual International ACM conference on Research and Development in Information Retrieval: SIGIR 2005, pp. 162-169.

Sebastiani, F. (2002). Machine learning in automated text categorization, ACM Computing Surveys, Vol. 34 No. 1, pp. 1-47.

Semet, F. and Taillard. E. (1993). Solving real-life vehicle routing problems effectively using taboo search, Annals of Operational Research, Vol. 41, No. 4, pp. 469-488.

Sinclair, M. (1993). Comparison of the performance of modern heuristics for combinatorial problems on real data. Computers and Operations Research, Vol. 20, No. 7, pp. 687-695.

Smith, M.P. and Smith, M. (1997). The use of genetic programming to build Boolean queries for text retrieval through relevance feedback, Journal of Information Science, Vol. 23, No. 6, pp. 423-431.

Sparck Jones, K., and van Rijsbergen, C.J. (1976). Information retrieval test collections. Journal of Documentation, Vol. 32, No. 1, pp59-75.

Stefik, M. (1995). Introduction to Knowledge Systems, Morgan Kaufmann, San Mateo.

Taylor, M., Zaragoza, H., Craswell, N., Robertson, S. and Burges, C. (2006). Optimisation methods for ranking functions with multiple parameters. In: Yu, P.S., Tsotras, V.J., Fox, E.A, and Liu, B. (eds) Proceedings of ACM Fifteenth conference on Information and Knowledge Management: CIKM 2006, pp. 585-593.

Tamine, L., Chrisment, C. and Boughanem, M. (2003). Multiple query evaluation based on an enhanced genetic algorithm, , Information Processing and Management, Vol. 39, No. 2, pp. 215-231.

Thangiah, S.R. (1995). An adaptive clustering method using a geometric shape for vehicle routing problems with time windows. In Eshelman, L.J. (ed), Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, Ca., Morgan Kaufmann, pp. 536-544

Trotman, A. (2005). Choosing document structure weights, Information Processing and Management, Vol. 41, No. 2, pp. 243-264.

Tuson, A.L. (2000). No Optimisation Without Representation: A Knowledge-Based Systems Approach to Evolutionary/Neighbourhood Search Optimiser Design. PhD Thesis, University of Edinburgh, U.K.

Vrajitoru, D. (1998). Crossover improvement for the genetic algorithm in information retrieval. Information Processing and Management, Vol. 34, No. 4, pp. 405-415.

Van Laarhoven P.J.M. and Aarts. E.H.L. (1988). Simulated Annealing: Theory and Applications, Kluwer, Dordrecht.

Voorhees, E. (1998). Variations in Relevance judgement and the measurement of retrieval effectiveness, In: Croft, W.B., Moffat, A, van Rijsbergen, C.J., Wilkinson, R. and Zobel, J. Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: SIGIR 1998, pp. 315-323.

Voorhees, E. and Harman D (2000). Overview of the Eighth Text REtrieval Conference. In: Voorhees, E. and Harman D (eds) NIST Special Publication 500-246: The Eighth Text REtrieval Conference, pp. 1-24.

Wartik, S. (1992). Boolean Operations. In: Frakes, W.B. and Baeza-Yates (eds), Information Retrieval: Data Structures and Algorithms, Prentice Hall, New Jersey, pp 264-292.

Walker, S., Robertson, S, and Boughanem, M. (1998). Okapi at TREC-6: automatic ad hoc, VLC, routing and filtering. In: Voorhees, E. and Harman, D (eds). Proceedings of the Fifth Text Retrieval Conference, Gaithersburg, November 1996, NIST SP 500-240, pp.125-136.

Wei L., S.E. Robertson, and A, Macfarlane (2006). Field-Weighted XML Retrieval Based on BM25, In: Fuhr, N., Lalmas, M., Malik, S. and Kazai, G. Proceedings of the 4rh International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, LNCS 3977, pp. 161-171.

Wolpert D.H. and Macready, W.G. (1995). No free lunch theorems for search, Technical report, SFI-TR-95-02-010, Santa Fe Institute.

Wong, S.K.M., Cai, Y.J. and Yao. Y.Y. (1993). In: Korfage, R, Rasmussen, E. and Willett, P. (eds). Proceedings of the 16th Annual ACM conference on Research and Development in Information Retrieval: SIGIR'93, pp. 107-115.

Yang, C., Yen, J. and Chen, H. (2000). Intelligent Internet searching agent based on hybrid simulated annealing. Decision Support Systems, Vol. 28, No. 3, pp. 269-277.

Zweben, M., Davis, E., Daun, B. and Deale, M. (1993). Scheduling and Rescheduling with Iterative Repair, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 6, pp. 1588-1596.

$$p(accept) = e^{\frac{quality(s_{curr}) - quality(s_{new})}{T_k}}$$

Equation 1

$$T_{k+1} = \alpha \cdot T_k \quad or \quad T_{k+1} = \frac{T_k}{(1 + \beta \cdot T_k)}$$

Equation 2

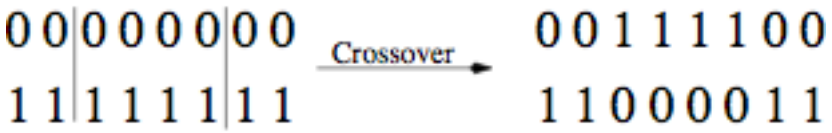

Figure 1: An Example of Two-Point Crossover

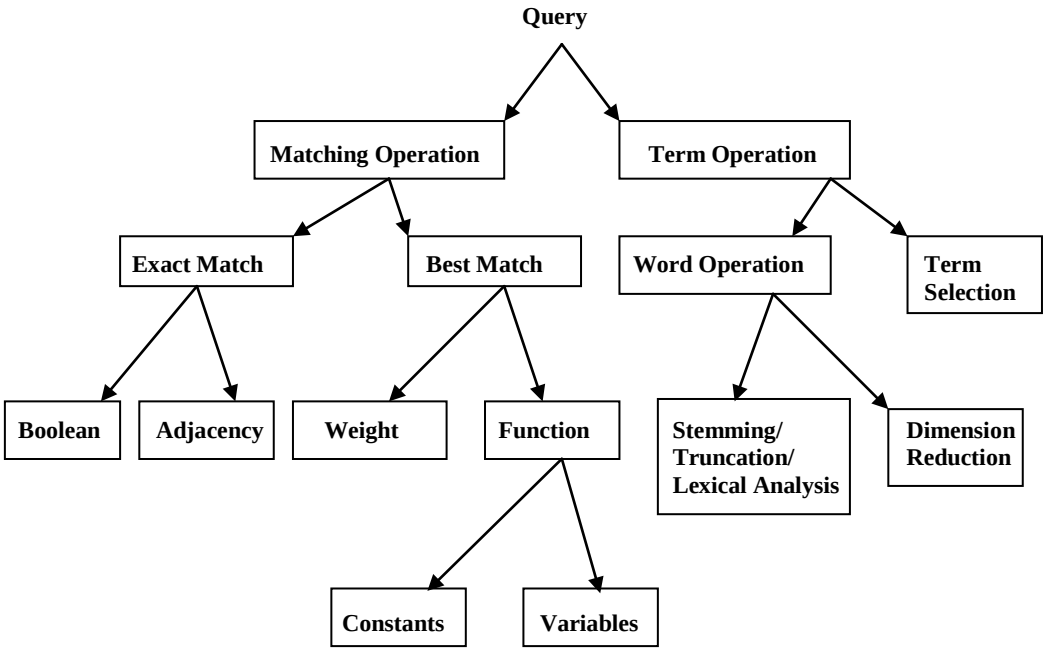

Figure 2: Query based framework for Local Search in IR

$$Function\ U = (A * u^1) + (B * u^2)$$

Equation 3