Motif-GRASP and Motif-ILS: Two New Stochastic Local Search Algorithms for Motif Finding

Mirela Andronescu
University of British Columbia
Computer Science Department
androne s@cs.ubc.ca

Baharak Rastegari
University of British Columbia
Computer Science Department
baharak@cs.ubc.ca

Abstract

Motif finding in biosequences is a very important and well-studied problem. However, the best algorithms to date either give a poor performance on some problem instances, or are computationally impractical. As it turns out that this problem is computationally hard, in this work, we propose *Motif-GRASP*, a GRASP algorithm, and *Motif-ILS*, an ILS algorithm, to solve the motif finding problem. We have tested the two algorithms on data sets generated with the *Motif-Generator* we have implemented, and found that they both algorithms work pretty well, with a better performance for the GRASP algorithm. Motif-GRASP also works well for the CRP real data set, and both algorithms perform poorly for simulated sets from the Challenge Problem [9]. The results presented in this work are promising, and maybe with more improvement, we can get results comparable with state of the art algorithms for this problem.

1 Introduction

A **motif** is a conserved pattern thought to exist in several biosequences, i.e. DNA, RNA or proteins. Examples include: (1) regulatory elements of gene expression; these are relatively short stretches of DNA (5 to 25 nucleotide-long), located in the non-coding sequence surrounding a gene; (2) the helix-turn-helix motif, which represents a large class of sequence-specific DNA binding structures for gene expression [3].

There have been more than one precise definition of the motif finding problem [3, 4]. We define the **problem** of motif finding as follows: Given N biosequences S_i of lengths n_i, $\forall i = 1 : N$ and a number L, find N sequences M_i of length L, one from each sequence S_i, such that they are as similar to each other as possible. M_i will be close variations of a general $motif$, $pattern$ or $signal\ M$.

The following is an example of 10 artificially created DNA sequences of different lengths n_i. They all share a common motif of length 8, separated by spaces from the rest of the sequences.

```
              AT CCGACCAC CTCCATGATGGCTAACACG
              AT CCCTCCGC TACGCACTGGAGCTGAC
            ATAG CCCTCCAC TAGCAATGGGCTA
TAGGCGTGAGGCTTAGG CCACCCGC CGTAGGC
    TAGGCTAGATCGG CCACCCAC GACTAGATGCCTGTAGCTAGTAGC
          TAGGCGA CCGCCCGC GATACGGCGCGCTAGTGCGTATACG
     TAGCGCTCAGCT CCGCCCTC AGTGC
            ATGC CCCTCCAC ATCGTCGCTAGATCGTCTAAC
         ATGCTAGC CCCTCCGC ATGCTGATTAGCTAG
       TAGCTTAAGC CCCTCCCC TAGTCGATGATGTCGATCG
```

The algorithms that are guaranteed to find the optimal motif in N sequences are based on exhaustive enumeration of all possible motifs M [7]. This leads to an exponential time complexity in the length of the motif. Many approximation algorithms and heuristics have been tried as well (see Section 2). We believe that a stochastic search algorithm can help us to find a good answer in reasonable time. We think that constructing a candidate solution with good quality, and then applying local search will result in a high quality answer. Therefore we built a Greedy Randomized Adaptive Search Procedure (GRASP) to solve this prolem. To find out whether the idea of constructive search plays a big role in the quality of the final result, we have also implemented an Iterative Local Search (ILS) method.

In Section 2, we first describe some previous work that has been done for solving this problem, especially the ones from which we used some ideas. Then in Section 3, we describe our model and algorithms in detail. In Section 4, we present the data sets we used to evaluate and test our algorithms, and then in Section 5 we describe the results we observed by running our algorithms on these data sets. In Section 6 we discuss the search space of two problem instances, in Section 7 we give some suggestions for future work, and finally conclude our work in Section 8.

2 Related Work

Several algorithms exist which are built on statistical models, greedy or local search methods. Most of them have tried to solve the basic problem (i.e. the problem we defined in the Introduction), in which the size of the motif is predefined. Few of them extend their work to address features that make the problem harder, such as: (1) the motif does not contain any insertions or deletions (it is one contiguous segment) and (2) every sequence in the set contains the motif exactly once. Some of them suffer from ending in a local optimum rather than finding the globally best motif.

The most relevant methods to our work are Gibbs Sampling [3], Random Projection [4] and SP-STAR algorithm [9], which we describe in the following subsections.

2.1 Gibbs Sampling

Lawrence et al. [3] used a Gibbs Sampling algorithm as a stochastic analog of expectation maximization (EM) methods that were previously used. The Gibbs Sampler proved to yield a more robust and faster optimization procedure than the EM methods.

The problem, as Lawrence et al. define it, is to locate and describe a pattern thought to be contained within a set of N biosequences. Their algorithm provides a procedure for automatic determination of the motif length by running the basic algorithm for a range of lengths and then choosing the best length. It is also able to find multiple motifs and insertions/deletions in patterns.

The **basic algorithm** looks for motifs of a specified length L in N sequences S_i. The following data structures are maintained:

1. The *pattern description* is a matrix of q_{ij}, representing the probabilistic model of frequencies of residue $j \in [1, A]$ for each position $i \in [1, L]$, where A is the size of residue alphabet (i.e. 4 for DNA/RNA, 20 for proteins). The *background frequencies* $p_j, \forall j \in [1, A]$ are defined analogously to pattern description and represents a probabilistic model of the residues that occur in sites not described by the pattern.

2. The *alignment set* is a set of positions $a_i, \forall i \in [1, N]$ for the common pattern within the N sequences.

The objective is to identify the most probable common pattern. Initially, random starting positions are chosen. Then the Gibbs Sampler iterates through the following two steps, until it converges:

1. One of the N sequences, z, is chosen uniformly at random or in specified order, and the values of q_{ij} and p_j are calculated according to all sequences except z. q_{ij}'s are calculated using the formula in Eq. 1, where c_{ij} is the count of residue j in position i. b_j is a "residue-dependent" pseudocount and B is the sum of the b_j. They have found that using $b_j = B.r_j$, where r_j is a frequency of residue j in the whole sequences, is effective. They have also found that the value $\sqrt{N}$ works well for B. Each p_j is calculated simply by finding the frequency of alphabet j in the whole set except sequence z and also except the positions belong to motif.

2. Every possible segment x of length L in sequence z is considered as a candidate motif. Then $A_x = Q_x/P_x$ is calculated, where Q_x is the probability of generating the subsequence x from the current motif, and P_x is the probability of generating the subsequence x from the background. Thus, A_x gives the probability that the pattern in sequence z starts at the starting position of x. One of them is selected randomly according to their values, and its position becomes the new start position for sequence z (a_z).

As mentioned above, A_x is the probability that the pattern starts at the position x in sequence z. The goal is to maximize the product of these ratios (for starting points in all sequences). Equivalently, one may maximize the sum of the logarithms of these ratios, which gives the function F in

Eq. 2.

$$q_{ij} = \frac{c_{ij} + b_j}{N - 1 + B}(1) \qquad F = \sum_{i=1}^{L} \sum_{j=1}^{A} c_{ij} log \frac{q_{ij}}{p_j}(2)$$

2.2 Random Projection

Buhler et al. [4] used Random Projection for solving the motif finding problem. They define the problem as follows: Given *N* sequences, of equal size n, look for a motif M, of length L, such that there is one occurence of M in each sequence, that is corrupted by exactly d point substitutions in random positions.

The Random Projection algorithm projects L-dimensional Hamming distance onto k dimensional subspace. Briefly, it works as follows: (1) choose a projection by selecting k positions uniformly at random; (2) for each L-tuple in input sequences, hash into bucket based on letters at k selected positions; (3) recover motif from bucket containing multiple L-tuples.

The bucket with the hash value $h(M)$ is called the *planted bucket*. It is expected that the planted bucket contains more motif instances than random sequences. They also set a low bucket size threshold s, and at the end of the algorithm they select buckets which contain more than s L-tuples, with the assumption that the motif should belong to one of them. At this point, the algorithm found some candidates for k positions of the motif M. The information in other $L - k$ positions are used as starting points for local refinement schema, e.g. EM or Gibbs sampler to find out the motif M. Thus, this algorithm eliminates some parts of the search space, which are less likely to contain optimal solutions, and produce a smaller search space for the EM or Gibbs Sampler to work on.

2.3 WINNOWER

Pevzner et al. [9] introduced the WINNOWER algorithm, in which they used the same problem definition as in the Random Projection algorithm.

The algorithm starts by constructing a graph $G(S, L, d)$. The vertices of this graph is the set of all subsequences of length L. There is an edge between 2 vertices if the Hamming distance between them is less than or equal to $2d$ and the vertices are subsequences from two different sequences. The problem is then reduced to finding large cliques in this graph. Finding large cliques in a graph is an NP-complete problem, however G is a multi-partite graph.

In their notation, a vertex u is a neighbour of a clique C if adding u to the list of vertices in C results in another clique in G. A clique is extendable if it has at least one neighbour in every part of the multi-partite graph G. An edge is called to be spurious if it does not belong to any extendable clique of size k. Edges which belong to a minimum number of extendable cliques are in maximal t-cliques, too. Their algorithm is based on the idea of removing edges which do not belong to the minimum number of extendable cliques. WINNOWER is an iterative algorithm that proceeds on G until it converges to a small collection of extendable cliques. This collection can be explored for the motif.

2.4 SP-STAR

Due to the substantial computational resources (time and memory) that WINNOWER needs, Pevzner et al. introduced another algorithm named SP-STAR [9].

They first introduced sum-of-pairs scoring as evaluation function. If W is a candidate signal and the best instances of W in sequences are $W_1, W_2, ...W_N$, then the sum-of-pairs scoring function will be $D(W,S) = \sum_{1 \leq i < j \leq N} d(W_i, W_j)$, where $d(W_i, W_j)$ denotes the number of different pairs in W_i and W_j. SP-STAR first finds a pattern W which minimizes $D(W,S)$ (for all W in S). Then it tries to improve the initial found signal by using a local improvement strategy.

They also extend their algorithm to work for unknown length signal. For this purpose they introduced another evaluation function, which is based on scoring the similiarity between each two instances. The similarity score for W is defined as:

$$S(W) = \sum_{1 \leq i < j \leq t} S(W_i, W_j) \binom{N}{2} (3),$$

where $S(W_i, W_j)$ is the similarity between W_i and W_j, which is defined as the sum of premiums for matches and penalties for mismatches (they found that using +2 as the match premium and -1 as the mismatch penalty works well). If the $S(W)$ score is positive, then the positions are significant.

Starting from some l_{max} (assume that the length of motif is in the range $[l_{max}, l_{min}]$, SP-STAR changes the scoring approach by scoring the best substring of length between l_{min} and l_{max}, which maximize the sum-of-column score. It then performs local search, in which the position of the best-scoring substrings are considered for finding the best instances of the majority in each string.

3 Our Algorithms

Our problem, as described in Section 1, is to find a set of $patterns$ (or $words$). This is equivalent to finding the starting positions a_i of these words in each sequence S_i. We consider all words M_i of equal length. Our algorithm will assume the length L of the motif is given. Once a_i and the length L are known, one can easily infer the words M_i.

Our two stochastic search algorithms are defined by the following components:

- a *candidate solution* is a set $\{a_1, a_2 \ldots a_N\}$, where $a_i \in [1, n_i - L + 1]$, $\forall i \in [1, N]$;
- all candidate solutions, made of all possible combinations of a_i assigments constitute the *search space*;
- two candidate solutions are *neighbours* if they have some a_i's in common. Thus, two candidate solutions are *k-exchange* neighbours if they differ in exactly *k* starting positions. In

our local search procedures, we use *one-exchange neighbourhood*, i.e. they differ in exactly one starting position, regardless if the new position is close to the old one or not.

- for the *evaluation function*, we considered two alternatives: (1) the F function that Lawrence et al. [3] used in the Gibbs Sampling algorithm (Eq. 2); (2) the similarity score function S that Pevzner et al. [9] used in the SPSTAR algorithm (Eq. 3). In the remainder of this paper, we refer to the former function as *Lawrence* function, and to the latter function as the *Similarity* function.

We explored the benefit in using a constructive search, followed by a local search procedure. This leads us to a GRASP algorithm [1, 2]. To experimentally test and observe the benefit of the constructive search, we have also built an ILS algorithm, which is not a constructive method, and compared it with GRASP.

The details of the GRASP algorithm and the ILS algorithm are described in the following two subsections. Because many of the motif finding problems are hard, a solution that finds an approximate result (e.g. finds $N-1$ correct starting positions and one incorrect position) might be good enough. For this reason, we need to evaluate the quality of the results. In this respect, a solution of quality 1 would mean a perfect solution, a solution of quality 0 would be completely wrong, and a solution of quality 0.9 would be pretty good.

We came up with a function to describe the solution quality, as follows: Let $r_1, r_2, \ldots r_N$ be the real starting positions of the motifs in the N sequences. We first define the SQ function to measure the quality of every single starting position separately: for each sequence i, $SQ(i) = 1 - |a_i - r_i|/L$ if $|a_i - r_i| < L$, and 0 otherwise. This means that for each sequence i, if the predicted starting point is near to the real one, the solution quality of that sequence is high. Then we define the Q function that measures the quality of the whole solution:

$$Q = \frac{\sum_{1 \leq i \leq N} SQ(i)}{N} \quad (4)$$

3.1 Motif-GRASP

We have implemented a constructive search algorithm inspired from GRASP, which we call *Motif-GRASP*. Figure 1 shows a pseudocode of our algorithm. The main idea is to construct a high quality candidate solution as a starting point of the local search.

In the *construct* procedure, we first choose a random starting point in the first sequence and update the data structures accordingly. Then we take the second sequence and calculate the weights A_x for all possible starting points x of the motif. The data structures that are updated and the way we calculate the weights A_x are dependant on the evaluation function F we use:

- F = *Lawrence* function: In this case we update the p and q matrices as our data structures. We also calculate the weights according to these data: $A_x = Q_x/P_x$.
- F = *Similarity* function: In this case, A_x is the similarity score. This means that we just

procedure *Motif-GRASP*
input: problem instance π (i.e. N seqs. S_i of length n_i, motif length L), objective function F
output: solution $\hat{s}$ (i.e. N positions a_i where motifs start)
s := 0;
$\hat{s}$:= s;
while not *terminate(π, s)* **do**
s := *construct* (π');
s' := *localSearch* (π', s);
if ($F(s') > F(\hat{s})$)
$\hat{s}$:= s';
end if
end while
return solution $\hat{s}$;
end procedure *Motif-GRASP.*

Figure 1: Pseudocode for Motif-GRASP; details are discussed in the text.

need to calculate the similarity score for this set of starting positions (where x is the starting position in the second sequence)

At each construction step, we take the k best A_x and randomly choose one of these x's as the starting point for the second sequence. Then we update the data structures according to the first two sequences and continue this procedure until starting points for all sequences are defined.

In the *localSearch* procedure, starting from the candidate solution which has been found in the construct part, we try to improve the value of the F function until we reach a local maximum. We use a one-exchange neighbourhood, i.e. the new candidate solution differs by one starting point from the previous candidate solution. We tried two different iterative improvement algorithms:

- *first-syst* is a first improvement method: every time a neighbour has a higher F value, it will become our new current candidate solution. We use a systematic procedure for searching in the neighbourhood region. In the first step of the local search, we start from the first sequence and let the starting point get different values from 1 to $n_i - L$. Once we found a higher value for F, we move to that candidate solution. If we didn't find any higher value for F, we will try the second sequence and continue until we reach a higher value for F. If, for the next step of the local search, we start again from the first sequence, then it may happen that the neighbour which has a good value of F, but differs in the starting position of the last sequences, takes a low chance to be observed. To solve this problem, for the next step of the local search we will not start from the first sequence again. Instead, assuming that the first improvement occured in sequence j in the previous step, we start searching the neighbours from sequence $j + 1$ in the new step.
- *pseudo-best* improvement: this is a hybrid of best and first improvement. It is the same as the First Improvement we described before, but every time we check neighbours which are all different in the starting point of the same sequence, if there are neighbours with higher F, we take the best one, and this will become our new candidate solution.

When the local search finishes, we check if the new local maximum is greater than the best so-

lution found so far. In this case, the best solution will be updated. These 3 steps, i.e. *construct, localSearch* and deciding the best solution, will be repeated for a given number of times, *Num-Restarts*(e.g. *Num-Restarts = 30*). This is the termination criterion which will make the routine end. The best solution found will be reported as the result.

There are four parameters for the *Motif-GRASP* algorithm, that we will tune in Section 5:

- *Num-Restarts*: The number of iterations for the 3 steps (*construct*, *localSearch* and updating the solution). A greater value for this parameter allows visiting more regions in the search space, but at a longer time cost. Finding a good value for *Num-Restarts* is essential and useful.
- *RCL* parameter is the number k we described above. A smaller value means that we are more greedy and select one of the few best candidate solutions in each step. It gives us more intensification. A greater value helps us to check more regions in the search space and increases diversification. A balance between intensification and diversification is required for good performance.
- The *Evaluation-Function* parameter: The possible values for this parameter are the two functions described above: *Lawrence* and *Similarity*.
- The *LS-Function* parameter: The possible values for this function will be *first-syst*, *pseudo-best* and *none*. The first were described before, and the last one means no local search after the construction step.

3.2 Motif-ILS

We have also implemented a non-constructive search algorithm, i.e. Iterated Local Search, that we call *Motif-ILS*. Figure 2 shows a pseudocode of our implementation. The *initialization* step consists of randomly picking a position a_i in each of the N input sequences S_i. This will give a candidate solution s. Then, starting with s, we search for a local maximum in the *localSearch* routine. Each new step in the localSearch procedure represents a one-exchange step: we pick a sequence j uniformly at random, and change the value of a_j by other valid value (i.e. in the range $1, n_j - L$), uniformly at random. These one-exchange steps are repeated until a local maximum is reached. For time efficiency reasons, in this localSearch procedure, we do not check all the possible neighbours (note that there are $\sum n_i - N$ neighbours). Instead, we consider we reached the local maximum when no improving step appears after a fixed number of steps (e.g. 30).

Then, a perturbative local search, followed by an acceptance criterion, iterates for a fixed number of steps (e.g. 1000). A new candidate solution $s^{'}$ is obtained by applying a *perturbation* to the previously found candidate solution s. Then, starting with $s^{'}$, the *localSearch* is performed, which will yield the candidate solution $s^{''}$.

Up to an input parameter, the perturbation procedure consists of a two-exchange neighbour, a three-exchange neighbour or a four-exchange neighbour. In this case, a *k-exchange neighbour* means that *k* different sequences are chosen unformly at random, and then their starting positions

```
procedure Motif-ILS
    input: problem instance π (i.e. N seqs S_i of length n_i, motif length L), objective function F
    output: solution ŝ (i.e. N positions a_i where motif starts)
    s := initialize(π);
    s := localSearch (π, s);
    ŝ := s;
    while not terminate (π, s) do
        s' := perturb (π, s);
        s'' := localSearch (π, s');
        if (F(s'') > F(ŝ))
            ŝ := s'';
        end if
        s := accept (π, s, s'');
    end while
    return solution ŝ;
end procedure Motif-ILS.
```

Figure 2: Pseudocode for the Motif Finding Iterated Local Search algorithm; details are discussed in the text.

are changed with other values, chosen uniformly at random from the possible values of each sequence.

The same as in the case of *Motif-GRASP*, we have two alternatives for the evaluation function: *Lawrence* function and *Similarity* function.

The *acceptance* criterion we use is as follows: with probability p (e.g. 0.9), s becomes s'' if the evaluation function value F of s'' is greater than the evaluation function value of s. With probability $1-p$ (e.g. 0.1), we continue with s'' (i.e. s is s''), regardless of the quality of s''. Thus, $\hat{s}$ will contain the best solution found, and will be returned at the end of the procedure, after the termination criterion (i.e. a specified number of steps) is satisfied.

In Section 5, we tune four parameters for *Motif-ILS*:

- The *Perturbation* parameter: the possible values are *2-ex*, *3-ex* and *4-ex*. *4-ex* means more diversification than *2-ex*.
- The *Accept-Probability* parameter is important to decide from which of the two local optimum s and s'' the search will continue.
- The *Evaluation-Function* parameter is identical to the corresponding parameter in *Motif-GRASP*. The two values are *Lawrence* function and *Similarity* function.
- The *Cutoff* parameter decides after how many iterations to terminate. A higher value will give the algorithm a higher chance to get a better result.

4 Data Sets

Finding data sets on which we can test our algorithms is not easy. To our best knowledge, there are very few available real sets. Even those that exist are hard to find.

Pevzner et al. [9] have created the *Challenge Problem*. This is a generator that creates sets of sequences containing a randomly planted motif of length L, and with exactly d mismatches between any two motifs. They showed that the Gibbs Sampling algorithm by Lawrence et al. [3] performs very poorly on Challege sets containing sequences longer than 300 residues. Unfortunately, the *Challenge Problem* is not publicly available. We obtained 20 sets from Sohrab Shah, but all these have sequences of length 600. These sets are known to be hard.

Given our need to have data sets of different characteristics, and given the lack of such data sets from other sources, we created a data set generator, that we call *Motif-Generator*. The following subsections describe the generator and the data sets we have created or obtained, and used in our testing.

To avoid confusion, in the remainder of this paper, different notations will be used for a set of sequences and for a set (or group) of data sets. For the former one, we conventionally use *Set* or *set*, and for the latter one we use *SET*.

4.1 *Motif-Generator*

The *Motif-Generator* we have created takes the following parameters as input: the motif length L, the number of sets N_S to create, the number of sequences N in each set, the minimum sequence length L_{min}, the maximum sequence length L_{max}, the sequence type T (i.e. DNA or PROTEIN) and the probability threshold p. Thus, the output is a SET of N_S sets, each containing N sequences of length between L_{min} and L_{max} and of type T. They have a motif of length L, planted at random positions. The motif base probability have a lower bound p. More details follow:

1. Given some threshold probability p (higher than 0.5), we first create the matrix of probabilities for the residues within the motif. For each position i in the motif, the probability of the residue which will appear most frequently will be generated uniformly at random in the interval $[p, 1]$. For the other residues, new different probabilities will be generated in the remaining interval. The probability of the last residue is the remaining value, such that they all sum up to 1. The following is an example of a probability matrix, where $L = 8$, $p = 0.70$ and $T =$DNA:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
A	**0.71**	0.09	0.20	0.22	0.05	**0.76**	0.10	0.00
C	0.11	0.05	0.00	**0.78**	0.04	0.19	0.06	0.02
G	0.06	0.04	**0.74**	0.00	**0.82**	0.03	**0.81**	**0.98**
T	0.12	**0.82**	0.06	0.00	0.09	0.02	0.03	0.00

   As the bolded probabilities show, the most probable motif is *ATGCGAGG*. The generated

sequences will contain slight variations of this motif.

2. We generate N sequences of length randomly chosen between a maximum and a minimum value (L_{min}, L_{max}). Then we choose a position to be the starting position of the motif in each sequence, uniformly at random. The residues within the motif will be generated according to the probability matrix described above, the residues outside the motif will be generated using a uniform distribution. We repeat this procedure N_S times, thus generating N_S sets, each in a different file, in FASTA format[11].

4.2 Artificially Created Data Sets

We have created five groups of data sets, each of them having a specific purpose:

- For tuning the parameters of *Motif-GRASP* and *Motif-ILS*, we created 50 sets, each of them containing 10 DNA sequences of length 50. The motif size L was set to 15, and the probability threshold to 70. In the remainder of this paper, we call this SET *Art-Tune-SET*. We give one set of these sets in the Appendix 1.
- For studying the influence of the number of sequences, n, on the performance of the algorithms, we created 10 DNA sets for each of the different n (n = 15, 20, 25, 30). In all of them $L = 15$, $p = 70$ and length of the sequences is 200. This SET will be called *Art-Num-Seq-SET*.
- To see the influence of sequence length on the performance, we created 10 DNA sets for L = 100, 200 ... 900. They contain 20 sequences each, L is 15 and p is equal to 70. Let the name of this SET be *Art-Seq-Len-SET*.
- We also want to see the influence of motif size on the performance of the algorithms. Therefore we created 10 DNA sets for each of L = 5, 10, 15, 20, 25. Each set contains 20 sequences of length 200, and p is equal to 70. We will refer to this SET as *Art-Motif-SET*.
- For observing the influence of the different value for p, we created 10 DNA sets for each of the different p (p = 50, 60, 70, 80, 90). Each set contains 20 sequences of length 200, each of them containing a motif of length 15. This SET will be further called *Art-Prob-SET*.

4.3 Simulated Data - Challenge Problem

We have obtained 20 sets generated by the *Challenge Problem* [9]. We have used 10 of them and we call this set *Challenge-SET*.

4.4 Real Data Set

We used the CRP data set as real data [11] (which is found in nature): *CRP-Set*. It consists of 18 DNA sequences of length 105, each containing 1 or 2 motifs of length 22. The sequences are shown in the appendix with the motif area separated from the rest of the sequences by spaces.

5 Evaluation

In this section, we perform empirical tests to evaluate the performance of the two algorithms described in the previous sections: Motif-GRASP and Motif-ILS. We start by tuning four parameters for each of Motif-GRASP and Motif-ILS on *Art-Tune-SET*. This step will find the optimal values for these parameters. Using these values, we further compare Motif-GRASP versus Motif-ILS. The winner, Motif-GRASP, will be tested on harder artificial sets. Then, some discussion on the *Challenge-SET* data is provided. We close this section by testing Motif-GRASP on the real data set *CRP-Set*.

We note that we have obtained *GIBBS*, the implementation of the Gibbs Sampling algorithm described in the Related Work section. Unfortunately, we were not able to test GIBBS and compare it with our algorithms because we did not understand exactly the input parameters it gets. We tried to test it on the *CRP-Set*, but, although several papers report *GIBBS* performs well on this set, the results it gave us were very poor.

All tests in this section measured the algorithm performance in terms of solution quality, unless otherwise stated. The solution quality was calculated as described in Section 3. For each input set, we ran the algorithm 10 times, and than took the average of the 10 solution qualities thus obtained.

5.1 Tuning Motif-GRASP Parameters

In this subsection, we tune the four parameters of Motif-GRASP: *Num-Restarts, RCL, Evaluation-Function* and *LS-Function*. We tested them on the 50 sets of *Art-Tune-SET*. Evaluating performance on many sets (in this case 50) definitely allow us to make more accurate tuning than evaluating on very few sets. However, this is far from being a thorough test, which would have required tests on sets with a variety of different characteristics.

For all the tests in this section, we varied one of the 4 parameters, while keeping the others fixed. We guessed some values for the parameters in our preliminary tests, and fixed them: *Num-Restarts = 10, RCL = 5, Evaluation-Function = Lawrence, LS-Function = first-syst*.

Testing the *Num-Restarts* Parameter

Intuitively, the higher the *Num-Restarts* is, the better the solution quality will be. But this comes at a cost of a greater time spent by the algorithm. Figure 3 shows the solution quality for tests on four different values for *Num-Restarts*: 5, 7, 9 and 11. The top-left figure and the top-right figure show 7 is better than 5, and 9 is better than 7, respectively. The bottom-left figure shows that the values 9 and 11 lead to roughly the same performance. Finally, the bottom-right figure shows clearly the difference in performance between values 5 and 11. Our conclusion is that after some point, the likelihood that an improvement will happen becomes very small. The results show the optimal value for the *Num-Restarts* parameter is around 10.

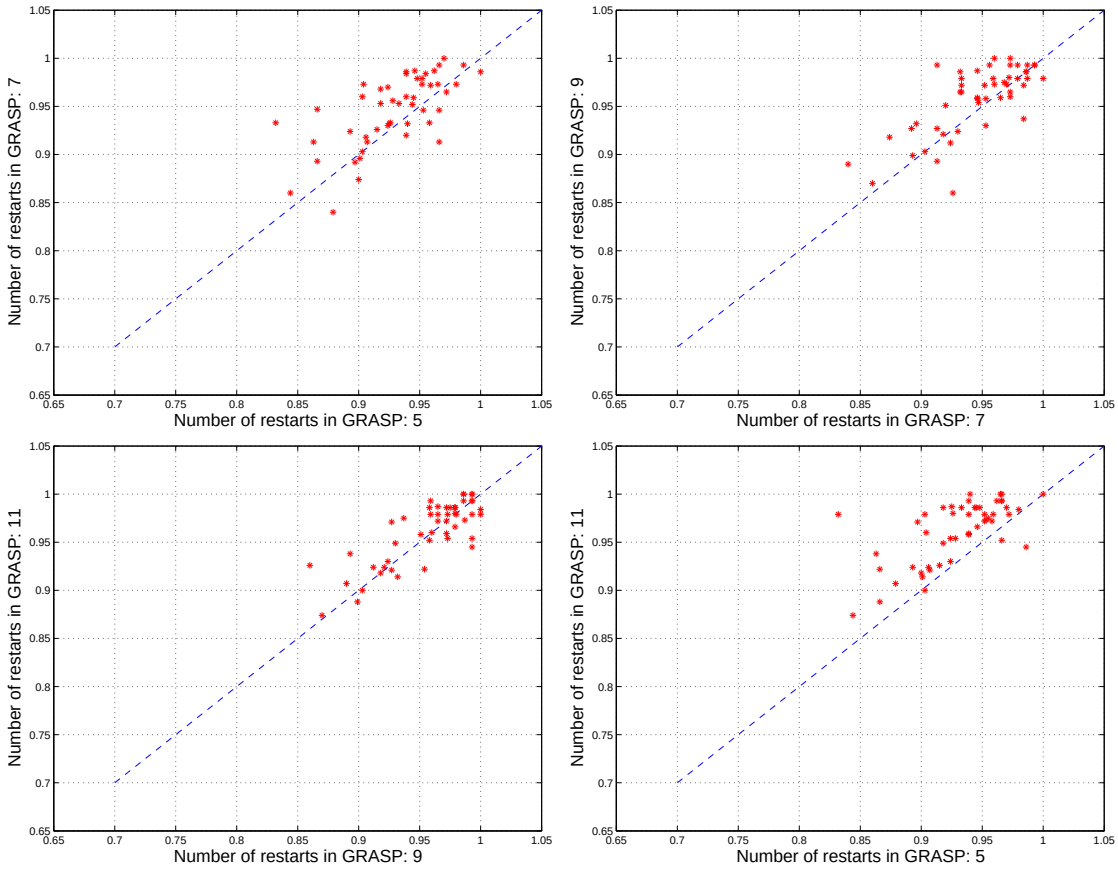

Figure 3: Comparison of performance in terms of solution quality for different values of the *Num-Restarts* parameter: 5, 7, 9 and 11. We consider the optimal value is 10.

Testing the *RCL* Parameter

The next parameter to test is the *RCL* parameter. Figure 4 shows tests for different values of *RCL*: 1, 3, 5, 7, 9, 11 and 13. As expected, the results show that if *RCL* is too low, the performance is poor, since the algorithm is too greedy. If *RCL* is too high, the algorithm is too randomized. A trade-off between greediness and randomization seems to be the best solution.

The top-left figure shows a value of 3 is clearly better than a value of 1. Note the performance values for *RCL=1* are in a broader range (i.e. $[0.78-0.99]$) than for the case *RCL=3* ($[0.85-1.00]$). A good way to visualize this is to note that the points in the top-left triangle are much more spread and far from the diagonal than the points in the bottom-right triangle.

The next figures show that the higher *RCL* is, the performance decreases gradually. We think the optimal value of *RCL* is around 3-5, and chose 5 to work with.

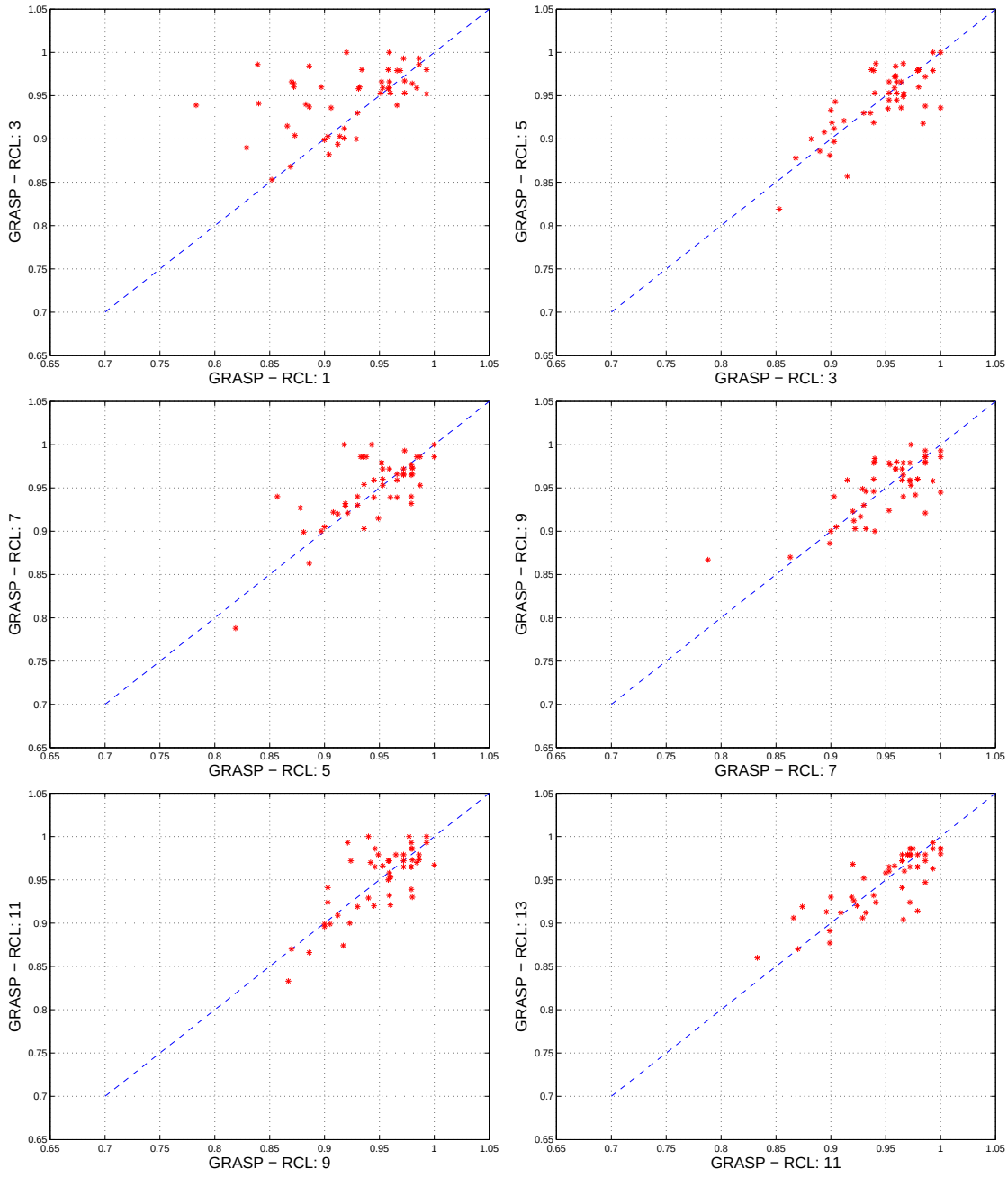

Figure 4: Comparison of performance in terms of solution quality for different values of the *RCL* parameter: 1, 3, 5, 7, 9 and 11. We consider the optimal value is 5.

Testing the *Evaluation-Function* Parameter

Next, we tested which of *Lawrence* and *Similarity* evaluation functions gives more accurate results. Figure 5 shows that the *Lawrence* function gives slightly better performance.

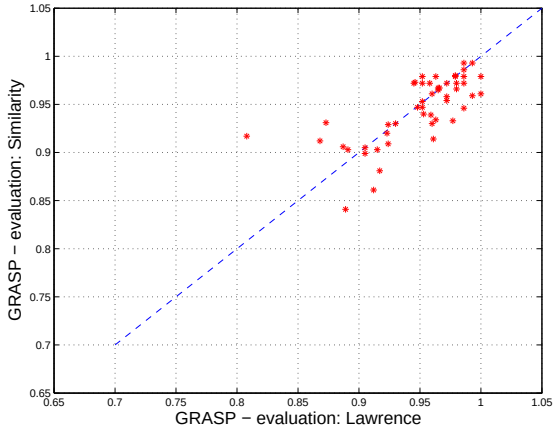

Figure 5: Comparison of performance in terms of solution quality for two different evaluation functions: *Lawrence* and *Similarity*. The former function is better than the latter.

Testing the *LS-Function* Parameter

The left plot in Figure 6 shows Motif-GRASP's performance for *no* local-search versus *pseudo-best* local search (red-stars cloud), and for *no* local-search versus *first-syst* local search (blue-crosses cloud). The plot clearly shows that the *first-syst* local search function gives a much better performance than the other options. The right graph shows the same values in a different way, where the green line corresponds to *no* local search, the blue line - to *pseudo-best* local search, and the red line - *first-syst* local search.

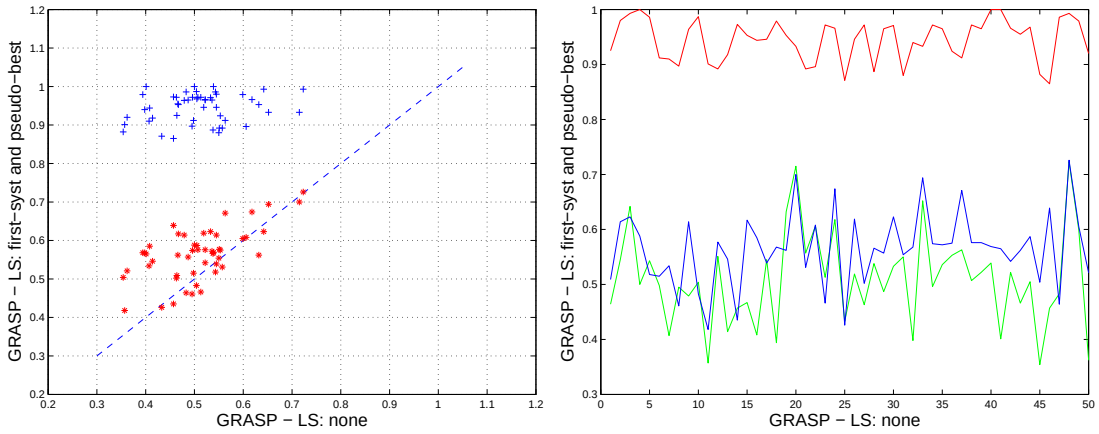

Figure 6: Comparison of performance in terms of solution quality for two different local search functions: *first-syst* and *pseudo-best*, and *no* local search. The *first-syst* function is definitely much better than the other one or than *no* local search function.

5.2 Tuning Motif-ILS Parameters

This section tests the four parameters of Motif-ILS: *Perturbation*, *Accept-Probability*, *Evaluation-Function* and *Cutoff*. Similarly to Motif-GRASP, we tested them on the 50 sets of *Art-Tune-SET*.

Interestingly, the points in the Motif-ILS plots are much more spread than the points in the Motif-GRASP plots, the range of the solution quality values being larger. This means that Motif-ILS performs less robustly than Motif-GRASP.

For all the tests in this section, we varied one of the 4 parameters, while keeping the others fixed. The fixed values, determined by preliminary tests, are: *Perturbation = 2-exchange*, *Accept-Probability = 80*, *Evaluation-Function = Lawrence*, *Cutoff = 100*.

Testing the *Perturbation* Parameter

For the perturbation function, we tried different k-exchange neighbourhoods, with k = 2, 3 and 4. Figure 7 shows that there is not much difference between 2-exchange, 3-exchange and 4-exchange. It seems that 3-exchange is slightly better than the other two. A higher k means a bigger jump in the search space. In the section *Search Space Analysis* we explain why these results are so close to each other.

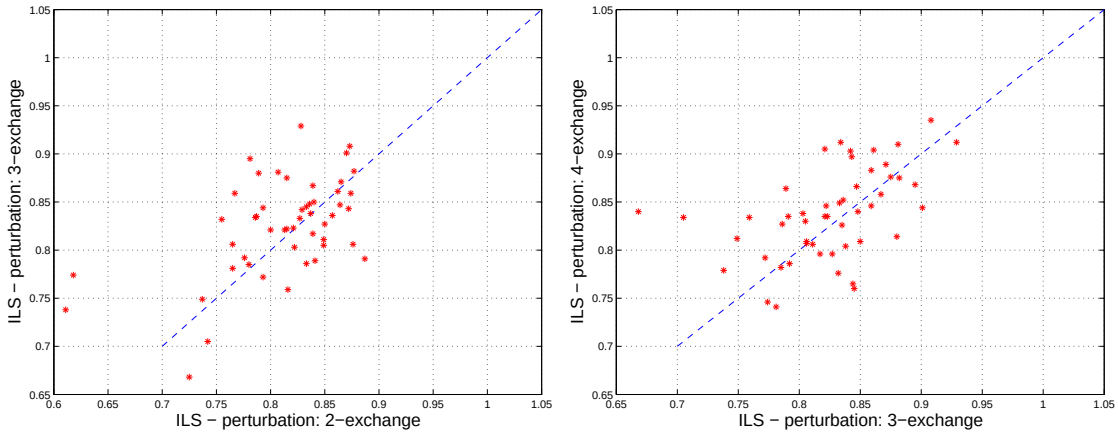

Figure 7: Comparison of performance in terms of solution quality for different perturbation functions: *2-exchange*, *3-exchange* and *4-exchange*. It appears that *3-exchange* is slightly better than *2-exchange* and *4-exchange*.

Testing the *Accept-Probability* Parameter

The values of the *Accept-Probability* parameter we tested were: 75%, 80%, 85% and 90%. The same as for the previous parameter, the results are very close to each other (Figure 8). For further tests, we chose the value 80%.

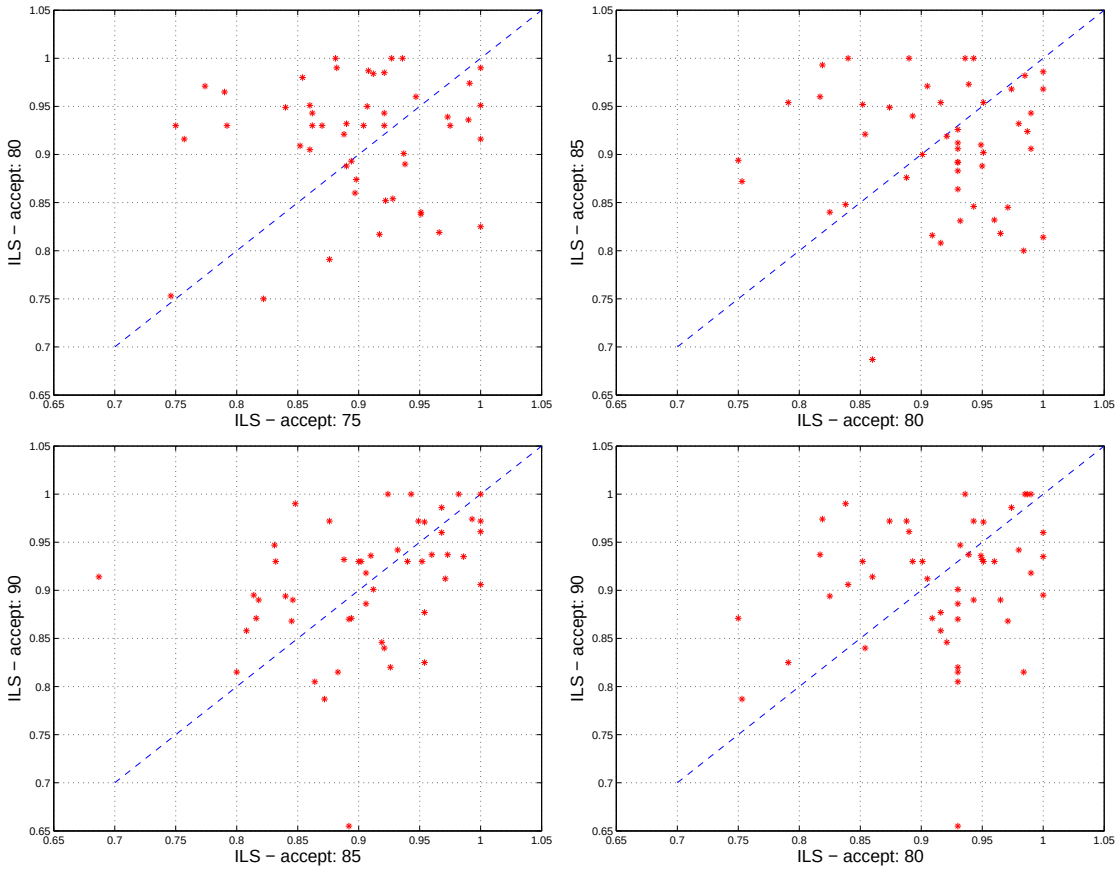

Figure 8: Comparison of performance in terms of solution quality for different values of the *Accept-Probability* parameter: 75%, 80%, 85% and 90%. It is not clear which if them is better, the results being very close to each other. The value we chose to work with is 80%.

Testing the *Evaluation-Function* Parameter

Similarly to Motif-GRASP, we tested which of the two evaluation functions: *Lawrence* and *Similarity* give better results. Similarly to the case of Motif-GRASP, the *Lawrence* function is better (Figure 9).

Testing the *Cutoff* Parameter

Definitely a higher *Cutoff* parameter will yield better solution quality, but at the cost of additional time. Figure 10 shows Motif-ILS performance when the *Cutoff* parameter is set to 100, 300 and 1000, respectively. We decided to choose the value 500 for further tests.

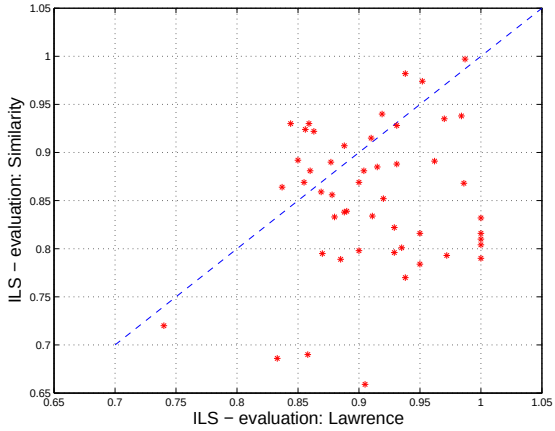

Figure 9: Comparison of performance in terms of solution quality for two different evaluation functions: *Lawrence* and *Similarity*. The former function is better than the latter.

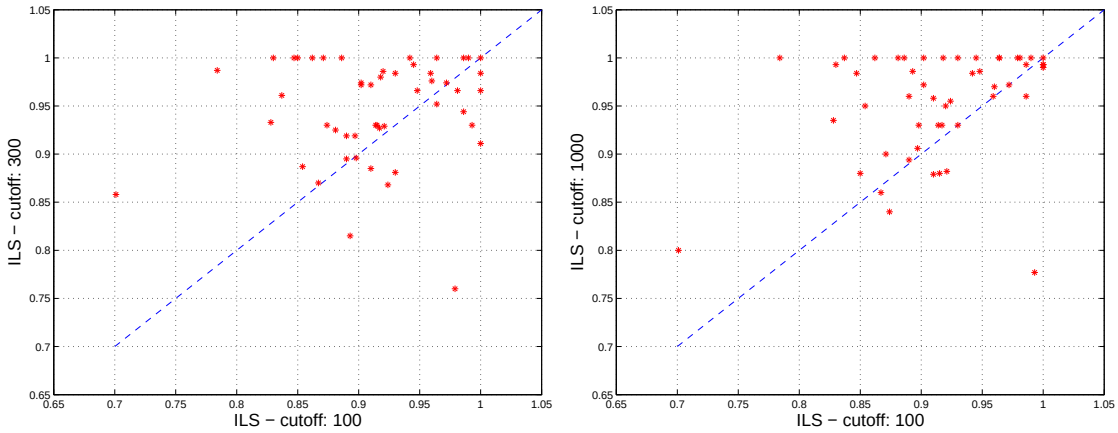

Figure 10: Comparison of performance in terms of solution quality for different *Cutoff* values: 100, 300 and 1000. The highest value we tested on (i.e. 1000) gives the best performance. However, because of the additional time cost, we chose the value 500.

5.3 Comparison between Motif-GRASP and Motif-ILS

After testing the specific parameters of Motif-GRASP and Motif-ILS separately, we now compare the two algorithms with each other. For all tests, we used the optimal values found in the previous two subsections. First, we compared them on the same set: *Art-Tune-SET*, and we measured the performance in terms of the average solution quality. Second, we compared Run Time Distributions (RTDs) [1] for four sets from *Art-Tune-SET*.

Motif-GRASP vs. Motif-ILS on *Art-Tune-SET*

Figure 11 shows the results in terms of the average solution quality on the *Art-Tune-SET*. As we expected, the GRASP algorithm performed better than ILS. It shows that the construct routine of GRASP has a considerable role in finding good quality solutions, and suggests that construction search methods are useful for this problem. Also, the range of solution quality is larger for *Motif-ILS* ($[0.78 - 1.00]$) than for *Motif-GRASP* ($[0.87 - 1.00]$). However, we notice that for this simple data set, Motif-ILS performs pretty well, too.

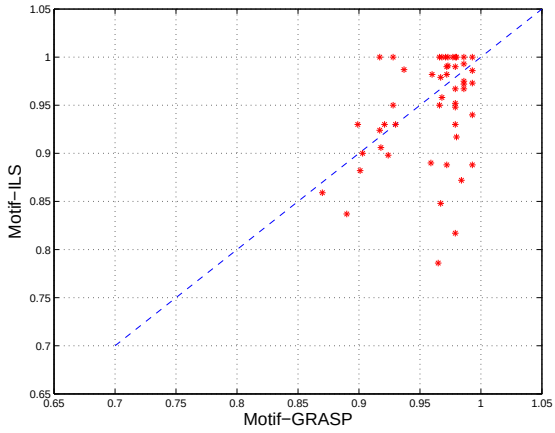

Figure 11: Comparison of performance in terms of solution quality for Motif-GRASP vs Motif-ILS on the *Art-Tune-SET* data set. Motif-GRASP performs slightly better, and the range of solution quality is narrower.

RTDs of Motif-GRASP and Motif-ILS

Next, we compared the Run Time Distributions of Motif-GRASP and Motif-ILS on four sets from the *Art-Tune-SET*. We ran the two algorithms 1000 times, using the optimal parameters found in the *Tuning Parameters* sections. Then, we took the instances that found the exact solution (i.e. had solution quality 1.00) and drew the graphs in Figure 12. The platform we used was Pentium III dual processor at 1GHz, running Linux 2.4.18.

Note that, as this problem is a combinatorial optimization problem, in each run, the programs continue searching until the termination criterion is satisfied. We measured the CPU time spent until the best solution was found. Therefore, if more time was spent to search for a better solution, but no better solution was found, this time was not considered.

Figure 12 shows that Motif-GRASP is faster than Motif-ILS. Moreover, out of the 1000 runs, Motif-GRASP was successful in 553, 742, 793 and 756 cases, respectively. Motif-ILS was successful in only 186, 237, 232 and 379 cases, respectively. This clearly shows Motif-GRASP's superiority when compared to Motif-ILS. The RTDs in Figure 12 contain the successful runs only. Note that a more thorough evaluation would be to show incomplete RTDs, which do not go to 100% probability of solve, but to the percentage of successful runs.

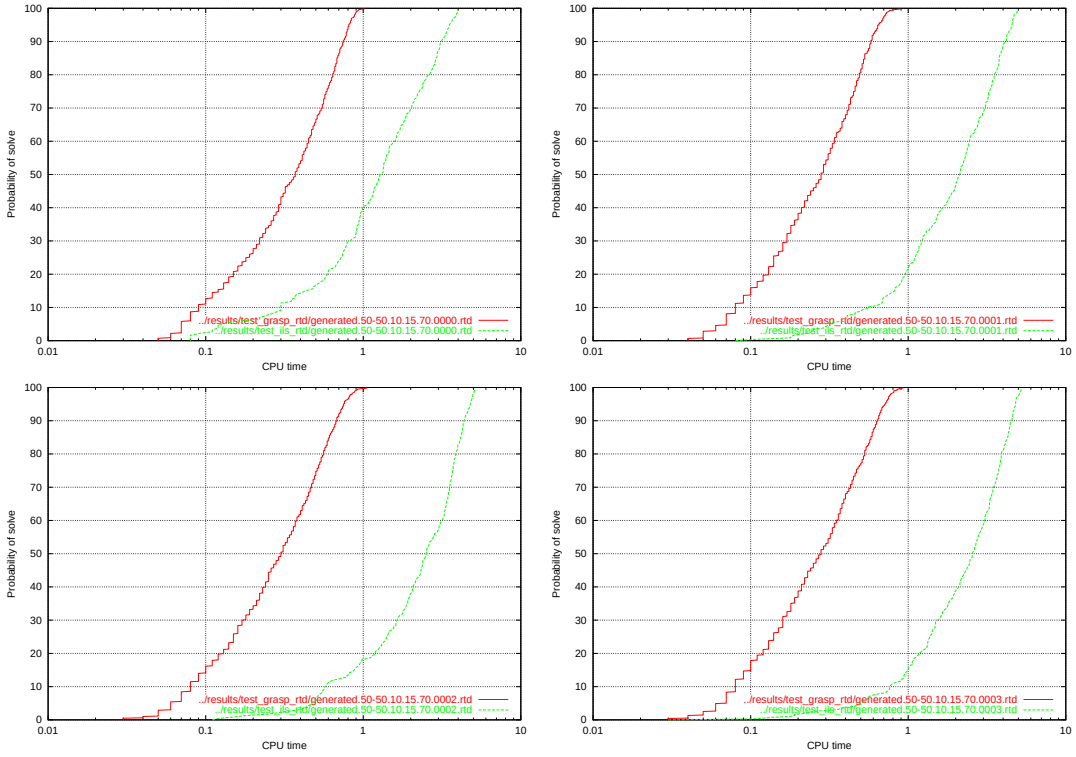

Figure 12: RTDs of Motif-GRASP vs. Motif-ILS on four sets from *Art-Tune-SET*. Motif-GRASP is clearly superior to Motif-ILS.

The representation we used also explains why the RTD top tails do not have the typical S shape (too few solutions found, especially in the case of Motif-ILS).

5.4 Performance of Motif-GRASP on Different Artificial Sets

Now, after we convinced ourselves that Motif-GRASP is better than Motif-ILS, we tested its performance on harder sets, that have different characteristics.

Figure 13 shows the results of Motif-GRASP (from left to right, top to bottom) on the four sets introduced in the Data section: *Art-Motif-SET*, *Art-Num-Seq-SET*, *Art-Prob-SET* and *Art-Seq-Len-SET*. They all contain several groups of 10 sets. In the figures, the groups are separated by green lines and every two adjacent groups are drown in different colors (red and blue) and symbols (stars and crosses). Moreover, each group corresponds to a $[x, x + 9]$ interval on the x-axis. For example, the interval 1-10 in the first figure shows the results for sets that contain motifs of length 5, the interval 11-20 contains the results for the sets with motif length 10 and so on.

The top-left figure shows that motifs of length 5 are hard to find, the performance of Motif-GRASP being poor, i.e. between 0 and 0.4. The sets with motif length 10 are easier, and on the sets with motif length 15, 20 and 25, Motif-GRASP gives a good performance (between 0.9 and 1).

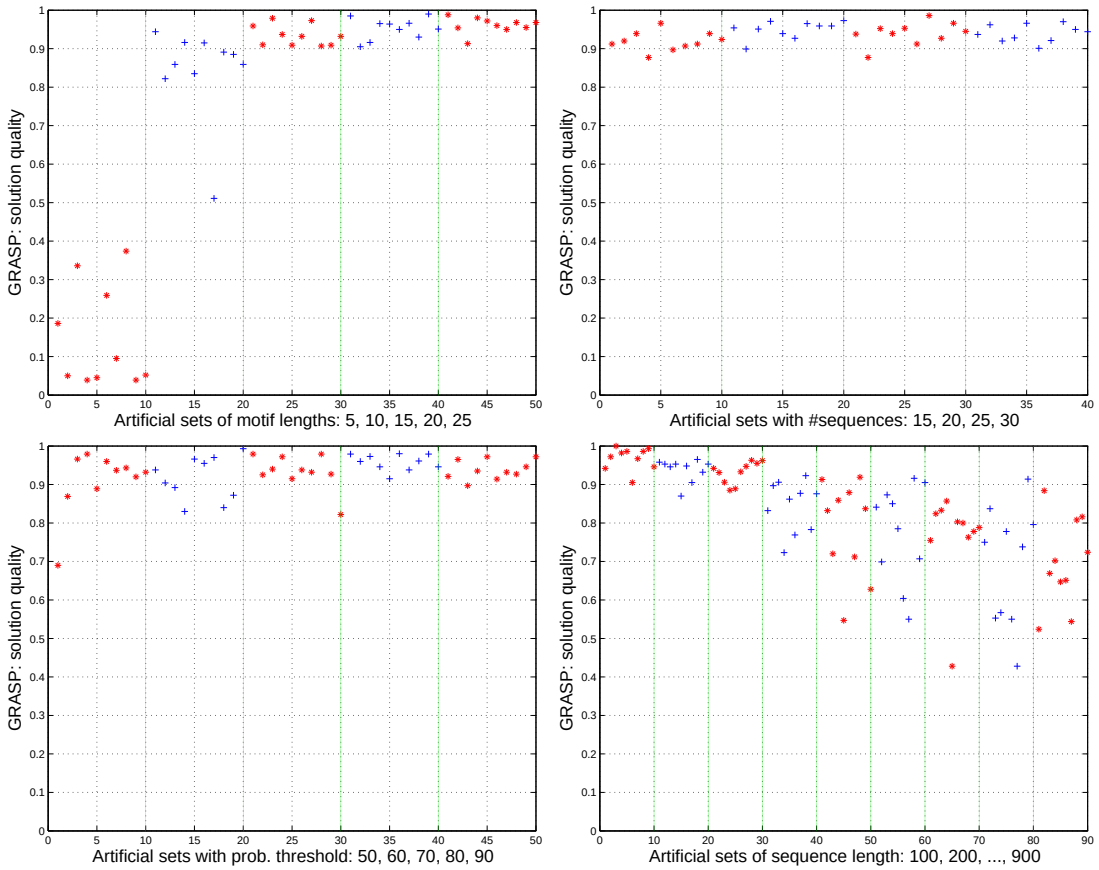

Figure 13: Performance of Motif-GRASP on sets with different characteristics. Details are explained in the text.

The top-right figure shows that Motif-GRASP behaves well for sets with different number of sequences, all of them having performance between 0.85 and 1.

The bottom-left figure shows the sets created with the probability threshold 50 and 60 are harder to solve, whereas the sets with a higher probability threshold, as expected, are easier.

The bottom-right figure shows the greater the sequence length in the set, the harder the sets are to solve. For longer sequences, using different values for the parameters (e.g. the *RCL* or *Num-Restarts*) might be necessary.

5.5 Performance of Motif-GRASP and Motif-ILS on the *Challenge-SET*

Pevzner et al [9] created the Challenge Problem and they report the Gibbs Sampling algorithm performs very poorly on these sets. For sets that have sequences of length 600, like our *Challenge-SET*, the GIBBS' performance is 0.12. The performance of Motif-GRASP and Motif-ILS is very poor, too. Figure 14 shows the performance of Motif-GRASP vs. Motif-ILS using the *Lawrence*

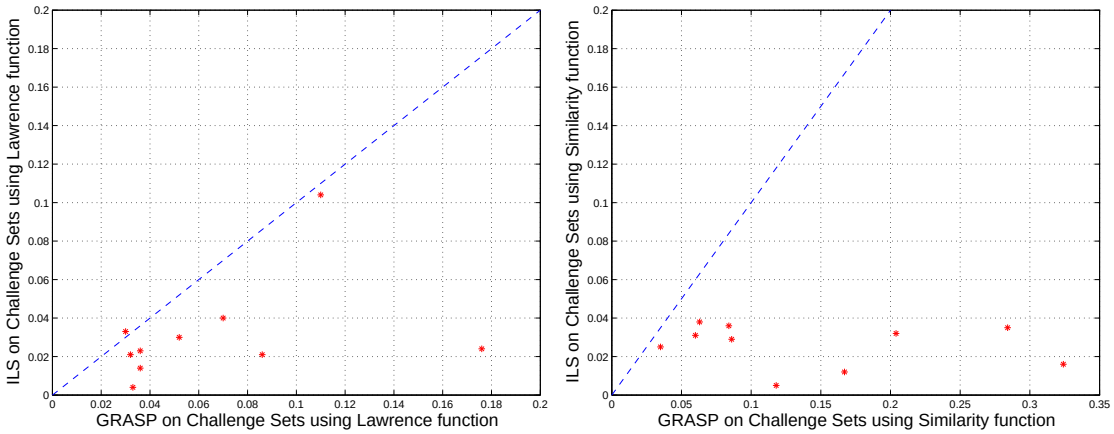

Figure 14: Performance of Motif-GRASP and Motif-ILS on the *Challenge-SET*, using *Lawrence* function and the *Similarity* function. In all cases, the performance is poor.

evaluation function (left) and using the *Similarity* evaluation function. All of them give average solution quality between 0.02 and 0.18.

5.6 Performance on Real Data: the *CRP-Set*

The only real data we could test on was the *CRP-Set*. We ran Motif-GRASP 50 times and obtained three types of good results that appeared more frequently. Table 1 describes these results. Motif-ILS did not give good results for this set. Column 2 shows the names of the 18 DNA sequences. Column 3 contains the real starting points of the motif in each sequence. Some sequences contain 2 positions of the motif, and the alternative starting points are shown in paranthesis. $L+R$ starting positions, in column 4, are the ones reported in [10]. Column 5 contains the best starting positions reported in [11].

Column 6 shows the first type of results we have got, which occured the most frequently (in 19 runs out of 50) and has quality solution of 0.82. For this type, all predicted starting points are shifted right by 2 positions comparing to the real starting points, except for *trn9cat* (sequence 17) that is predicted incorrectly.

Column 7 shows the second type of results, having quality solution of 0.85, which occurs in 8 (out of 50) of runs. For this type, all predicted starting points are shifted left by 1 position comparing to the real starting points, except the starting points for *eco gale* (sequence 7) and *trn9cat* (sequence 17), which are predicted incorrectly.

Finally, the last column shows the third type of results which occurs in 8 (out of 50) of runs. This is the best result according to the solution quality measure (i.e. 0.89) and the predicted starting positions. For this type, all starting points were determined correctly, except for *eco gale* (sequence 7) and *trn9cat* (sequence 17).

The solution quality of the remaining 15 runs was worse than those presented here.

No	Sequence name	Real Sites	L+R	[11]	Type 1	Type 2	Type 3
1	cole 1	61 (17)	61	61	63	60	61
2	eco arapob	55 (17)	55	55	57	54	55
3	eco bglrl	76	76	76	78	75	76
4	eco crp	63	63	63	65	62	63
5	eco cya	50	70	50	52	49	50
6	eco deop	7 (60)	7	7	9	6	7
7	eco gale	42	42	24	44	**23**	**24**
8	eco ilvbpr	39	39	39	41	38	39
9	eco lac	9 (80)	9	9	11	8	9
10	eco male	14	14	14	16	13	14
11	eco malk	61 (29)	29	61	63	60	61
12	eco malt	41	41	41	43	40	41
13	eco ompa	48	48	28	50	47	48
14	eco tnaa	71	71	71	73	70	71
15	eco uxul	17	17	17	19	16	17
16	pbr-p4	53	53	53	55	52	53
17	trn9cat	1 (84)	84	5	**57**	**4**	**5**
18	(tdc)	78	78	78	78	77	78

Table 1: Starting points for the common motif in the CRP data set. The table contains the names of the DNA sequences, the real starting points, starting points reported by two other algorithms, and the three type of results which are found more frequently by the Motif-GRASP algorithm.

6 Search Space Analysis

In this section we try to get an idea of how the search spaces for two problem instances look like. We use an easy set from the *Art-Tune-SET*, and a hard set from the *Challenge-SET*. First, we use the *Motif-ILS* algorithm and watch the steps it performs between a local minimum and a jump to a new local minimum. Next, we draw a Fitness-Distance Correlation (FDC) plot and calculate the correlation coefficient.

6.1 Search Space Samples Using Motif-ILS Trajectory

Figure 15 shows four samples from the search space of one relatively easy set from *Art-Tune-SET*. (The set is given in Appendix 1). For this purpose, we ran Motif-ILS 4 times with *Cutoff* = 50. This means that 50 local-search - perturbation cycles were performed. For each step, two evaluation function values are drawn in the plots: (1) the value for the local maximum, and (2) the value for the new candidate solution after a new jump in the search space was performed. Note that these figures do not show how hard is to find a local maximum (i.e. how many local search steps are performed), but only the difference between the candidate solution quality in a local maximum and after a jump. The graphs show that there are many local maxima. For this set, the optimal evaluation function value is 132.78.

Figure 16 shows a sample of the search space for a hard set from the *Challenge-Set*. As in Fig-

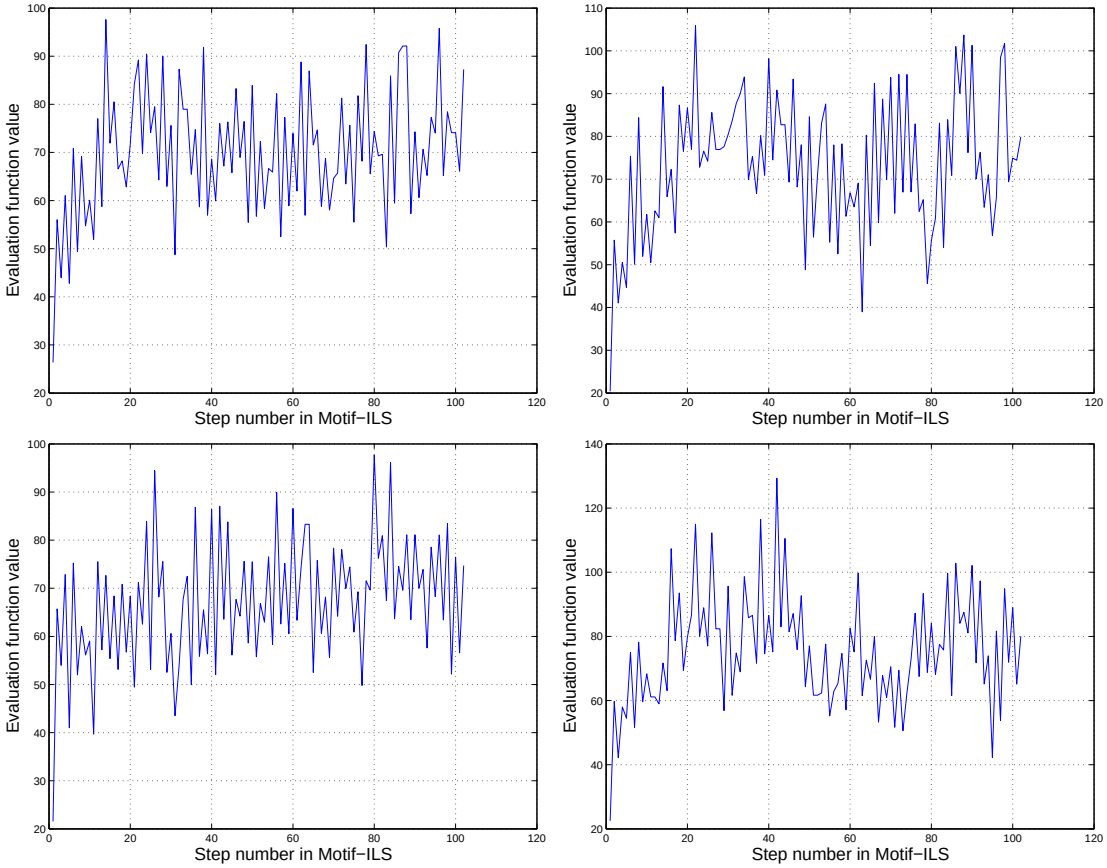

Figure 15: Motif-ILS search trajectory for one set from *Art-Tune-SET*. Each graph represents a trajectory starting from a different random pick. The graphs show that there are many local maxima and the level of ruggedness is high.

ure 15, we can see that, again, there are many local maxima.

6.2 Measuring Fitness-Distance Correlation

Figure 17 shows FDC plots for the same two sets used in the previous subsection. We created these plots in the following way: we performed a random pick of a candidate solution and then applied local search until we found a local maximum. This point was a point in the plot. We repeated this step n times. Clearly, a higher n will give a more accurate FDC plot. For the set from the *Art-Tune-SET* (left), $n = 10,000$. For the set from the *Challenge-SET* (right), $n = 1,000$, because reaching a local maximum takes much longer. We note that this yields an incomplete FDC plot for this case.

The distance is measured as the number of sequences for which the starting positions in this local maximum solution are different from the real starting positions. Thus, if all positions are correct, the distance will be 0, and if all positions are wrong, the distance will be N, where N is the number

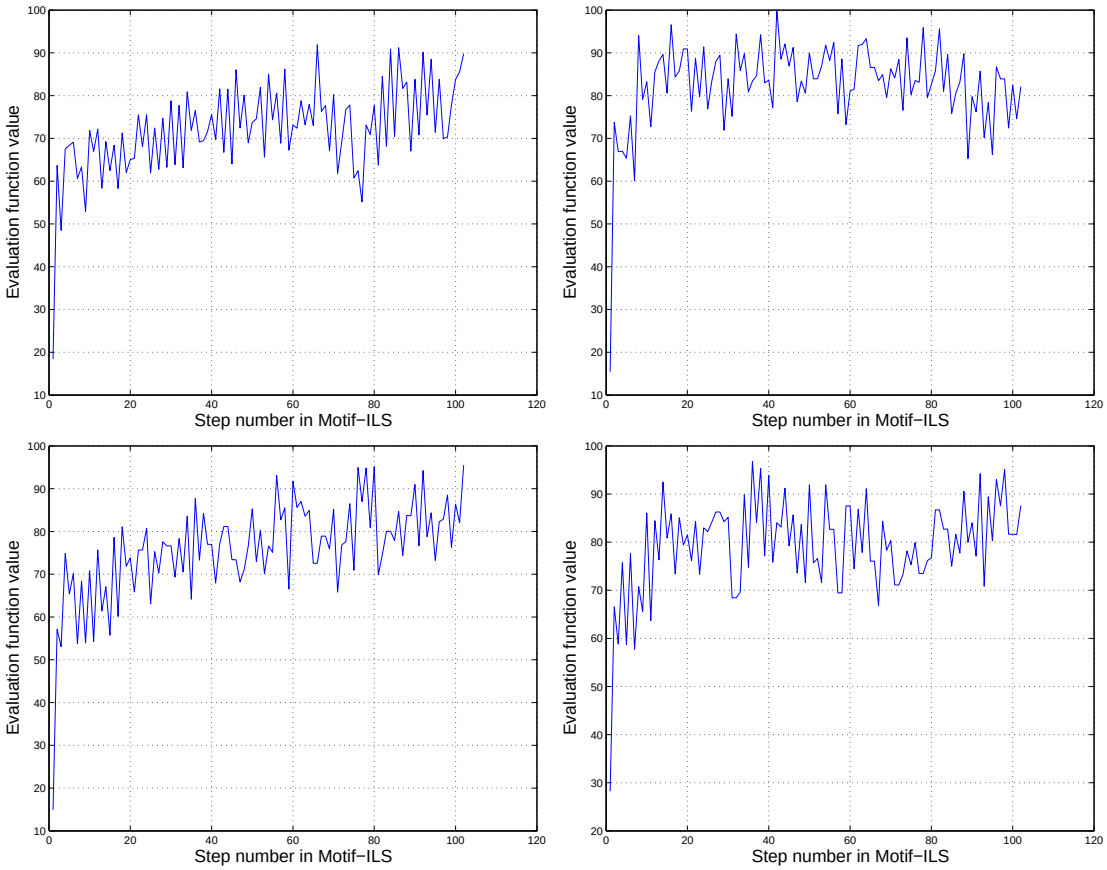

Figure 16: Motif-ILS search trajectory for one set from the *Challenge-SET*. As in Figure 15, the level of ruggedness is high.

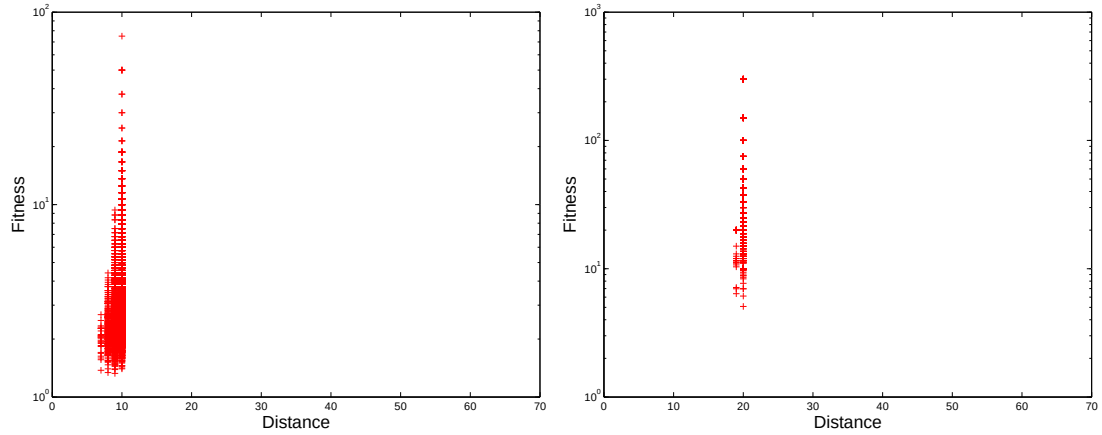

Figure 17: FDC plots for a set from *Art-Tune-SET* and for a set from *Challenge-SET* (incomplete).

of sequences in the set. So, if the distance is x, where $0 \leq x \leq N$, the minimum number of steps to the global optimum is x, i.e. by changing one position to the correct one at a time.

The fitness is measured as $1/quality$, where $quality$ is the solution quality value measured as described in Section 3. The $quality$ is a value between 0 and 1, thus $1/quality$ will be a value between 1 and infinity. (For practical reasons, we set a bound to this value). The higher the solution quality, the lower $1/quality$ will be.

In the left figure, we can see the good-quality candidate solutions are at the bottom of the plot. There are many local maxima with the highest possible distance (i.e. 10), and their quality ranges from a pretty good one to a very bad one. We note that the local maximum candidate solutions which have a lower distance also have a better quality. The Fitness-Distance correlation coefficient for this set was 0.76, which shows that the problem is pretty easy, but still has many local maxima. For this case, perturbation is better than restart, and this explains why both Motif-GRASP and Motif-ILS work pretty well for this instance.

Although it is incomplete, the right figure shows that most of the local maxima are concentrated around N, i.e. 20. The FDC coefficient for this set is 0.55, that shows clearly that this set is much harder to solve than the previous one. None of Motif-GRASP and Motif-ILS perform well on this set.

7 Conclusions and Future work

In this paper we have studied two new stochastic search algorithms for the motif finding problem in biosequences: a constructive method, *Motif-GRASP*, and a non-constructive method, *Motif-ILS*. We performed many tests on artificially generated data and a few tests on simulated and real data. The results show that a constructive method is a good fit to this problem. The search space analysis we have performed explained why this is the case: there are many local maxima, with different solution qualities, at a range of distances from the global maximum. This makes the search hard. However, on the *CRP-Set*, our *Motif-GRASP* performed nearly as good as Gibbs Sampler and the algorithms reported in [10, 11] . We think these are promising results, and with further improvement, which could be done in the future, maybe we can achieve performances comparative to the state-of-the-art algorithms for this problem.

As future work, first of all, testing our algorithm on CRP data set showed that the *Lawrence* evaluation function does not necessarily give a maximum value for global optimum (real motif positions). This means that trying to find a solution with better (greater) evaluation function, may not always lead to the real answer. Therefore finding an evaluation function which has not this weakness might improve the performance of the algorithm.

Other improvements of our Motif-GRASP are appropriate, such as: using a more randomized local search function, tuning the parameters on more diverse data sets, changing some parameters adaptively. Also, search space analysis is very useful for algorithm design. Doing such analysis more thoroughly may help us understand why the sets from the *Challenge-SET* are so hard to solve.

Trying other and more complex local search method can also be considered as future work. It might happen that a more complex algorithm as local search leads us to a higher quality candidate solution.

There are also some important limitations of our algorithms, that can be addressed in the future:

- Unknown motif length: Currently our algorithm requires the motif length as input. It does not work for data sets in which the length of motif is unknown. However, in real life, the length of the motif is not exactly known. All we know is that they cannot be too low or too high. The estimated value is between 5 and 25.
- Insertions and Deletions : We assumed the motif in each sequence is a contiguous segment. Our algorithm cannot handle the situation when the motif is partitioned into more than one contiguous segment.
- Each sequence has exactly on motif: More occurences of the motif in a sequence or no occurence in some sequences in the set can appear in real data. In real life, we do not know whether or not some sequence in the set has the motif, or how many times some motif appears.

Acknowledgements. We gratefully thank Holger Hoos for giving us the idea of tackling this problem for our course project, for useful discussions and for the feedback. We also thank Sohrab Shah for providing us the *Challenge-SET* and the *CRP-Set*, and for giving us access to his previous achievements.

References

[1] H. H. Hoos and T. Stützle, *Stochastic Local Search - Foundations and Applications*, Morgan Kaufmann Publishers, to appear.

[2] T. A. Feo and M. G. Resende, *Greedy randomized adaptive search procedures*, Journal of Global Optimization 6 (1995), 109-133.

[3] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald and J. C. Wootton, *Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment*, Science, 1993 Oct 8;262(5131):208-14.

[4] J. Buhler and M. Tompa, *Finding Motifs Using Random Projection*, Proceedings of the Fifth Annual Conference on Computational Molecular Biology, 2001

[5] M. Blanchette, *Algorithms for phylogenetic footprinting*, Proceedings of the Fifth Annual Conference on Computational Molecular Biology, 2001

[6] T. L. Bailey and C. Elkan, *Fitting a mixture model by expectation maximization to discover motifs in biopolymers*, Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, pp. 28-36, AAAI Press, 1994.

[7] M. Blanchette, B. Schwikowski and M.Tompa, *An exact algorithm to identify motifs in orthologous sequences from multiple species*, Proceedings of the Eigth International Conference on Intelligent Systems for Molecular Biology, 2000, AAAI Press

[8] Y-J. Hu, *Finding subtle motifs with variable gaps in unaligned DNA sequences*, Computer Methods and Programs in Biomedicine 70 (2003) 11-20

[9] P.A. Pevzner and S.H. Sze *Combinatorial Approaches to Finding Subtle Signals in DNA Sequences*, American Association for Artificial Intelligence, AAAI Press, 2000

[10] C.E. Lawrence and A.A. Reilly *An Expectation Maximization (EM) Algorithm for the Identification and Characterization of common sites in unaligned biopolymer sequences*, Proteins, 7:41-51,1990.

[11] S. Shah, *Making sense of 'junk': An EM algorithm for discovering motifs in intergenic DNA*, Project report for the 530A course at Computer Science, UBC, December 2002.

[12] S. Shah, *Computational motif finding: A review of current methods in pattern finding in biosequences*, September 2002.

Appendix: Sequence sets that share a common motif

This Appendix shows some sample data sets we used in our tests. The fragments inside the brackets are the real motifs shared by each set. The sequences are aligned such that the motifs start at the same position in the line.

CRP-Set **The sets consist of 18 DNA sequences of length 105, each containing a motif of length 12.**

```
TAATGTTTGTGCTGGTTTTTGTGGCATCGGGCGAGAATAGCGCGTGGTGTGAAA
GACTGT (TTTTTGATCGTTTTCACAAAAA) TGGAAGTCCACAGTCTTGACAG

GACAAAAACGCGTAACAAAAGTGTCTATAATCACGGCAGAAAAGTCCA
CATTGA (TTATTTGCACGGCGTCACACTT) TGCTAT GCCATAGCATTTTTATCCATAAG

ACAAATCCCAATAACTTAATTATTGGGATTTGTTATATATAACTTTATAAATTCCTAAAATTACACAAA
GTTAAT (AACTGTGAGCATGGTCATATTT) TTATCAAT

CACAAAGCGAAAGCTATGCTAAAACAGTCAGGATGCTACAGTAATACATTGATGTA
CTGCAT (GTATGCAAAGGACGTCACATTA) CCGTGCAGTACAGTTGATAGC

ACGGTGCTACACTTGTATGTAGCGCATCTTTCTTTACGGTCAAT
CAGCAA (GGTGTTAAATTGATCACGTTTT) AGACCATTTTTTCGTCGTGAAACTAAAAAAACC

AGTGAA (TTATTTGAACCAGATCGCATTA) CAGTGATGCAAACTTGTAAGTAGATTTCCTTA
ATTGTGATGTGTATCGAAGTGTGTTGCGGAGTAGATGTTAGAATA

GCGCATAAAAAACGGCTAAATTCTTGTGTAAACGA
TTCCAC (TAATTTATTCCATGTCACACTT) TTCGCATCTTTGTTATGCTATGGTTATTTCATACCATAAGCC

GCTCCGGCGGGGTTTTTTGTTATCTGCAATTC
AGTACA (AAACGTGATCAACCCCTCAATT) TTCCCTTTGCTGAAAAATTTTCCATTGTCTCCCCTGTAAAGCTGT

AA
CGCAAT (TAATGTGAGTTAGCTCACTCAT) TAGGCACCCCAGGCTTTACACTTTATGCTT
CCGGCTCGTATGTTGTGTGGAATTGTGAGCGGATAACAATTTCAC

ACATTAC
CGCCAA (TTCTGTAACAGAGATCACACAA) AGCGACGGTGGGGCGTAGGGGCAAG
GAGGATGGAAAGAGGTTGCCGTATAAAGAAACTAGAGTCCGTTTA

GGAGGAGGCGGGAGGATGAGAACACGGCTTCTGTGAACTAAACCGAGGTCATGT
AAGGAA (TTTCGTGATGTTGCTTGCAAAA) ATCGTGGCGATTTTATGTGCGCA

GATCAGCGTCGTTTTAGGTGAGTTGTTAATAAAG
ATTTGG (AATTGTGACACAGTGCAAATTC) AGACACATAAAAAAACGTCATCGCTTGCATTAGAAAGGTTTCT

GCTGACAAAAAAGATTAAACATACCTTATACAAGACTTTTT
TTTCAT (ATGCCTGACGGAGTTCACACTT) GTAAGTTTTCAACTACGTTGTAGACTTTACATCGCC
```

```
TTTTTTAAACATTAAAATTCTTACGTAATTTATAATCTTTAAAAAAAGCATTTAATATTGCTCC
CCGAAC (GATTGTGATTCGATTCACATT) TAAACAATTTCAGA

CCCATGAGAG
TGAAAT (TGTTGTGATGTGGTTAACCCA) ATTAGAATTCGGGATTGACATGT
CTTACCAAAAGGTAGAACTTATACGCCATCTCATCCGATGCAAGC

CTGGCTTAACTATGCGGCATCAGAGCAGATTGTACTGAGAGTGCAC
CATATG (CGGTGTGAAATACCGCACAGAT) GCGTAAGGAGAAAATACCGCATCAGGCGCTC

       (CTGTGACGGAAGATCACTTCGC) AGAATAAATAAATCCTGGTGTCCCTGTTGAT
ACCGGGAAGCCCTGGGCCAACTTTTGGCGAAAATGAGACGTTGATCGGCACG

GATTTTTATACTTTAACTTGTTGATATTTAAAGGTATTTAATTGTAATAACGATACTCTGGAAAGTATTGA
AAGTTA (ATTTGTGAGTGGTCGCACATAT) CCTGTT
```

A set from the *Art-Tune-SET* The sets consists of 10 DNA sequences of length 50, each containing a motif of length 15.

```
  CCCCTCGAACGTATGTCAGGCAATCTGCGGT (GTGACAATATCCTAG) CCAT

                              GATGT (GTTACAATATCCTTG) GGGGGCCATTATAGCCTTCGCTACCCCGCA

 TACTCTGAATGTGACCCGCTCACCAGTCTCGC (GAGACAACATCCTAA) ACA

    GCTTGAGCGCTATCCGCAGCGAAGACCGG (TTGACATTAACCTTG) GTGTTT

                        TAAGCTGGTAC (GTGACAATATCCGTG) TAGAATCTCCGCCGCGCGAGCACA

                           CAGGCCCT (GAGAGAATATCCTTG) TTAACTTCAGACTTAAAATGATCCGAA

    TCTTTCGGCTCCTAATCCAATCATTACCC (GTGACAATATCCTTC) TTAATT

                             GTTTAAT (GAGACAATATCCTTG) GGGCACGTCCACAATTAAACCTCAGGTT

 GTAAACATGGGTTCGCTCCTGGCACAAATAGC (ATGACAATATCCATG) TGC

GGCTGAGGCATCTGTGGGCTCAAACGAGCACTG (TTGACAGTAACCATC) AG
```