Pacific Symposium on Biocomputing 4:482-493 (1999)

Application of parameter optimization to molecular comparison problems

C. Lemmen, A. Zien, R. Zimmer, and T. Lengauer
GMD – German National Research Center for Information Technology,
Institute for Algorithms and Scientific Computing (SCAI),
Schloß Birlinghoven, D-53754 Sankt Augustin, Germany,
{*Christian.Lemmen,Alexander.Zien,Ralf.Zimmer,Thomas.Lengauer*}*@gmd.de*

Various bioinformatics comparison problems require optimizing several different properties simultaneously. Often linear objective functions combine the values for different properties of solution candidates into a single score to allow for multivariate optimization. In this context, an essential question is how each property should be weighted. Frequently, no apparent measure is available to serve as a model for the score. However, if preferences of certain solution candidates over others in a training set are available, the implied partial ordering may be used to best possibly adjust the weights. We apply different strategies to optimize the parameterization of empirical scoring functions used for two molecular comparison problems, *protein threading* and *small molecule superposition*. Using well established evaluation methods, it can be shown that the results of both comparison methods are significantly improved by systematically choosing appropriate weights for the scoring function contributions.

1 Introduction

Many analysis procedures in bioinformatics apply empirical scoring schemes that involve linear scoring functions composed of several terms. Two examples from the work in our group are the protein threading tool 123D [1] and the small molecule superposition tool FlexS [2]. Usually, the terms, which we will call scoring *contributions*, model different aspects of the optimization problem on a phenomenological basis. In this context, it is unclear how the different contributions should be weighted. Because of the absence of a suitable measure that could serve as a model for the score, regression methods are not applicable. Instead, we will calibrate the score by giving a preference to certain so-called *reference solutions*, e.g. solutions derived from experimental data.

We have introduced novel methods [3] that can be used in such cases. More precisely, our methods deal with the following problem. Given a data set, each point s of which is characterized by a descriptor $c = c(s)$ which is a d-vector of scoring contributions, determine a weight vector w such that the reference solutions rank highest according to the following definition of the score

$$score(s) = wc = \sum_{p=1}^{d} w_p * c_p. \quad (1)$$

Previous related work can be found in [4]. Maiorov and Crippen introduced the idea of calibrating weights such that a single reference solution ranks top.

Here, we consider two problem variants, both related to pattern recognition [5]. We developed three methods to tackle these problems: an algorithm (called CALP) based on polyhedral intersections, an algorithm that is distantly related to the classical *perceptron* algorithm [6] (called VLSH), and an algorithm that is based on a linear programming approach for the pattern recognition problem [7] (called VALP). Recently, Akutsu and Tashimo [8] suggested a similar linear programming formulation and applied it to protein threading.

Our calibration methods have been described in detail previously [3]. In this paper, we discuss the application of the methods to two molecular comparison problems using an iterative procedure to estimate weights and to compute the corresponding solutions. Our goal is to improve the performance and sensitivity of the underlying comparison methods for threading and superposition. In small molecule superposition we perform predictions that aim at approximately reproducing experimentally observed data that serve as reference solutions. The quality of a predicted solution is measured in terms of a positional error (rmsd). An appropriate weighting of the different scoring contributions should score the reference solutions highest. With respect to threading we are interested in reproducing a predefined fold classification [a]. I.e., given a certain threading procedure and objective function, the weighting should score highest a set of similar folds which serve as (multiple) reference solutions for the protein sequence. We show that for both applications significant improvements can be achieved over results obtained without systematic calibration. E.g., we can improve the fold recognition rate in threading experiments by about 40%. Our procedure is able to compute reasonable weights for the different scoring contributions [b].

2 Calibration methods

In this section, we briefly summarize our calibration methods. A more detailed description can be found in [3].

Recall that we compare solutions s that are scored with real values (cf. equation 1). Here, $c = (c_1, \ldots, c_d)$ is a d-vector of scoring contributions and w is a d-vector of weights. In essence, the goal of our methods is to determine weights such that our preselected reference solutions score higher than other solutions. We confine our algorithms to samples of the respective solution space, since complete coverage is usually infeasible. Let S^i be a sample of solutions for the problem instance i in a certain limited training set [c]. Reference solutions in S_i have been chosen a-priori. Details on how to choose instances,

[a] according to structural criteria (SCOP), independent of threading scores

[b] i.e. meaningful from a structural point of view

[c] E.g. in threading, i denotes a protein sequence and S_i a set of threadings of that sequence.

samples and reference solutions are given in the application sections.

In general, it is impossible to adjust the weights such that all reference solutions score higher than all other elements of S_i. The assumed reasons for this are the deficit of a simple scoring function to accurately model reality as well as experimental errors in the reference solutions. Therefore, we relax the calibration goal and consider the following two problem variants:

Definition 1 [Violated Inequality Minimization (VIM)]:

> Here, we assume a single reference solution $\hat{s}^i$ for each training instance i, and ask for minimizing the overall number of non-reference solutions that score higher than the corresponding reference.

Definition 2 [Cone Intersection Maximization (CIM)]:

> Here, we allow for several reference solutions for each training instance, and aim at maximizing the number of instances for which an arbitrary reference solution scores highest.

Both problems are computationally hard[d]. We have developed two approximation algorithms for VIM, and one for CIM. The methods for VIM can also be applied to pattern recognition (PR) problems, and perform competitively with established PR algorithms[3].

Using formula (1), VIM can be written as a system of linear inequalities

$$wx > 0, \text{ with } x := c(\hat{s}^i) - c(s). \tag{2}$$

Such an inequality is defined for each non-reference solution $s \in S_i$ of each training instance i. The vector x is normal to a hyperplane that divides the d-dimensional space of weight vectors into two half-spaces. The regions in d-space for which a fixed set of these inequalities holds is a polyhedral (*cone*) which amounts to the intersection of the respective positive half-spaces. In order to satisfy all inequalities simultaneously, the weight vector must be chosen from the intersection of all positive half-spaces (called *solution cone*).

Frequently, the inequality systems derived from application data are inconsistent, implying an empty solution cone. Minimizing the number of violated inequalities (VIM) amounts to finding a cone that is the intersection of as many positive half-spaces as possible. However, since the number of cones grows exponentially in d, cones cannot be examined exhaustively.

2.1 VIM Line Search Heuristic (VLSH)

Starting from a given point w in weight space, VLSH explores a polynomial number of cones as follows. For each inequality violated by w, a line through

[d] proven to be NP-complete[3]

w and perpendicular to the corresponding hyperplane is constructed. For every cone intersected by the line, the number of satisfied inequalities is calculated and the best cone found so far is stored. Finally, w is replaced by an inner point of the best cone found after processing all lines. This greedy strategy is iterated until no further improvement is achieved.

We consider two different choices of starting points. VLSH(centr) indicates that the centroid of the vectors x is used. VLSH(VALP) applies VLSH to the result of VALP (see below).

2.2 VIM Approximation Linear Program (VALP)

The basic technique used in VALP has already been applied successfully to the pattern recognition problem[7], which is closely related to VIM[3]. We formulate an approximation of the number of violated inequalities that can be expressed by means of a linear objective function and linear constraints. This enables us to use existing linear programming (LP) solvers to optimize the resulting objective function.

For each inequality, we define a measure of error. If the inequality holds, the error is zero, otherwise it amounts to the slack that must be added to the left side of inequality (2) in order to fulfill it. The sum of the error terms for all inequalities serves as a linear substitute for the number of violated inequalities.

2.3 CIM Approximation Linear Program (CALP)

The CIM problem considers several separate inequality systems, one for each reference solution of each training instance i. Any such inequality system states that the considered reference solution scores higher than any other candidate from the respective sample S^i. The goal is to find a weight vector w lying within the intersection of as many solution cones as possible. This maximizes the number of instances with a reference solution scoring highest.

In order to address CIM problems, we use an approach that is analogous to VALP. Here, we define a measure of error for each solution cone. It is zero, if the cone is hit, and the Euclidean distance to the most distant positive half-space, otherwise. The sum of these error terms over all cones serves as a substitute for the number of cones not hit by w. Again, minimizing the sum of errors fits the framework of linear programming.

3 Applications

3.1 Small molecule superposition

A typical task in pharmaceutical research is to align small molecules structurally in 3D space (*ligand superposition*). This is done to allow for a detailed

comparison of local physico-chemical properties of the molecules. A common assumption is that molecules with similar distributions of properties in space behave similarly [9].

Methods Our program FLEXS [e] takes a combinatorial approach to the ligand superposition problem [2]. It allows to fit a flexible *test ligand* onto a rigid *reference ligand* applying the following protocol. First, the flexible ligand is decomposed into small and relatively rigid portions (*fragments*). Second, an anchor fragment of the test ligand is selected. Third, using a discrete surface approximation, possible positions of the anchor on top of the reference molecule are determined. Finally, in an iterative incremental construction procedure, the anchor placements are extended by adding the remaining fragments of the test ligand step by step considering a discrete set of possible conformations for each fragment. The number of partial placements, generated in this way, grows exponentially with the number of added fragments. A greedy strategy is applied in each iteration to select a suitable subset of placements which is used for the next iteration. In order to do so, partial placements are scored and sorted by score.

Relevant parameters The scoring contributions of FLEXS comprise a matching term (c_{match}) that accounts for intermolecular interactions, a van der Waals overlap term (c_{vdW}), a topology matching term (c_{stm}), and five overlap terms considering Gaussian functions that model different physico-chemical properties [f] p (c_p). I.e., the objective function to be maximized is:

$$w_{\text{match}} \cdot c_{\text{match}} + w_{\text{vdW}} \cdot c_{\text{vdW}} + w_{\text{stm}} \cdot c_{\text{stm}} + \sum_{\text{property } p} w_p \cdot c_p \qquad (3)$$

Hence, we consider an eight-dimensional descriptor c for every placement. The choice of weights for each term critically influences the optimal superposition found. There is experimental evidence on *reference superpositions* for test cases from X-ray data. Obviously, the similarity score should achieve its maximum at the observed orientation in those cases. For two ligands that bind to the same protein, the reference superposition is obtained by superimposing the backbones of the corresponding protein-ligand complexes. The rigid reference ligand is taken in this orientation. For the flexibly fitted test ligand the root-mean-square deviation (rmsd, for short) from its orientation in the reference superposition can be calculated.

Our data set comprises 284 ligand pairs (instances) obtained from protein-

[e] available via http://cartan.gmd.de/FlexS (case sensitive!)

[f] electron density, hydrophobicity, partial charge, H-bonding donor potential, and H-bonding acceptor potential

ligand complexes of 14 proteins [g]. This selection covers the whole range of drug-size molecules from 18 to 158 atoms.

Calibration In order to specify a calibration problem we define instances as ligand pairs from the training set. S_i is a sample set of superpositions generated by FLEXS. For the VIM specification the reference superposition $\hat{s}_i$ is derived as detailed above. For the CIM specification the set of reference superpositions for an instance i is extended to those superpositions from the sample S^i with an rmsd to $\hat{s}^i$ of at most 1.5 Å. For sample set generation the scoring used for the greedy selection is simply replaced by a random number. The final set of complete superpositions is evaluated for the different scoring contributions. The training set comprises about one third of the entire test set (90 ligand pairs). Thus, the ability of the method to generalize to novel data is implicitly accounted for in the results.

Results In order to evaluate the performance of the calibrated scoring functions we distinguish by rmsd between four classes of superpositions [h]. Furthermore, in order to account for the ranking of the generated superpositions, we distinguish between superpositions found at the first rank, within the top ten ranks, and among all ranks [i], respectively. The corresponding figures can be found in Table 1. Carrying out several iterations of data generation and calibration will be subject of future work.

As a point of reference, the results using the original parameterization ($w = (1, ..., 1)$) are provided together with the results using the calibrated scoring functions obtained by the different methods. The average number of candidates handled by the algorithms is about 25.000 and the runtime required for candidate generation (60%) plus calibration (40%) is about 2.5 hours.

The data show that the application of any of the calibration methods generates a parameterization of the empirical scoring function that is superior to the original scoring scheme with respect to any of the indications tabulated. VALP shows a slightly better and CALP a slightly worse performance than VLSH. However, as desired, the density of placements with low rmsd on the best ranks (1st–10th) increases markedly. Note that flexible ligand superposition has to be considered an extremely difficult task [2] [j]. With this respect, 12% more superpositions with an rmsd <1.5 Å and about 13% additional superpositions

[g] immunoglobuline, streptavidin, trypsin, glycogen phosphorylase, concanavalin, HIV-protease, elastase, thermolysin, carboxypeptidase, thrombin, dihydrofolate reductase, human rhinovirus, and fructose bisphosphatase

[h] <1.0 Å, <1.5 Å, <2.0 Å, and >2.0 Å; of course the smaller the rmsd the better

[i] Since the greedy strategy is applied after each expansion step also the final number of placements generated by FLEXS amounts to at most k, all of which are plausible placements.

[j] In contrast to protein-ligand docking ligand, superpositioning relies purely on a comparison of ligand structures.

Table 1: Results using FLEXS with the original and any of the generated scoring schemes labeled by the type of algorithm given in the first column. The remaining four columns show the percentage of instances having superpositions with four specific rmsd. The difference in percent between the original and the generated scoring schemes is provided in brackets.

method	**1st rank**							
	<1.0 Å		<1.5 Å		<2.0 Å		>2.0 Å	
original	16.2%		32.4%		43.3%		56.7%	
VALP	17.6%	(1.4)	35.9%	(3.5)	47.5%	(4.2)	52.5%	(-4.2)
VLSH(VALP)	17.3%	(1.1)	35.9%	(3.5)	47.9%	(4.6)	52.1%	(-4.6)
CALP	16.5%	(0.3)	35.2%	(2.8)	47.9%	(4.6)	52.1%	(-4.6)
method	**1st–10th rank**							
	<1.0 Å		<1.5 Å		<2.0 Å		>2.0 Å	
original	24.3%		41.5%		52.1%		47.9%	
VALP	27.5%	(3.2)	46.5%	(5.0)	58.8%	(6.7)	41.2%	(-6.7)
VLSH(VALP)	27.1%	(2.8)	45.8%	(4.3)	59.2%	(7.1)	40.8%	(-7.1)
CALP	26.1%	(1.8)	45.4%	(3.9)	57.7%	(5.6)	42.3%	(-5.6)
method	**all ranks**							
	<1.0 Å		<1.5 Å		<2.0 Å		>2.0 Å	
original	33.1%		53.5%		63.0%		37.0%	
VALP	33.5%	(0.4)	53.5%	(0.0)	65.1%	(2.1)	34.9%	(-2.1)
VLSH(VALP)	33.1%	(0.0)	53.9%	(0.4)	65.1%	(2.1)	34.9%	(-2.1)
CALP	33.1%	(0.0)	53.5%	(0.0)	64.4%	(1.4)	35.6%	(-1.4)

with an rmsd <2.0 Å appears to be a promising first step. It is encouraging that all experiments indicate improvements in all categories, especially since the test set is about two times larger than the training set.

3.2 Threading of protein sequences and structures

Threading methods map protein sequences onto known protein structures in order to find the most compatible fold using empirical objective functions. These objective functions (similarity measures) are statistically derived from known protein structures using the inverse Boltzmann law [10]. Many proposals for such empirical potentials have been made in recent years [11]. General agreement has not been reached yet, neither about the scoring function nor about the threading method. It is apparent, however, that such scoring functions have to involve several contributions, the exact balancing of which influences the outcome and the performance of the respective threading experiment in non–trivial ways.

Here, we show that the careful choice of such weighting parameters is crucial by applying calibration methods which significantly improve the *fold*

recognition rate. An orthogonal approach to the determination of weights is taken by parametric alignment methods [12,13]. Unfortunately, for more than two parameters, the parametric alignment methods are still much too slow to allow for optimal parameter selection.

Methods Many methods have been proposed for threading [11]. In order to perform a systematic analysis of relevant parameters and parameter settings for the calibration methods a very fast program is necessary. We therefore choose the 123D [1] method, as implemented in the ToPLign package [k]. It is extremely fast and optimizes an interesting new empirical scoring potential. 123D uses a dynamic programming algorithm to compute, for a given sequence *seq* and given weights w, an optimal ranking list of a set of folds $\mathcal{F}$. The ranking is derived from the scores of the corresponding optimal alignments. 123D has been developed to optimize a new type of scoring potentials (called *contact capacity potentials* CCP [1]), but also exploits standard scoring contributions. One goal of the experiment described here is to evaluate the CCP potential and to derive an optimal weighting of the contributions in order to improve similarity searches.

123D threads a sequence of length $n = 150$ amino acids in about 15 minutes CPU time on a current workstation/PC against the entire set of about 13.000 PDB [14] chains. A threading run against the representative set of 251 proteins used in this study requires about 20 seconds.

Relevant parameters For the scope of this paper we consider six terms of the objective function used in 123D: a sequence score (*seqp*), a secondary structure preference (*ssp*), a 'local' contact capacity potential (*ccpl*), a 'global' contact capacity potential (*ccpg*), and an affine gap penalty function with gap insertion costs (*gi*) and gap extension costs (*ge*) [1]. These terms are evaluated independently for three types of secondary structure elements: alpha helix (H), beta strand/extended (E), and all other conformations/loops (L). The reason for this distinction is the different degree of structural conservation, and the corresponding level of confidence, in secondary structures, structural cores, and loops. The following objective function to be maximized combines the 18 relevant terms for an alignment A [l]:

$$score^A(seq, f) = \sum_{\substack{i \in \{H,E,L\} \\ j \in \{seqp, ssp, ccpl, ccpg, gi, ge\}}} w_{ij} * c_{ij}^A(seq, f)$$

For any setting of the weights w, 123D guarantees to compute optimal

[k] available via http://cartan.gmd.de/ToPLign.html (case sensitive!)

[l] The scoring terms $c_{ij}^A(seq, f)$ for a given alignment A are defined by summing the respective scoring terms (dependent on amino acids, contacts and the like) over the positions of the alignment A.

sequence–structure alignments, i.e. an alignment of the sequence seq with the fold $f \in \mathcal{F}$ that achieves the maximal score $\max_A score^A(seq, f)$. Hence, for any threading instance with given weights w and sequence seq, 123D computes $|\mathcal{F}|$ optimal alignments and the corresponding scoring contributions. These are used as a sample of candidates in the calibration procedure.

Our data set is derived from (HS25) [15] [m]. We used the SCOP classification in order to select 81 proteins from 11 families each with 5 or more members as training set `TR` and an additional 74 proteins from 24 families as test set `TS`.

Calibration For the calibration approach described here, we interpret a fold recognition experiment as follows: Given a problem instance consisting of an objective function $score = wc$ (with defined weights w), a sequence seq, and a set of representative folds $\mathcal{F} = \{f_1, \ldots, f_N\}$ [n], the threading problem is: Compute the set of all optimal alignments of seq with all folds of $\mathcal{F}$ with respect to score. A 'similar' fold f with sufficiently high 'confidence' in the similarity may be used to predict a putative fold for seq and in some cases allows for using f as a template structure to derive a 3D model for seq. For the fold recognition problem, in addition, a partition $\mathcal{FAM}$ of $\mathcal{F}$ into disjoint fold families $\mathcal{F}_f$ is given. We say a sequence seq is recognized by its family $\mathcal{F}_{seq}$, if any member of $\mathcal{F}_{seq}$ (excluding the native fold f_{seq}) scores highest. In this setup, the training set `TR` is the union of certain families and the test set `TS` is another union of families disjoint from `TR`. The candidates for the calibration of weight parameters are all alignments computed via the threading of the sequences from `TR`. Reference solutions are those aligned sequence structure pairs being classified into the same family [o]. This situation perfectly fits the CIM problem definition. The goal of the calibration, as defined by CIM, is to find weights w such that the number of recognized folds is maximized. Note that, in general, changing weights leads to changed optimal alignments and, thus, to new sets of candidates. Therefore, we repeat cycles of threading and subsequent re-estimation of weights.

In order to apply the algorithms developed for VIM to the fold recognition problem, we need to define a single reference structure for each problem instance seq. Here, we initially choose the most similar structure from $\mathcal{F}_{seq}$ according to structural superposition. Despite VIM aiming at scoring the reference structures higher than all others it may happen that different family

[m] available from EMBL: ftp://ftp.embl-heidelberg.de/pub/databases/pdb_select/

[n] Note that for our purposes here, we have $seq \in \mathcal{F}$, i.e. we know the structure of all sequences involved.

[o] We exclude native sequence structure pairs, since fold recognition is intended to be applied to sequences of unknown structure. Furthermore, 123D easily recognizes native structures for a broad range of possible parameterizations of the scoring function, any of which would equally well solve the resulting calibration problem.

Table 2: Number of correctly recognized folds as defined in the text from the training set (top half) and test set (bottom half). This measure of success is shown for the standard weights (iteration zero) and for the following iteration cycles applying the different calibration algorithms. As a point of reference, a purely sequence based approach recognizes 31 of 81 folds for the training set and 17 of 74 for the test set.

method	**iterations on the training set**							
	zero	1st	2nd	3rd	4th	5th	6th	7th
CALP	35	34 (-1)	36 (+1)	30 (-5)	30 (-5)	30 (-5)	31 (-4)	28 (-7)
VALP	35	36 (+1)	44 (+9)	46 (+11)	49 (+14)	48 (+13)	46 (+11)	48 (+13)
VLSH(centr)	35	37 (+2)	41 (+6)	41 (+6)	38 (+3)	46 (+11)	47 (+12)	41 (+6)
VLSH(VALP)	35	36 (+1)	48 (+13)	45 (+10)	45 (+10)	45 (+10)	49 (+14)	49 (+14)
method	**iterations on the test set**							
	zero	1st	2nd	3rd	4th	5th	6th	7th
CALP	26	28 (+2)	24 (-2)	17 (-9)	19 (-7)	15 (-11)	16 (-10)	17 (-9)
VALP	26	29 (+3)	31 (+5)	29 (+3)	31 (+5)	30 (+4)	30 (+4)	30 (+4)
VLSH(centr)	26	30 (+4)	27 (+1)	34 (+8)	33 (+7)	31 (+5)	31 (+5)	32 (+6)
VLSH(VALP)	26	29 (+3)	31 (+5)	32 (+6)	33 (+7)	32 (+6)	34 (+8)	31 (+5)

members score higher for some sequences. Accordingly, we reselect the reference solution after each iteration of threading and calibration as follows: taking the latest threading result, the highest scoring member of $\mathcal{F}_{seq}$ becomes the new reference structure for *seq*. In our experiments, this strategy appears to converge quickly, both with respect to the recognition rate and the resulting weights.

Results For the threading parameter calibration we have performed several experiments, each iterating parameter calibration[p] followed by the recomputation of the threading alignments. Table 2 exhibits the recognition rates using the different calibration methods during the iterations. The results are given for both the training and test sets. Consistently for all calibration methods except CALP, we observe a significant increase of the recognition rate. However, we also find quite a few differences in the convergence and the stability of the respective optimized weightings.

Table 2 shows that 35 out of 81 sequences are found with all parameters having equal weight. Calibration with VLSH(VALP) leads to 36 recognized sequences in the first iteration. A substantial increase (to 48 cases) in the recognition rate due to the reranking of the alignments is obtained in the second cycle. Unfortunately, this is not fully sustained in the subsequent recomputations. Finally, in iteration 7, we obtain a parameter setting which not only yields 47 recognized sequences after recalibration, but even 49 hits after recom-

[p] using one of the different methods as described in section 2

putation with these calibrated parameters. Overall, this is an improvement of almost 40% starting from 35 recognized sequences. Additionally, for the much more difficult test set, we start with a recognition rate of 26 sequences out of 74, which is already a remarkable improvement over the 17 sequences recognized with pure sequence alignment. The sustained performance increases to well above 30 sequences with a peak rate of 34 recognized folds, i.e. doubling the rate of pure sequence alignment.

In all experiments, the sequence scores are weighted quite heavily as compared to the other contributions. This is remarkable for two reasons: First, the recognition rate for pure sequence alignment is only around 30 recognized sequences[q]. Thus, threading with the trivial weights already results in an improvement by about 15% above the sequence alignment recognition rate. Using our new weighting, the improvement in recognition rate increases to more than 50% as compared to sequence alignment. Second, the sequence identities among the family members are at most 25% by definition. Moreover, if we analyze the sequence identities not for optimal alignments but structural superpositions the percent identities drop well below the number which is to be expected by chance, i.e. below 17–18% (often even below 10%).

A more detailed analysis of the recognized sequences over the iterations also reveals that the procedure is well-behaved along its way to an improved parameter setting. In the first major improvement of the recognition rate from 36 to 48, only one sequence previously recognized is lost and 13 additional sequences are found. Just 5 sequences find a different family member scoring highest as compared to the previous run. Only the second iteration shows some divergence in that 7 sequences are lost and 4 new sequences are found leading to a net loss of three recognized sequences[r].

4 Conclusions and Outlook

We applied novel calibration techniques to the calibration of empirical scoring functions of two important molecular comparison problems. Significant improvements could be achieved for both applications.

Using a carefully parameterized scoring function for threading improved the recognition rate on our test data from 35 to 49 sequences (43% to 60%, respectively), i.e. an increase of about 40%. In small molecule superposition significant improvements of the performance of our approach could be achieved, too. This demonstrates the usefulness of parameter calibration and the capabilities of our methods.

[q] depending on gap parameters and scoring matrix

[r] In three cases different top ranked family members are recognized.

The efficiency of our approaches, both of the application software and the calibration methods, is essential for carrying out the proposed parameter optimizations and to achieve the presented improvements. Using prominent alternative approaches, as for example THREADER [16] for protein threading or GASP [17] for ligand superpositioning, would result in a tedious procedure because of the computational costs. However, even with our fast application methods simple gradient-based parameter optimization would require many iterations and thus days and weeks of computing time.

Currently, our calibration approaches detect local optima, at best. Accordingly, improved strategies to find global optima have to be developed in future work. Also, the choice of a representative training set and an appropriate sampling of solution candidates have the potential to further improve the scoring functions and will be subject of future research.

Acknowledgements

We thank our colleague Theo Mevissen for helping to prepare the threading data.

References

1. N. Alexandrov, R. Nussinov, and R. Zimmer. In Lawrence Hunter and Teri E. Klein, editors, *Pacific Symposium on Biocomputing'96*, pages 53–72. World Scientific Publishing Co., 1996.
2. C. Lemmen, T. Lengauer, and G. Klebe. *J. Med. Chem.*, 1998. In press.
3. A. Zien, C. Lemmen, and T. Lengauer. *SIAM J. Opt.*, 1998. Submitted.
4. V.N. Maiorov and G.M. Crippen. *J. Mol. Biol.*, 227:876–888, 1992.
5. J.T. Tou and R.C. Gonzalez. Addison-Wesley, Massachusetts, USA, 1974.
6. F. Rosenblatt. Technical report, Cornell Aeronautical Laboratory, 1957.
7. K.P. Bennett and O.L. Mangasarian. *Optimization Methods and Software*, 1:23–34, 1992.
8. T. Akutsu and H. Tashimo. In *Proceedings of the Pacific Symposium on Biocomputing '98*, pages 413–424, 1998.
9. H. Kubinyi. *Perspectives in Drug Discovery and Design*, 9–11:225–252, 1998.
10. M. Sippl. *J. Mol. Biol.*, 213:859–883, 1990.
11. E.E. Lattman, editor. Supplement 1 to Proteins, 1997.
12. M. Vingron and M.S. Waterman. *J. Mol. Biol.*, 235(1):1–12, 1994.
13. R. Zimmer and T. Lengauer. In M. Waterman, editor, *First Annual International Conference on Computational Molecular Biology (RECOMB'97)*, pages 344–353. ACM Press, 1997.
14. F.C. Bernstein et al. *J. Mol. Biol.*, 112:535–542, 1977.
15. U. Hobohm and C. Sander. *Protein Science*, 3:522–524, 1994.
16. D.T.Jones, W.R.Taylor, and J.M.Thornton. *Nature*, 358:86–89, 1992.
17. G. Jones, P. Willett, and R.C. Glen. *J. Comput.-Aided Mol. Design*, 9:532–549, 1995.