

Side chain-positioning as an integer programming problem

Olivia Eriksson¹, Yishao Zhou² and Arne Elofsson³

¹ Stockholm Bioinformatics Center, Stockholm University, SE-106 91 Stockholm, Sweden E-mail:olivia@sbc.su.se

² Department of Mathematics, Stockholm University, SE-106 91 Stockholm, Sweden E-mail:yishao@matematik.su.se

³ Stockholm Bioinformatics Center, Stockholm University, SE-106 91 Stockholm, Sweden, Fax: +46-8-15 8057, Tel: +46-8-16 1553 E-mail:arne@sbc.su.se

Abstract. An important aspect of homology modeling and protein design algorithms is the correct positioning of protein side chains on a fixed backbone. Homology modeling methods are necessary to complement large scale structural genomics projects. Recently it has been shown that in automatic protein design it is of the uttermost importance to find the global solution to the side chain positioning problem [1]. If a sub-optimal solution is found the difference in free energy between different sequences will be smaller than the error of the side chain positioning. Several different algorithms have been developed to solve this problem. The most successful methods use a discrete representation of the conformational space. Today, the best methods to solve this problem, are based on the dead end elimination theorem. Here we introduce an alternative method. The problem is formulated as a linear integer program. This programming problem can then be solved by efficient polynomial time methods, using linear programming relaxation. If the solution to the relaxed problem is integral it corresponds to the global minimum energy conformation (GMEC). In our experimental results, the solution to the relaxed problem has always been integral.

1 Introduction

Within the near future the approximate fold of most proteins will be known, thanks to structural genomics projects. The approximate fold of a protein is not enough though, to obtain a full understanding of a molecular mechanism or to be able to utilize the structure in drug design. For this a complete model of the protein is often needed. The main procedure today to obtain a complete model is by “homology modeling”. The process of homology modeling often includes the positioning of amino acid side chains on a fixed backbone of a protein. Another area that has recently become important is the area of automatic “protein design”. Here the goal is to obtain a sequence that folds to a given structure. Mayo and coworkers have shown that it is possible to perform automatic designs [1]. One crucial step in their procedure is to find the optimal side chain conformation.

There are two common features to most algorithms that try to solve this problem. The first one is to discretize the allowed conformational space into “rotamers” representing the statistically dominant side chain orientations in naturally occurring proteins [2, 3]. The rotamer approximation reduces the conformation space and makes it possible to use a discrete formulation of the problem. The second feature is the use of an energy function that can be divided into terms depending only on pairwise interactions between different parts of the protein. The total energy of a protein in a specific conformation, E_C , can therefore be described as

$$E_C = E_{backbone} + \sum_i E(i_r) + \sum_i \sum_{j>i} E(i_r j_s) \quad (1)$$

$E_{backbone}$ is the self-energy of the backbone, i.e. the interaction between all atoms in the backbone. $E(i_r)$ is the self-energy of side chain i in its rotamer conformation i_r , including its interaction with the backbone. $E(i_r j_s)$ is the interaction energy between side chain i in the rotamer conformation i_r and the side chain j in the rotamer conformation j_s . In this study we will keep the backbone of the protein fixed and only change the side chain rotamers. The term $E_{backbone}$ will therefore not contribute to any difference in energy between two protein conformations and can be ignored. The problem we want to solve can thus be defined as; *given the coordinates of the backbone and a specific rotamer library and energy function, find the set of rotamers that minimizes the energy function.* The solution space of this problem obviously increases exponentially with the number of residues included.

Fig. 1. Schematic drawing of different rotamers in a fantasy protein

1.1 Methods used today

There are several types of solution methods to the side chain positioning problem. Here we briefly review three of them.

Stochastic algorithms Several groups have developed stochastic algorithms, such as Monte Carlo simulations [4] and Genetic Algorithms [5] to solve the side chain positioning problem. The rotamer approximation together with the energy function makes it possible to view the problem as a discrete energy landscape where each point represents a specific rotamer combination and an assigned energy. To find a global minimum of the energy landscape these algorithms sample solutions semi-randomly and then move from one possible solution to another in a manner that depends both on the nature of the energy landscape and on specific rules for movement. With this approach you only need to compute the energy for the sampled conformations. Another advantage of these algorithms is that you can allow the structure of the protein to vary continuously if you want. One limitation, on the other hand, is that you are never guaranteed to have found the global minimum.

Pruning algorithms The most frequently used algorithms in this category are based on the dead end elimination theorem (DEE) [6–9]. DEE-based methods iteratively use rejection criteria. If a criterion is fulfilled it guarantees that a certain rotamer or combination of rotamers cannot be a part of the GMEC. This reduces the conformation space significantly and hopefully at the end a single solution remains. If DEE-based methods converge to a single solution they are guaranteed to have found the global minimum energy. During the last few years methods based on this theorem have developed considerably and can now solve most side chain positioning problems [10]. However in protein design there is a limit beyond which this method fails to converge and other inexact methods have to be used [10]. The reason for this is that protein design requires a large rotamer library [9].

Mean-field algorithms A third approach is to use mean field algorithms [11]. Here all rotamers of all side chains can be thought of as existing at the same time, but with different probabilities. Each residue is considered in turn and the probabilities for all rotamers of that side chain are updated, based on the mean field generated by the multiple side chains at neighbouring residues. The procedure is repeated until it converges. The predicted conformation of a side chain is chosen to be the rotamer with the highest probability. An advantage of self consistent mean field algorithms is that the computational time scales linearly with the number of residues. Unfortunately, there is no guarantee that the minimum of the mean-field landscape corresponds to the true GMEC.

2 The side chain positioning problem formulated as an integer program.

The side chain positioning problem can be formulated as a classical *mathematical programming problem*. To this end it is necessary to introduce some notations and background. Mathematical programming concerns maximizing (or minimizing) a function of *decision variables*, which we denote by x , in the presence of *constraints*, which restrict the variables x to belong to a set F of admissible values, the *feasible set*. The function to optimize, *the objective function*, is denoted $f(x)$. A maximization problem can easily be turned into a minimization problem by changing the sign of $f(x)$. A general mathematical programming problem can then be stated as

$$\begin{aligned} & \text{Minimize} && f(x) \\ & \text{subject to} && g_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

In that case the decision variables only can take integer values we have an *integer programming problem*. The easiest programming problems have continuous variables and linear constraints and objective functions. These are called *linear programming (LP) problems*. Please refer to [12, 13] for a thorough introduction to linear programming and integer programming.

It is possible [8], to rewrite the energy function (1) so that it only contains terms depending on pairs of side chains, i. e. the energy of a conformation can be written as

$$E_C = \sum_i \sum_{j>i} E'(i_r, j_s). \quad (2)$$

This benefits our problem formulation. Let the decision variables be

$$x_{i_r, j_s} = \begin{cases} 1 & \text{if side chain } i \text{ is in rotamer } r \text{ and side chain } j \text{ is in rotamer } s \\ 0 & \text{else} \end{cases} \quad (3)$$

where $i = 1 \dots n_{sc}$, $j = 2 \dots n_{sc}$, $i < j$ and n_{sc} is the number of side chains. The total energy of the system (not just one conformation as before) can then be calculated as a sum, consisting of all possible variables, one for each rotamer combination, i_r, j_s , times their respective energy contribution to the total energy $e_{i_r, j_s} = E'(i_r, j_s)$, see equation (2). The energy e_{i_r, j_s} is calculated assuming both of these rotamers are included in the conformation. The *total* energy, E_{tot} is then:

$$E_{tot} = \sum_i \sum_r \sum_{j>i} \sum_s e_{i_r, j_s} x_{i_r, j_s} \quad (4)$$

where

$$\sum_r \sum_s x_{i_r, j_s} = 1 \text{ for all } i \text{ and } j, \quad i < j \quad (5)$$

$$\sum_q x_{h_q, i_r} = \sum_p x_{g_p, i_r} = \sum_s x_{i_r, j_s} = \sum_t x_{i_r, k_t} \quad (6)$$

for all g, h, i, j, k and $r, h, g < i < j, k$

$$x_{i_r, j_s} \in \{0, 1\} \quad (7)$$

The first condition (5) together with (7) are to assure, that a certain pair of side chains, i and j , only can exist in *one* rotamer-state, i_r and j_s . If you consider a certain pair of side chains, say i and j , and think of all possible combinations of rotamer states that these two side chains could possibly take, then only *one* of these combinations can exist. This could for example be i_r and j_s .

The second condition (6) states that one side chain can only exist in one rotamer state, independent of the rotamer states of other side chains. This assures that it can not exist in one state in relation to one side chain and another state in relation to another side chain.

This allows us to formulate a minimization problem as:

$$\begin{aligned} \text{Minimize:} \quad & E_{tot} \quad (8) \\ \text{subject to:} \quad & (5) (6) (7) \end{aligned}$$

This is a linear integer programming problem. We can rewrite it in matrix form:

$$z_{IP} = \min\{cx \mid Ax = b, x \in \{0, 1\}^n\} \quad (9)$$

where A is an $m \times n$ matrix and b and c vectors of appropriate dimensions. In our case all components of A are 0, 1 or -1 , the components of b 0 or 1 and c consists of real values. The condition (6) can of course be implemented in different ways to obtain this form. We refer to the Appendix for a description of our approach. A small example in Table 1 shows what the integer program can look like for a small side chain positioning problem and our implementation.

In general, integer problems are hard to solve. By relaxing the integral constraints, that is, allowing $0 \leq x_{i_r, j_s} \leq 1$ instead of $x_{i_r, j_s} \in \{0, 1\}$, we can turn this problem into a LP-problem to which there exist several efficient algorithms. If the solution to the LP relaxed problem is integral, $x_{i_r, j_s} \in \{0, 1\}$ then it is an optimal solution to the original problem [13]. This means that if we solve the relaxation of the side chain positioning problem (8) and the solution turns out to be integral, it corresponds to the GMEC.

To be precise, the *linear programming relaxation* is as follows

$$z_{LP} = \min\{cx \mid Ax = b, 0 \leq x \leq 1\} \quad (10)$$

The feasible solution set of the problem, $F(LP)$, defined by $Ax = b$ and $0 \leq x \leq 1$, forms a convex polyhedron, as it is an intersection of a finite number of hyper-planes. The LP-problem is well defined in that either it is infeasible, unbounded, or has an optimal solution. An optimal solution to an LP problem can always be found in an extreme point of the polyhedron (if there exists an

branch and bound methods can be used in combination with LP to get an integer solution.

Time complexity Let n_{sc} be the number of side chains and n_{rot} be the average number of rotamer states for each side chain. Then the integer programming formulation of the problem (8) will give approximately $n_{sc}^2 n_{rot}$ number of constraints. For the simplex algorithm it is usually counted on that the number of iterations grows linearly with the number of constraints of the problem and the work for each iteration grows as the square of the number of constraints [15]. That is the computational time of solving the relaxed problem by the simplex method should be in the order of $O(n_{sc}^6)$. If we look at the average number of rotamers for each residue the time complexity for the simplex algorithm ought to be $O(n_{rot}^3)$. This is under the assumption that the solution to the relaxed problem is integral and therefore the final solution.

3 The dead end elimination theorem

Dead end elimination methods seeks to systematically eliminate bad rotamers and combination of rotamers until a single solution remains. This is done by the iterative use of different criteria of elimination, which make it possible to exclude rotamers from being a part of the GMEC.

It is necessary for the DEE methods that the energy description is pairwise as described in equation (1). Let us consider one residue, say residue i and let $\{C'\}$ be the set of all possible conformations of the rest of the protein, that is, all possible combinations of rotamers of all side chains *except* side chain i . Now consider two rotamers of residue i , namely i_r and i_t . If every protein conformation with i_r is higher in energy than the corresponding conformation with i_t for all possible configurations of non- i residues then,

$$E(i_r) + \sum_{j,j \neq i} E(i_r j_s) > E(i_t) + \sum_{j,j \neq i} E(i_t j_s) \quad \text{all } C' \quad (11)$$

and the rotamer i_r cannot be a member of the GMEC. The inequality (11) is not easy to check computationally since $\{C'\}$ can be huge. Instead one can get a more practical, but weaker, condition by just comparing that case where the left side of (11) is *highest* in energy with the case where the right side is *lowest* in energy. This can be done by the following inequality known as the DEE-theorem introduced by Desmat et al, 1992 [6]

$$E(i_r) + \sum_{j,j \neq i} \min_s E(i_r j_s) > E(i_t) + \sum_{j,j \neq i} \max_s E(i_t j_s) \quad i \neq j. \quad (12)$$

The theorem states that the rotamer i_r is a dead ending, thus precluding i_r to be a member of the GMEC, if the inequality (12) holds true for any rotamer $i_t \neq i_r$ of the side chain i . For a proof of this theorem we refer to [6]. In words inequality (12) says that rotamer i_r must be a dead ending if the energy of the

system taken at i_r :s most favorable interactions with all the other residues are bigger than the energy of another rotamer (i_t) at its worst interactions. The best and worst interactions are given by “choosing” that rotamer that gives the lowest respectively the highest value for each interaction term. Desmat also described how the criteria could be extended to the elimination of pairs of rotamers inconsistent with the GMEC.

A more powerful version of these criteria was introduced by Goldstein [7]. It subtracts the right-hand side from the left in equation (12) before applying the min operator.

$$E(i_r) - E(i_t) + \sum_{j,j \neq i} \min_s (E(i_r j_s) - E(i_t j_s)) > 0 \quad (13)$$

This criterion says that i_r is a dead ending if we can always lower the energy by taking rotamer i_t instead of i_r while keeping the other rotamers fixed. Goldstein’s criterion can also be extended to pairs of rotamers inconsistent with the GMEC. These criteria are used iteratively and one after each other until no more rotamers can be eliminated. If the method converges to a single solution, this is guaranteed to be the GMEC. In computational time calculation using the rotamer-pairs is significantly more expensive than with single rotamers.

Other improvements of the DEE-method have been done [9, 17]. They have made it possible to eliminate more rotamers and to accelerate the process of elimination. Today DEE based methods can handle most side chain positioning problems [10]. One still reach a limit though, in design problems where this method fails to converge in a reasonable amount of time [10]. This means that the conformational space can not be reduced to a single solution. In this study we have only utilized Goldstein’s criterion for single residues (13) .

4 Methods

The energy function used in this study consists of van der Waals forces between atoms that are more than three bonds away from each other. Van der Waals parameters were taken from CHARMM22 [18] parameter set (par_all22_prot). In this study we have focused on the algorithmic part of the side chain positioning problem, i. e. finding the global minimum energy conformation of the model. The next step would be to use a more appropriate energy function. We used a backbone-independent rotamer library of R. Dunbrack [3] (bbind99.Aug.lib), that contains a maximum of 81 rotamers per residue. Further, all hydrogen atoms were ignored. All calculations have been performed using the backbone coordinates of the lambda repressor (PDB-code: 1r69). The energy contributions of each rotamer pair to the total energy were calculated in advance and stored in a vector c , see (10). The two matrices A and b were constructed. An example of these matrices for a small problem can be seen in Table 1. These matrices were then used as the input to the linear programming algorithms. First a simplex algorithm, lp_solve_3.1 [19], was used. Secondly the problem was solved using a mixed integer programming (*mip*) algorithm from the CPLEX package [20]. This

algorithm is designed to solve problems with both integral and real variables. It makes use of the fact that we have an integer problem, and finds structures in the input that could be used to reduce the problem. After a relaxation of the integer constraints it solves the obtained LP-program by the simplex method. If the solution is not integral a branch and bound procedure takes place. We have also tried other algorithms from the CPLEX package such as primal and dual; simplex, hybrid network and hybrid barrier solvers. The *mip* solver gave the best results.

For comparison we have also performed exhaustive search for up to 10 side chains and implemented a single-residue based DEE-algorithm, using Goldstein’s criterion (13). We also implemented a combination of the DEE-theorem and the linear programming (CPLEX-*mip*). Here we used condition (13) iteratively until no more rotamers could be eliminated and then applied linear programming on the rest of the solution space to find the GMEC.

n_{sc}	\bar{n}_{rot}	Number of conformations	% Remaining rot. after DEE
4	15	$10^{3.8}$	0
5	27	$10^{6.2}$	0
6	24	$10^{7.1}$	0
8	15.7	$10^{8.1}$	4.2
10	16.2	$10^{10.5}$	3.3
12	21	$10^{13.4}$	5.0
13	19.6	$10^{13.8}$	5.0
15	27.8	$10^{17.6}$	3.0
20	28.8	$10^{22.9}$	9.9
27	25.9	$10^{30.5}$	14.3
33	27.5	$10^{38.2}$	17.0
35	28.5	$10^{41.0}$	16.0
43	26.5	$10^{48.1}$	15.4

Table 2. Some of the problems used in the study; n_{sc} is the number of free side chains, \bar{n}_{rot} is the average number of rotamers per side chain. The fraction of rotamers left after the DEE-search and the total number of conformations in the problems are also shown. The protein used is the lambda repressor (PDB-code:1r69)

All algorithms were tested on identical systems, and we assured that the correct solution was found. In Table 2 the size of the conformational space of some of the problems can be seen. As several different programs were used for the calculations, it is not trivial to compare the absolute computational time needed. It is, however, possible to compare their complexity.

5 Results and Discussion

Today there are mainly three types of methods used for the side chain positioning problem, stochastic methods, mean-field algorithms and DEE-theorem based methods. Stochastic methods and mean-field algorithms always find a solution,

however this is not guaranteed to be optimal. If a DEE method converges to one solution, it is guaranteed to be the global optimal solution. However, for larger problems DEE methods do not always converge. Therefore, the remaining solutions have to be tested by e.g. exhaustive search. Here we introduce a novel method using linear programming. If the solution from the LP-method is integral it corresponds to the GMEC.

Integral solutions In our experiments, see below, the solutions were controlled for integrality and so far we have never found a fractional solution. The *mip* method from the CPLEX package is made for integer programs. It uses a simplex solver, and after this branch and bound with simplex if the solution is not integral, see Section 4. In our experiments the branch and bound part of the algorithm was never used. We have also used primal and dual; simplex, hybrid network and hybrid barrier solvers from the CPLEX package. We received integral results from all these solvers. To examine if the energy function has any effect on the integrality of the solutions we have tried different nonsense energy functions on a test case with the simplex algorithm. All the solutions found were integral.

Fig. 2. Experimental studies of the complexity of the two linear programming algorithms versus the number of side chains. The circles represent the simplex method and the crosses the *mip* method. The complexity is approximately $O(n_{sc}^5)$ and $O(n_{sc}^3)$ respectively. For comparison a curve representing the exhaustive search is also included.

Experimental time complexity First the number of free side chains was increased, n_{sc} , to examine the time behavior of the linear programming methods. The computational time for the *mip* method and the simplex method (from

lp_solve_3.1) are shown in Figure 2. It can be seen that the time scales approximately as $O(n_{sc}^3)$ for the *mip* method and as $O(n_{sc}^5)$ for the simplex method. This agrees quite well with the estimation of the complexity, see Section 2 where the computational time was calculated to scale as $O(n_{sc}^6)$ for the simplex method. Further, it shows the superiority of the *mip*-method. The reason that the *mip*-method has a better complexity is due to the preprocessor, which finds structures in the input that can reduce the problem size and increase the speed. This means that there probably exists better ways to implement the conditions of (8).

In protein design the average number of rotamers for each side chain, n_{rot} , is often very large. Therefore, it is interesting to study the complexity of the LP-algorithms versus n_{rot} . Our estimation of the time complexity for the simplex method on the side chain positioning problem is $O(n_{rot}^3)$, see Section 2. This agrees quite well with our experimental study, see Figure 3, where the complexity is approximately $O(n_{rot}^{2.2})$ both for the simplex and *mip* algorithms.

Fig. 3. Experimental studies of the complexity of the two linear programming algorithms used in this study versus the number of rotamers. The number of side chains is 8. The circles represent the simplex method, the crosses the *mip* method and the stars a DEE-algorithm with a final exhaustive search, see methods. For a problem of this size the DEE-algorithm almost converges. Therefore, the computational time of the exhaustive search is negligible. The complexity is approximately $O(n_{rot}^{2.2})$ (*mip*), $O(n_{rot}^{2.3})$ (simplex) and $O(n_{rot}^{2.0})$ (DEE).

Comparison with the DEE-method The DEE-method is a pruning method that consecutively eliminate parts of the conformation space. First, simple criteria with a low complexity are used and then slower methods are applied until convergence. For large design problems DEE-methods do not converge in a reasonable amount of time [10]. This means that the time complexity for large

system could be exponential and not polynomial. All this makes it difficult to calculate an overall time complexity of DEE-methods. In a study by Pierce et al [17], the time complexity of DEE was estimated in terms of the cost of the nested loops required to implement each approach. For the different criteria their estimation was between $O(n_{sc}^2 n_{rot}^2)$ and $O(n_{sc}^3 n_{rot}^5)$.

Fig. 4. Experimental studies of the complexity of the *mip* linear programming algorithm and the Dead End Elimination algorithm, versus the number of side chains. The circles represent the linear programming *mip* method alone, the crosses a combination of DEE and LP, the stars the DEE with a final exhaustive search and the '+'-signs the DEE-algorithm alone. The complexity for *mip* and the combined methods are approximately $O(n_{sc}^3)$ and $O(n_{sc}^{3.5})$ respectively. The complexity of the DEE part of the computation is $O(n_{sc}^{3.2})$, the dashed line.

To perform a comparison between DEE and LP a part of the DEE method, Goldstein's criterion for single residues was implemented. However, we did not implement other criteria. The comparison of the two methods can therefore only be seen as an indication of their relative performance.

In our implementation, where the DEE did not converge to one solution, the remaining conformational space was searched exhaustively. When the problem contained more than approximately 8 residues the DEE algorithm did not converge. In Figure 4, it can be seen that for larger systems our DEE implementation does not show a polynomial complexity. With a more complete DEE-method it ought to be possible to limit the remaining conformational space so that a single solution is obtained for much larger systems. However, the complexity for the DEE algorithm would then be larger.

What might be more interesting is to compare the DEE implementation with the *mip*-linear programming method. If one only consider the complexity of the DEE-algorithm before the exhaustive search is started, the complexity is approximately equal to the LP-method, see Figure 4. However, while the

LP-method has found the GMEC, the DEE-algorithm has not converge to a single solution for most problems, Table 2. In Figure 4, the time complexity of a combination of this DEE-algorithm and the LP-algorithm *mip* is also shown. Here, the time for the combined method is less than for *mip* alone, but the complexity is a little bit better for *mip*.

We have also made a study of the DEE-algorithm (Goldstein’s singles criterion) with an increasing number of rotamers, see Figure 3. Here the problem was small enough (8 residues) for the DEE-method to eliminate almost all rotamers. The complexity of the DEE-algorithm was $O(n_{rot}^2)$, i.e. almost identical with the LP-methods. However, the DEE-algorithm was faster.

6 Conclusions

Here, we have introduced a novel solution method to the problem of optimal side chain positioning on a fixed backbone. It has earlier been shown that finding the optimal solution to this problem is essential for the success of automatic protein designs [1]. The state of the art methods include several rounds using different versions of the Dead End Elimination theorem, with an increasing time complexity. This method do not necessarily converge to a single solution but the conformational space is reduced significantly and the remaining solutions can be searched.

By using linear programming (LP) we are guaranteed to find an optimal solution in polynomial time. If this solution is integral it corresponds to the global minimum energy conformation. This far in our studies the solutions have always been integral.

Linear programming is a well studied area of research and many algorithms for fast solutions are available. We obtained the best results using the *mip*-method from the CPLEX package. The time complexity for the *mip*-method to find the GMEC was $O(n_{sc}^3 n_{rot}^{2.2})$, while our DEE implementation (Goldstein’s singles criterion), had a similar complexity of $O(n_{sc}^3 n_{rot}^2)$. However, for the *mip*-method the GMEC was found while for the DEE-method a fraction of the conformation space remained to be searched. More advanced DEE implementations converge to a single solution for larger problems, but they use more time consuming criteria, the worst with an estimated complexity of $O(n_{sc}^3 n_{rot}^5)$ [17]. As the complexity for the *mip*-method has a smaller dependency on the number of rotamers, the use of LP-algorithms might be best for problems with many rotamers, as in protein design. One reason for the effectiveness of the *mip*-algorithm is most likely due to the preprocessing of the problem. Therefore, a reformulation of the input matrices (10) to the simplex algorithm, perhaps as a network [21], might improve the complexity even further.

7 Acknowledgements

We thank Jens Lagergren for the idea of using linear programming and helpful discussions. This project was supported in part by Swedish Natural Science Research Council (NFR) and the Strategic Research Foundation (SSF).

References

1. Dahiyat, B.I. & Mayo, S.L.: De novo protein design: fully automated sequence selection. *Science* **278** (1997) 82–87
2. Ponder, J.W. & Richards, F.M.: Tertiary Templates for Proteins: Use of Packing Criteria in the Enumeration of Allowed Sequences for Different Structural Classes *J. Mol. Biol.* **193** (1987) 775–791
3. Dunbrack, R. & Karplus, M.: Backbone-dependent rotamer library for proteins - application to sidechain prediction. *J. Mol. Biol.* **230** (1993) 543–574
4. Metropolis, N., Metropolis, A.W., Rosenbluth, M.N. & Teller, A.H.: Equation of state computing on fast computing machines. *J. Chem. Phys.* **21** (1953) 1087–1092
5. LeGrand, S.M. & Mers Jr, K.M. The genetic algorithm and the conformational search of polypeptides and proteins. *J. Mol. Simulation* **13** (1994) 299–320
6. Desmet, J., De Maeyer, M., Hazes, B. & Lasters, I.: The dead end elimination theorem and its use in side-chain positioning. *Nature* **356** (1992) 539–542
7. Goldstein, R.F.: Efficient rotamer elimination applied to protein side-chains and related spin glasses. *Biophysical J.* **66** (1994) 1335–1340
8. Lasters, I., De Maeyer, M. & Desmet J.: Enhanced dead-end elimination in the search for the global minimum energy conformation of a collection of protein side chains. *Protein Eng.* **8** (1995) 815–822
9. Gordon, D.B. & Mayo, S.L.: Radical performance enhancements for combinatorial optimization algorithms based on the dead-end elimination theorem. *J. Comp. Chem.* **19** (1998) 1505–1514
10. Voigt, C.A., Gordon, D.B. & Mayo, S.L.: Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design. *J. Mol. Biol.* **299** (2000) 789–803
11. Koehl, P. & Levitt, M.: De Novo Protein Design. I. In search of stability and specificity. *J. Mol. Biol.* **293** (1999) 1161–1181
12. Lueneberger, D.G.: *Linear and nonlinear programming*. Addison-Wesley publishing company (1973)
13. Nemhauser, G.L. & Wolsey, L.A.: *Integer and Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization (1988)
14. Korte, B. & Vygen, J.: *Combinatorial optimization. Theory and algorithms* Springer-Verlag (1991)
15. Lindeberg, P.O.: *Opimeringslارا: en introduction*. Department of Optimization and Systems Theory, Royal institute of Technology, Stockholm (1990)
16. Karmarkar, N.: A New Polynomial Time Algorithm for Linear Programming. *Combinatorica* **4** (1984) 375–395
17. Pierce, N.A., Spriet, J.A., Desmet, J. & Mayo, S.L.: Conformational Splitting: A More Powerful criterion for Dead-End Elimination. *J. Comp. Chem.* **21** (2000) 999–1009
18. Brooks, B.R., Brucoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S. & Karplus, M.: CHARMM: A program for Macromolecular Energy, Minimization and Dynamics Calculations. *Journal of Computational Chemistry* **4** (1983) 187–217
19. Berkelaar, M: lp_solve_3.1: Program with the Simplex algorithm. ftp://ftp.es.ele.tue.nl/pub/lp_solve (1996)
20. www.cplex.com
21. Rockafellar, R.T.: *Network Flows and Monotropic Optimization*. Wiley Interscience (1984)

A Implementation of the integer program

To transform the integer program describing the side chain positioning problem (8) into the following form

$$z_{IP} = \min\{cx \mid Ax = b, x \in \{0, 1\}^n\} \quad (14)$$

we have formulated the condition (6) as

$$\begin{aligned} \sum_s x_{i_r, j_s} - \sum_t x_{i_r, (j+1)_t} &= 0, \quad i = 1 \dots (n_{sc} - 2), j = 2 \dots (n_{sc} - 1), i < j \\ \sum_r x_{i_r, j_s} - \sum_t x_{(i+1)_t, j_s} &= 0, \quad i = 1 \dots (n_{sc} - 2), j = 3 \dots n_{sc}, (i + 1) < j \\ \sum_r x_{i_r, j_s} - \sum_t x_{j_s, (i+2)_t} &= 0, \quad i = 1 \dots (n_{sc} - 2), j = i + 1. \end{aligned}$$