Automated bond order assignment as an optimization problem

Anna Katharina Dehof1,∗, Alexander Rurainski1, Quang Bao Anh Bui2, Sebastian Böcker2, Hans-Peter Lenhof1 and Andreas Hildebrandt1

1Center for Bioinformatics, Saarland University, 66041 Saarbrücken and 2Chair for Bioinformatics, Friedrich-Schiller-University Jena, 07743 Jena, Germany

ABSTRACT

Motivation: Numerous applications in Computational Biology process molecular structures and hence strongly rely not only on correct atomic coordinates but also on correct bond order information. For proteins and nucleic acids, bond orders can be easily deduced but this does not hold for other types of molecules like ligands. For ligands, bond order information is not always provided in molecular databases and thus a variety of approaches tackling this problem have been developed. In this work, we extend an ansatz proposed by Wang et al. that assigns connectivity-based penalty scores and tries to heuristically approximate its optimum. In this work, we present three efficient and exact solvers for the problem replacing the heuristic approximation scheme of the original approach: an A∗, an ILP and an fixed-parameter approach (FPT) approach.

Results: We implemented and evaluated the original implementation, our A∗, ILP and FPT formulation on the MMFF94 validation suite and the KEGG Drug database. We show the benefit of computing exact solutions of the penalty minimization problem and the additional gain when computing all optimal (or even suboptimal) solutions. We close with a detailed comparison of our methods.

Availability: The A∗ and ILP solution are integrated into the opensource C++ LGPL library BALL and the molecular visualization and modelling tool BALLView and can be downloaded from our homepage www.ball-project.org. The FPT implementation can be downloaded from http://bio.informatik.uni-jena.de/software/

Contact: anna.dehof@bioinf.uni-sb.de

Supplementary information: Supplementary data are available at Bioinformatics online.

Received on August 12, 2010; revised on December 7, 2010; accepted on December 22, 2010

1 INTRODUCTION

Correct bond order information is essential for many algorithms in Computational Structural Biology and Theoretical Chemistry, since bonds do not only define the connectivity of atoms in a molecule but also define structural aspects like rotatability of individual parts. However, bond order information can often not be directly inferred from the available experimental data. Even important molecular databases, like the Protein Data Bank (PDB) (Berman et al., 2003) and the Cambridge Structural Database (Allen, 2002), are known to contain erroneous data for connectivity and bond order information (Labute, 2005) or to even omit them entirely. For proteins and nucleic acids, bond orders can be easily deduced due to their building block nature, but this does not hold for other kinds of molecules like ligands. The problem is made much worse by the fact that quite often, the bond-order assignment for a given molecule is not unique, even when neglecting symmetries in the molecule.

The chemical reasons for this effect are complex and out of scope of this work; here we just want to state that the concept of integer bond orders is only an approximation to a full quantum chemical treatment, and cannot explain all effects occurring in molecules. Important examples are aromatic or delocalized bonds, leading to important resonance structures (cf. Fig. 1). In addition, formal charges are often not contained in the input files, but atoms carrying a formal charge will obviously show a different bonding pattern.

One body of opinion tries to overcome these obstacles by hand curation, which clearly provides the highest reliability. On the other hand, manual data curation does not scale well to large numbers of molecules, and it does not help in conditions where modifications are systematically applied to molecules, e.g. in computational combinatorial chemistry.

In the past decades, the problem of assigning bond orders automatically has been addressed by a number of different approaches. Early methods in the field strongly rely on the correctness of atomic coordinates and focus on reference bond lengths and valence angles (Baber and Hodgkin, 1992) or additionally consider functional group detection (Hendlich et al., 1997) and further molecular features like hybridization states and charges (Lang et al., 1992; van Aalten et al., 1996; Zhao et al., 2007). The main drawbacks of these approaches are the dependence on correct atomic coordinates and the algorithms’ heuristic nature.

In contrast, exact solvers proposed previously represent the bond order assignment problem as a Maximum Weighted Matching for non-bipartite graphs (Labute, 2005) or as an integer linear programming problem that generates valid Lewis structures (electron dot structures) with minimal formal charge on each atom (Froeyen and Herdewijn, 2005).

Recently, Wang et al. (2006) have presented an elegant novel approach to the problem, which is implemented in the established Antechamber package, a suite of tools used for the preparation of input structures for molecular mechanics studies. In this approach, a chemically motivated, expert generated penalty function is used to score bond order assignments. This function is then heuristically optimized. However, this procedure has two drawbacks: the score of a resulting assignment is not guaranteed to be optimal and the algorithm provides only one solution while there can be more than one assignment with optimal score. Figure 1 exemplarily
of a molecule now yields the total penalty score \( tps = \sum_{i=1}^{n} apsi \) where \( n \) denotes the number of atoms. The smaller the \( tps \) of a given bond order assignment, the more reasonable it is. Unfortunately, this problem is NP-hard (Böcker et al., 2009). In Wang et al. (2006), minimization now proceeds in a heuristic and greedy manner.

3 METHODS

3.1 Integer linear program (ILP)

To compute a bond order assignment with guaranteed globally minimal \( tps \), we formulated the problem as an ILP (Papadimitriou and Steiglitz, 1998) as described below.

\[ \min \sum_{s=1}^{N} \sum_{i=1}^{n} P_{s,i}(b(i)) \cdot y_{s} \]

To ensure that each atom is assigned exactly one valence state, we add the additional linear constraints \( \sum_{s=1}^{N} y_{s} = 1 \) for all \( a \in A \). In addition, we have to ensure that the sum of its bond orders equals its chosen valence. These constraints can be formulated as

\[ \sum_{v \in V(i)} y_{s,v} \cdot \sum_{b \in B(i)} k_{v,b} \cdot y_{s} = \sum_{v \in V(i)} y_{s,v} \cdot \sum_{b \in B(i)} k_{v,b} \cdot y_{s} \]

for all \( a \in A \), because the left-hand side evaluates to valence \( v \) if and only if \( y_{s,v} = 1 \). The full resulting ILP can be found in the Supplementary Material. Additional solutions can be found if for each bond order assignment \( r = (x_{a},y_{s}) \), we add the constraint \( \sum_{s=1}^{N} y_{s} = 1 \) for \( a \in A \). In our experiments, we have seen a drastic increase in running time if more than one solution is computed. Thus, the ILP approach is not well suited for obtaining co-optimal or suboptimal bond order assignments.

3.2 The \( A^* \) approach

In order to be able to efficiently enumerate all feasible solutions—optimal and suboptimal ones—we formulated the bond order assignment problem
as an A* search algorithm. This allows enumeration of all assignments in the order of increasing penalty and hence, for instance, to compare the assignments of all solutions for a given molecule up to a user-defined penalty threshold. In addition, such an A* algorithm is simpler to implement, and often easier to extend, than an ILP approach, for instance, it is easily possible to influence the order in which solutions with equal score are computed.

As a combinatorial optimization problem, the bond order assignment problem can be represented by a tree, where each layer stands for one of the decisions that have to be made. In our case, the tree has $k$ layers, where $k$ is the number of bonds that have to be assigned. A node at layer $i$ has $\mu_i$ children, where $\mu_i$ is the number of possible bond orders, typically 3, and each edge is labeled with its corresponding order. Hence, by tracing the path from the root to a node at layer $i$, we can determine the values of the first $i$ bonds in this particular partial assignment represented by the node $w$. Thus, the root node corresponds to a completely unassigned molecule with only unknown bond orders, while the leave nodes correspond to complete bond order assignments. If we only add child nodes and if the resulting valence state is valid, the leaf nodes correspond to the feasible bond order combinations. In order to discriminate between the different combinations, each leaf is assigned its total penalty score.

Visiting all nodes in the tree, the optimal bond order assignment can be found in a brute-force manner with exponential running time. If, additionally, all intermediate nodes are assigned the atomic penalty score of the partial bond order assignment they represent, a greedy search will yield an assignment with heuristically good (but not necessary optimal) total penalty score, ensuring that the first leaf reached is optimal (roughly speaking, if all intermediate nodes are assigned the atomic penalty score of the partial bond order assignment they represent, a greedy search will yield an upper bound of the atomic valence of an atom $a$). The maximum order of an unassigned bond with respect to atom $a$ is given by

$$l(a) = v_a(a) + \sum_{b \in R(a)} l = v_a(a) + \lceil B(a) \rceil$$

Thus, we can formulate a tighter search heuristic by

$$h^*(a) = \sum_{b \in A(a) \cup B(a)} \min_{p \in B(a)} \{ P(a,b) \}$$

An even tighter version of the search heuristic would also take the already assigned bond orders of the neighboring atoms of $a$ into account. The maximum order of an unassigned bond with respect to atom $a$ is given by

$$t(a) = \max\{V(a), l(a) + 1\}$$

Denoting by $a_1, a_2$ the atoms connected by an unassigned bond $b$, its maximum bond order equals

$$b_{\max}(b) = \min\{t(a_1), t(a_2)\},$$

yielding an upper bound of the atomic valence of an atom $a$

$$a_{\text{up}}(a) = \max\{V(a), t(a) + \sum_{b \in B(a)} b_{\max}(b)\}.$$ 

Thus, a tighter version of the search heuristic is given by:

$$h^*(a) = \sum_{b \in A(a) \cup B(a)} \min_{p \in B(a)} \{ P(a,b) \}$$

The function $g^*$ sums the atomic penalties of all completely assigned atoms in the partial bond order assignment represented by node $w$, whereas $h^*$ considers all atoms with at least one bond of unassigned bond order. For the atoms in this set, we compute the minimal atomic penalty possible under the current partial assignment independently of the other atoms in the set. Each atom can choose its preferred value for each unassigned bond without considering overall consistency. Obviously, $h^*$ is optimistic. All three heuristics are implemented in our code.

### 3.3 The fixed-parameter approach (FPT)

In this approach, we consider each molecule as a molecule graph $G = (U, E)$, where each vertex represents an atom and each edge represents a bond. A molecule graph is called a tree if it is connected and has no cycles. The bond order assignment problem can be solved in polynomial time using dynamic programming. We omit the details, and concentrate on the more general case of graphs that are tree-like. A tree decomposition of a graph $G = (U, E)$ consists of an index set $I$, a set of bags $X_i \subseteq U$ for $i \in I$ and a tree $T$ with node set $I$ such that:

1. every vertex $a \in U$ is contained in at least one bag $X_i$.
2. for every edge $\{u, v\} \in E$, there is at least one bag $X_i$ such that $u, v \in X_i$.
3. for two nodes $i, j$ of the tree $T$, if $a \in X_i$ and $a \in X_j$, then $a \in X_k$ for all nodes $k$ of the path from $i$ to $j$ in $T$.

The width of this tree decomposition equals $\omega - 1$ for $\omega = \max\{|X_i| - 1\}$. The treewidth of $G$ is the minimum width of any tree decomposition of $G$. The treewidth of a tree equals one.

Given a molecule graph $G$, we first compute a tree decomposition of $G$. We will see below that the running time and the required space of our algorithm grow exponentially with the width of the decomposition. Unfortunately, computing a tree decomposition with minimum width is again an NP-hard problem (Arnborg et al., 1987). Fortunately, there exist heuristic and exact algorithms to compute such tree decompositions efficiently in practice (Bodlaender et al., 2006; Gogate and Dechter, 2004).

To simplify the description of our algorithm, we use nice tree decompositions. Here, we assume the tree $T$ to be rooted. A nice tree decomposition is a tree decomposition satisfying:
A.K. Dehof et al.

Fig. 2. A graph (top left) with a tree decomposition (bottom left) and a corresponding nice tree decomposition (right). Dashed lines illustrate connected components sharing common vertices.

(1) Every node of T has at most two children.
(2) If a node i has two children j and k, then \(X_i = X_j \cup X_k\); in this case, i is called a join node.
(3) If a node i has one child j, then one of the following two conditions must hold:
   (a) \(|X_i| = |X_j| + 1\) and \(X_i \subseteq X_j\); in this case, X_i is called an introduce node.
   (b) \(|X_i| = |X_j| - 1\) and \(X_i \subseteq X_j\); in this case, X_i is called a forget node.

Here, introduce nodes and forget nodes are viewed as moving bottom-up from the leaves to the root. We can easily transform a tree decomposition into a nice tree decomposition, in time linear in the size of the tree decomposition.

Figure 2 illustrates a tree decomposition and a corresponding nice tree decomposition of a graph. It can be easily verified that the union of all bags in the tree decomposition as well as all bags in the nice tree decomposition contains every vertex of the graph, and every edge of the graph exists in at least one bag of the tree decompositions. Furthermore, all bags sharing a common vertex induce a connected subgraph in the tree decomposition.

The tree T is rooted at an arbitrary bag. Above this root, we add additional forget nodes, such that the new root contains a single vertex. Let \(X_1\) denote the atoms in the molecule graph \(G\) and \(X_2\) denote the single vertex contained in \(X_1\). Analogously, we add additional introduce nodes under every leaf of \(T\), such that the new leaf also contains a single vertex. Let \(X_j = \{a_1, a_2, \ldots, a_l\}\) be the atoms inside bag \(X_j\), where \(l \geq 0\).

In our presentation below, we want to avoid double indices, so we refer to the atoms inside bag \(X_j\) as \(a_1, a_2, \ldots, a_l\). It should be understood that these are different atoms for each bag. For simplicity of presentation, we also assume that the molecular graph induced by \(a_1, a_2, \ldots, a_l\) is fully connected and, thus, contains all bonds \(a_1a_2, a_1a_3, \ldots, a_{l-1}a_l\).

Let \(Y_i\) denote the atoms in the molecular graph \(G\) that are contained in the bags of the subtree of \(T\) below bag \(X_i\). To save memory in the dynamic programming below, we will not use the bond order \(b_{ik}\) between atoms \(a_i, a_j\) but instead, the free bond order \(b_{ik} = b_{ij} - 1\ in [0, 1, 2]\). Then, the valence of an atom is the sum of free bond orders over all incident bonds, plus the degree of the atom in the molecule graph. We assign a score matrix \(D_i\) to each bag \(X_i\) of the tree decomposition: let \(D_i(v_1, v_2, \ldots, v_l)\) be the minimum score over all valence assignments to the vertices in \(X_i \subseteq X_j\) if for every \(l = 1, \ldots, k\), \(v_k\) valences of atom \(a_l\) have been consumed by the atoms in \(Y_i \subseteq X_i\), and free bond orders \(b_{ij} = b_{ij} - 1\) are assigned to bonds \(a_{j1}a_{j2}, a_{j3}a_{j4}, \ldots, a_{jl}a_{jl+1}\). Using this definition, we delay the scoring of any vertex to the forget node where it is removed from a bag. We can compute the minimum score among all assignments using the root bag \(X_1 = \{a_1\}\) as

\[
\min_{v_1, \ldots, v_l} [D_0(v, v) + D_i(v_1)] = \min_{v_1, \ldots, v_l} [D_0(v, v) + D_i(v_1)].
\]

Our algorithm begins at the leaves of the tree decomposition and computes the score matrix \(D_i\) for every node \(X_i\) when score matrices of its children nodes have been computed. We initialize the matrix \(D_i\) of each leaf \(X_i = \{a_1\}\) with \(D_i(v_1) = 0\) if \(v_1 = 0\), and \(D_i(v_1) = \infty\ otherwise.\)

During the bottom-up traversal, the algorithm distinguishes if \(X_i\) is a forget node, an introduce node or a join node, and computes \(D_i\) as follows:

**Introduce nodes:** let \(X_i\) be the parent node of \(X_j\) such that \(X_j = \{a_1, a_2, \ldots, a_l\}\). Then

\[
D_i(v_1, v_2, \ldots, v_l) = \begin{cases} 
D_j(v_1, v_2, \ldots, v_l) & \text{if } v_1 = 0 \\
\infty & \text{otherwise.}
\end{cases}
\]

**Forget nodes:** let \(X_i\) be the parent node of \(X_j\) such that \(X_j = \{a_1, a_2, \ldots, a_l\}\). Then

\[
D_i(v_1, v_2, \ldots, v_l) = \min_{v_1, \ldots, v_l} \left\{ D_j(v_1, v_2, \ldots, v_l) + \min_{b_{ik} = 0} \left[ \frac{1}{2} \sum b_{ik} \right] \right\}
\]

where \(\min_{b_{ik} = 0} \left[ \frac{1}{2} \sum b_{ik} \right] \) denotes the degree of vertex \(a_k\).

**Join nodes:** let \(X_i\) be the parent node of \(X_j\) and \(X_k\) such that \(X_j = X_k = X_i\). Then

\[
D_i(v_1, v_2, \ldots, v_l) = \min_{v_1, \ldots, v_l} \left\{ D_j(v_1, v_2, \ldots, v_l) + D_k(v_1, v_2, \ldots, v_l) \right\}
\]

For simplicity of the presentation of our algorithm, we assumed above that every two vertices in each bag of the tree decomposition are connected by an edge, but in reality, the degree of a vertex in a molecule graph cannot exceed the maximum valence \(d\). In our presentation, the number of edges in a bag is upper bounded by \(\omega\). Given a nice tree decomposition of a molecule graph \(G\), the algorithm described above computes an optimal assignment for the bond order assignment problem on \(G\) in time \(O(2^n \cdot 3^d \cdot \omega n)\), where \(n = 1 + \max_{a} \max V(a)\) is the maximum (open) valence of an atom plus one, \(m = 1 + \omega \) and \(d = 1 + \max \omega = 1\) is size and width of the tree decomposition, \(d\) is the maximum degree in the molecule graph, and \(\beta = \min \{\frac{1}{2} \sum b_{ik} \} \) (Böcker et al., 2009).

We implemented our algorithm in Java and used the method QuickBB in the library LibTW implemented by van Dijk et al. (http://www.treewidth. com) to compute the optimal tree decomposition of a molecule graph. After computing the optimal tree decomposition, we transformed it into a nice tree decomposition. Running times reported for the fixed-parameter approach (FPT) algorithm include the running times of computing the optimal nice tree decomposition. To save memory, we use hash maps instead of arrays to implement score matrices \(D\). During the course of the dynamic programming algorithm, we do not have to compute or store entries \(D(v_1, v_2, \ldots, v_l)\) with \(v_1 + \sum b_{ik} > \max V(a)\) for some \(i\), because such entries will never lead to a feasible bond order assignment. Furthermore, we find that the following trick speeds up our algorithm in practice: we initialize an integer \(k = 0\) and do not store matrix entries with score exceeding \(\beta\). If the score of the optimal solution is at most \(k\), this optimal solution will be found. Otherwise, we call our algorithm again with increasing \(k\), until an optimal solution is found. If not only the optimal solutions but also a certain number of suboptimal solutions are required, we call our algorithm repeatedly with increasing \(k\), until all required suboptimal solutions are found.
Antechamber approach. In Section 4.2, we compare the results of the
et al. and KEGG Drug (Goto
perception.
3D positions that we found very reasonable for testing bond order
was originally designed to test the MMFF94 force field parameters,
orders, adding hydrogens where valences had to be completed, and
by the authors of the MMFF94 force field by assigning bond
fulfilling those constraints, we chose the MMFF94 validation suite
diagrams or SMILES expressions. To provide a diverse test dataset
contain 3D ligand structures as well as those only storing structure
pattern of single and double bonds. In contrast to structure-based
molecules from the KEGG drug set.
In Section 4.1, we compare the total penalty score
tps of the results of our exact solvers with that of the results of the original Antechamber approach. In Section 4.2, we compare the results of the different approaches to the expert generated, hand-crafted reference assignments and study the implications of the ambiguity of two or more co-optimal solutions.

4 DISCUSSION
For proteins and DNA, bond orders can be simply inferred by
of the authors of the MMFF94 force field by assigning bond
orders, adding hydrogens where valences had to be completed, and
minimizing the resulting complexes. The MMFF94 validation suite
was originally designed to test the MMFF94 force field parameters,
and thus yields a diverse set of molecules with hand-curated
connectivity information, hydrogens and bond order assignment and
3D positions that we found very reasonable for testing bond order
topology alone. Unfortunately, hydrogens are missing in the KEGG
data bases, and were added for our tests using standard rules for
completing free valences as performed by OpenBabel (Guha et al.,
2006). Furthermore, 2550 files of the KEGG Drug set contain more
than one molecule, and each molecule may appear in more than one
file. To prevent a skewed analysis, we split up the dataset into
unique connected components. Ignoring molecules with less than
four atoms (e.g. water), this preparation led to a test set of 7424
molecules from the KEGG drug set.

In Section 4.1, we compare the total penalty score
tps of the results of our exact solvers with that of the results of the original Antechamber approach. In Section 4.2, we compare the results of the different approaches to the expert generated, hand-crafted reference assignments and study the implications of the ambiguity of two or more co-optimal solutions.

All algorithms—A*, ILP, FPT and Antechamber—are applied to the
two test sets such as MMFF94 and KEGG Drug. Computing all
optimal solutions for all 761 molecules of the MMFF94 dataset,
the total running time was 252.0 s for the ILP, 227.1 s for the
A* algorithm and 24.9 s for the FPT algorithm. The antechamber
heuristic took 7.9 s to compute one solution for all molecules
(cf. Supplementary Material). All reported running times were
averaged over 20 repetitions. Thus, the ability to provide all optimal
exact solutions and to use user-editable SMARTS strings for penalty
class assignment takes its toll: the heuristic antechamber approach
is the fastest of the methods, about an order of magnitude faster than
ILP and A*.
Still, all running times are sufficiently small to allow
the routine usage in high-throughput applications.

4.1 Comparison to Antechamber
In order to evaluate whether solving the optimization exactly makes
a difference in practice, we first focus on the following properties:

(1) how often do manual, heuristic and exact approaches produce
an optimally scored solution;
(2) how often do the exact approaches find a solution with a
smaller tps than the heuristic;
(3) how often does each approach fail to find a feasible solution.

Evaluation on the MMFF94 validation suite (761 molecules in total) was done as follows: the Antechamber bond perception
algorithm as well as our own algorithms—A*, ILP and FPT—were
run for each input molecule. Note that all exact algorithms will
in principle compute the same solutions, and only the order of co-optimal solutions can differ. If both Antechamber and our algorithms
computed bond order assignments (i.e. none of the approaches
failed), we compared these to test if the Antechamber assignment
is optimal.

For 734 molecules (96.45%), the solution found by the heuristic
Antechamber approach is optimal. For nine molecules (1.18%), the
exact algorithms indeed find bond order assignments with a total
penalty score less than that of the solution provided by Antechamber
(cf. the Supplementary Material). For 14 cases (1.83%), our
algorithms computed an optimal bond order assignment, whereas
the heuristic Antechamber bailed out. In four cases (0.53%),
neither Antechamber nor our algorithms computed a bond order
assignment, due to missing atom types in the penalty table. In no
case, Antechamber computed a solution but our algorithms did not.
In total, Antechamber bailed out in 18 cases (2.30%), and in 23
cases (3.02%) we improved upon Antechamber (no solution by
Antechamber or better solution by our algorithms).

The comparison of our algorithms to the Antechamber approach
on the KEGG Drug set (7424 molecules in total) looks very similar.
For 7202 molecules (97.01%), the bond order assignment found
by Antechamber is optimal. For 13 molecules (0.18%) containing
PO₄, Antechamber reproducibly provided infeasible solutions,
whereas our algorithms computed optimal assignments. For 27
cases (0.36%), our algorithms computed an optimal assignment but
Antechamber bailed out. In 180 cases (2.42%), both approaches
bailed out, as not all atom types are contained in the original
penalty table given in Wang et al. (2006). In total, Antechamber bailed out
in 207 cases (2.79%), and we improved upon Antechamber in 40
cases (0.54%).
As a second step in the analysis, we compare the results produced by all approaches to the reference assignment. For our own solvers, which are able to enumerate all optimal (FTP, ILP) or even all feasible solutions (A*), we only recorded the first one.

As can be seen in Table 2, our methods are able to significantly reproduce more bond order assignments of the MMFF94 validation suite than the original Antechamber approach. While Antechamber correctly recomputed 37.05% of the molecules, the exact methods reconstructed between 53.88% and 61.89% of the reference bond order assignments as the first solution. Similar results can be seen on the KEGG Drug set: Antechamber correctly reproduced 41.96% of the bond order assignments, compared to 49.95–56.9% for the exact methods. Obviously, all results returned by the exact solvers are optimal and hence, the differences in these numbers are due to systematic differences in the order in which each algorithm enumerates the solutions. In the case of the A* algorithm, this order can easily be tweaked by adapting the heuristic part of the scoring functions. By design, our A* heuristics tend to avoid the occurrences of larger bond orders, but this strategy could be further fine tuned. Note that the FPT algorithm can easily be modified to simulate this behaviour, as computing all optimal solutions does not significantly increase running times. For the ILP approach, in contrast, running times would increase considerably. In future, we plan to sort co-optimal solutions with respect to another objective function before writing the output. This might possibly further increase the quality of our results, and is the topic of ongoing research.

Considering that bond order assignments need not be unique, it makes sense to provide the user not only with the first solution but with all optimal ones (or even some suboptimal ones). In this case, taking all optimal solutions into account, we find that our algorithms find the reference solution in 78.71% of the cases on the MMFF94 validation suite and in 85.21% on the KEGG Drug set. A complete comparison is given in Table 2.

Obviously, the performance of all approaches is limited by the quality of the penalty table: the definition of the atom classes, their allowed valence states, and the choice of the valence state’s penalties have a significant influence on the performance. As can be seen in Table 2, the current penalty table does not cover all molecules in the reference datasets—for four molecules in the MMFF94 set and for 180 in the KEGG set, the required atom classes are missing. Hence, in our own implementations, we use SMARTS expressions stored in an XML file to define the penalty classes, which allows a user to easily add atom types or tune the results to his needs. To guarantee a fair comparison between the solvers, we ensured that for all tests in this article, our implementation used exactly the same penalty classes as Antechamber. Improvements to the penalty table, and a systematic study of their influence, are the focus of future work.

### 5 Conclusion

Automated bond order assignment is an important problem when working with user-generated molecules, molecular databases or computational combinatorial chemistry. Especially fully automated pipelines in high-throughput applications depend on reliable bond order assignments. The modern and extensible approach realized in Antechamber is based on sound chemical principles and has proven to be a very valuable tool. In this work, we have shown three different exact solvers as alternatives to the heuristic approach pursued by Wang et al. (2006): an A* algorithm, an ILP formulation and a fixed parameter approach. While we found in our evaluations that the heuristic solver works surprisingly well—roughly 97% of all cases in our tests—it still can be significantly improved using exact techniques. If we keep in mind that bond order assignments are in many cases non-unique—different resonance structures, for instance, might have the same probability to occur—the ability to systematically enumerate all solutions becomes an invaluable tool.

When bond order assignments are important, it might be worthwhile to enumerate all optimal assignments, run whatever procedure is supposed to work with the results in the next step, and average over the results.

Comparing the three different exact strategies, each of them has its advantages and disadvantages. If computational efficiency is required, the best choice is clearly the FPT, where running times are almost on par with the Antechamber heuristic. The A* algorithm, on the other hand, is even simpler to implement than the heuristic and can be very easily extended through the heuristic cost function. Both approaches can compute co-optimal

### Table 1

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference (%)</th>
<th>Solver reproduces (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antechamber</td>
<td>282 (37.05)</td>
<td>282 (37.05)</td>
</tr>
<tr>
<td>ILP</td>
<td>471 (61.89)</td>
<td></td>
</tr>
<tr>
<td>A*</td>
<td>455 (59.79)</td>
<td>599 (78.71)</td>
</tr>
<tr>
<td>FPT</td>
<td>410 (53.88)</td>
<td></td>
</tr>
<tr>
<td>Antechamber</td>
<td>3115 (41.96)</td>
<td>3115 (41.96)</td>
</tr>
<tr>
<td>ILP</td>
<td>4224 (56.90)</td>
<td></td>
</tr>
<tr>
<td>A*</td>
<td>3708 (49.95)</td>
<td>6326 (85.21)</td>
</tr>
<tr>
<td>FPT</td>
<td>3777 (50.88)</td>
<td></td>
</tr>
</tbody>
</table>

The third column denotes the number of molecules for which the algorithms return the original bond order assignment as first solution, the fourth column the number of molecules for which the algorithms return it at as any of their optimal solutions.

### Table 2

<table>
<thead>
<tr>
<th>Test set</th>
<th>Method</th>
<th>Reference is first solution (%)</th>
<th>Solver reproduces (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMFF94</td>
<td>Antechamber</td>
<td>282 (37.05)</td>
<td>282 (37.05)</td>
</tr>
<tr>
<td></td>
<td>ILP</td>
<td>471 (61.89)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A*</td>
<td>455 (59.79)</td>
<td>599 (78.71)</td>
</tr>
<tr>
<td></td>
<td>FPT</td>
<td>410 (53.88)</td>
<td></td>
</tr>
<tr>
<td>KEGG</td>
<td>Antechamber</td>
<td>3115 (41.96)</td>
<td>3115 (41.96)</td>
</tr>
<tr>
<td></td>
<td>ILP</td>
<td>4224 (56.90)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A*</td>
<td>3708 (49.95)</td>
<td>6326 (85.21)</td>
</tr>
<tr>
<td></td>
<td>FPT</td>
<td>3777 (50.88)</td>
<td></td>
</tr>
</tbody>
</table>
and sub-optimal solutions without significantly increasing running times, and geometric information can be employed to provide a more sensible ordering of the results. The ILP approach, finally, is trivial to implement when external solvers can be used. However, enumerating all solutions requires a certain sophistication and can easily spoil the running time. An additional advantage of our method is their easy extensibility. For example, adding missing hydrogens or even bonds is possible but will require more elaborate, e.g. structure based, scoring to handle the exponential number of combinations. Such a scoring scheme only requires modifications of the \( f(x) \) definition. Algorithmically, the bond order assignment problem bears close resemblance to the side chain optimization problem, where similar solution strategies have been developed [Althaus et al. (2002); Leach and Lemon (1998); Xa et al. (2005)]. Future work will study whether modern probabilistic approaches [see, e.g. Yanover et al. (2008)] for this problem will also be appropriate for bond order assignment.

**ACKNOWLEDGEMENT**

Implementation of the FPT algorithm was done by Kai Duhrkop and Patrik Seебer.

**Funding**: A.H. acknowledges financial support from the Intel Visual Computing Institute (IVCI) of Saarland University; A.H. and H.P.L. financial support from DFG, research group ‘Parameterized Algorithms in Bioinformatics’ (BO 1910/5).

**Conflict of Interest**: none declared.

**REFERENCES**


