BIOINFORMATICS ORIGINAL PAPER

Vol. 27 no. 5 2011, pages 619–625
doi:10.1093/bioinformatics/btq718

Structural bioinformatics

Advance Access publication January 17, 2011

Automated bond order assignment as an optimization problem

Anna Katharina Dehof[1,*], Alexander Rurainski[1], Quang Bao Anh Bui[2], Sebastian Böcker[2], Hans-Peter Lenhof[1] and Andreas Hildebrandt[1]

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

Associate Editor: Burkhard Rost

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 open-source 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 different 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 those 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

*To whom correspondence should be addressed.

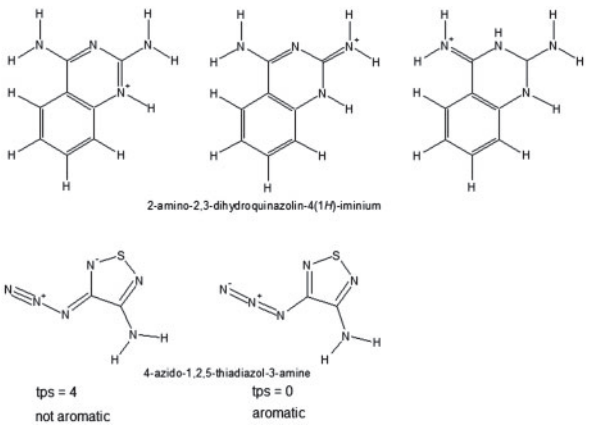

Fig. 1. Top: different co-optimal bond order assignments. In the left structure, both NH_2 groups are connected via a single bond and thus freely rotatable. In the middle and right structure, one group is connected via a double bond. Bottom: heuristic and optimal bond order assignments. The left structure is the solution provided by Antechamber with a *tps* of 4. Its 5-ring does not fulfill the aromaticity criterion of AM1-BCC (Jakalian *et al.*, 2002). The right structure is the solution computed with our exact solvers. Its *tps* is 0 and the 5-ring meets the AM1-BCC aromaticity criterion.

shows two cases where theses two drawbacks may lead to a wrong interpretation of molecular properties (flexibility and aromaticity). Unfortunately, the optimization problem introduced by Wang *et al.* (2006) is NP-hard (Böcker *et al.*, 2009). In this work, we propose three exact approaches that solve the problem to provable global optimality by discrete optimization techniques: we give an integer linear program (IPL) formulation, and an A* approach and a fixed-parameter algorithm for enumerating all optimal or, if desired, all feasible solutions. In addition, our approach can easily be extended to, e.g. include structural information, add missing hydrogens or even missing bonds. The A* and ILP solution are integrated into the opensource C++ LGPL library BALL (Kohlbacher and Lenhof, 2000) (Hildebrandt *et al.*, 2010) and the molecular visualization and modelling tool BALLView (Moll *et al.*, 2006) and can be downloaded from our homepage www.ball-project.org. The FPT implementation can be downloaded from http://bio.informatik.unijena.de/software/.

2 SCORING BOND ORDER ASSIGNMENTS

The idea behind the bond order assignment algorithm proposed in Wang *et al.* (2006) is to cast the problem into a discrete optimization problem. Finding the most probable consistent bond order assignment for a given molecule is addressed by minimizing a total penalty score *tps*, where each atom is assigned an atomic valence *av* that is defined as the sum over all bond orders *bo* of all bonds connected to the atom under consideration: $av = \sum_{i=1}^{con} bo_i$ Here, *con* denotes the number of bonded atoms. The distance of the calculated *av* to the atom's most desirable valence value is measured by the atomic penalty score *aps*: the possible valences of an atom and the corresponding distance penalty scores are stored in a penalty table that uses a rule-based atom type classification and was derived by Wang *et al.* (2006). The sum over all atomic penalty scores of a molecule now yields the total penalty score $tps = \sum_{i=1}^{n} aps_i$ 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.

- A is the set of all atoms of the molecule under consideration.
- $B(a)$ is the set of bonds of atom $a \in A$ and B denotes the set of all bonds of the molecule.
- $V(a) \subset \mathbb{N}$ contains the possible valences of atom $a \in A$.
- $P(a, v)$ is the penalty value for atom $a \in A$ and valence $v \in V(a)$.

Our approach uses two different classes of variables. For each bond $b \in B$, we introduce indicator variables $x_{b_i} \in \{0, 1\}$ symbolizing whether the corresponding bond b is set to the bond order $i \in \{1, \ldots, \mu\}$, where μ is the maximum bond order considered (in the following, we will set μ to 3, allowing single, double and triple bonds). To ensure that each bond $b \in B$ is assigned exactly one bond order, we add the linear constraints $\sum_{i=1}^{\mu} x_{b_i} = 1$ for all $b \in B$. Then the sum over all bond orders of all bonds $b \in B(a)$ can be computed as $\sum_{b \in B(a)} \left(\sum_{i=1}^{\mu} x_{b_i} \cdot i \right)$.

The second class of variables focuses onto the atomic valences: for all atoms a and corresponding possible valences v according to the penalty table P, we introduce indicator variables $y_{a,v} \in \{0, 1\}$. Each $y_{a,v}$ symbolizes whether the corresponding valence is chosen or not, i.e. penalty $P(a, v)$ contributes to the score if and only if $y_{a,v} = 1$. Thus, the objective function of our score minimization problem can be formulated as a linear function in $\mathbf{y}$ with penalty prefactors:

$$\min \sum_{a \in A} \sum_{v \in V(a)} P(a, v) \cdot y_{a,v}.$$

To ensure that each atom is assigned exactly one valence state, we add the additional linear constraints $\sum_{v \in V(a)} y_{a,v} = 1$ 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

$$\sum_{v \in V(a)} y_{a,v} \cdot v = \sum_{b \in B(a)} \left(\sum_{i=1}^{\mu} x_{b_i} \cdot i \right)$$

for all $a \in A$, because the left-hand side evaluates to valence v if and only if $y_{a,v} = 1$. The full resulting ILP can be found in the Supplementary Material. Additional solutions can be found if for each bond order assignment $s = \{x_{b_i} | x_{b_i} = 1\}$ computed so far we add the constraint(s) $\sum_{x_{b_i} \in s} x_{b_i} \leq k \forall s$ where k denotes the number of bonds in the molecule. Please note that we used $x_{b_i} \in \{0, 1\}$ instead of $x_b \in \{1, \ldots, \mu\}$ to model the bond orders. This simplifies the enumeration of additional solutions.

For the solution of ILPs to provable global optimality, several strategies can be chosen, like the popular pure branch and bound approaches or branch and cut methods (Papadimitriou and Steiglitz, 1998). We employed the open source solver lp_solve (http://lpsolve.sourceforge.net), which uses a simplex algorithm-based branch and bound approach (Papadimitriou and Steiglitz, 1998). 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 μ children, where μ 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 w 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 in linear running time. It can be shown that, if at each intermediate node more information is provided, finding an optimal solution can be guaranteed with greatly improved expected running time. This leads to the popular A*-search algorithm (Hart *et al.*, 1968), which employs a search heuristic to guide the algorithm in descending the tree. More formally, the algorithm associates with each node w a function $f(w)=g^*(w)+h^*(w)$, where $g^*(w)$ describes the score corresponding to the decisions already made and $h^*(w)$ is the so-called search heuristic. For the purposes of the A*-search algorithm, the search heuristic must be an admissible estimate of the score of the best leaf that can be reached starting from node w and descending further down the tree. Here, admissible means that it needs to be 'optimistic': for all nodes w, the estimated cost $h^*(w)$ may never be greater than the lowest real cost to reach a goal node. Given the additional information provided by h^*, the A*-search algorithm always expands one of the nodes with the most promising score, ensuring that the first leaf reached is optimal (roughly speaking, if the algorithm would visit a leaf with worse score first, the search-heuristic would have overestimated the penalty of the real optimal solution, which an admissible heuristic never does).

To map the bond order assignment problem formally to an A*-search, we need further notations that are adapted to the partial bond order assignments corresponding to each node w in the search tree. We denote the set of all assigned bonds in the node w by $W(B)$, the assigned bonds connected to atom a in node w by $W(a)$, and the set of atoms for which all bonds are already assigned with a bond order by K. Then, the functions g^* and h^* can be defined by:

$$g^* = \sum_{a\in K} P\big(a, v_w(a)\big) \tag{1}$$

$$h^* = \sum_{a\in A\setminus K} \min_{i\in V(a)} \big\{P\big(a,i\big)\big\} \tag{2}$$

where $V(a)\subset\mathbb{N}$ contains the possible valences of atom $a\in A$ according to the penalty table P. The search heuristic given in (2) is far too optimistic and can be tightened significantly. Thus, we define the bond order of an assigned bond by $bo(b)$. A partial bond order assignment induces a simple lower bound $v_w(a):=\sum_{b\in W(a)} bo(b)$ for the valence of atom a. Assuming a single bond for each unassigned bond of atom a, a tighter lower bound for the valence is given by

$$lo(a):=v_w(a)+\sum_{b\in B(a)\setminus W(a)} 1 = v_w(a)+|B(a)\setminus W(a)|$$

Thus, we can formulate a tighter search heuristic by

$$h^* = \sum_{a\in A\setminus K} \min_{lo(a)\le i\le size(P(a))} \big\{P\big(a,i\big)\big\} \tag{3}$$

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

$$t(a):=\max\{V(a)\}-lo(a)+1$$

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

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

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

$$up(a):=\min\left\{\max\{V(a)\},\quad v_w(a)+\sum_{b\in B(a)\setminus W(a)} bo_{\max}(b)\right\}$$

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

$$h^* = \sum_{a\in A\setminus K} \min_{lo(a)\le i\le up(a)} \big\{P\big(a,i\big)\big\} \tag{4}$$

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. Given a molecule graph that is a tree, 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 $G=(U,E)$ 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 $u\in U$ 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,k of the tree T, if $u\in X_i$ and $u\in X_k$ then $u\in X_j$ also holds for all nodes j of the tree along the path from i to k in T.

The *width* of this tree decomposition equals $\omega-1$ for $\omega:=\max\{|X_i|\,|\,i\in I\}-1$. 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:

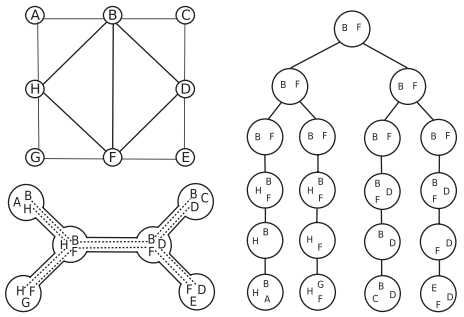

Fig. 2. A graph (top left) with a tree decomposition (bottom left) and a corresponding nice tree decomposition (right). Dashed lines illustrate connected components sharing common vertices.

(1) Every node of T has at most two children.

(2) If a node i has two children j and k, then $X_i = X_j = X_k$; 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 $X_j \subset X_i$; in this case X_i is called an *introduce node*.

   (b) $|X_i| = |X_j| - 1$ and $X_i \subset X_j$; 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_r denote the new root of the tree decomposition and v_r denote the single vertex contained in X_r. Analogously, we add additional introduce nodes under every leaf of T, such that the new leaf also contains a single vertex. Let $X_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,k}\}$ be the atoms inside bag X_i, where $k \leq \omega$. In our presentation below, we want to avoid double indices, so we refer to the atoms inside bag X_i as $a_1, a_2, \ldots, a_k$. It should be understood that these are different atoms for each bag. For simplicity of presentation, we also assume that the molecular subgraph induced by $a_1, a_2, \ldots, a_k$ is fully connected and, thus, contains all bonds $a_1a_2, a_1a_3, \ldots, a_{k-1}a_k$.

Let Y_i denote the atoms in the molecule 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 $\tilde{b}_{i,j}$ between atoms a_i, a_j but instead, the *free bond order* $b_{i,j} := \tilde{b}_{i,j} - 1 \in \{0, 1, 2\}$. 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 $D_i[v_1, \ldots, v_k; b_{1,2}, \ldots, b_{k-1,k}]$ be the minimum score over all valence assignments to the vertices in $Y_i \setminus X_i$ if for every $l = 1, \ldots, k$, v_l valences of atom a_l have been consumed by the atoms in $Y_i \setminus X_i$, and free bond orders $b_{1,2}, \ldots, b_{k-1,k}$ are assigned to bonds $a_1a_2, a_1a_3, \ldots, a_{k-1}a_k$. 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 $X_r = \{a_r\}$ as $\min_{v_r \in V(a_r)} \{P(a_r, v_r) + D_r[v_r]\}$.

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_j of each leaf $X_j = \{a_1\}$ with $D_j[v_1; \cdot] = 0$ if $v_1 = 0$, and $D_j[v_1; \cdot] = \infty$ 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 $X_j = \{a_1, \ldots, a_{k-1}\}$ and $X_i = \{a_1, \ldots, a_k\}$. Then

$$D_i[v_1, \ldots, v_k; b_{1,2}, \ldots, b_{k-1,k}] = \begin{cases} D_j[v_1, \ldots, v_{k-1}; b_{1,2}, \ldots, b_{k-2,k-1}] & \text{if } v_k = 0, \\ \infty & \text{otherwise.} \end{cases} \quad (5)$$

Forget nodes: let X_i be the parent node of X_j such that $X_j = \{a_1, \ldots, a_k\}$ and $X_i = \{a_1, \ldots, a_{k-1}\}$. Then

$$D_i[v_1, \ldots, v_{k-1}; b_{1,2}, \ldots, b_{k-2,k-1}] = \min_{\substack{b_{1,k}, \ldots, b_{k-1,k} \in \{0,1,2\} \\ v_k \in \{0, \ldots, \max\{V(a_k)\} - \deg(a_k)\}}} \left\{ P\left(a_k, v_k + \deg(a_k) + \sum_{l=1}^{k} b_{l,k}\right) + D_j[v_1 - b_{1,k}, \ldots, v_{k-1} - b_{k-1,k}, v_k; b_{1,2}, \ldots, b_{k-1,k}] \right\} \quad (6)$$

where $\deg(a_k)$ denotes the degree of vertex a_k.

Join nodes: let X_i be the parent node of X_j and X_h such that $X_i = X_j = X_h$. Then

$$D_i[v_1, \ldots, v_k; b_{1,2}, \ldots, b_{k-1,k}] = \min_{\substack{v'_l = 0, \ldots, v_l \\ \text{for } l = 1, \ldots, k}} \left\{ D_j[v'_1, \ldots, v'_k; b_{1,2}, \ldots, b_{k-1,k}] + D_h[v_1 - v'_1, \ldots, v_k - v'_k; b_{1,2}, \ldots, b_{k-1,k}] \right\} \quad (7)$$

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 $d \leq 7$ of an atom in the molecule graph. Therefore, the number of edges in a bag is upper bounded by ωd. 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 $O(\alpha^{2\omega} \cdot 3^{\beta} \cdot \omega \cdot m)$, where $\alpha = 1 + \max_{a \in A} \{\max V(a)\}$ is the maximum (open) valence of an atom plus one, m and $\omega - 1$ are size and width of the tree decomposition, d is the maximum degree in the molecule graph, and $\beta := \min\{\binom{\omega}{2}, \omega d\}$ (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 $D_j[v_1, \ldots, v_k; b_{1,2}, \ldots, b_{k-1,k}]$ with $v_l + \sum_j b_{l,j} > \max V(a_l)$ for some l, 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 k. 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

or k arrives its upperbound $\sum_{a\in A}\max_{v\in V(a)}P(a,v)$. Further details can be found in (Böcker *et al.*, 2009).

4 DISCUSSION

For proteins and DNA, bond orders can be simply inferred by matching the given state to a database containing the bond orders for all amino acids and nucleotides. Hence, we focus the evaluation of our algorithms on small and medium-sized molecules (for instance, drug-like molecules). Such molecules can be found in large numbers in several established ligand databases, such as ZINC (Irwin and Shoichet, 2005), ASTEX (Nissink *et al.*, 2002), KEGG Ligand and KEGG Drug (Goto *et al.*, 2002), the MMFF94 validation suite (Halgren, 1996) or the Cambridge Structural Database (Allen, 2002). But evaluating the correctness of our bond order algorithms poses certain constraints: we need ligand structures that contain not only the connectivity information but also preassigned bond orders and explicit hydrogens. As a further requirement, aromatic bonds should be given in kekulized form, i.e. replaced by a suitable pattern of single and double bonds. In contrast to structure-based bond assignment approaches, however, we can use databases that contain 3D ligand structures as well as those only storing structure diagrams or SMILES expressions. To provide a diverse test dataset fulfilling those constraints, we chose the MMFF94 validation suite and the KEGG Drug Database to provide our ground truth.

The MMFF94 validation suite (Halgren, 1996) provides 761 small drug-like molecules, mainly derived from the Cambridge Structural Database (Allen, 2002). The molecules were thoroughly prepared by 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 perception.

The KEGG Drug Database (Goto *et al.*, 2002), provided by the Kanehisa Laboratories, offers a remarkable number of drug-like molecules for diverse applications in bioinformatics. The atoms molecular coordinates are 2D, which is suitable for representation by structure diagrams, but is unsuited for structure-based bond order assignment as performed by most of the former approaches. It thus represents a perfect test scenario for bond order assignment from 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 PO_4, 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%).

Table 1. Comparison of our exact solvers with the original heuristic implementation of Antechamber and the expert generated solutions ('reference solution') for the molecules of the MMFF94 validation set and the KEGG Drug set

	MMFF94 (%)		KEGG (%)	
Heuristic solution is optimal	734	(96.45)	7202	(97.01)
Heuristic solution is suboptimal	9	(1.18)	15	(0.20)
Heuristic found no feasible solution	18	(2.37)	207	(2.79)
Exact solvers found no feasible solution	4	(0.53)	180	(2.42)
Optimal solution agrees with reference	599	(78.71)	6326	(85.21)

Table 2. Performance of the original Antechamber implementation and our exact solvers on the test sets using the penalty table as defined in Wang *et al.* (2006)

Test set	Method	Reference is first solution (%)	Solver reproduces reference (%)
MMFF94	Antechamber	282 (37.05)	282 (37.05)
	ILP	471 (61.89)	
	A*	455 (59.79)	599 (78.71)
	FPT	410 (53.88)	
KEGG	Antechamber	3115 (41.96)	3115 (41.96)
	ILP	4224 (56.90)	
	A*	3708 (49.95)	6326 (85.21)
	FPT	3777 (50.88)	

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.

A complete comparison for both test sets is given in Table 1. Please note that the test datasets were chosen such that they are relatively wellsuited to the heuristic Antechamber approach, e.g. they contain relatively few large or complex ring systems.

4.2 Comparison to reference assignments

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 1, 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

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 methods 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 *tps* 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); Xu *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 Dührkop and Patrik Seeber.

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 (BIZ4:1-4). Q.B.A.B. financial support from DFG, research group 'Parameterized Algorithms in Bioinformatics' (BO 1910/5).

Conflict of Interest: none declared.

REFERENCES

Allen,F.H. (2002) The Cambridge Structural Database: a quarter of a million crystal structures and rising. *Acta Crystallogr. B*, **58** (Pt 3 Pt 1), 380–388.

Althaus,E. *et al.* (2002) A combinatorial approach to protein docking with flexible side chains. *J. Comput. Biol.*, **9**, 597–612.

Arnborg,S. *et al.* (1987) Complexity of finding embedding in a k-tree. *SIAM J. Algebra. Discr.*, **8**, 277–284.

Baber,J.C. and Hodgkin,E.E. (1992) Automatic assignment of chemical connectivity to organic molecules in the cambridge structural database. *J. Chem. Inform. Comput. Sci.*, **32**, 401–406.

Berman,H. *et al.* (2003) Announcing the worldwide protein data bank. *Nat. Struct. Biol.*, **10**, 980.

Böcker,S. *et al.* (2009) Computing bond types in molecule graphs. In *Proceedings of Computing and Combinatorics Conference (COCOON 2009)*, Vol. 5609 of *Lecture Notes Computer Science*, Springer, pp. 297–306.

Bodlaender,H.L. *et al.* (2006) On exact algorithms for treewidth. *Technical Report UU-CS-2006-032*, Institute of Information and Computing Sciences, Utrecht University, Utrecht, Netherlands.

Froeyen,M. and Herdewijn,P. (2005) Correct bond order assignment in a molecular framework using integer linear programming with application to molecules where only non-hydrogen atom coordinates are available. *J. Chem. Inf. Model*, **45**, 1267–1274.

Gogate,V. and Dechter,R. (2004) A complete anytime algorithm for treewidth. In *UAI '04: Proceedings of the 20th conference on Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, Virginia, United States, pp. 201–208.

Goto,S. *et al.* (2002) LIGAND: database of chemical compounds and reactions in biological pathways. *Nucleic Acids Res.*, **30**, 402–404.

Guha,R. *et al.* (2006) The blue obelisk-interoperability in chemical informatics. *J. Chem. Inf. Model*, **46**, 991–998.

Halgren,T. (1996) MMFF VI. MMFF94s option for energy minimization studies. *J. Comp. Chem.*, **17**, 490–519.

Hart,P.E. *et al.* (1968) A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybernetics SSC4*, **4**, 100–107.

Hendlich,M. *et al.* (1997) BALI: automatic assignment of bond and atom types for protein ligands in the brookhaven protein databank. *J. Chem. Inform. Comput. Sci.*, **37**, 774–778.

Hildebrandt,A. *et al*. (2010) BALL - Biochemical Algorithms Library 1.3. *BMC Bioinformatics*, **11**, 531.

Irwin,J.J. and Shoichet,B.K. (2005) ZINC–a free database of commercially available compounds for virtual screening. *J. Chem. Inf. Model*, **45**, 177–182.

Jakalian,A. *et al.* (2002) Fast, efficient generation of high-quality atomic charges. AM1-BCC model: II. parameterization and validation. *J. Comput. Chem.*, **23**, 1623–1641.

Kohlbacher,O. and Lenhof,H.P. (2000) BALL–rapid software prototyping in computational molecular biology. *Bioinformatics*, **16**, 815–824.

Labute,P. (2005) On the perception of molecules from 3D atomic coordinates. *J. Chem. Inf. Model*, **45**, 215–221.

Lang,E. *et al.* (1992) Automatic assignment of bond orders based on the analysis of the internal coordinates of molecular structures. *Anal. Chim. Acta*, **265**, 283–289.

Leach,A.R. and Lemon,A.P. (1998) Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. *Proteins*, **33**, 227–239.

Moll,A. *et al.* (2006) BALLView: a tool for research and education in molecular modeling. *Bioinformatics*, **22**, 365–366.

Nissink,J.W.M. *et al.* (2002) A new test set for validating predictions of protein-ligand interaction. *Proteins*, **49**, 457–471.

Papadimitriou,C.H. and Steiglitz,K. (1998) *Combinatorial Optimization: Algorithms and Complexity*. Courier Dover Publications, Mineola, New York, United States.

van Aalten,D.M. *et al.* (1996) PRODRG, a program for generating molecular topologies and unique molecular descriptors from coordinates of small molecules. *J. Comput. Aided Mol. Des.*, **10**, 255–262.

Wang,J. *et al.* (2006) Automatic atom type and bond type perception in molecular mechanical calculations. *J. Mol. Graph. Model*, **25**, 247–260.

Xu,J. *et al.* (2005) A tree-decomposition approach to protein structure prediction. *Proc. IEEE Comput. Syst. Bioinform. Conf.*, pp. 247–256.

Yanover,C. *et al.* (2008) Minimizing and learning energy functions for side-chain prediction. *J. Comput. Biol.*, **15**, 899–911.

Zhao,Y. *et al*. (2007) Automatic perception of organic molecules based on essential structural information. *J. Chem. Inf. Model*, **47**, 1379–1385.