Computation **2015**, *3*, 285-298; doi:10.3390/computation3020285

OPEN ACCESS

computation

ISSN 2079-3197

www.mdpi.com/journal/computation

Article

Computational Recognition of RNA Splice Sites by Exact Algorithms for the Quadratic Traveling Salesman Problem

Anja Fischer [1], Frank Fischer [2], Gerold Jäger [3], Jens Keilwagen [4], Paul Molitor [5] and Ivo Grosse [5,6,*]

[1] Department of Mathematics, TU Dortmund, D-44227 Dortmund, Germany;
E-Mail: anja.fischer@mathematik.tu-dortmund.de

[2] Institute of Mathematics, University of Kassel, D-34132 Kassel, Germany;
E-Mail: frank.fischer@mathematik.uni-kassel.de

[3] Department of Mathematics and Mathematical Statistics, University of Umeå, S-90187 Umeå, Sweden; E-Mail: gerold.jaeger@math.umu.se

[4] Institute for Biosafety in Plant Biotechnology, Julius Kühn-Institut, D-06484 Quedlinburg, Germany;
E-Mail: jens.keilwagen@jki.bund.de

[5] Institute of Computer Science and Universitätszentrum Informatik, Martin Luther University Halle-Wittenberg, D-06120 Halle, Germany; E-Mail: paul.molitor@informatik.uni-halle.de

[6] German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, D-04103 Leipzig, Germany

* Author to whom correspondence should be addressed; E-Mail: ivo.grosse@informatik.uni-halle.de; Tel.: +49-345-5524774; Fax: +49-345-5527039.

Academic Editors: Marnix Medema and Rainer Breitling

Received: 3 December 2014 / Accepted: 12 May 2015 / Published: 3 June 2015

Abstract: One fundamental problem of bioinformatics is the computational recognition of DNA and RNA binding sites. Given a set of short DNA or RNA sequences of equal length such as transcription factor binding sites or RNA splice sites, the task is to learn a pattern from this set that allows the recognition of similar sites in another set of DNA or RNA sequences. Permuted Markov (PM) models and permuted variable length Markov (PVLM) models are two powerful models for this task, but the problem of finding an optimal PM model or PVLM model is NP-hard. While the problem of finding an optimal PM model or PVLM model of order one is equivalent to the traveling salesman problem (TSP), the problem of finding an optimal PM model or PVLM model of order two is equivalent to the quadratic TSP (QTSP). Several exact algorithms exist for solving the QTSP, but it is unclear if these algorithms are capable of solving QTSP instances resulting from RNA splice sites of at least 150 base pairs in a reasonable time frame. Here, we investigate the performance of three exact algorithms for solving the QTSP for ten datasets of splice acceptor sites and splice donor sites of five different species and find that one of these algorithms is capable of solving QTSP instances of up to 200 base pairs with a running time of less than two days.

Keywords: splice site; permuted Markov model; permuted variable length Markov model; quadratic traveling salesman problem; combinatorial optimization; dynamic programming; branch and bound; branch and cut; integer linear programming

1. Introduction

Gene regulation in higher organisms is accomplished at several levels such as transcriptional regulation and post-transcriptional regulation by several cellular processes such as transcription initiation and RNA splicing. In order to better understand these processes, it is desirable to have a good understanding of how transcription factors recognize DNA binding sites and how the spliceosome recognizes RNA splice sites.

Many approaches for the computational recognition of transcription factor binding sites or RNA splice sites rely on statistical models, and two popular models for this task are permuted Markov (PM) models [1] and permuted variable length Markov (PVLM) models [2]. The advantage of these models over traditional Markov models such as the position weight matrix model [3,4] or the weight array matrix model [5] is their capability of modeling dependencies among neighboring and non-neighboring positions [1,2].

There is evidence that dependencies are not restricted to pairs of positions, which can be modeled by PM models and PVLM models of order one, and that PM models and PVLM models of order two, which are capable of modeling dependencies among triplets of neighboring and non-neighboring positions, outperform PM models and PVLM models of order one [1,2]. However, learning optimal PM models and PVLM models of orders one and two is NP-hard, where NP stands for non-deterministic polynomial-time [6], so heuristics have been used for this problem [1,2].

While transcription factor binding sites are typically short and rarely exceed 20 base pairs (bp), it has been shown that the recognition of RNA splice sites can be improved by modeling longer sequences of at least 150 bp [7]. Hence, it would be desirable to develop exact algorithms capable of learning PM models and PVLM models for RNA splice sites of at least 150 bp in practically acceptable running times.

Learning a pattern from data by a statistical model is often performed in two steps. First, an appropriate model structure reflecting realistic and yet tractable conditional independence assumptions must be learned, and second, appropriate model parameters of the conditional probability distributions must be estimated. Several estimators have been proposed for the second task including the widely used maximum likelihood estimator (MLE). Likewise, several learning principles have been proposed for the first task, and one popular choice is the maximum likelihood principle (MLP).

For PM models and PVLM models of order one, the task of learning the maximum likelihood model results in the traditional Hamiltonian path problem (HPP) or the related traditional traveling salesman problem (TSP). For more powerful PM models and PVLM models of order two, the task of learning the maximum likelihood model results in the quadratic Hamiltonian path problem (QHPP) or the related quadratic traveling salesman problem (QTSP), which are extensions of the traditional linear HPP and TSP, respectively.

Several heuristics for approximately solving QTSPs associated with PM models and PVLM models of order two exist [8,9]. However, these heuristics yield quite different solutions with quite different likelihoods, and there are no guarantees about the deviations of the suboptimal solutions from the optimal solution. Moreover, the running times of some of the heuristics are so high that we focus on exact solution methods in this work.

There are several exact algorithms for solving QTSPs [8–10], which provide optimal solutions in acceptable running times at least for small instances. Specifically, it has been shown that a branch-and-cut algorithm tailored to the QHPP and the QTSP can solve these problems for random instances up to size 30 in a few hours [9].

Here, we investigate the performance of this algorithm and two other exact algorithms for learning PM models and PVLM models of order two for RNA splice sites. Specifically, we investigate the performance of these three algorithms for ten datasets of splice acceptor sites and splice donor sites of five different species by measuring their running times for increasing lengths of the studied RNA splice sites until the running times exceed 1.5×10^5 seconds (s), corresponding to approximately two days.

The paper is organized as follows. In Section 2, we summarize the works by Ellrott *et al.* [1] and Zhao *et al.* [2], who introduce PM models and PVLM models, respectively, and who show how to use these models for the recognition of DNA and RNA binding sites. We provide a formal definition of the QTSP in Section 3 and present three exact algorithms for solving this optimization problem in Section 4. We study the running times of these three algorithms for ten datasets of RNA splice sites in Section 5 and conclude the paper in Section 6.

2. Permuted Markov Models and Permuted Variable Length Markov Models

Nucleotide sequences of transcription factor binding sites and splice sites are not statistically independent, and one of the challenges of bioinformatics is to devise statistical models that can capture dependencies among neighboring and non-neighboring positions that are omnipresent in both transcription factor binding sites and splice sites. Two powerful models for the recognition of transcription factor binding sites and splice sites that are capable of capturing such dependencies are PM models and PVLM models proposed by Ellrott *et al.* [1] and Zhao *et al.* [2]. The key idea of both models is to allow a permutation of the ordering of nucleotides of a binding site and then to define a Markov model or a variable length Markov model over that permuted binding site.

Consider a dataset D of sequences x of length L over a discrete, finite alphabet Σ, and consider a PM model or PVLM model for modeling these sequences. The central structure of these models is a permutation π of positions $1, \ldots, L$. We denote the predecessors of π_ℓ in permutation π by $\text{pre}(\pi, \ell)$. Specifically, $\text{pre}(\pi, \ell)$ returns $\pi_{\ell-1}$ (if $\ell > 1$) for PM models and PVLM models of order one, $(\pi_{\ell-1}, \pi_{\ell-2})$

(if $\ell > 2$) for PM models and PVLM models of order two and, in general, $(\pi_{\ell-1}, \ldots, \pi_{\ell-d})$ (if $\ell > d$) for PM models and PVLM models of order d, while pre(π, ℓ) returns $(\pi_1, \ldots, \pi_{\ell-1})$ for $\ell \leq d$.

Given a permutation π, the log-likelihood of a PM model can then be written as a sum of L position-specific terms, where the ℓ-th term depends only on position π_ℓ and its predecessor pre(π, ℓ). Specifically, the log-likelihood of a single sequence x given permutation π can be written as:

$$\log P(x|\pi) = \sum_{\ell=1}^{L} \log P(x_{\pi_\ell}|x_{\text{pre}(\pi,\ell)})$$

and the likelihood of an independent and identically distributed dataset D given permutation π can be written as:

$$\log P(D|\pi) = \sum_{x \in D} \sum_{\ell=1}^{L} \log P(x_{\pi_\ell}|x_{\text{pre}(\pi,\ell)})$$

Using maximum likelihood estimators for the model parameters, the conditional probabilities on the right-hand side can be replaced by ratios of absolute frequencies as follows. For a dataset D of N sequences x, we denote the absolute frequency of observing oligonucleotide y at position i by:

$$N_i(y) := |\{x \in D \colon x_i = y\}|$$

the absolute frequency of observing oligonucleotide y at position i and nucleotide z at position j by:

$$N_{i,j}(y, z) := |\{x \in D \colon x_i = y \wedge x_j = z\}|$$

the relative frequency of observing oligonucleotide y at position i by:

$$P_i(y) := \frac{N_i(y)}{N}$$

and the conditional relative frequency of observing nucleotide z at position j given oligonucleotide y at position i by:

$$P_{j,i}(z|y) := \frac{N_{i,j}(y, z)}{N_i(y)}$$

Based on that, the maximum log-likelihood of dataset D given permutation π can now be rewritten as:

$$\begin{aligned}
\log P(D|\pi) &= \sum_{\ell=1}^{L} \sum_{y \in \Sigma^{|\text{pre}(\pi,\ell)|}} \sum_{z \in \Sigma} N_{\text{pre}(\pi,\ell),\pi_\ell}(y, z) \log P_{\pi_\ell,\text{pre}(\pi,\ell)}(z|y) \\
&= N \sum_{\ell=1}^{L} \sum_{y \in \Sigma^{|\text{pre}(\pi,\ell)|}} P_{\text{pre}(\pi,\ell)}(y) \sum_{z \in \Sigma} P_{\pi_\ell,\text{pre}(\pi,\ell)}(z|y) \log P_{\pi_\ell,\text{pre}(\pi,\ell)}(z|y) \\
&= -N \sum_{\ell=1}^{L} H(X_{\pi_\ell}|X_{\text{pre}(\pi,\ell)}) \qquad (1)
\end{aligned}$$

where $\Sigma^{|\text{pre}(\pi,\ell)|}$ denotes the set of sequences over Σ of length $|\text{pre}(\pi, l)|$, and $H(X_{\pi_\ell}|X_{\text{pre}(\pi,\ell)})$ denotes the frequency estimator of the conditional Shannon entropy of nucleotide X_{π_ℓ} at position π_ℓ given oligonucleotide $X_{\text{pre}(\pi,\ell)}$ at position $\text{pre}(\pi, \ell)$ in units of nats. Hence, Equation (1) states that the maximum log-likelihood of dataset D given permutation π is given by a sum of L position-specific terms, where the ℓ-th term depends only on positions $\text{pre}(\pi, \ell)$ and π_ℓ.

Computing the maximum log-likelihood $\log P(D|\pi)$ of a dataset D given a permutation π is straightforward, but finding an optimal permutation π that maximizes the log-likelihood $\log P(D|\pi)$ over all $L!$ permutations π is non-trivial. Mathematically, this optimization problem can be stated as:

$$\hat{\pi} := \underset{\pi}{\text{argmax}} \{\log P(D|\pi)\} \tag{2}$$

and we present the connection to the QTSP in the following section.

3. Quadratic Traveling Salesman Problem

In this section, we formally introduce the QHPP and the QTSP. Given a complete directed graph $G = (V, A)$ with a set of nodes $V = \{1, \ldots, L\}, L \geq 3$, a set of arcs $A = V^{(2)} := \{(i, j) \colon i, j \in V,\ i \neq\ j\}$, and an associated set of two-arcs $V^{(3)} := \{(i, j, k) \colon i, j, k \in V, |\{i, j, k\}| = 3\}$ (a two-arc (i, j, k) is an ordered sequence of pairwise distinct nodes i, j, k), a sequence of nodes $(v_1, v_2, \ldots, v_k)$ is called a path if all $v_i \in V, i \in \{1, \ldots, k\}$, are pairwise distinct. Similarly, a sequence of nodes $(v_1, v_2, \ldots, v_k, v_1)$ is called a cycle in G if all $v_i \in V, i \in \{1, \ldots, k\}$, are pairwise distinct. A path $(v_1, v_2, \ldots, v_k)$ or a cycle $(v_1, v_2, \ldots, v_k, v_1)$ in G is called Hamiltonian if $k = L$, and a Hamiltonian cycle is also called a tour.

For given arc weights $c_l \colon V^{(2)} \to \mathbb{R}$ and two-arc weights $c_q \colon V^{(3)} \to \mathbb{R}$, the total costs of a path $Q = (v_1, \ldots, v_k)$ w. r. t. to c_l or c_q are:

$$c_l(Q) := \sum_{i=1}^{k-1} c_l((v_i, v_{i+1})) \quad \text{and} \quad c_q(Q) := \sum_{i=1}^{k-2} c_q((v_i, v_{i+1}, v_{i+2}))$$

respectively.

Similarly, the total costs of the corresponding cycle $C = (v_1, \ldots, v_k, v_1)$ w. r. t. to c_l or c_q are:

$$c_l(C) := c_l(Q) + c_l((v_k, v_1)) \quad \text{and} \quad c_q(C) := c_q(Q) + c_q((v_{k-1}, v_k, v_1)) + c_q((v_k, v_1, v_2))$$

respectively.

With these definitions, the weighted Hamiltonian path problem (HPP) is to:

$$\begin{array}{ll} \text{minimize} & c_l(Q) \\ \text{subject to} & Q \text{ is a Hamiltonian path in } G \end{array}$$

and the weighted quadratic Hamiltonian path problem (QHPP) is to:

$$\begin{array}{ll} \text{minimize} & c_q(Q) \\ \text{subject to} & Q \text{ is a Hamiltonian path in } G \end{array}$$

The TSP that asks for a cost-minimal tour w. r. t. c_l can be formulated in a similar way by replacing the Hamiltonian path Q with a Hamiltonian cycle C, and the QTSP that asks for a cost-minimal tour

w. r. t. c_q can be formulated analogously by replacing the Hamiltonian path Q with a Hamiltonian cycle C. That is, the TSP is to:

$$\begin{aligned} &\text{minimize} && c_l(C) \\ &\text{subject to} && C \text{ is a Hamiltonian cycle in } G \end{aligned}$$

and the QTSP is to:

$$\begin{aligned} &\text{minimize} && c_q(C) \\ &\text{subject to} && C \text{ is a Hamiltonian cycle in } G \end{aligned}$$

i.e., the QTSP differs from the TSP only in that the cost of a tour depends on each triple, and not only on each pair, of nodes that are traversed in succession in the tour.

An HPP can be easily transformed into a TSP on a larger graph by inserting an additional artificial node and by setting the costs of the additional arcs appropriately. Likewise, a QHPP can be easily transformed into a QTSP on a larger graph by inserting an additional artificial node and by setting the costs of the additional two-arcs appropriately.

For modeling the QHPP, we consider a QTSP on the complete graph $\overline{G}$ with set of nodes $\overline{V} = V \cup \{0\}$, where the artificial node 0 is used to model the first summands in Equation (1) and to allow for a complete definition of $c_l \colon \overline{V}^{(2)} \to \mathbb{R}$ and $c_q \colon \overline{V}^{(3)} \to \mathbb{R}$. Specifically, we set:

$$c_l((j,k)) := \begin{cases} H(X_k) & j = 0 \\ H(X_k|X_j) & j,k \in V \\ 0 & \text{otherwise} \end{cases}$$

$$c_q((i,j,k)) := \begin{cases} H(X_k) & j = 0 \\ H(X_k|X_j) & i = 0 \\ H(X_k|X_{ij}) & i,j,k \in V \\ 0 & \text{otherwise} \end{cases}$$

for arcs $(j,k) \in \overline{V}^{(2)}$ and two-arcs $(i,j,k) \in \overline{V}^{(3)}$, respectively.

Based on these definitions, an optimal permutation π of problem (2) can be obtained by solving a QTSP.

4. Exact Algorithms

In this section, we describe three exact algorithms for solving the NP-hard QTSP. First, we present a simple algorithm based on dynamic programming in Section 4.1. Second, we present a branch-and-bound algorithm based on some combinatorial lower bounds in Section 4.2. Finally, we present a branch-and-cut algorithm based on integer programming in Section 4.3.

4.1. Dynamic Programming Algorithm

The following dynamic programming (DP) algorithm for solving the QTSP is an extended version of a DP algorithm for solving the TSP [11]. For the sake of simplicity, we solve the QHPP on the graph $\overline{G}$ with the additional node 0 as the fixed first node of the path, so we consider only the nodes in V in the remainder of this subsection.

For a subset $W \subseteq V$ and two nodes $j, k \in W$ and $j \neq k$, we compute the optimal costs $B_{W,j,k}$ for any path visiting all nodes of W, with j and k being the last two nodes in this order by the recursion:

$$B_{W,j,k} = \begin{cases} c_q((0, j, k)) & \text{iff } |W| = 2 \\ \min\limits_{i \in W \setminus \{j,k\}} B_{W \setminus \{k\},i,j} + c_q((i, j, k)) & \text{otherwise} \end{cases}$$

We perform this recursion bottom-up for all subsets $W \subseteq V$, *i.e.*, we start with all subsets of size two, then continue with all subsets of size three, *etc.*, which ensures that the partial solutions needed in the recursion are already computed.

The minimal costs for a quadratic Hamiltonian path can finally be determined by:

$$B^* = \min_{j,k \in V, j \neq k} B_{V,j,k}$$

and the corresponding path can be derived by backtracking.

4.2. Branch-and-Bound Algorithm

The branch-and-bound (BnB) algorithm was first described by Land and Doig in 1960 [12]. The idea for solving minimization problems with discrete decisions, e.g., whether an arc is contained in a tour or not, is the following. Fixing parts of the solution and creating several subproblems (branching), we compute local lower bounds for each of the subproblems that are compared to the currently best solution. If the lower bound of one of the subproblems is larger than the value of the currently best solution, we can delete this subproblem, because it cannot lead to an optimal solution. Furthermore, we update the best solution if we find a better solution during the subproblem calculations. The hope is that, instead of testing all possible cases, the solution space can be reduced significantly by the lower bound calculations.

The BnB algorithm for the QTSP traverses all possible tours in the worst case. In the branching part, we create subproblems by fixing subpaths. Each time we consider a new subpath, we compute a lower bound for a QTSP solution containing this subpath. Here, we use as a lower bound the solution of some cycle cover problem (CCP), which asks for an arc-cost minimal set of cycles $K = \{C_1, \ldots, C_k\}$ such that each node is contained in exactly one cycle. As the objective function for the CCP, we use the coefficients:

$$c_l((u, v)) := \min_{w \in V \setminus \{u,v\}} c_q((u, v, w))$$

for the arcs $(u, v) \in V^{(2)}$. With this definition of the arc costs, we ensure that the optimal value of the CCP is a lower bound of the optimal value of the associated QTSP, because for each two-arc $(u, v, w') \in V^{(3)}$ contained in an optimal QTSP solution, we only count the costs $c_q((u, v, w))$ with

$w = \operatorname{argmin}_{\bar{w}\in V\setminus\{u,v\}} c_q((u,v,\bar{w}))$, associated with the arc (u,v) in the CCP. The advantage of the described transformation to a CCP is that the CCP can be solved in polynomial time by the Hungarian method [13], which allows us to derive lower bounds in polynomial time. We start all tours with a fixed node v_1, which is chosen in such a way that the sum over all values $c_q((v_1,u,w))$ with $(v_1,u,w)\in V^{(3)}$ is maximal. The motivation for this choice is that we hope that the lower bounds in the first steps are rather large, allowing us to prune several subproblems close to the beginning.

4.3. Branch-and-Cut Algorithm

Branch-and-cut (BnC) algorithms have been very successful for solving the TSP [10,14–16]. Given an integer programming (IP) formulation, we consider a relaxation of the problem by dropping the integrality constraints of the variables and possibly some of the constraints. As we will see below, the standard formulations for the TSP and the QTSP contain an exponential number of constraints. Hence, it is practically impossible to add all constraints to the model at once. The BnC approach combines linear programming for solving the relaxation with a so-called cutting-plane approach, which starts with a restricted number of constraints and adds them successively during the solution process if they are violated. In order to improve the formulation, one usually separates further cutting planes that tighten the current relaxation to obtain even stronger bounds. Furthermore, the BnC approach combines the cutting plane approach based on linear programming with a BnB approach.

The standard IP formulation for the TSP of Dantzig *et al.* [14] uses binary arc variables $r_a \in \{0,1\}, a \in A$, with the interpretation $r_a = 1$ if and only if the arc a is contained in the tour and zero otherwise. It reads:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(u,v)\in A} c_l((u,v))\cdot r_{(u,v)} && \\
\text{subject to} \quad & \sum_{\substack{v\in V:\\(u,v)\in A}} r_{(u,v)} = \sum_{\substack{v\in V:\\(v,u)\in A}} r_{(v,u)} = 1 & u\in V \quad & (3)\\
& \sum_{(u,v)\in S^{(2)}} r_{(u,v)} \le |S|-1 & \emptyset\neq S\subsetneq V \quad & (4)\\
& r\in\{0,1\}^A & & (5)
\end{aligned}$$

The degree constraints (3) ensure that each node is entered and left exactly once by the tour. Cycles of a length less than L are forbidden by the well-known subtour elimination constraints (SEC) (4). Finally, condition (5) ensures the integrality of the variables.

Concerning the QTSP, each two-arc $(u,v,w)\in V^{(3)}$ is contained in a tour if and only if the two associated arcs (u,v) and (v,w) are both contained in the tour. Hence, we can formulate the QTSP as an IP with quadratic objective function:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(u,v,w)\in V^{(3)}} c_q((u,v,w))\cdot r_{(u,v)}\cdot r_{(v,w)}\\
\text{subject to} \quad & \text{constraints (3), (4), and (5)}
\end{aligned}$$

Linearizing this objective function by replacing each product $r_{(u,v)} \cdot r_{(v,w)}$ with a new binary variable $t_{(u,v,w)} \in \{0,1\}$ for each two-arc $(u,v,w) \in V^{(3)}$, we derive the following linear IP formulation for the QTSP:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(u,v,w)\in V^{(3)}} c_q((u,v,w)) \cdot t_{(u,v,w)} \\
\text{subject to} \quad & \text{constraints (3), (4), and (5)} \\
& r_{(u,v)} = \sum_{\substack{w\in V:\\(u,v,w)\in V^{(3)}}} t_{(u,v,w)} = \sum_{\substack{w\in V:\\(w,u,v)\in V^{(3)}}} t_{(w,u,v)}, \qquad (u,v)\in A \qquad (6) \\
& t \in \{0,1\}^{V^{(3)}} \qquad (7)
\end{aligned}$$

Constraints (6) couple the arc variables and the two-arc variables. If an arc $(u,v) \in A$ is contained in the tour, there must be a node $w \in V$ so that the path (u,v,w) is part of the tour and a node $w' \in V$ so that the path (w',u,v) is part of the tour.

In the BnC approach, we use constraints (3) and (6), $r_a \in [0,1], a \in A$, and $t_{(u,v,w)} \in [0,1]$, $(u,v,w) \in V^{(3)}$, as a basic relaxation. In order to obtain an exact algorithm for the QTSP, it is sufficient to separate the standard subtour elimination constraints (4). However, in order to speed up the algorithm, we add further cutting planes known to be valid for the QTSP.

Lemma 1: See [17]:

1. *The triangle inequalities of Type I:*

$$t_{(u,v,w)} + t_{(w,u,v)} \leq r_{(u,v)} \tag{8}$$

are valid for the QTSP for all $u,v,w \in V, |\{u,v,w\}| = 3, L \geq 4$.

2. *The triangle inequalities of Type II:*

$$\sum_{(u,v)\in S^{(2)}} r_{(u,v)} - \sum_{(u,v,w)\in S^{(3)}} t_{(u,v,w)} \leq 1 \tag{9}$$

are valid for the QTSP for all $S \subseteq V, |S| = 3$.

3. *The strengthened subtour elimination constraints:*

$$\sum_{(u,v)\in S^{(2)}} r_{(u,v)} + \sum_{\substack{u,v\in S, w\in V\setminus S:\\(u,w,v)\in V^{(3)}}} t_{(u,w,v)} \leq |S| - 1 \tag{10}$$

are valid for the QTSP for all $S \subset V, 2 \leq |S| < \frac{L}{2}, L \geq 5$.

The triangle inequalities (8) forbid cycles of length three. If $L \geq 4$, at most one of the two two-arcs (u,v,w) and (w,u,v) can be contained in a tour. However, if one of these two two-arcs is contained in the tour, then also arc (u,v) is contained in the tour. The validity of inequalities (9) follows from the fact that at most two of the corresponding arcs can be contained in a tour, but if two arcs are contained in the tour, also one of the two-arcs is contained in the tour. Inequalities (10) are a strengthened version of inequalities (4), where we do not only count the direct connections between two nodes $u, v \in S, u \neq v$, but also the connections with one node between them.

Inequalities (8)–(10) are even facet-defining for the polytope associated with the QTSP [17] if L is sufficiently large. Considering the separation problems for the three inequality classes, a violated inequality of the first two classes can be determined in polynomial time, because there are only $\mathcal{O}(L^3)$ such inequalities. A maximally-violated subtour elimination constraint (4) can be determined in polynomial time by minimum-cut calculation [18], because constraints (4) are equivalent to $\sum_{u \in S, v \in V \setminus S} r_{(u,v)} \geq 1, S \subset V, 2 \leq |S| \leq L - 2$. In contrast to this, it has been shown in [17] that determining a maximally-violated constraint of type (10) is NP-hard. For this reason, we explicitly separate all inequalities (10) for sets $S \subset V$ with $|S| = 2$ and use a heuristic approach for separating inequalities (10) in general in our experiments.

5. Experimental Study

In this section, we study the running times of the three algorithms presented in Section 4 on ten datasets of splice acceptor sites and splice donor sites from five different species of lengths up to 200 bp. We implemented all algorithms in C++ and Java using the following subroutines. For the BnB algorithm, we used the CCP solver implemented by Jonker and Volgenant [19], which is based on the Hungarian method. For the BnC algorithm, we used the BnC solver Gurobi 5.6.3 [20], which we extended by problem-specific cutting planes. For the separation of the subtour elimination constraints (4), we used the software package Lemon [21]. If a cut of value less than one is found, we separate constraints (10) instead of constraints (4). Furthermore, we built separators for inequalities (8) and (9) and for inequalities (10) with sets $S \subset V, |S| = 2$, which are based on complete enumeration. We carried out all experiments on a PC with an Intel® Core$^{\text{TM}}$ i7 CPU 920 with 2.67 GHz and 12 GB RAM.

From each of the ten datasets, we extract all splice sites, truncate the surrounding sequences to length L symmetrically around the splice sites, and apply each of the three algorithms presented in Section 4 to each of these ten datasets containing subsequences of length L. We stop each algorithm if its running time exceeds 1.5×10^5 s, corresponding to approximately two days, and record its running time otherwise. Finally, we plot the running times as a function of the sequence length L ranging from 4 bp to 200 bp for each of the three algorithms and each of the ten datasets in Figure 1.

Figure 1 shows that the running time of the DP algorithm increases super-exponentially with L, as expected theoretically. For the ten datasets studied, the DP algorithm is capable of solving the QTSP for $L = 20$ bp in approximately 70 s, whereas it is not capable of solving the QTSP for $L \geq 30$ bp in less than 1.5×10^5 s.

Figure 1 further shows that the BnB algorithm is faster than the DP algorithm for short sequences of $L \leq 10$ bp, but that its running time increases dramatically with growing L so that the BnB algorithm is not capable of solving the QTSP for $L = 20$ bp in less than 1.5×10^5 s for any of the ten datasets studied.

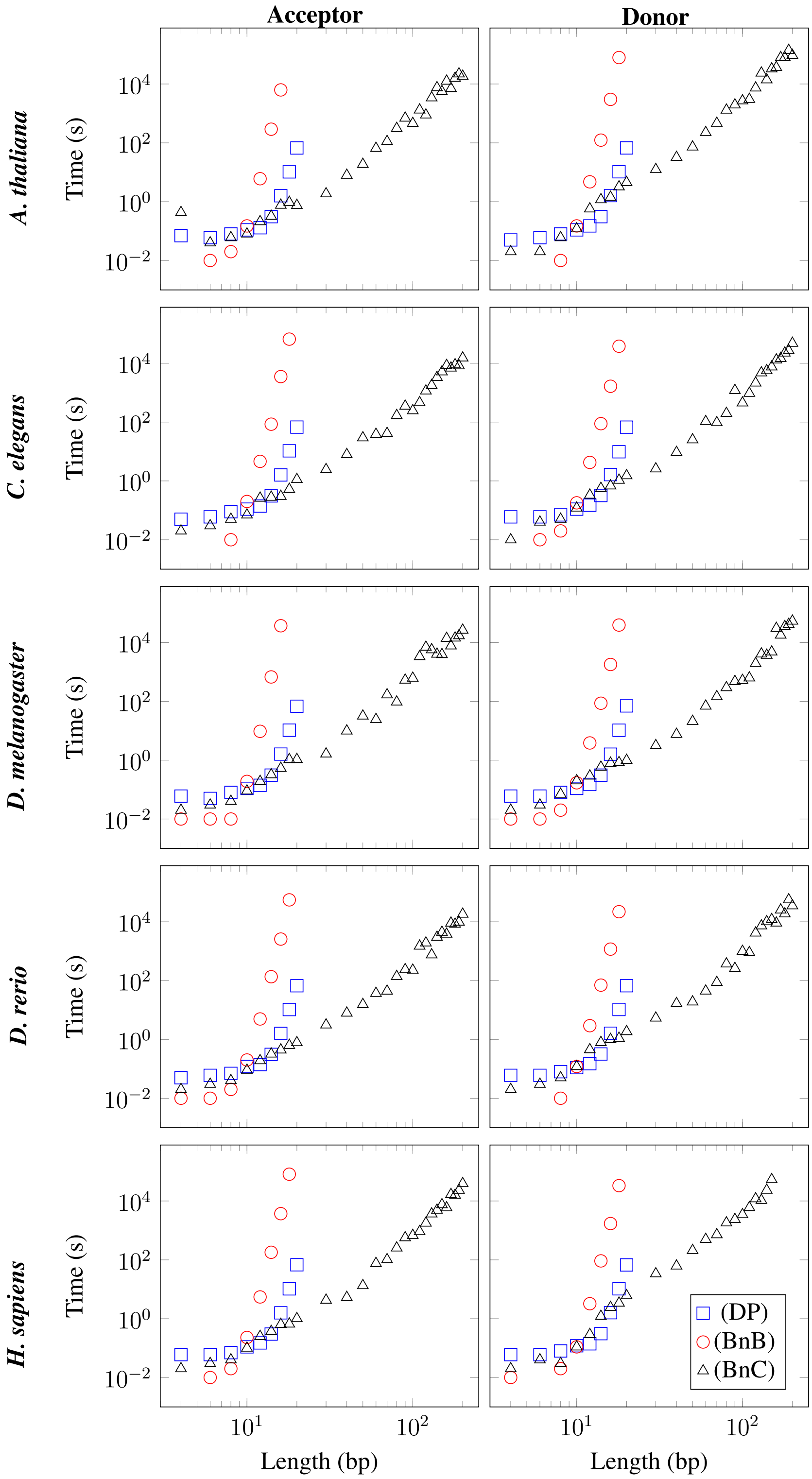

Figure 1. Running times of the three algorithms presented in Section 4 for five datasets of splice acceptor sites and five datasets of splice donor sites from *A. thaliana*, *C. elegans*, *D. melanogaster*, *D. rerio*, and *H. sapiens* of [22]. We plot the running time of each of the three algorithms as a function of the sequence length L ranging from 4 bp to 200 bp for each of the ten datasets provided the running time exceeds 10^{-2} s and does not exceed 1.5×10^5 s, corresponding to approximately two days. DP, dynamic programming; BnB, branch-and-bound; BnC, branch-and-cut.

Finally, we observe in Figure 1 that the running time of the BnC algorithm is comparable to that of the DP algorithm for short sequences of $L \leq 10$ bp, but that its running time increases substantially more slowly with L than that of the DP algorithm for $L \geq 10$ bp. Surprisingly, the BnC algorithm is capable of solving the QTSP for $L = 200$ bp in less than 1.5×10^5 s for nine of the ten datasets studied. The average running times for $L = 50$ bp, 100 bp, 150 bp, and 200 bp can be found in Table 1.

Table 1. Average running times of the BnC algorithm for $L = 50$ bp, 100 bp, 150 bp, and 200 bp. For $L = 200$ bp, we compute the average over all instances except for the *H. sapiens* splice donor sites.

Sequence Length *L* (bp)	**50**	**100**	**150**	**200**
running time (s)	45	1032	13,731	38,757

The running times of the BnC algorithm are significantly shorter than those of the other two algorithms. Even if we assume an only exponential increase of the running time of the DP algorithm with L and a crude theoretical lower bound of 2^L, we obtain a running time for the DP algorithm of more than 10^{56} s for $L = 200$ bp, stating that the BnC algorithm is at least 50 orders of magnitude faster than the currently fastest algorithm for solving the QTSP for splice donor sites and splice acceptor sites of $L = 200$ bp.

The surprisingly small running times of the BnC algorithm can be explained as follows. There are usually less than one hundred nodes in the BnC tree. Hence, for the ten datasets studied, the cutting planes introduced in Section 4 are very effective and much better than for the random instances studied in [9].

6. Conclusions

The computational recognition of RNA splice sites is an important task in bioinformatics, and two popular models for this task are permuted Markov models and permuted variable length Markov models. However, learning permuted Markov models and permuted variable length Markov models is NP-hard and, thus, a challenging problem for the recognition of RNA splice sites, because it could be shown that sequences of at least 150 bp surrounding the splice sites should be taken into account for a reliable recognition of RNA splice sites. Hence, learning permuted Markov models and permuted variable length Markov models could be performed only heuristically in the past.

Learning optimal permuted Markov models and optimal permuted variable length Markov models of order two, however, is equivalent to the quadratic traveling salesman problem (QTSP), and a recently-developed BnC algorithm based on integer programming has been capable of solving randomly-generated instances up to instance sizes of 30. Hence, we have investigated in this paper the capability of this algorithm and two additional algorithms for learning permuted Markov models and permuted variable length Markov models, a DP algorithm and a BnB algorithm, for ten datasets of splice acceptor sites and splice donor sites.

We have found that the BnC algorithm based on integer programming is by orders of magnitude more effective than the DP algorithm and the BnB algorithm. Specifically, we have found that the BnC algorithm is capable of learning permuted Markov models and permuted variable length Markov models from splice sites of lengths up to 200 bp in less than two days.

Acknowledgments

We thank DFG (Grant No. GR 3526/3) and Universitätszentrum Informatik Halle-Wittenberg for financial support.

Author Contributions

Ivo Grosse and Jens Keilwagen formulated the bioinformatics problem as QTSP and developed the DP approach, Gerold Jäger and Paul Molitor developed the BnB approach, and Anja Fischer and Frank Fischer developed the BnC approach. Anja Fischer, Frank Fischer, Gerold Jäger, and Jens Keilwagen processed the data and performed the study. All authors discussed the study and the results, and all authors wrote the manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Ellrott, K.; Yang, C.; Sladek, F.M.; Jiang, T. Identifying transcription factor binding sites through Markov chain optimization. *Bioinformatics* **2002**, *18*, 100–109.
2. Zhao, X.; Huang, H.; Speed, T.P. Finding short DNA motifs using permuted Markov models. *J. Comput. Biol.* **2005**, *12*, 894–906.
3. Stormo, G.D.; Schneider, T.D.; Gold, L.M.; Ehrenfeucht, A. Use of the “perceptron” algorithm to distinguish translational initiation sites. *Nucleic Acids Res.* **1982**, *10*, 2997–3010.
4. Staden, R. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res.* **1984**, *12*, 505–519.
5. Zhang, M.; Marr, T. A weight array method for splicing signal analysis. *Comput. Appl. Biosci.* **1993**, *9*, 499–509.
6. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W.H. Freeman and Company: New York, NY, USA, 1979.
7. Sonnenburg, S.; Schweikert, G.; Philips, P.; Behr, J.; Rätsch, G. Accurate splice site prediction using support vector machines. *BMC Bioinform.* **2007**, *8*, doi:10.1186/1471-2105-8-S10-S7.
8. Jäger, G.; Molitor, P. Algorithms and Experimental Study for the Traveling Salesman Problem of Second Order. In *Lecture Notes in Computer Science*; Proceedings of the Second International Conference on Combinatorial Optimization and Applications, COCOA 2008, St. John’s, NL, Canada, 21–24 August 2008; Yang, B., Du, D., Wang, C.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5165, pp. 211–224.

9. Fischer, A.; Fischer, F.; Jäger, G.; Keilwagen, J.; Molitor, P.; Grosse, I. Exact Algorithms and Heuristics for the Quadratic Traveling Salesman Problem with an Application in Bioinformatics. *Discret. Appl. Math.* **2014**, *166*, 97–114.
10. Applegate, D.L.; Bixby, R.E.; Chvátal, V.; Cook, W.J. *The Traveling Salesman Problem: A Computational Study*; Princeton Series in Applied Mathematics; Princeton University Press: Princeton, NJ, USA, 2007.
11. Bellman, R. Dynamic Programming Treatment of the Travelling Salesman Problem. *J. ACM* **1962**, *9*, 61–63.
12. Land, A.H.; Doig, A.G. An automatic method of solving discrete programming problems. *Econometrica* **1960**, *28*, 497–520.
13. Kuhn, H.W. The Hungarian Method for the Assignment Problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97.
14. Dantzig, G.; Fulkerson, R.; Johnson, S. Solution of a large-scale traveling-salesman problem. *Oper. Res.* **1954**, *2*, 393–410.
15. Grötschel, M.; Holland, O. Solution of large-scale symmetric travelling salesman problems. *Math. Program. Ser. A* **1991**, *51*, 141–202.
16. Padberg, M.; Rinaldi, G. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Rev.* **1991**, *33*, 60–100.
17. Fischer, A. An Analysis of the Asymmetric Quadratic Traveling Salesman Polytope. *SIAM J. Discret. Math.* **2014**, *28*, 240–276.
18. Hong, S. A Linear Programming Approach for the Traveling Salesman Problem. Ph.D. Thesis, John Hopkins University, Baltimore, MD, USA, 1972.
19. Jonker, R.; Volgenant, A. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* **1987**, *38*, 325–340.
20. Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual, 2014. Available online: http://www.gurobi.com (accessed on 19 May 2015).
21. LEMON Graph Library 1.2.2. Available online: http://lemon.cs.elte.hu/trac/lemon (accessed on 19 May 2015).
22. Computational Biology Center. http://cbio.mskcc.org/public/raetschlab/user/behr/splicing/ (accessed on 19 May 2015).