



Internal Report 2011–08

August 2011

Universiteit Leiden

Opleiding Informatica

Scaffolding of next-generation sequencing assemblies
using diverse information sources

Alexey A. Gritsenko

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

ABSTRACT

Motivation: The increased interest in genome sequencing, fueled by the ongoing rapid development of *high-throughput sequencing* technologies, calls for means of obtaining high quality draft genomes. The millions of reads usually involved are first assembled into contigs. These are then scaffolded, i.e. ordered and oriented using additional information. A number of scaffolders have been developed, including the feature-rich Bambus scaffolder which allows for usage of multiple scaffolding information sources. However, none of the available scaffolding algorithms employ continuous scaffolding constraint satisfiability measures and none provide quality estimates for the constructed scaffolds.

Results: We propose a natural formulation of the contig scaffolding problem that places no limits on scaffolding information sources and operates with a continuous measure of order and distance scaffolding constraint satisfiability, in the form of a Mixed-Integer Quadratic Programming problem. A proof of concept algorithm based on an expectation-maximization procedure and an Unconstrained Binary Quadratic Programming approximation of the original problem was implemented and evaluated on simulated scaffolding problems with real and simulated paired reads, showing promising results for small scaffolding problems (number of contigs $n < 400$) in several experimental setups.

Availability: <http://code.google.com/p/tud-scaffolding>

Contact: grizenko.a@gmail.com

1. INTRODUCTION

High-throughput sequencing (HTS) technologies, such as offered by Illumina (Illumina, Inc.), 454 (454 Life Sciences) and IonTorrent (Life Technologies) produce millions of short DNA reads with typical lengths of 36-500 bp. Using assembly algorithms (Zerbino and Birney, 2008; Miller *et al.*, 2008; Simpson *et al.*, 2009; Peng *et al.*, 2010) these reads can be joined into longer sequences called *contigs* (see Figure 1 (a)). Contigs are usually shorter than the sequenced chromosomes, as it is impossible to unambiguously assemble a genome with long repeats (i.e. longer than the read length) because the order of its non-repeat regions can not be fixed without additional information. Scaffolding of assembled contigs is a crucial step for obtaining high quality draft genomes. It is concerned with joining contigs into longer supercontigs with gaps called *scaffolds* and orienting, ordering contigs from these scaffolds and determining gap sizes between consecutive contigs (see Figure 1 (b)). Scaffolding is enabled by utilization of additional data, such as paired reads (reads of known relative orientation and physical distance distribution).

A plethora of additional information is available for scaffolding, such as related genomes, restriction maps (maps of known restriction sites in the genome) and RNA-seq data (HTS of cDNA for obtaining information about sample RNA content). However, most scaffolders support only a single information source, missing out on the available information. Formally scaffolding is defined as a problem of maximizing the number of satisfied constraints (Huson *et al.*, 2002; Pop *et al.*, 2003). The conditions for considering constraints satisfied differ between scaffolders, but they are commonly binary (i.e. require exact contig order and orientation and put a threshold on contig distances). Such an approach neither provides information about the degree of constraint sat-

isfaction which is useful for quality assessment of *de novo* constructed scaffolds, nor takes the degree of constraint satisfiability into account during optimization. The latter limits usefulness of the number of satisfied constraints as an indirect measure of expected scaffold correctness. Contig scaffolding (Huson *et al.*, 2002) and related problems (Kent and Haussler, 2001; Pop *et al.*, 2003) are known to be NP-hard. Therefore, scaffolding algorithms usually simplify the problem and use elaborate heuristics to obtain approximate solutions. A common approach of separately choosing contig orientation and order leads to suboptimal scaffolds.

Many scaffolding algorithms have been developed. Most modern assemblers use paired reads to perform scaffolding following the assembly process: Velvet (Zerbino and Birney, 2008), CABOG (Miller *et al.*, 2008), IDBA (Peng *et al.*, 2010). While this allows for exploiting information about read placement given by the assembly during the scaffolding stage, such an approach lacks the ability to use additional data sources. To our knowledge Bambus (Pop *et al.*, 2003) is the only standalone (i.e. not coupled with an assembler) scaffolder capable of using multiple data sources. However, other standalone scaffolders for paired read data (Huson *et al.*, 2002; Dayarian *et al.*, 2010; Gao *et al.*, 2011; Boetzer *et al.*, 2011) and optical restriction maps (Nagarajan *et al.*, 2008) exist. To tackle the complexity of the problem Huson *et al.* (2002) implemented a greedy scaffold merging heuristic; Pop *et al.* (2003); Dayarian *et al.* (2010) find orientation and order of contigs separately, where the underlying subproblems are solved using greedy algorithms or meta-heuristics; and Boetzer *et al.* (2011) propose a greedy scaffold extension solution. The only exception is the algorithm of Gao *et al.* (2011), a fixed-parameter tractable scaffolding algorithm for fixed paired read insert size and minimum contig length based on a different definition of the contig scaffolding problem. Their definition considers a scaffolding constraint on two contigs satisfied if the contigs have the desired orientation and order and the distance between them is below a fixed threshold (i.e. no lower bound on gap sizes is used). None of the existing algorithms combines all the features desired in a scaffolder, signifying the need for a new scaffolding approach.

We propose a novel formulation of the contig scaffolding problem as a *Mixed-Integer Quadratic Programming* (MIQP) problem suitable for multiple scaffolding information sources, which combines contig orientation, order and gap sizes in a single optimization criterion, implements continuous variables for scaffolding constraint satisfaction and assigns a single quality value to any scaffold - the value of the optimization objective function. We also propose a proof of concept optimization algorithm for solving the defined problem. An iterative expectation-maximization search which utilizes genetic algorithms for solving the *Unconstrained Binary Quadratic Programming* (UBQP) problems (Beasley, 1998) during the expectation phase for determining contig orientation and heuristics for solving large *Mixed-Binary Programming* (MBP) problems. The heuristic is based on solving related *Linear Programming* (LP) problems using the CPLEX (IBM ILOG, 2011) optimization package.

The algorithm was implemented in C++ and evaluated on simulated scaffolding problems for the bacterium *Escherichia*

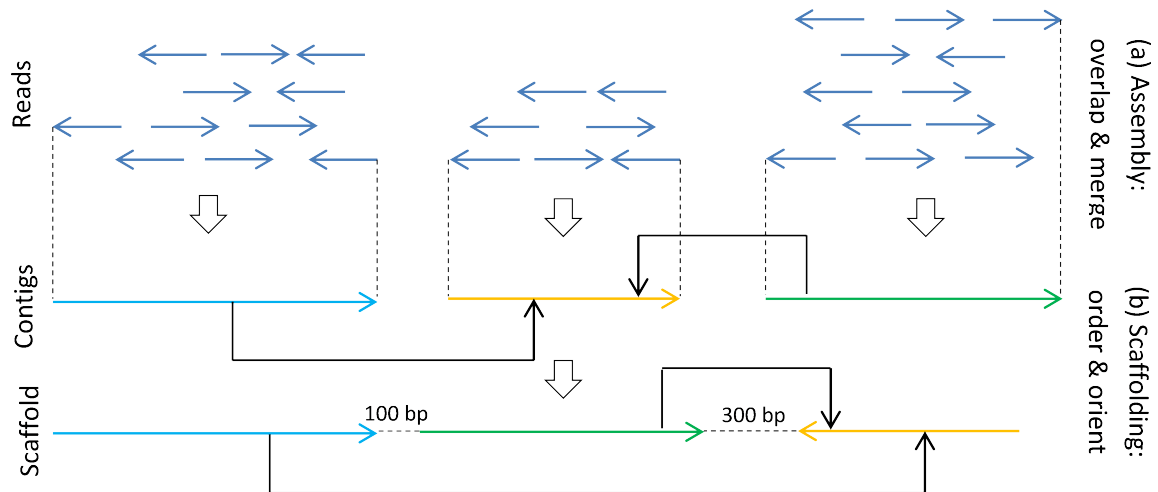


Figure 1: Sequence assembly and contig scaffolding work-flow. a) Sequencing reads (dark blue arrows) are joined together in contigs (light blue, orange and green arrows) based on sequence similarity in a process called assembly. As the originating DNA strands for the reads are not known, both read orientations are considered; therefore, the orientation of the assembled contigs is also not known, but forward orientation (depicted with arrowhead pointing to the 3' end) is assumed. b) The assembled contigs are ordered and oriented to produce scaffolds, for which the scaffolding constraints (black arrows) are satisfied.

coli strain K-12 MG1655. We used Illumina and 454 real and simulated paired reads as a scaffolding information source, but the implementation can be extended to use additional data. The algorithm showed promising results for small scaffolding problems, but was unsuccessful on real-sized problems. Detailed analysis of the algorithm's behavior in several experimental setups, identifying the causes of poor performance and potential solutions is offered.

2. METHODS

2.1 Data representation

Contig scaffolding is possible only with the help of additional information to devise constraints for orienting and ordering contigs and estimating gap sizes. It is convenient to define these constraints as pair-wise relations between contigs (later called *contig links*). Paired reads and paired RNA-seq data naturally define constraints for pairs of contigs (when paired reads map to different contigs), whereas related genome and restriction map information sources can be used to create contig links by considering neighbor-contigs in the alignment to the genome/map.

Gap size estimation requires availability of constraints on contig distances which are usually approximate: for paired reads they are derived using insert size which follows a Gaussian distribution; they are not available for paired RNA-seq reads due to intron splicing; in optical mapping the distances obtained contain errors (which are modeled as Gaussian distributions), resulting from the limited resolution of the optical device; and in case of alignment to a related genome only approximate distances can only be obtained, depending on the evolutionary distance between the organisms. For capturing the uncertainty inherent to the information sources, it is natural to define the distance constraints for contig pairs using Gaussian distributions.

To enable usage of multiple information sources, it is proposed to convert pair-wise contig relations to a common rep-

resentation as abstract contig links $l_i = (a_{l_i}, b_{l_i})$ of weight ω_{l_i} that suggest the relative orientation and order of contigs a_{l_i} , b_{l_i} and give the distance between them as a Gaussian distribution $N(\mu_{l_i}, \sigma_{l_i})$ with σ_{l_i} chosen with regard to the data source (e.g. it is necessary to a large σ_{l_i} for RNA-seq data). The weight ω_{l_i} reflects the importance or ambiguity of the constraint (e.g. the level of trust in the information source) and is used in *link bundling*, a procedure for combining several links that agree on contig order, orientation and distance into a single link of higher weight. Each scaffolding information source requires its own procedure for conversion into abstract contig links. This paper focuses on paired reads, as they are the most prominent scaffolding information source; the conversion procedures for paired end reads and mate pair reads are given in Section A.

Since DNA is double-stranded, scaffolders are free to reconstruct any of the two strands (see Figure 2). The strands of input contigs or the strands for which the contig links are derived are not known (see description for Figure 1). It is, therefore, necessary to take this ambiguity inherent to the problem into account. This is done by considering *relative contig orientation* (i.e. equal if the contigs should come from the same DNA strand and opposite if they should come from different strands) and *relative order*, defined as the order of contigs a_{l_i} and b_{l_i} when contig a_{l_i} has forward orientation. For convenience of presentation relative order and orientation are written as

- $e_{l_i} = \begin{cases} 0, & a_{l_i} \text{ and } b_{l_i} \text{ must have opposite orientation} \\ 1, & a_{l_i} \text{ and } b_{l_i} \text{ must have equal orientation} \end{cases}$
- $r_{l_i} = \begin{cases} 0, & a_{l_i} \text{ must follow } b_{l_i} \\ 1, & b_{l_i} \text{ must follow } a_{l_i} \end{cases}$ when a_{l_i} has forward orientation.

To avoid dependence on contig orientation the distance between two contigs is defined as paired read external insert

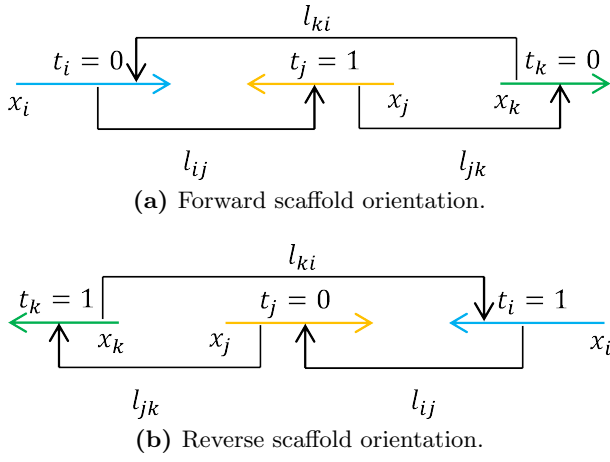


Figure 2: Variable assignment for forward and reverse orientations for a three-contig scaffold (see Section 2.2 for definition of variables t and x) with three links: $e_{l_{ij}} = 0$, $e_{l_{jk}} = 0$, $e_{l_{ki}} = 1$ and $r_{l_{ij}} = 1$, $r_{l_{jk}} = 0$, $r_{l_{ki}} = 0$. Contig links are satisfied in both forward (a) and (b) scaffold orientations.

size (i.e. the length of the gap between the contigs plus the contig lengths). Most other definitions of contig distance are not invariant to contig orientation (i.e. give different distance values for different contig orientations).

2.2 Optimization formulation

The traditional contig scaffolding problem formulation defines when contig links are satisfied and maximizes the number of such links. Our formulation is similar to the traditional definition of the contig scaffolding problem (Huson *et al.*, 2002), but modified to allow a continuous measure of contig link satisfaction. The proposed formulation is given as an MIQP problem with variables assigned to contigs and contig links. The integer (binary) variables naturally arise from the combinatorial nature of the contig scaffolding problem. To relax the NP-hard MIQP problem we substitute integer variables with continuous where possible. This section only defines the optimization variables and objective functions; the complete derivation is available in Section B.

2.2.1 Scaffold contigs

Given a set of contigs $C = \{c_1, c_2, \dots, c_n\}$ to be scaffolded we associate with every contig c_i a number of optimization variables (see Figure 2):

- $t_i = \begin{cases} 0, & c_i \text{ has forward orientation} \\ 1, & c_i \text{ has reverse orientation} \end{cases}$ is used to define contig orientations in the scaffold.
- $x_i \in \mathbb{R}^+$ corresponds to the 5' position of c_i in the scaffold (the input contigs and the constructed scaffold are viewed as having 5' to 3', i.e. forward orientation).

Naturally x_i should be an integer variable, but it is relaxed to simplify the problem and can be rounded during creation of the scaffold nucleotide sequence.

Assignment of coordinate variables to the 5' end of the contigs makes any distance measure that can be defined between two contigs dependent on the contig orientation. In fact, other coordinate assignments (for example, assigning the coordinate to the left end of the contig despite of its orientation) are also orientation-dependent. Due to this dependency care has to be taken when deriving distance constraints, as expressions for distance change depending on orientation (see Section B.1).

2.2.2 Contig links

Contigs from the previous subsection are connected by contig links $L = \{l_1, l_2, \dots, l_m\}$. Each contig link $l_i = (a_{l_i}, b_{l_i})$ enforces orientation, distance and order constraints on contigs a_{l_i} and b_{l_i} , as described as in Section 2.1. With every link we associate optimization variables:

- $\xi_{l_i} = \left\{ \begin{matrix} \vec{\xi}_{l_i}, \overleftarrow{\xi}_{l_i} \end{matrix} \right\}, \Delta_{l_i} = \left\{ \begin{matrix} \vec{\Delta}_{l_i}, \overleftarrow{\Delta}_{l_i} \end{matrix} \right\} \in \mathbb{R}^{+2}$, distance and order constraints' slack variables correspondingly for forward and reverse contig pair orientations. These variables allow for violation of the optimization constraints and are penalized in the objective function depending on the orientation of contig pairs (a_{l_i}, b_{l_i}) .
- $\alpha_{l_i}, \beta_{l_i} \in \{0, 1\}$, distance and order constraints' switch variables (0, constraint disabled; 1, constraint enabled) used for disabling the optimization constraints when their associated penalties are higher than the maximum allowed penalties S_ξ (for distance constraints) and S_Δ (for order constraints).

Due to association of binary variables with contig links, the number of links becomes a critical parameter influencing the optimization search space size. A modified version of the link bundling procedure in Huson *et al.* (2002); Gao *et al.* (2011) is used to minimize the number of contig links (see Section D).

2.2.3 Objective function

Box 1 (a) summarizes the objective function used in this formulation. The function consists of three parts responsible for contig orientation, order and distance correspondingly. Function $g(t)$ adds weights of contig links with satisfied orientation to the objective and functions $h(t, \vec{\xi}, \overleftarrow{\xi}, \alpha)$ and $p(t, \vec{\Delta}, \overleftarrow{\Delta}, \beta)$ penalize the objective function with penalties proportional to values of the slack variables ξ (distance constraints' slack variables) and Δ (order constraints' slack variables). The penalties are allowed to be at most $\frac{\omega_{l_i}}{2}$ so that together the distance and the order penalties are bounded by contig link weight ω_{l_i} .

The objective function does not directly depend on variables x_i , but their influence is expressed through the slack variables ξ_{l_i} and Δ_{l_i} which are connected to the former using optimization constraints (see Section B for constraint derivation). Functions $h(t, \vec{\xi}, \overleftarrow{\xi}, \alpha)$ and $p(t, \vec{\Delta}, \overleftarrow{\Delta}, \beta)$ in Box 1 do not depend on variables α_{l_i} and β_{l_i} , but these variables are introduced when expressions for the minimum (e.g. $\min \left(\overleftarrow{\xi}_{l_i}, S_\xi \right)$) are expanded (see Section B.4).

$$\begin{aligned}
f(t, \vec{\xi}, \overleftarrow{\xi}, \vec{\Delta}, \overleftarrow{\Delta}, \alpha, \beta) &= g(t) - h(t, \vec{\xi}, \overleftarrow{\xi}, \alpha) - p(t, \vec{\Delta}, \overleftarrow{\Delta}, \beta) \rightarrow \max && \text{Joined objective} \\
g(t) &= \sum_{i=1, \overline{m}}^{e_{l_i}=0} (t_{a_{l_i}} + t_{b_{l_i}} - 2t_{a_{l_i} b_{l_i}}) \omega_{l_i} + \sum_{i=1, \overline{m}}^{e_{l_i}=1} (1 - t_{a_{l_i}} + t_{b_{l_i}} - 2t_{a_{l_i} b_{l_i}}) \omega_{l_i} && \text{Orientation} \\
h(t, \vec{\xi}, \overleftarrow{\xi}, \alpha) &= \sum_{i=1, \overline{m}}^{e_{l_i}=0} q_{a_{l_i} b_{l_i}} \left((1 - t_{a_{l_i}}) \min(\vec{\xi}_{l_i}, S_\xi) + t_{a_{l_i}} \min(\overleftarrow{\xi}_{l_i}, S_\xi) \right) \frac{\omega_{l_i}}{2S_\xi} + && \text{Distance \& orientation} \\
&\quad + \sum_{i=1, \overline{m}}^{e_{l_i}=1} (1 - q_{a_{l_i} b_{l_i}}) \left((1 - t_{a_{l_i}}) \min(\vec{\xi}_{l_i}, S_\xi) + t_{a_{l_i}} \min(\overleftarrow{\xi}_{l_i}, S_\xi) \right) \frac{\omega_{l_i}}{2S_\xi} \\
p(t, \vec{\Delta}, \overleftarrow{\Delta}, \beta) &\equiv h(t, \vec{\Delta}, \overleftarrow{\Delta}, \beta) && \text{Order \& orientation}
\end{aligned}$$

(a) Original optimization problem. Expression $q_{a_{l_i} b_{l_i}} \equiv t_{a_{l_i}} + t_{b_{l_i}} - 2t_{a_{l_i} b_{l_i}}$ equals 1 when contigs a_{l_i} and b_{l_i} have opposite orientation, and 0 otherwise.

$$\begin{aligned}
f(x, \xi, \Delta, \alpha, \beta) &= g - h(\xi, \alpha) - p(\Delta, \beta) \rightarrow \max && \text{Objective} \\
g &= \sum_{i=1, \overline{m}}^{t_{a_{l_i}} \oplus t_{b_{l_i}} \neq e_{l_i}} w_{l_i} \equiv \text{const} && \\
h(\xi, \alpha) &= \frac{1}{2S_\xi} \sum_{i=1, \overline{m}} \min(\xi_{l_i}, S_\xi) \omega_{l_i} && \text{Distance} \\
p(\Delta, \beta) &= \frac{1}{2S_\Delta} \sum_{i=1, \overline{m}} \min(\Delta_{l_i}, S_\Delta) \omega_{l_i} && \text{Order}
\end{aligned}
\qquad
\begin{aligned}
f(x, \xi, \Delta) &= g - h(\xi) - p(\Delta) \rightarrow \max && \text{Objective} \\
g &= \sum_{i=1, \overline{m}}^{t_{a_{l_i}} \oplus t_{b_{l_i}} \neq e_{l_i}} w_{l_i} \equiv \text{const} && \\
h(\xi) &= \frac{1}{2S_\xi} \sum_{i=1, \overline{m}} \xi_{l_i} \omega_{l_i} && \text{Distance} \\
p(\Delta) &= \frac{1}{2S_\Delta} \sum_{i=1, \overline{m}} \Delta_{l_i} \omega_{l_i} && \text{Order}
\end{aligned}$$

(b) Fixed optimization problem. As the values for variables t_i are known, they are excluded from the optimization.

(c) Fixed optimization problem relaxation. Binary variables α_{l_i} and β_{l_i} are excluded from the optimization, as minima of the slack variables are not taken in the objective function.

Box 1: Objective functions for the proposed optimization problem formulations.

2.3 Problem splitting

The proposed MIQP formulation used with link bundling is intractable even for toy scaffolding problems (number of contigs $n < 10$) when approached directly (several unsuccessful attempts to solve it using the CPLEX optimization suite were made). Therefore, an approach that simplifies the problem by optimizing contig orientations and contig distances/order in alternating steps is proposed (Section 2.3.3).

2.3.1 Fixed optimization problem

Suppose the optimization problem formulated in Section 2.2 has a fixed contig orientation (i.e. the values of variables t_i are known). This *fixed optimization problem* summarized in Box 1 (b) (derivation is available in Section C), is a mixed-binary LP problem that is much simpler than the original problem, though still NP-hard. Its hardness comes from the binary variables α_{l_i} , β_{l_i} used in the expansion of terms $\min(\xi, S_\xi)$ and $\min(\Delta, S_\Delta)$. A heuristic based on solving a related continuous LP is proposed for obtaining values of the binary variables and ultimately a solution for the fixed optimization problem.

Fixed optimization heuristic. In this problem, binary variables play the role of switches determining which distance constraints (variables α_{l_i}) and order constraints (variables β_{l_i}) are taken into account during scaffold optimization. Therefore, the problem of obtaining their optimal values is

identical to the problem of selecting a set of distance and order constraints that gives the optimum objective function value. A problem where all constraints are taken into account during scaffold optimization (i.e. $\alpha_{l_i} = \beta_{l_i} = 1$) can be used for deciding on constraints that should be disregarded by the optimization. Such a problem can be viewed as a relaxation of the original problem. It is equivalent to a fixed optimization formulation that instead optimizes the functions

$$\begin{aligned}
h(\xi) &= \frac{1}{2S_\xi} \sum_{i=1, \overline{m}} \xi_{l_i} \omega_{l_i} \\
p(\Delta) &= \frac{1}{2S_\Delta} \sum_{i=1, \overline{m}} \Delta_{l_i} \omega_{l_i}.
\end{aligned}$$

This problem is summarized in Box 1 (c). It constitutes a continuous LP problem with $(n + 2m)$ continuous variables and $3m$ constraints (see Section C) for which an optimal solution can be efficiently obtained using any suitable optimization package (CPLEX was used in this research). It is parsimonious to assume that the majority of the contig linking information will be correct, so in the optimal solution incorrect contig links (that should be disregarded by the optimization) will be associated with large values for the corresponding slack variables. It is therefore proposed

to choose values for α_{l_i} and β_{l_i} as

$$\alpha_{l_i} = \begin{cases} 0 & , \quad \xi_{l_i} > S_\xi \\ 1 & , \quad \xi_{l_i} \leq S_\xi \end{cases}$$

$$\beta_{l_i} = \begin{cases} 0 & , \quad \Delta_{l_i} > S_\Delta \\ 1 & , \quad \Delta_{l_i} \leq S_\Delta \end{cases}$$

and resolve the optimization problem with the new values. This can be done efficiently, as the new problem will be a continuous LP problem. The proposed heuristic allows obtaining a solution for the fixed optimization problem in polynomial time and performs quite well in practice (see Section 3.4 for an evaluation of the heuristic).

In practice, after resolving the problem with fixed values α_{l_i} and β_{l_i} , new links with penalties higher than the maximum can be identified. It is attractive to use an approach that iteratively disables constraints with high penalties and re-solves the problem, however it leads to aggressive link disabling behavior, as already after the first iteration re-enabling some of the disabled constraints increases the objective value. Therefore, a non-iterative (i.e. performing only one iteration) heuristic was chosen.

Scaffold extraction. For a set of contig links $L = \{l_i\}$ a contig multi-graph can be constructed by adding node c_i to the graph for every contig and an undirected edge (a_{l_i}, b_{l_i}) for every bundled link. In absence of information joining the connected components of this graph, each component must correspond to one or more separate scaffolds. However, an additional post-processing step is required for solutions of the fixed optimization problem, as depending on the values of α_{l_i} and β_{l_i} some contig links may be disabled. For caution we consider a contig link disabled if at least one of its constraints is not taken into account (i.e. $\alpha_{l_i} = 0 \vee \beta_{l_i} = 0$) and modify the contig graph accordingly. In practice this gives more accurate scaffolds than disabling a contig link iff $\alpha_{l_i} = 0 \wedge \beta_{l_i} = 0$, as more fragmented scaffolds lead to less potential problems. Separate scaffolds are then extracted for each connected component of the modified graph.

2.3.2 Unconstrained Binary Quadratic Programming

The fixed optimization formulation and heuristic provide a means of solving the contig scaffolding problem when contig orientations are known. Consider the reverse problem, when contig link penalties (determined by slack variables Δ_{l_i} , ξ_{l_i} and switches α_{l_i} , β_{l_i}) are known or equal to zero and the optimal contig orientation is asked for. It is implicitly assumed that the complete set of penalties corresponds to some valid assignment of contig coordinates. As link penalties are bounded by the link weights, the case of non-zero penalties can be easily transformed into the case of zero penalties by subtracting penalties from the link weight. From here on only the zero penalty case is considered. The proposed problem is artificial, because link penalties indirectly depend on contig orientation and on each other and, therefore, cannot be fixed. Nonetheless it provides the necessary elements for an expectation-maximization algorithm.

In the case when all penalties are equal to zero the proposed

MIQP formulation (Section B) is reduced to optimizing

$$f(t) \equiv g(t) = \sum_{i=1, \overline{m}}^{e_{l_i}=0} (t_{a_{l_i}} + t_{b_{l_i}} - 2t_{a_{l_i}} t_{b_{l_i}}) \omega_{l_i} + \sum_{i=1, \overline{m}}^{e_{l_i}=1} (1 - t_{a_{l_i}} - t_{b_{l_i}} + 2t_{a_{l_i}} t_{b_{l_i}}) \omega_{l_i} \rightarrow \max$$

free of any constraints. Thus the problem is to find an assignment of binary variables t_i that maximizes quadratic function $f(t)$. This can be converted into a UBQP problem which is concerned with maximizing function $q(x) = x^t Q x$, where x is a binary vector of length n and Q a quadratic real $n \times n$ matrix (Beasley, 1998; Merz and Freisleben, 1999; Katayama *et al.*, 2000; Merz and Katayama, 2004). The conversion requires constructing a matrix that defines a quadratic form equivalent to function $f(t)$. Starting from a zero matrix Q and $C = 0$ it can be constructed by updating it for every contig link as

- For $e_{l_i} = 0$:

$$q_{a_{l_i} a_{l_i}} \leftarrow q_{a_{l_i} a_{l_i}} + \omega_{l_i}$$

$$q_{b_{l_i} b_{l_i}} \leftarrow q_{b_{l_i} b_{l_i}} + \omega_{l_i}$$

$$q_{a_{l_i} b_{l_i}} \leftarrow q_{a_{l_i} b_{l_i}} - 2\omega_{l_i}$$

- For $e_{l_i} = 1$:

$$q_{a_{l_i} a_{l_i}} \leftarrow q_{a_{l_i} a_{l_i}} - \omega_{l_i}$$

$$q_{b_{l_i} b_{l_i}} \leftarrow q_{b_{l_i} b_{l_i}} - \omega_{l_i}$$

$$q_{a_{l_i} b_{l_i}} \leftarrow q_{a_{l_i} b_{l_i}} + 2\omega_{l_i}$$

$$C \leftarrow C + \omega_{l_i}$$

The resulting matrix defines a quadratic form equivalent but not identical to function $f(t)$; the two are connected as $f(t) = t^t Q t + C$ (i.e. they differ by a constant) and therefore attain maxima for the same t .

UBQP problems arise in many fields and are well studied. Due to their NP-hardness heuristical and meta-heuristical (including simulated annealing and genetic algorithms) approaches were devised. Although more recent and elaborate strategies exist, for ease of implementation a genetic algorithm approach was used in this research. A memetic algorithm from Merz and Katayama (2004) capable of solving UBQP problems with 2500 variables through combination of randomized k -opt local search, an elaborate population initialization strategy and a specialized crossover operator was adopted, with minor changes. New individuals generated using the population initialization strategy were added to the population on every restart until the population reached the desired size of 40 individuals (a value reused from Merz and Katayama (2004)). A restart was triggered if there were no objective function improvements in the last 30 generations and the last restart was at least 30 generations ago.

Matrices Q corresponding to functions f of the MIQP formulation are typically very sparse (a consequence of scaffolds'

linear structure). Therefore, all enhancements for sparse matrices described in Merz and Katayama (2004) were implemented. Additionally, genetic operators applied to individuals independent of each other (local search, population generation, mutation and crossover) were implemented in parallel using OpenMP (Dagum and Menon, 1998).

2.3.3 Expectation-Maximization

The fixed optimization heuristic and the UBQP approach can be combined into an *Expectation-Maximization* (EM) - like procedure for solving the original MIQP formulation. In the expectation step, UBQP is used for obtaining expected contig orientations based on the current set of contig link penalty values. In the maximization step the penalty values are updated by solving the fixed optimization problem for contig orientations obtained using UBQP. The proposed procedure is outlined in four steps:

1. Associate with every contig link l_i penalty variables $\Theta_{l_i}^\xi$ (distance penalty), $\Theta_{l_i}^\Delta$ (order penalty) and set them to zero. These variables are used to keep track of the last obtained constraint penalties values for all contig links (i.e. even if the latter are not present in the current fixed optimization problem).
2. Solve the UBQP problem with the current penalty values $\Theta_{l_i}^\xi$ and $\Theta_{l_i}^\Delta$ to obtain contig orientations t_i (obviously no penalty is expected the first time orientation is determined).
3. Solve the fixed optimization problem for the orientations found in step 2 and update link penalties for all links with satisfied relative orientation by setting them equal to the corresponding penalties obtained by solving the fixed optimization problem (see Section B.4 for penalty derivation):

$$\Theta_{l_i}^\xi \leftarrow \begin{cases} \frac{\omega_{l_i}}{2} & , \alpha_{l_i} = 0 \\ \frac{\xi_{l_i}}{2S_\xi} \omega_{l_i} & , \alpha_{l_i} = 1 \end{cases}$$

$$\Theta_{l_i}^\Delta \leftarrow \begin{cases} \frac{\omega_{l_i}}{2} & , \beta_{l_i} = 0 \\ \frac{\Delta_{l_i}}{2S_\Delta} \omega_{l_i} & , \beta_{l_i} = 1 \end{cases}$$

4. The procedure is considered to be converged if the UBQP solution did not change from the last iteration. If not converged, go to step 2 and use the current values of t_i as a starting point for the search.

After this procedure, the contig coordinate values from the solution of the last fixed optimization problem together with contig orientations used in it define the obtained scaffolds. In 97% of experiments the EM procedure converged within 7 iterations, but occasionally a few more steps were required (at most 22 iterations). These statistic were gathered for problems of sizes $100 \leq n \leq 1500$ solved during algorithm evaluation.

In essence the EM algorithm is an iterative approximation of the original MIQP problem with an UBQP problem. The latter is a less difficult and better studied problem that can be efficiently solved even for large instances.

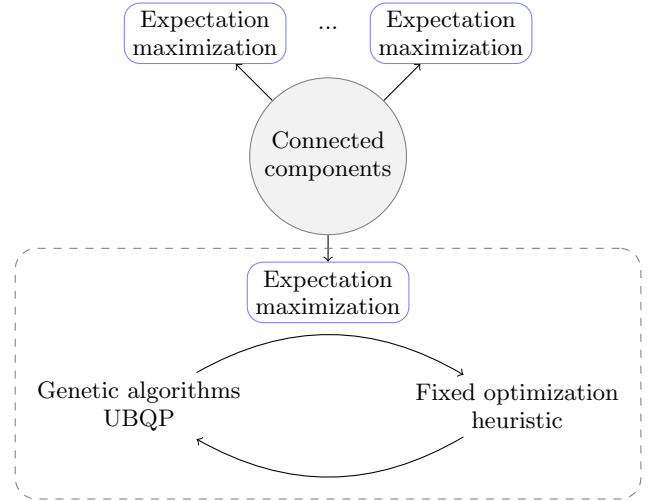


Figure 3: Schematic diagram of the proposed scaffolding algorithm.

2.4 Connected components

Section 2.3.1 describes a procedure for extraction of multiple scaffolds from a fixed optimization procedure using a contig multi-graph. This graph, constructed for the initial set of contigs and bundled links prior to scaffolding, can be used to separate the problem into smaller subproblems by extracting contigs and contig links for each connected component. The solution of the initial problem is then given by the union of subproblem scaffolds. This divide-and-conquer approach is possible because contigs in different connected components do not influence each other and can only be reconstructed into disjoint scaffolds.

The combined algorithm proposed for solving the formulated scaffolding problem is summarized in Figure 3.

2.5 Simulation

Simulated scaffolding problems were used for evaluation of the combined algorithm and its parts. The problems were created by selecting genome regions with known ground-truth sequence and introducing gaps in them; orientation of the resulting contigs was randomly flipped and the contigs were shuffled to allow for a realistic simulation. Illumina and 454 paired reads were used as the most abundant scaffolding information source. The simulation procedure for real and simulated reads is described in detail in Section G.

2.6 Evaluation

The problems were solved using the proposed algorithms to obtain one more scaffolds (referred to as the *found set of scaffolds*). They were compared to the ground-truth scaffolds given by the simulation. The ground-truth scaffolds were obtained by solving the fixed optimization problem with the orientation (variables t_i) and the coordinates (variables x_i) fixed at values given by the simulation procedure. This allows for obtaining a set of scaffolds, which are more similar to the best scaffolds that can be constructed from the available data (e.g. the ground-truth scaffolds obtained in such a way can be more fragmented due to absent or inconsistent linking information).

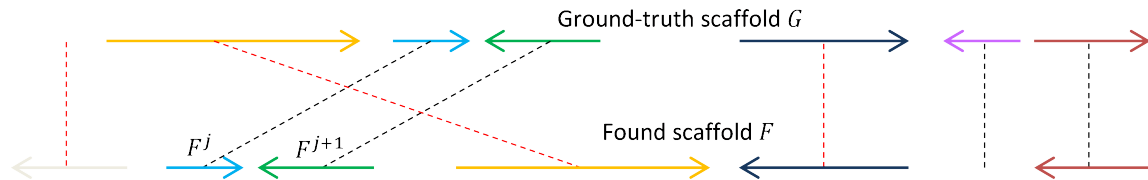


Figure 4: Breakpoints for the ground-truth (top) and the found (bottom) scaffolds. The scaffolds have 3 breakpoints: gray contig does not belong to the ground-truth scaffold; contig order in the green-orange pair is not conserved; and orientation of the dark blue contig in the orange-blue pair does not match. The blue-red pair has correct relative orientation and is not counted as a breakpoint.

Evaluation of the results requires a procedure for comparing the ground truth set of scaffolds to the found set of scaffolds based on correctness and completeness. The notion of scaffold correctness from Gao *et al.* (2011) was adopted. They define a procedure for comparing scaffolds to a genome, which is based on aligning scaffold contigs to the genome and considering consecutive pairs of contigs in the scaffold. An ordered pair is counted as a breakpoint if the order or orientation of its contigs are different in the genome. This procedure was modified to compare sets of scaffolds (see Section H for the complete description).

The new procedure (see Figure 4) counts a consecutive pair of contigs (F^j, F^{j+1}) as a breakpoint if:

1. F^j or F^{j+1} do not belong to the ground truth scaffold G ;
2. the order of (F^j, F^{j+1}) is different in G ;
3. the *relative* orientations of F^j and F^{j+1} in the found scaffold F and the ground truth scaffold G do not match.

It extends to comparing sets of scaffolds by matching every found scaffold to a ground truth scaffold (multiple found scaffolds can be matched to the same ground-truth scaffold) and counting breakpoints in the matches.

3. RESULTS

3.1 Implementation

Tools for contig scaffolding using the optimization formulation with the continuous constraint satisfiability measure and algorithm evaluation were implemented in C++. They perform the scaffolding problem simulation, conversion of paired read data into abstract contig links (with automated read alignment using BWA and Novoalign) and solving the contig optimization formulation. Solutions for the fixed optimization problems and their heuristic relaxations are found using the CPLEX Concert API for C++ (IBM ILOG, 2009). The orientation, order and coordinates of the found scaffolds are output by the solver in plain text format.

The tools are distributed under the MIT License. Source code is freely available from the SVN repository: <http://code.google.com/p/tud-scaffolding/>.

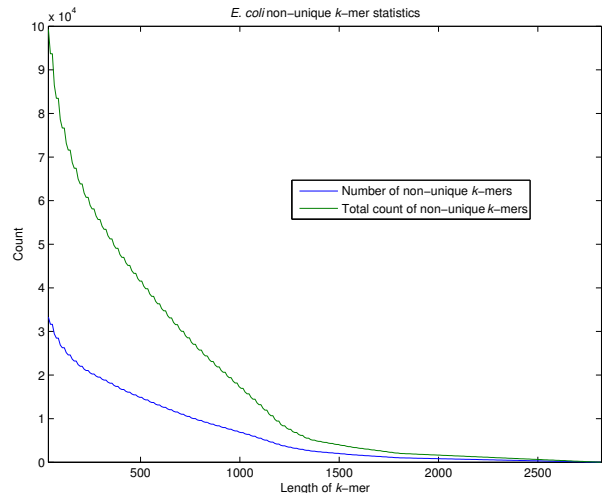


Figure 5: Repetitive k -mer counts for *E. coli*. The counts are based on the forward DNA strand of the genome. k -mer lengths in the range $[36; 2820]$ were analyzed.

3.2 Data

Bacterium *Escherichia coli* strain K-12 MG1655 data was used in evaluation of the proposed algorithms. Its complete reference sequence was obtained from GenBank (entry U00096.2) and read sets were downloaded from NCBI Short Read Archive. Read dataset statistics are presented in Table 1.

Figure 5 plots the number of non-unique k -mers and their total count against k -mer length. Although *E. coli* contains only one repeat (a two-fold repeat of length 2815 bp confirmed using REPuter (Stoye *et al.*, 2006) *de novo* repeat discovery software), its number of non-unique regions is much higher. This characteristic of the *E. coli* genome complicates scaffolding, as read lengths of the available datasets fall within the non-unique length region (i.e. ambiguous contig links will be derived from reads with multiple mapping locations).

3.3 Experimental setup

Several classes of scaffolding problems with different contig lengths and gap sizes (described in Table 2) were used for evaluation of the proposed methods: the Sim-split and Real-split setups approximating absence of contig gaps through introduction of 1 bp gaps; the Sim-small and Real-small setups with gaps that can be spanned by the majority of read pairs in the available datasets; and the Sim-large and

Table 1: Read set statistics for *E. coli*. Accession column gives the NCBI Short Read Archive (SRA) accession numbers.

Technology	Accession	Pairs	Read length	Coverage	Insert size, bp
Illumina	SRX000430	7,043,376	36 ± 0 bp	107	488.182 ± 18.11
454	SRX000348	62,654	108 ± 52 bp	2	3,278 ± 907.5

Table 2: Scaffolding problem simulation settings. Problems with simulated and real reads (read statistics not shown) were generated. Gaps following the specified Gaussian distribution were introduced such that lengths of the resulting fragments were greater than a given minimum length. Fragment overlaps were not allowed.

Reads	Name	Gap size, bp	Min length, bp	Illumina			454		
				Read length, bp	Insert size, bp	Coverage depth	Read length, bp	Insert size, bp	Coverage depth
Simulated	Sim-large	350 ± 100	1000	36 ± 0	488 ± 18	107	108 ± 52	3,278 ± 908	2
	Sim-small	50 ± 0	1000	36 ± 0	488 ± 18	107	108 ± 52	3,278 ± 908	2
	Sim-split	1 ± 0	100	36 ± 0	488 ± 18	107	108 ± 52	3,278 ± 908	2
Real	Real-large	350 ± 100	1000	N/A	N/A	N/A	N/A	N/A	N/A
	Real-small	50 ± 0	1000	N/A	N/A	N/A	N/A	N/A	N/A
	Real-split	1 ± 0	100	N/A	N/A	N/A	N/A	N/A	N/A

Real-large setups with gap sizes that are spanned by fewer reads. For every problem of n contigs a single region of length $(2500 \cdot n + 10000)$ bp was randomly chosen from the *E. coli* genome for introduction of $(n - 1)$ gaps. Simulated error-free reads in the problems mimicked the available read datasets from Table 1 (i.e. used similar read length and insert size distributions).

Problems with $101 \leq n \leq 1501$ contigs (with a step of 100) were considered. All problems were designed to contain a single scaffold spanning all contigs. However, due to uneven read coverage and large gap sizes, the problems often contained several scaffolds (see Section 2.6). Nevertheless the size of the largest scaffold was close to the number of contigs in the problem (on average the largest scaffold contained 92% of the contigs). Unless stated otherwise, only contig links derived from reads with high ($Q \geq 20$) mapping quality and a minimum length of 30 bp were kept in the optimization problems. The latter criterion filters out very short 454 reads. Mapping quality provided by BWA and Novoalign is a Phred-scaled estimate of the probability that the read is incorrectly mapped; a quality value of 20 corresponds to a 0.01 chance of incorrect mapping.

3.4 Fixed optimization heuristic evaluation

The heuristic approach used for solving the fixed optimization problem was evaluated separately from the combined algorithm. Problem instances were obtained by fixing contig orientations at their true values (i.e. t_i were set according to simulation). Scaffold correctness, contiguity and objective function value were then evaluated.

Figure 6 shows scaffold correctness for the Sim-split and Sim-small simulation setups, for which influence of ambiguous contig links is investigated. The Sim-small setup with the ambiguous contig links used in the optimization (referred to as Sim-small-ambiguous) shows scaffold correct-

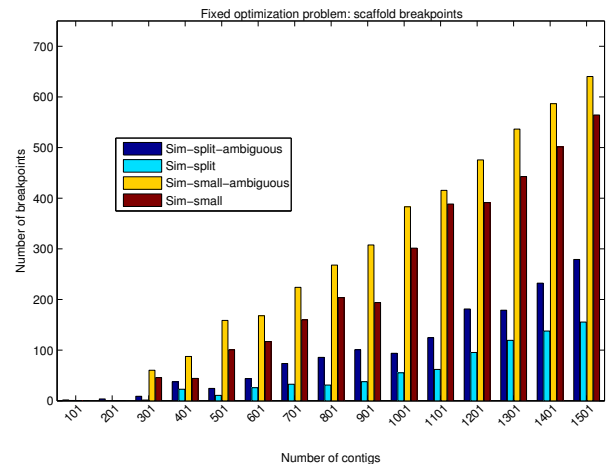


Figure 6: Scaffold correctness for simulated problems plotted as the average number of breakpoints over 10 scaffolding problems. Simulation setups are described in Table 2.

ness close to random. Mapping quality filtering was not using in the ambiguous setup, as quality filtering removes ambiguous links (reads with multiple mapping locations are assigned mapping quality of 0). The scaffolding errors result from incorrect or ambiguous linking data with high weights. Satisfying constraints corresponding to these links gives higher objective function values than the ground truth scaffold. Figure 23 shows a part of the ground-truth scaffold for a 10-contig scaffolding problem generated according to Sim-small-ambiguous. The solution found for this problem contains 3 breakpoints resulting from satisfying contig links that are disabled in the ground-truth solution (i.e. the dashed links originating from contig 5). These contig links are derived from paired reads with multiple mapping loca-

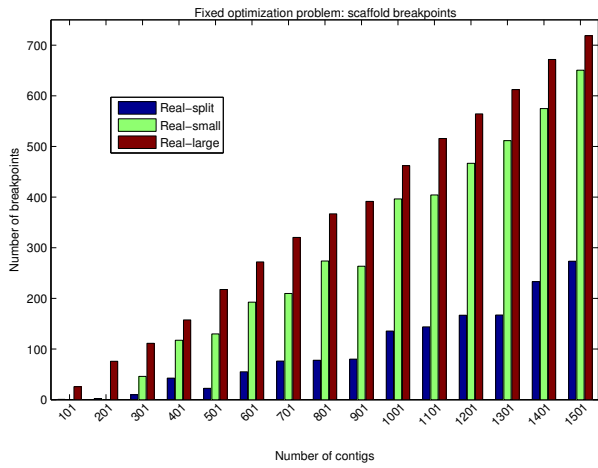


Figure 7: Scaffold correctness for simulated problems with real reads. See Figure 6 description.

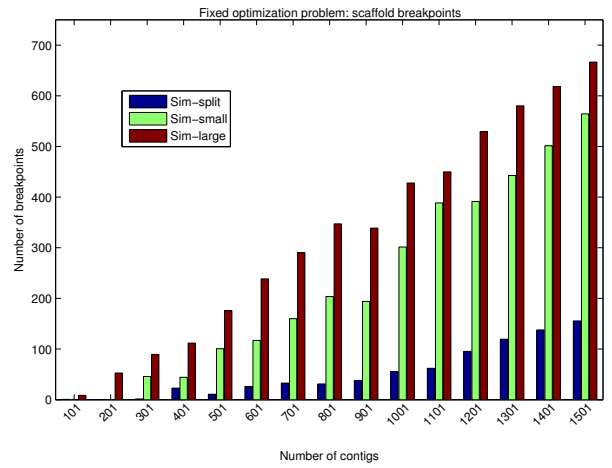


Figure 9: Scaffold correctness for simulated problems with simulated reads. See Figure 6 description.

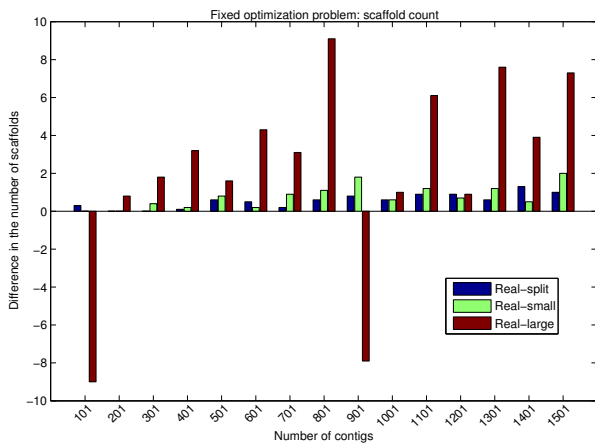


Figure 8: Scaffold contiguity for simulated problems with real reads. Average over 10 scaffolding problems of the found number of scaffolds minus the ground truth number of scaffolds is plotted.

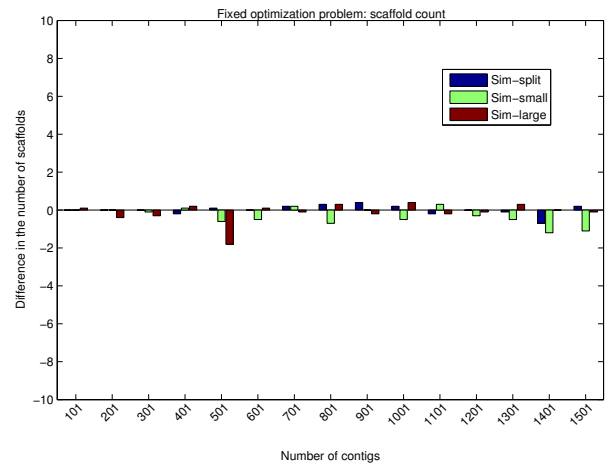


Figure 10: Scaffold contiguity for simulated problem with simulated reads. See Figure 8 for description.

tions and, therefore, are ambiguous. Removing ambiguous links in this example allows for finding an error free scaffold. The negative influence of ambiguous links is further investigated in the Sim-small setup, which solves the problems from Sim-small-ambiguous without the ambiguous links. Figure 6 shows an improvement (also seen for the Sim-split and Sim-split-ambiguous setups) in scaffolding accuracy when ambiguous links are filtered out. Therefore, all further experiments do not employ ambiguous links.

Scaffold correctness for problems with real and simulated setups is plotted in Figures 7 and 9 correspondingly. The problem with real and simulated reads have similar scaffold correctness, but the number of breakpoints on simulated reads is lower. This behavior is explained by the absence of read errors in simulated reads. There are two explanations for the high numbers of errors on the Real-large and Sim-large experiments. First, the large gap size used in simulation and the low coverage of reads with large insert size (454 data) do not allow for having high-weight links



Figure 11: Illustration of errors caused by large contig gaps. Gaps are depicted as suspension points.

with scaffolding constraints leading to a correct scaffold. Instead, numerous low-weight, contradictory links are present by read pairs mapping contigs by chance. These links cannot be satisfied and are disabled by the fixed optimization heuristic. Disabling links leads to more fragmented scaffolds, as confirmed by the scaffold size evaluation in Figure 8 where the Real-large setup clearly stands out.

Second, a negative effect on all scaffolds is that reads falling into gaps (or contig boundaries in the Real-split and Sim-split setups) are either aligned somewhere else in the scaffold or miss alignments which would identify them as ambiguous (see Figure 11). Failing to identify reads as ambiguous prevents from filtering (ambiguous) links derived from them or

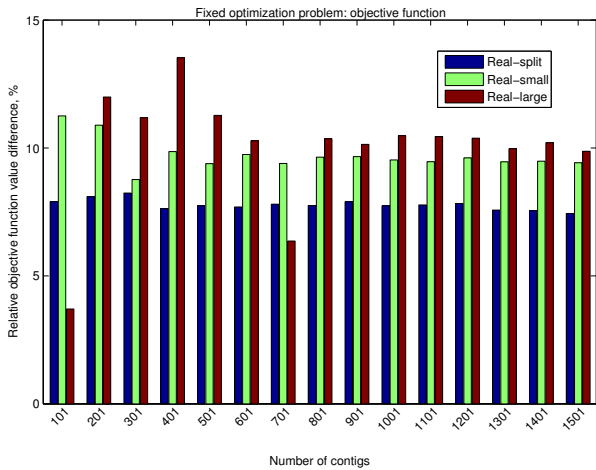


Figure 12: Optimization objective function for simulated problem with real reads. Average relative distance between objective function values corresponding to the found and the ground truth scaffold (calculated as $\frac{f_{\text{found}} - f_{\text{ground}}}{\max(f_{\text{found}}, f_{\text{ground}})}$) over 10 scaffolding problems is plotted.

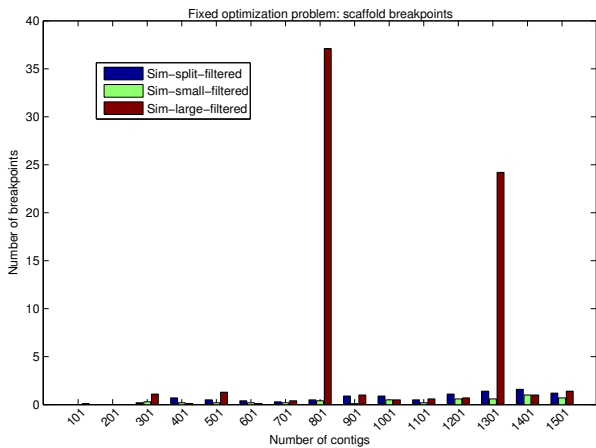


Figure 13: Scaffold correctness for simulated problems with filtered simulated reads plotted as the average number of breakpoints over 10 scaffolding problems.

from adequately decreasing link weights. This error source has an increasing influence on scaffolds with large gaps. The described effect was illustrated by performing additional experiments according in which all read pairs with at least one read originating from a contig gap were discarded. The results were obtained only for the problems with simulated reads (shown in Figure 13), as the originating read positions are not known for the real reads. The filtered simulated setups show nearly-error free scaffolds for considered all problem sizes, suggesting that the formulation can produce high quality scaffolds when the effects of ambiguous data are minimized.

In all experiments, the scaffolds found always gave a higher objective function value than the ground-truth solution (as shown in Figures 12 and 14; data for filtered setups is not shown due to similarity). This sometimes allowed for con-

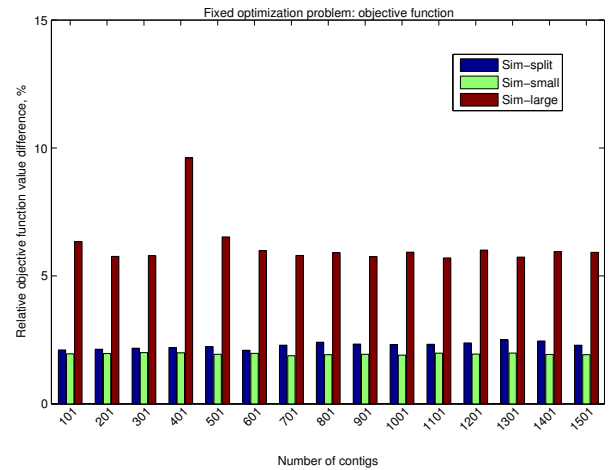


Figure 14: Optimization objective function for simulated problem with simulated reads. See Figure 12 for description.

structing more complete scaffolds than the ground-truth, although on average the ground truth solution and the found solution had the same number of scaffolds (see Figures 8 and 10; data for filtered setups is not shown due to similarity).

An in-depth evaluation of the heuristic part of the fixed optimization is performed in Section E. Although runtime measurements were not the aim of the experiments it can be noted that the running time depends on the number of contigs and contig links and goes up to 9 seconds for problems of size $n = 1500$.

3.5 Expectation-Maximization algorithm evaluation

The combined algorithm (from Figure 3) proposed for solving the scaffolding optimization problem was evaluated on simulated scaffolding problems from Section 3.4. To this end the EM algorithm was applied to every connected component of the problem. The amount of computation during the maximization part of the algorithm (based on GA) was limited by terminating the GA after 50 restarts. Scaffold correctness was also measured using the contig orientation mismatch between the ground truth set of scaffolds and the found set, similar to the breakpoint count (see Section H.2). Since contig orientations are determined using the EM procedure, considering orientation mismatch allows for evaluation of the EM algorithm separately from the fixed optimization heuristic.

Figures 15 and 17 shows numerous scaffolds with the number of breakpoints similar to that of random solutions. This is clearly seen for the setups Real-large, Real-small and Sim-small, although the setups Sim-large and Real-split show similar correctness for scaffolding problems larger than 200 contigs (Sim-large starts showing poor performance right away). The much better (although still poor) scaffold quality of the Sim-large setup compared to that of the Real-large setup can be explained by the large number of scaffolds it gives (see Figure 18). The number of breakpoints agrees with the orientation mismatch counts from Figures 19 and 20 suggesting that the EM procedure and GA are responsi-

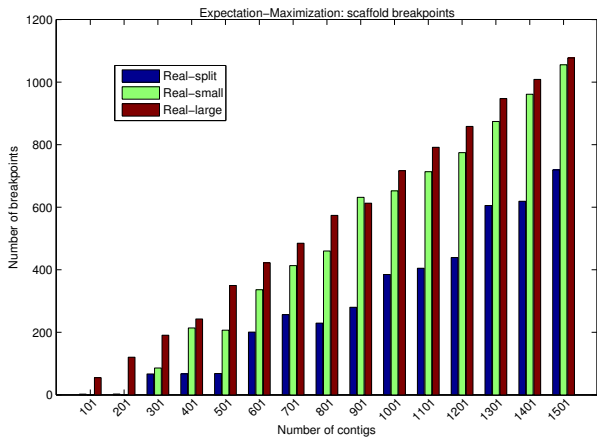


Figure 15: Scaffold correctness for simulated problems with real reads (scaffolds were obtained using the EM algorithm). See Figure 6 for description.

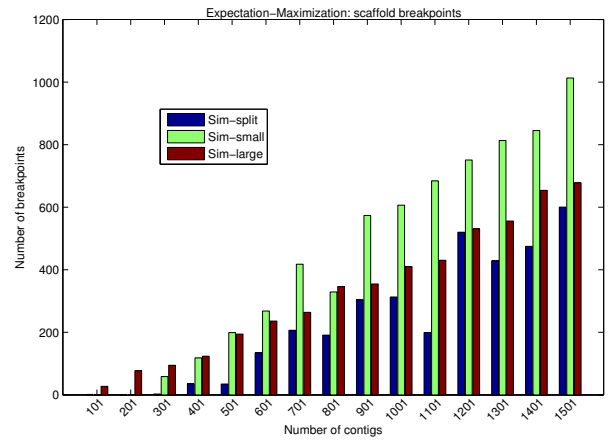


Figure 17: Scaffold correctness for simulated problems with simulated reads (scaffolds were obtained using the EM algorithm). See Figure 6 for description.

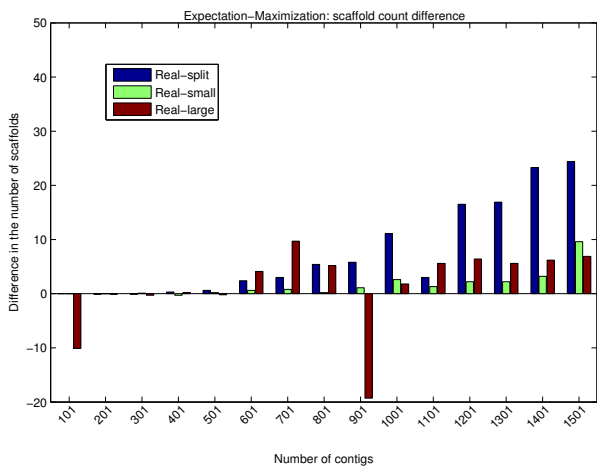


Figure 16: Scaffold contiguity for simulated problems with real reads (scaffolds were obtained using the EM algorithm). See Figure 8 for description.

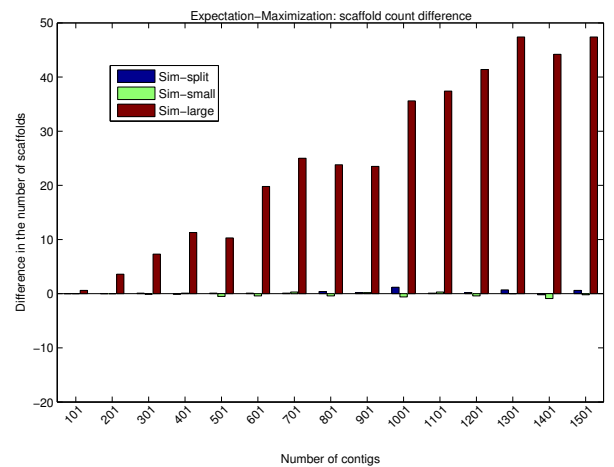


Figure 18: Scaffold contiguity for simulated problems with simulated reads (scaffolds were obtained using the EM algorithm). See Figure 8 for description.

ble for the poor scaffold quality.

Inability of the GA to find high quality UBQP solutions during the expectation step of the algorithm as a possible cause for poor scaffold quality was examined by repeating Sim-split experiments with a GA termination limit of 200 restarts (shown as Sim-split-long in Figure 21). The two setups do not yield significant differences in breakpoint count or orientation mismatch (the latter is not shown), suggesting that increasing the computational budget of the GA is will not yield significantly better result.

The arithmetic differences of objective function values obtained by the EM algorithm and the fixed optimization heuristic (used as described in Section 3.4) were computed (data shown in Section F). Most of the time these differences are positive, i.e. the EM algorithm finds solutions with slightly (relative differences smaller than 0.02% in magnitude) better objective function values. The low number of breakpoints (shown in Figure 22) and orientation mismatches for

the filtered setups with scaffold count similar to their unfiltered counterparts suggest that contig links derived from read pairs located in contig gaps or overlapping contig splits make it favorable to select contig orientations different from the ground truth. This is supported by the results for the Real-split and Sim-split setups, which have smaller numbers of reads falling into gaps (due to gap size of 1 bp) and on average give more accurate scaffolds than the other setups.

Although scaffold correctness for large scaffolding problems is too poor for real applications, the scaffolder produces relatively accurate results for small problems ($n < 400$). However, it cannot yet be applied to real scaffolding problems. Independent of problem size, the EM algorithm rarely required many iterations to converge. Nevertheless the GA used for solving the UBQP at the expectation step presents a computational bottleneck. The time spent on GA varied between setups (the amount of local search and the number of iterations between consecutive restarts are individual for every problem), but usually took ≈ 5 hours on large

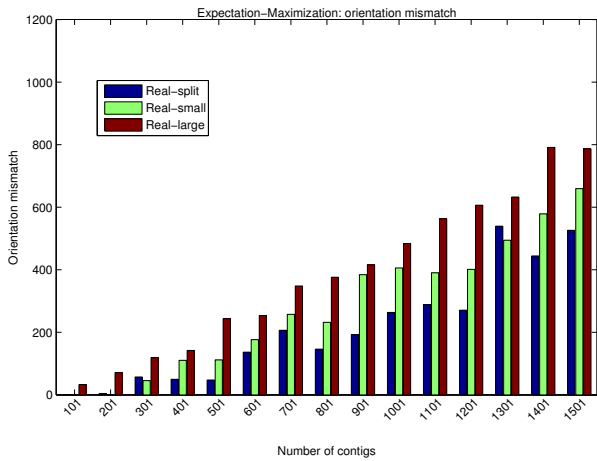


Figure 19: Scaffold orientation mismatch for simulated problems with real reads (scaffolds were obtained using the EM algorithm). The average number of orientation mismatches over 10 scaffolding problems is plotted.

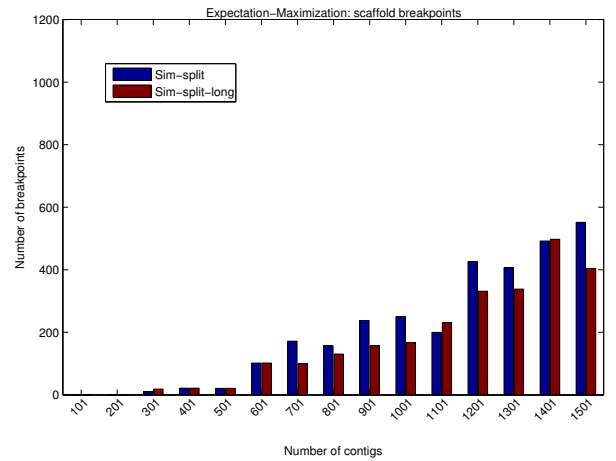


Figure 21: Scaffold correctness for simulated problems with increased GA computational budget. See Figure 6 for description.

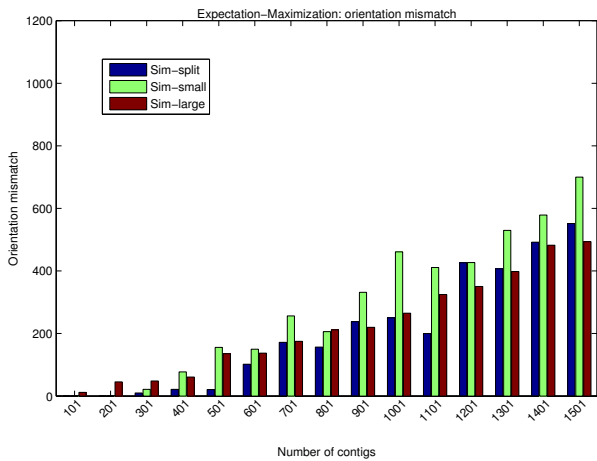


Figure 20: Scaffold orientation mismatch for simulated problems with simulated reads (scaffolds were obtained using the EM algorithm). See Figure 19 for description.

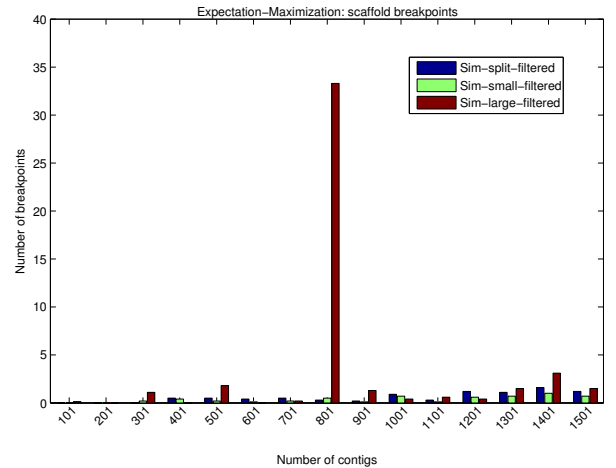


Figure 22: Scaffold correctness for simulated problems with filtered simulated reads (scaffolds were obtained using the EM algorithm). See Figure 6 for description.

problems ($n > 1400$). The GA was run in a single thread, so a speedup is possible if more cores are used as the GA algorithm is implemented using OpenMP.

4. DISCUSSION

We introduced a novel formulation of the contig scaffolding problem that enables a continuous measure of the degree to which scaffolding constraints are satisfied. As a MIQP problem, the formulation assigns objective function values to scaffolds, which can naturally be treated as scaffold quality measures in the absence of a ground truth. Neither the formulation of the optimization problem, nor the algorithms proposed for solving it, make assumptions concerning the origin of the abstract contig links. Thus our formulation is not limited to paired reads as the only information source and can easily be extended to other data, including strobe sequencing from Pacific Biosciences.

The proposed expectation-maximization approach provides the basis for separating the MIQP formulation into a linear constrained programming problem and an unconstrained quadratic programming problem. The latter is currently solved using genetic algorithms and presents a computational bottleneck. It can be overcome by using more efficient algorithms (such as MST2 (Nesterov, 1997) or GES (Pardalos *et al.*, 2008) capable of solving UBQP problems with 7000 variables) or devising an algorithm that specifically exploits the scaffolding nature of the UBQP problem.

Although in theory the proposed formulation is capable of constructing large scaffolds, in practice it is sensitive to ambiguous contig linking information with its increasing influence for larger scaffolds and scaffolds with large contig gaps. These complications only permit treating results of this research as a proof of concept that the novel formulation works. Further work should be focused on the disambiguation of contig linking information, by

- limiting read alignment to contig ends (Boetzer *et al.* (2011) use this as a speedup heuristic) in the paired read conversion procedure;
- repetitive read filtering using read k -mer frequency prior to the conversion procedure; and/or
- detection of reads with alignments spanning contig gaps and alignments with overhang. Such alignments are not supported by BWA or Novoalign. The formulation is particularly susceptible to ambiguous links derived from such reads

Furthermore, scaffolding accuracy is expected to improve with better sequencing data (longer paired reads and larger insert sizes, as produced by existing sequencing technologies), but additional experiments on simulated and real scaffolding problems are required to verify this.

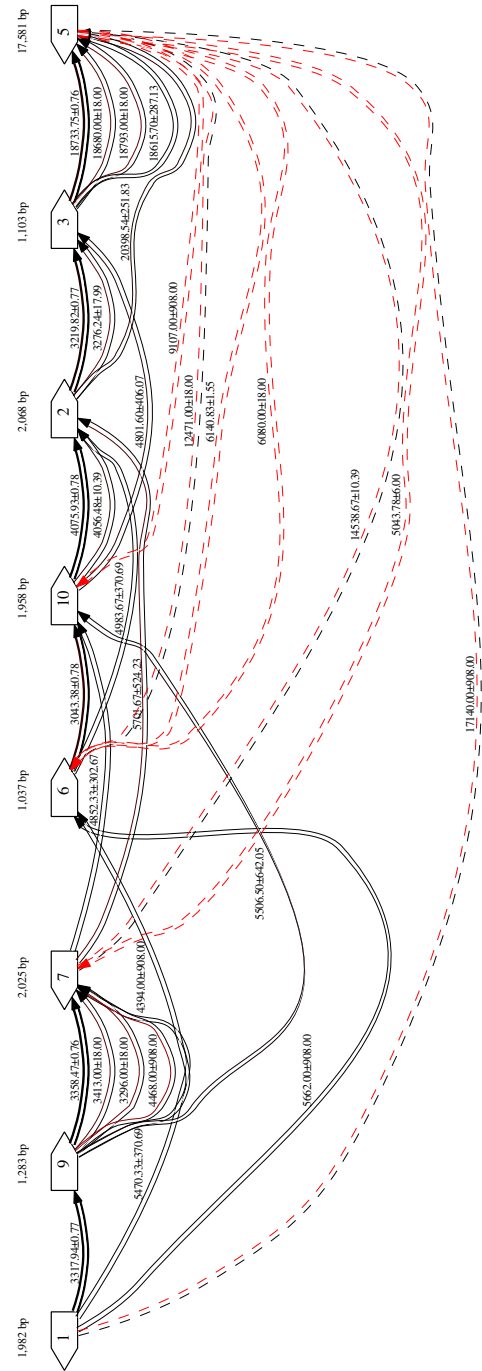


Figure 23: Ground-truth scaffold visualization (with ambiguous links). Contigs (depicted as rectangles with arrowheads defining orientation) are connected by contig links (double lines with arrowheads pointing to upstream contig). Thickness of contig links depicts contig relative weight and color shows the extent to which link constraints are satisfied (based on slack values, from black to red). The upper and lower lines correspond to distance and order constraints correspondingly. Links with disabled order or distance constraints are drawn using dashed lines. Slacks are obtained as described in Section 2.6. Visualization is done using the GraphViz (Gansner and North, 2000) package.

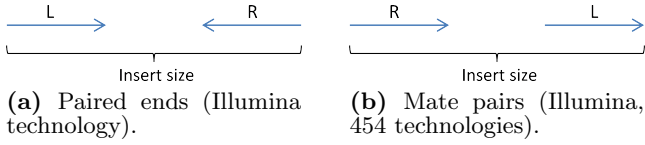


Figure A.1: Paired reads, coming from different technologies. L and R represent left and right reads correspondingly; curly brackets show paired read insert size.

Supplementary material

A. PAIRED READ INFORMATION CONVERSION

To convert paired read information into contig links, the left and right reads are separately mapped to the contigs. Genomic repeats allow for finding multiple equally good alignments for reads originating from these repeats. Therefore, considering all alignment hits allows for detecting reads falling into the repeats and reducing weight ω_{l_i} of contig links derived from them. BWA (Li and Durbin, 2009) and Novoalign (<http://www.novocraft.com/>) were used for aligning shorter NGS reads (Illumina, SOLiD) and longer reads (454) respectively as they are able to report multiple hits and output alignments in SAM format. The functionality of aligning reads, converting the alignments to BAM format and processing them was built into a converter application. BAM files were created from SAM files using SAMtools (Li *et al.*, 2009) and processed using the BamTools API (Barnett *et al.*, 2011).

After left and right reads are independently aligned to the contigs with all read mappings recorded, the alignments for every paired read library are processed separately. Paired reads, depending on the sequencing platform they originate from, have a certain orientation, order and distance separation. Figure A.1 shows the relations between paired reads for existing paired read protocols (paired reads and mate pairs). These relations between read pairs can be extended to the contigs to which these reads map (assuming the reads in a pair map to different contigs). In genome sequencing it is not possible to know from which of the two complementary strands a certain read pair arises. This has to be taken into account when read pairs are converted to contig links. The following sections describe constraints derived from read mapping. Only pair alignments, where two reads align to different contigs were considered. Non-unique alignments are treated in two ways:

- for a read pair whose left and right read have l and r alignments correspondingly, a total of $n = l \cdot r$ paired alignments are possible. If n is larger than the maximum number of allowed links per read pair (by default 5), the read pair is discarded;
- otherwise, every combination of single-end alignments is processed to produce a contig link. The weights of these links are divided by the number of combinations n . The links are also marked as ambiguous to allow for special treatment in the optimization formulation of the scaffolding problem.

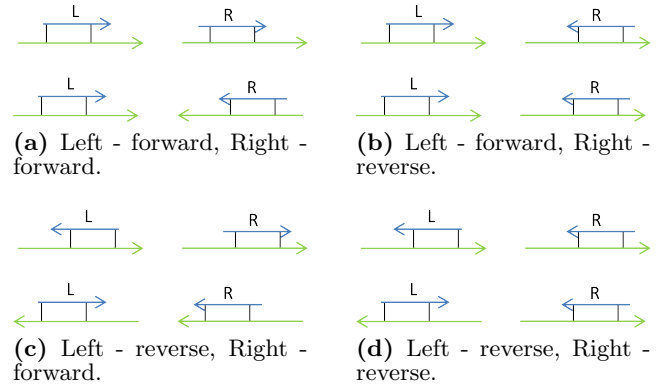


Figure A.2: Orientation derivation for paired ends. In every subfigure the upper picture shows the input mapping of paired reads (blue) to contigs (green) and the lower picture shows how the contig orientation must be changed in order to satisfy the paired end orientation constraints (the input contig orientation is assumed to be forward). In cases (a) and (d) contigs must have opposite; and in cases (b) and (c) equal relative orientations.

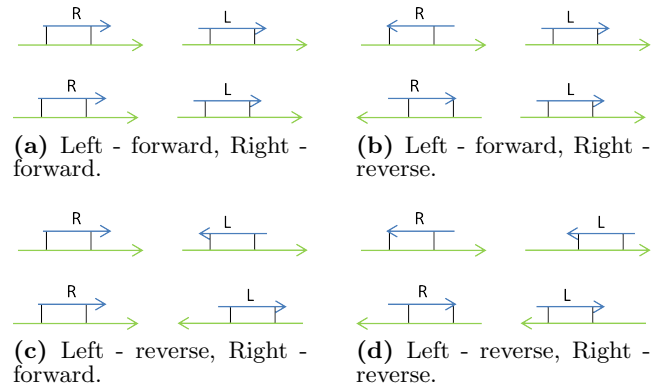


Figure A.3: Orientation derivation for mate pairs. See Figure A.2 for description. In cases (a) and (d) contigs must have equal; and in cases (b) and (c) opposite relative orientations.

A.1 Orientation constraints

Figure A.1 shows that in paired ends the left read has forward orientation and the right read has reverse orientation (relative to the DNA molecule they were sequenced from). Since the DNA molecule could in turn have originated from any of the two DNA strands of the genome, it can only be stated that paired reads have *opposite* orientation. Similarly mate pairs have *equal* orientation. Extending these read orientation constraints to contig orientation constraints is not straightforward and requires considering all possible read pair mapping orientations for mate pairs and paired ends. Figures A.2 and A.3 show derivation of orientation constraints for paired ends and mate pairs correspondingly. The idea behind the derivation is to flip the orientation of the contigs until the orientation of the reads mapped to them (which changes with the contig orientation) becomes correct.

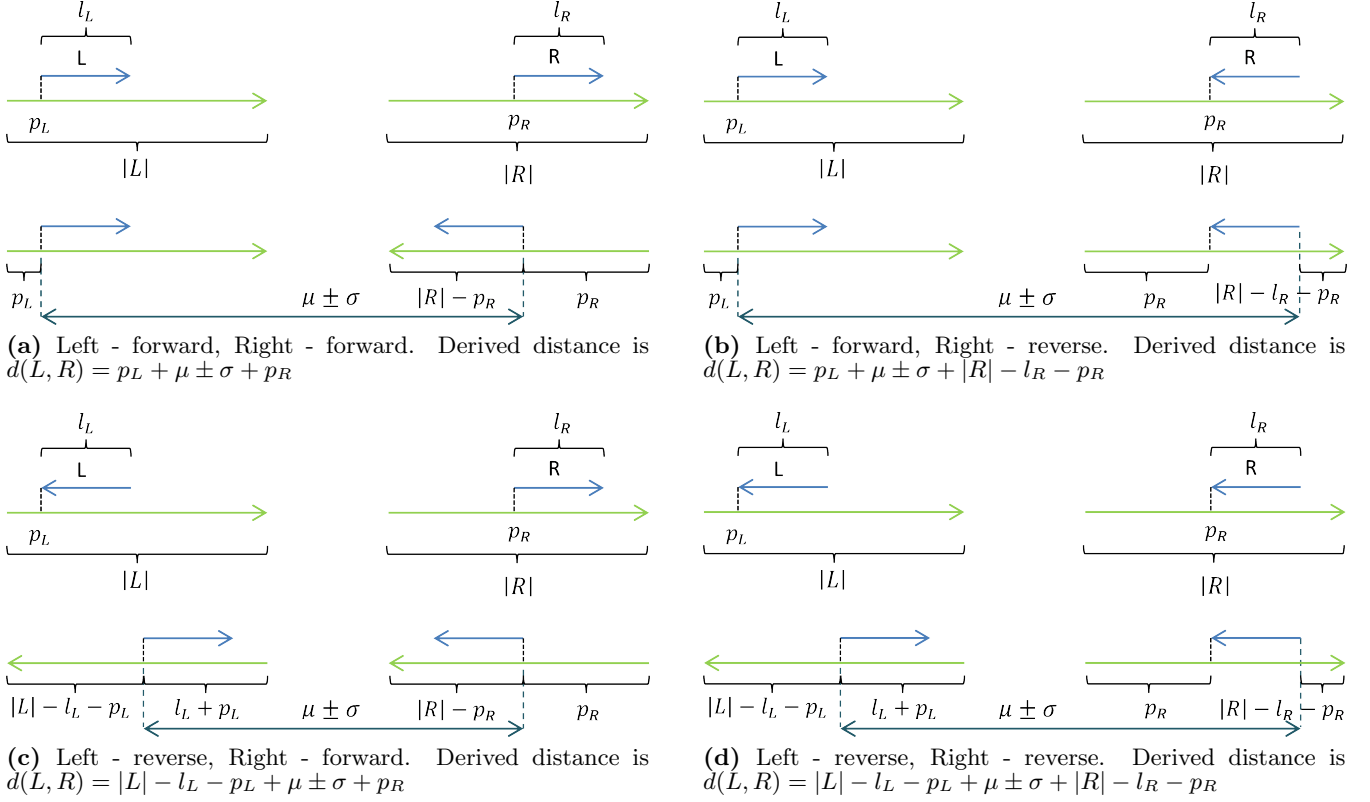


Figure A.4: Distance derivation for paired ends. See Figure A.2 for description. For ease of presentation Gaussian distributions $N(\mu, \Sigma)$ in distance expressions were replaced by $\mu \pm \sigma$.

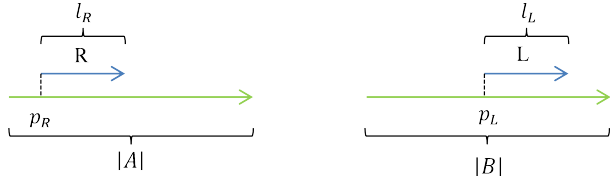


Figure A.5: Strict order constraints. Contigs and reads are depicted with green and blue arrows correspondingly. The read coordinate order constraint (for mate pairs) is given as $x_A + p_R + l_R < x_B + p_L$, where x_A and x_B are scaffold coordinates of contigs A and B correspondingly (5' ends). This gives the order constraint $x_B - x_A > p_R + l_R - p_L$ (note that the right side can also be negative). This derivation can be extended to paired ends and all other read/contig orientations.

A.2 Distance constraints

The insert size property of paired reads, describing the distance between the reads (including the read lengths) contains variations inherent to the process of creating the read library during sequencing. These variations are modeled using Gaussian distributions $N(\mu, \sigma)$ which can be used to derive distance distributions between the contigs aligning these reads. Recall that distance between contigs was defined as a natural extension of the insert size for paired reads. Such definition has the advantage of being invariant to contig order and orientation, allowing to define only a single distance

for a pair of contigs. However care has to be taken when calculating this distance from contig coordinates, as contig order and orientation change the formulas. This fact substantially complicates the optimization problem formulation (as addressed in Section B).

Since the insert size is defined for read pair orientations depicted in Figure A.1, which in general are not the orientation in which reads get mapped to contigs, the contig distance constraints will be given by different formulas for every combination of read orientation. Figures A.4 and A.6 give derivations for all possible read mapping orientations for paired ends and mate pairs correspondingly.

A.3 Order constraints

For paired reads it is known which of the two reads in the pair is upstream of the other. In *canonical read pair orientation* (as shown in Figure A.1) for paired end reads the right read must be upstream of the left read, whereas for mate pairs the left read must be upstream of the right read. These 'consecution' constraints can be transformed into read order constraints by requiring that coordinate of the right end of the downstream read is smaller than the left coordinate of the upstream read (i.e. that reads do not overlap). In turn, these read coordinate order constraints together with read mapping locations can be extended to contig coordinate order constraints as shown in Figure A.5. Such a strict extension however creates different contig coordinate order constraints for read pairs with consistent read coordinate or-

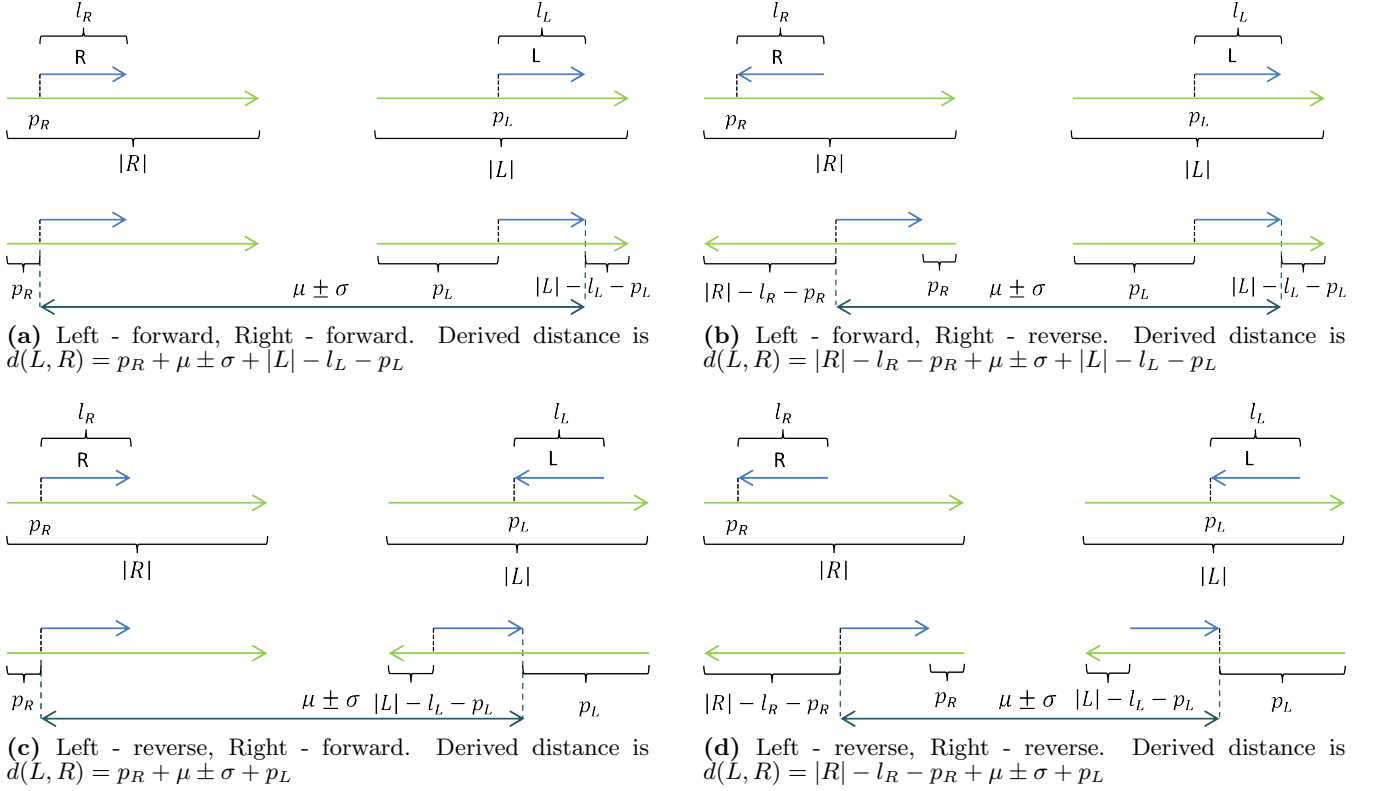


Figure A.6: Distance derivation for mate pairs. See Figure A.2 for description.

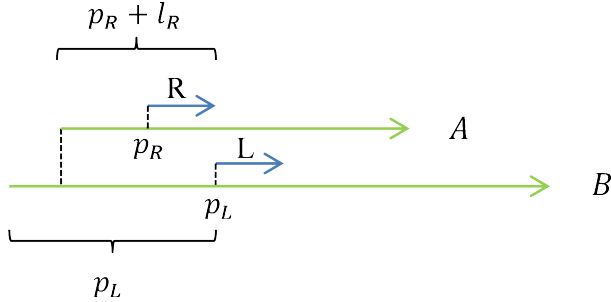


Figure A.7: Assumption used in the proposed contig coordinate constraint extension is that the depicted situation is not possible. Contigs and reads are depicted with green and blue arrows correspondingly. The read coordinate constraints (for mate pairs) are satisfied, whereas the contig order constraints are not: $x_B < x_A$ (the opposite must hold). Note that this situation occurs only when the left part of inequality in Figure A.5 is negative.

der constraints, which is undesirable as it makes contig link bundling problematic. Therefore, difficulty a different extension of read coordinate order to contig coordinate order is proposed: if contig c_i is downstream of contig c_j (written as $c_i < c_j$) then for coordinates of the left ends of these contigs y_{c_i} and y_{c_j} the inequality $y_{c_i} < y_{c_j}$ must hold. Note that this extension is less general than the one previously proposed and *assumes* that the case depicted in Figure A.7 is not possible.

The order constraints are dependent on the contig orientations and make sense only when the relative contig orientation constraints are satisfied. Figures A.8 and A.9 show possible order constraints for every combination of contigs with correct relative orientation for paired end and mate pair protocols correspondingly.

B. OPTIMIZATION FORMULATION

B.1 Distance constraints

Knowledge of the read pair distance distribution offers a continuous measure of link distance satisfiability and gives advantages over threshold or fixed interval formulations. Naturally, it is desired to use Gaussian probability distributions given by the contig links in distance constraints, however such constraint formulation does not lead to a MIQP. When contig distance for a link is given by a Gaussian distribution, it is desirable (i.e. gives higher probability value) that the distance is close to the link mean distance, but it can be further away. Moreover, it should be possible to violate optimization constraints when necessary (for instance, the constraints make sense only when contigs a_{i_i} and b_{i_i} have correct relative orientation). This implies using both the mean and the standard deviation of the distribution in the optimization constraints. The following expression is used for the constraints:

$$\frac{|d(a_{i_i}, b_{i_i}) - \mu_{i_i}|}{\sigma_{i_i}} \leq \xi_{i_i}, \quad (1)$$

where $d(a_{i_i}, b_{i_i})$ is the distance between contigs a_{i_i} , b_{i_i} and $\xi_{i_i} \geq 0$ are slack variables allowing the constraint to be

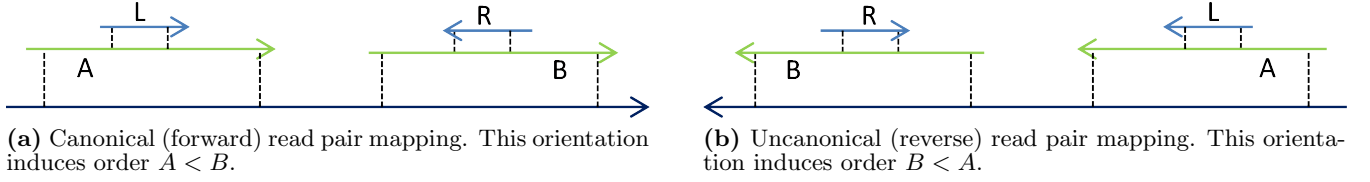


Figure A.8: Order derivation for paired ends. In every subfigure mapping of paired reads (blue) to contigs (green) and the genome (dark blue) is shown. It is assumed that the given contig orientation satisfies paired read relative orientation constraint.

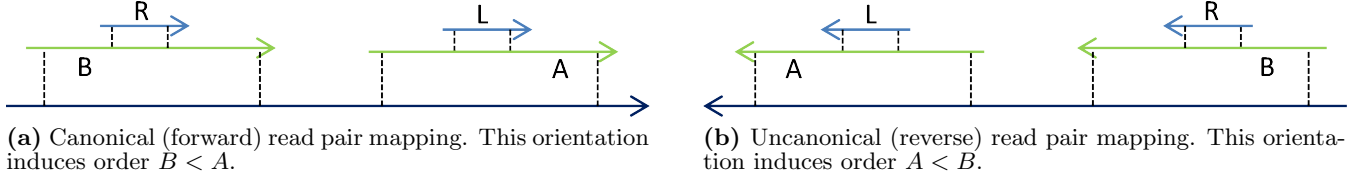


Figure A.9: Order derivation for paired ends. See Figure A.8 for description.

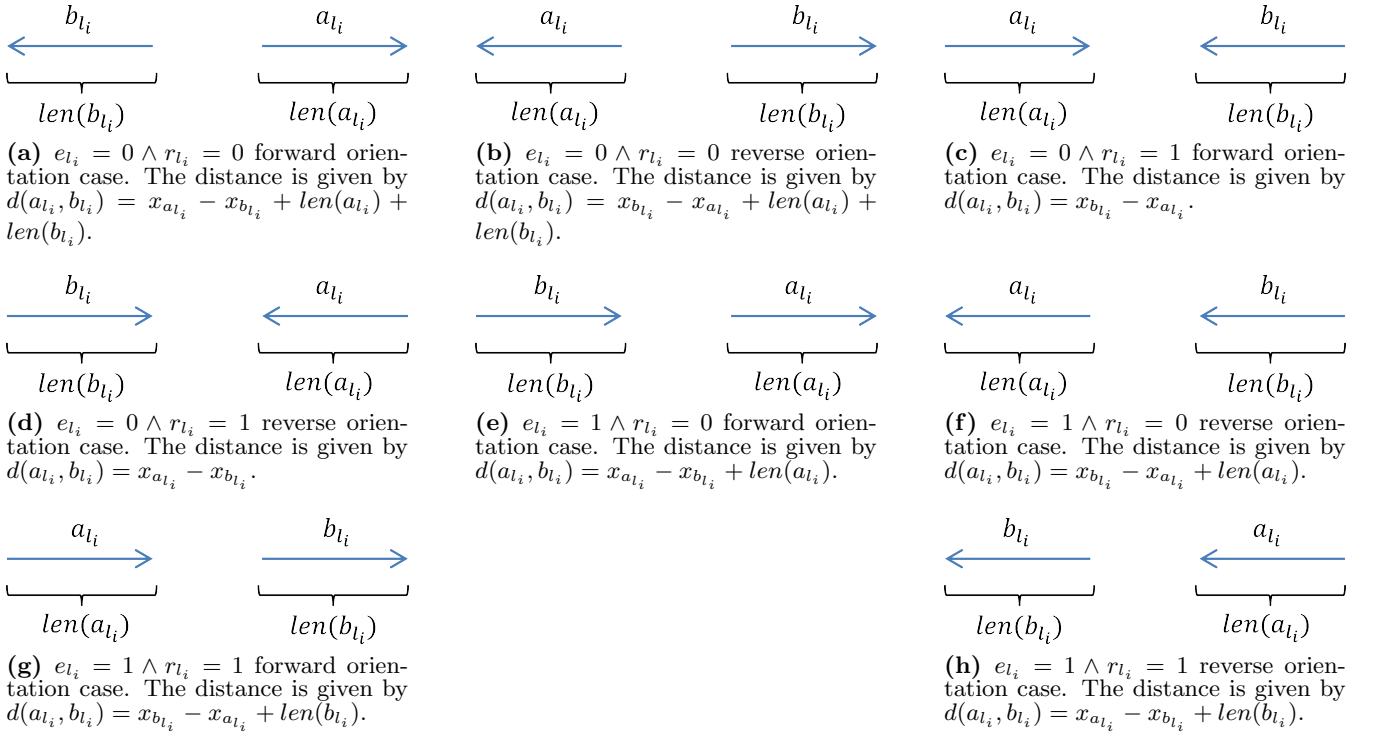


Figure B.1: Contig distance expression derivation for optimization constraints. Blue arrows represent contigs pointing from 5'-end to 3'-end. Subfigures (c) and (d) correspond to the paired end, whereas (e) and (f) correspond to mate pair protocols.

violated and indicating to what extent the constraint is violated (the slack variables should be penalized by the objective function). It is attractive to penalize the difference $\frac{|d(a_i, b_i) - \mu_{l_i}|}{\sigma_{l_i}}$ directly in the objective function, thus reducing the number of optimization constraints. However, such a formulation complicates further design of the optimization problem and requires additional constraints to cope with the absolute value sign. An alternative formulation using

the squared difference instead of the absolute value sign increases the complexity of the optimization function by increasing its degree.

The formula for $d(a_i, b_i)$ depends on contig orientations and order and its general case expression is not suitable for use in constrained programming. It is, therefore, proposed to make two assumptions concerning the contigs:

- contigs a_{l_i} and b_{l_i} have correct relative orientation e_{l_i} , a natural assumption as the distance constraint makes sense only if the relative orientation is satisfied;
- contigs a_{l_i} and b_{l_i} have correct order r_{l_i} , an assumption that allows simplifying the distance function expression (note that the formulation does not guarantee that this assumption holds).

Every combination of e_{l_i} and r_{l_i} gives rise to a different expression for $d(a_{l_i}, b_{l_i})$. These expressions are derived in Figure B.1. Note that they are also correct for the case when contigs overlap, but do not make sense if the contig order is violated (order of the left contig ends changes).

Differences in expressions for forward and reverse contig pair orientation prevent the formulation of a single distance expression for every pair of e_{l_i} and r_{l_i} . Instead separate constraints for two possible contig pair orientations are added to the optimization problem with different slack variables $\overset{\rightarrow}{\xi}_{l_i} \geq 0$ and $\overset{\leftarrow}{\xi}_{l_i} \geq 0$, corresponding to the *canonical* (when contig a_{l_i} has forward orientation, i.e. $t_{a_{l_i}} = 0$) and *un-canonical* (when contig a_{l_i} has reverse orientation, i.e. $t_{a_{l_i}} = 1$) contig pair orientations respectively. Later, only one of them must be penalized depending on the contig pair orientation. Under the assumption that the relative contig pair orientation is correct (which was made earlier), the orientation of the contig pair is given by the orientation of contig a_{l_i} , i.e. by the value of $t_{a_{l_i}}$ (or alternatively by the orientation of contig b_{l_i} and value of $t_{b_{l_i}}$). Thus for every combination of e_{l_i} and t_{l_i} inequality (1), which is equivalent to

$$\begin{cases} d(a_{l_i}, b_{l_i}) \leq & \sigma_{l_i} \xi_{l_i} + \mu_{l_i} \\ d(a_{l_i}, b_{l_i}) \geq & -\sigma_{l_i} \xi_{l_i} + \mu_{l_i} \end{cases}$$

can be written using Figure B.1 as:

- For $e_{l_i} = 0 \wedge r_{l_i} = 0$:

$$\begin{cases} x_{a_{l_i}} - x_{b_{l_i}} \leq & \overset{\rightarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & -\overset{\rightarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \leq & \overset{\leftarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & -\overset{\leftarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \end{cases}$$

- For $e_{l_i} = 0 \wedge r_{l_i} = 1$:

$$\begin{cases} x_{b_{l_i}} - x_{a_{l_i}} \leq & \overset{\rightarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & -\overset{\rightarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} \\ x_{a_{l_i}} - x_{b_{l_i}} \leq & \overset{\leftarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & -\overset{\leftarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} \end{cases}$$

- For $e_{l_i} = 1 \wedge r_{l_i} = 0$:

$$\begin{cases} x_{a_{l_i}} - x_{b_{l_i}} \leq & \overset{\rightarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & -\overset{\rightarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \leq & \overset{\leftarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & -\overset{\leftarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) \end{cases}$$

- For $e_{l_i} = 1 \wedge r_{l_i} = 1$:

$$\begin{cases} x_{b_{l_i}} - x_{a_{l_i}} \leq & \overset{\rightarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(b_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & -\overset{\rightarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(b_{l_i}) \\ x_{a_{l_i}} - x_{b_{l_i}} \leq & \overset{\leftarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(b_{l_i}) \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & -\overset{\leftarrow}{\sigma}_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(b_{l_i}) \end{cases}$$

B.2 Order constraints

In contig order optimization constraints the relationship “contig a_{l_i} must follow contig b_{l_i} ” is treated as in Sections A.3 and B.1. However since contig overlaps in assembly contigs are uncommon (as repeats are rarely resolved by the assemblers), and if they occur this can lead violation of the order constraints (e.g. in case when the constraints are derived from paired reads), the relationship is strengthened to forbid contig overlaps and is treated as described in Figure B.2. The general form for these constraints is $x_{a_{l_i}} - x_{b_{l_i}} \geq C$, where C is a constant (possibly equal to zero). As for distance constraints, order constraints can be partially violated; in some cases it should be possible to violate the constraint completely (e.g. inconsistent contig relative orientation). This can be done by introducing slack variables $\Delta_{l_i} \geq 0$ weighed by the length of the downstream contig for every constraint (the weight is used for normalization to have the slack variables on the same scale):

$$x_{a_{l_i}} - x_{b_{l_i}} \geq C - \Delta_{l_i} \cdot \text{len}(b_{l_i}). \quad (2)$$

As for distance optimization constraints, we assume that contigs a_{l_i} and b_{l_i} have correct relative orientation. This is a valid assumption as the order constraints make sense only when the orientation constraints are satisfied. It permits determining for a pair (a_{l_i}, b_{l_i}) whether or not it has canonical orientation and allows choosing one of the two opposite contig orders (see Figures A.8 and A.9).

The opposite contig order constraints arise from different contig pair orientations. For the cases described in Figure B.2 the following pairs of opposing constraints arise:

- For $e_{l_i} = 0 \wedge r_{l_i} = 0$:

$$\begin{cases} x_{a_{l_i}} - x_{b_{l_i}} \geq & -\text{len}(b_{l_i}) \cdot \overset{\rightarrow}{\Delta}_{l_i}, \quad t_{a_{l_i}} = 0 \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & -\text{len}(a_{l_i}) \cdot \overset{\leftarrow}{\Delta}_{l_i}, \quad t_{a_{l_i}} = 1 \end{cases}$$

- For $e_{l_i} = 0 \wedge r_{l_i} = 1$:

$$\begin{cases} x_{b_{l_i}} - x_{a_{l_i}} \geq & \text{len}(a_{l_i}) \cdot \left(1 - \overset{\rightarrow}{\Delta}_{l_i}\right) + \text{len}(b_{l_i}), \quad t_{a_{l_i}} = 0 \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & \text{len}(a_{l_i}) \cdot \left(1 - \overset{\leftarrow}{\Delta}_{l_i}\right) + \text{len}(b_{l_i}), \quad t_{a_{l_i}} = 1 \end{cases}$$

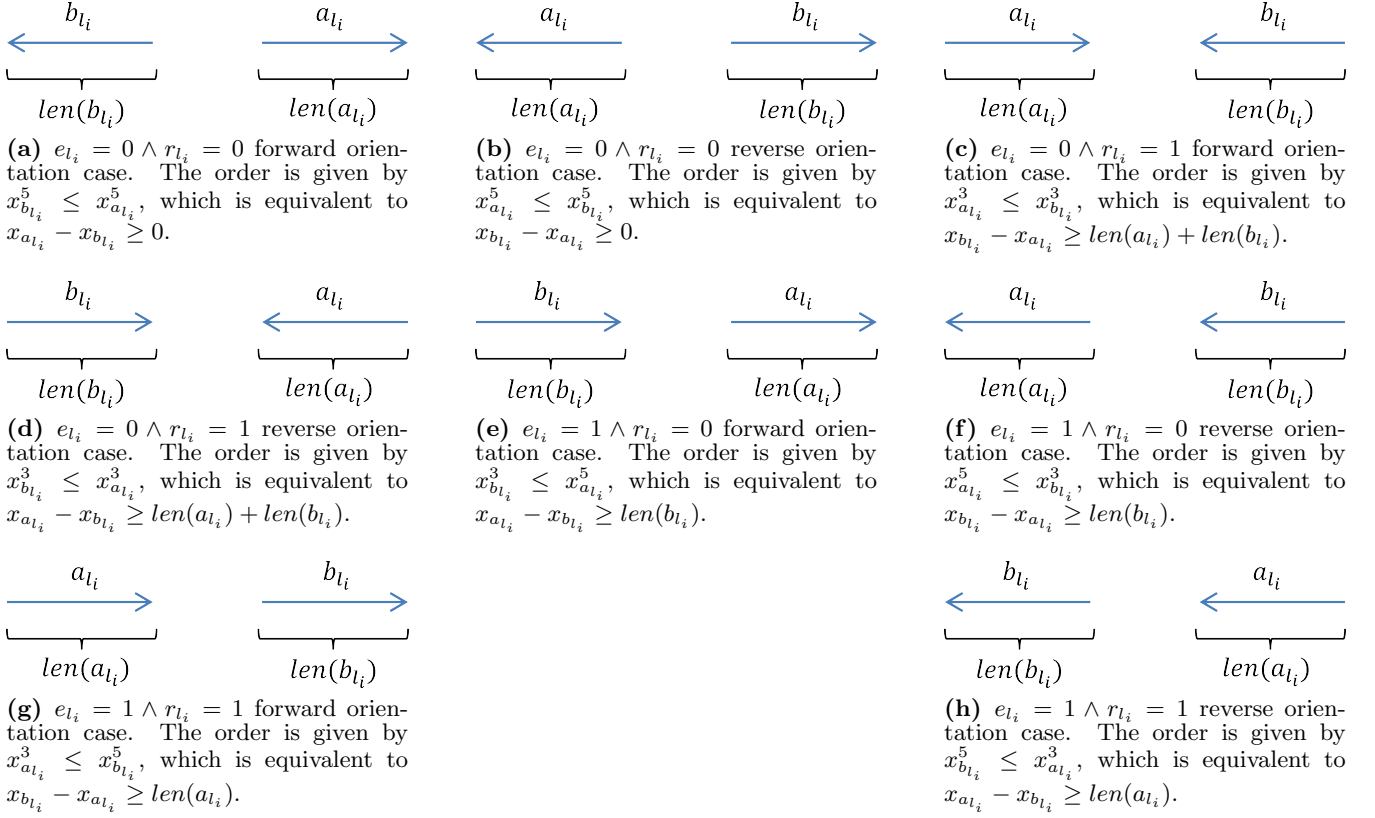


Figure B.2: Contig order expression derivation for optimization constraints. See Figure B.1 for description. Variables x_a^3 and x_a^5 denote 3' and 5' coordinates of contig a correspondingly.

- For $e_{l_i} = 1 \wedge r_{l_i} = 0$:

$$\begin{cases} x_{a_{l_i}} - x_{b_{l_i}} \geq \text{len}(b_{l_i}) \cdot (1 - \vec{\Delta}_{l_i}), & t_{a_{l_i}} = 0 \\ x_{b_{l_i}} - x_{a_{l_i}} \geq \text{len}(b_{l_i}) - \text{len}(a_{l_i}) \cdot \overleftarrow{\Delta}_{l_i}, & t_{a_{l_i}} = 1 \end{cases}$$

- For $e_{l_i} = 1 \wedge r_{l_i} = 1$:

$$\begin{cases} x_{b_{l_i}} - x_{a_{l_i}} \geq \text{len}(a_{l_i}) \cdot (1 - \vec{\Delta}_{l_i}), & t_{a_{l_i}} = 0 \\ x_{a_{l_i}} - x_{b_{l_i}} \geq \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \cdot \overleftarrow{\Delta}_{l_i}, & t_{a_{l_i}} = 1 \end{cases}$$

Here $\vec{\Delta}_{l_i} \geq 0$ and $\overleftarrow{\Delta}_{l_i} \geq 0$ are slack variables for the constraints associated with the canonical and uncanonical contig pair orientations, respectively. It is proposed to add the opposing constraints into the optimization together and penalize only one of them depending on $t_{a_{l_i}}$.

B.3 Orientation constraints

Implementing relative contig orientation constraints as optimization problem constraints is not straightforward. It is proposed to include those constraints as part of the objective function. For this purpose a function equal to the sum of weights ω_{l_i} of contig links with satisfied orientation is

constructed:

$$g(t) = \sum_{i=1, m}^{e_{l_i}=0} q_{a_{l_i} b_{l_i}} \omega_{l_i} + \sum_{i=1, m}^{e_{l_i}=1} (1 - q_{a_{l_i} b_{l_i}}) \omega_{l_i} \rightarrow \max,$$

where $q_{a_{l_i} b_{l_i}} = \begin{cases} 0, & a_{l_i} \text{ and } b_{l_i} \text{ are equally oriented} \\ 1, & a_{l_i} \text{ and } b_{l_i} \text{ are oppositely oriented} \end{cases}$. The maximum of this function is attained for contig orientations satisfying as many orientation constraints as possible. This leads to a high quality contig orientation if the majority of the contig links are correct. The definition of $q_{a_{l_i} b_{l_i}}$ permits expressing it as

$$q_{a_{l_i} b_{l_i}} = |t_{a_{l_i}} - t_{b_{l_i}}| = (t_{a_{l_i}} - t_{b_{l_i}})^2 = t_{a_{l_i}}^2 + t_{b_{l_i}}^2 - 2t_{a_{l_i}} t_{b_{l_i}} = t_{a_{l_i}} + t_{b_{l_i}} - 2t_{a_{l_i}} t_{b_{l_i}}. \quad (3)$$

Expanding (3) in the previous expression for $g(t)$ gives

$$\begin{aligned} &= \sum_{i=1, m}^{e_{l_i}=0} (t_{a_{l_i}} + t_{b_{l_i}} - 2t_{a_{l_i}} t_{b_{l_i}}) \omega_{l_i} + \\ &+ \sum_{i=1, m}^{e_{l_i}=1} (1 - t_{a_{l_i}} - t_{b_{l_i}} + 2t_{a_{l_i}} t_{b_{l_i}}) \omega_{l_i} \end{aligned}$$

with $g(t) \rightarrow \max$.

This function will constitute a part of the final objective function of the optimization problem. The function is quadratic

in its arguments and hence falls in the class of quadratic programming problems, for which efficient solvers are available.

B.4 Distance slack penalty function

In order to take the distance optimization constraints from Section B.1 into account, the slack variables $\vec{\xi}_{l_i}$ and $\overleftarrow{\xi}_{l_i}$ must be penalized as a part of the objective function. Since distance constraints make sense only when the relative contig orientation is correct (otherwise the distance expression is invalid), this is the only case when the slack must be penalized. Moreover, for every contig link only one of the two slack variables must be penalized depending on the contig pair orientation.

The influence of slack penalties should reflect the importance of the corresponding contig links. This can be done by making penalties proportional to contig weights. Care has to be taken to avoid situations when a low-weight (incorrect) contig link is completely violated and its penalty influences the entire scaffold. This can be achieved by defining a maximum allowed slack S_ξ and working with values $\min\left(\overleftarrow{\xi}_{l_i}, S_\xi\right)$ and $\min\left(\vec{\xi}_{l_i}, S_\xi\right)$ instead of the corresponding slack variables.

The resulting penalty should not be bigger than the weight of the contig (which is added to the objective in function $g(t)$). It is proposed to limit the penalty by $\frac{\omega_{l_i}}{2}$ (half of the link weight) to allow using the same maximum penalty for the order slack variables (Section B.2), as this leads to equal influence of the distance and order slack variables on the objective function. Therefore, the distance penalty is given by

$$\frac{\omega_{l_i}}{2} \cdot \frac{\min(\xi_{l_i}, S_\xi)}{S_\xi}.$$

A default value $S_\xi = 1$ is chosen because at higher values of slack the order of two contigs is completely violated (and hence the penalty should be maximal).

A function matching the described requirements is given by (4) and can be rewritten as (5). Here $q_{a_{l_i} b_{l_i}}$ is defined as in (3). For brevity, we consider only the derivation for the first inner sum. In this expression the minimum of a variable and a constant is taken twice. To our knowledge the only way to implement this behavior in constrained programming framework is to replace $\min\left(\vec{\xi}_{l_i}, S_\xi\right)$ and $\min\left(\overleftarrow{\xi}_{l_i}, S_\xi\right)$ by

expressions $\alpha_{l_i} \vec{\xi}_{l_i} + (1 - \alpha_{l_i}) S_\xi$ and $\alpha_{l_i} \overleftarrow{\xi}_{l_i} + (1 - \alpha_{l_i}) S_\xi$, where α_{l_i} is a binary switch variable (as introduced in Section 2.2.2). It is possible to use a single binary variable for both minimum expressions as depending on the value of $t_{a_{l_i}}$

only one of them is taken into account in function $h(t, \vec{\xi}, \overleftarrow{\xi})$. Expansion of these terms and $q_{a_{l_i} b_{l_i}}$ gives rise to a polynomial of fourth degree (6) containing products of binary variables (e.g. $\alpha_{l_i} t_{a_{l_i}} t_{b_{l_i}} S_\xi$) an products of binary variables multiplied by a single continuous variable (e.g. $\alpha_{l_i} t_{a_{l_i}} t_{b_{l_i}} \overleftarrow{\xi}_{l_i}$). Therefore, it is not directly suitable for MIQP optimization but can be simplified by applying the so-called *big-M formulation*, as follows.

Every product $\prod_{i=1}^n b_i$, $b_i \in \{0, 1\}$ in the polynomial should

be replaced by a binary variable π with the following additional optimization constraints:

$$\begin{aligned} \pi &\leq b_i \\ \pi &\geq 0 \\ \pi &\geq \sum_{i=1}^n b_i - n. \end{aligned}$$

Similarly every product $a \prod_{i=1}^n b_i$, $a \in \mathbb{R}$, $b_i \in \{0, 1\}$ should be replaced by a continuous variable π with additional constraints:

$$\begin{aligned} \pi &\leq M b_i \\ \pi &\geq 0 \\ \pi &\geq a - nM + M \sum_{i=1}^n b_i, \end{aligned}$$

where M is an upper bound on a (hence the name big-M). Applying this transformations effectively reduces the degree of expression (6) to one, at the expense of adding auxiliary optimization variables and constraints. The described technique permits rewriting $h(t, \vec{\xi}, \overleftarrow{\xi})$ as a linear function (not shown), which makes it suitable for MIQP optimization. Henceforth any reference to this function implies its linear form.

B.5 Order slack penalty function

The requirements for distance and order slack penalty functions are identical. It is, therefore, possible to use the functional form from Section B.4 to penalize slacks $\vec{\Delta}_{l_i}$ and $\overleftarrow{\Delta}_{l_i}$. Following (Li and Durbin, 2009; Gao *et al.*, 2011) the value of maximum allowed slack S_Δ was chosen as 6 (i.e. 6 standard deviations away from the distance mean). Instead of α_{l_i} , variables β_{l_i} are used. The resulting function and its linear form are referred to as $p(t, \vec{\Delta}, \overleftarrow{\Delta})$.

B.6 Objective function

The joint objective function of the optimization problem combines functions $g(t)$, $h(t, \vec{\xi}, \overleftarrow{\xi})$ and $p(t, \vec{\Delta}, \overleftarrow{\Delta})$:

$$f(t, \xi, \overleftarrow{\xi}, \Delta, \overleftarrow{\Delta}) = g(t) - h(t, \vec{\xi}, \overleftarrow{\xi}) - p(t, \vec{\Delta}, \overleftarrow{\Delta}) \rightarrow \max$$

s.t. the constraints from Sections B.1, B.2, B.4, B.5 are met. The resulting optimization problem is a MIQP that contains $(2n + 4m)$ optimization variables ($(n + 2m)$ of which are binary), $(3n + 14m)$ optimization constraints and an exceptionally large number of auxiliary variables (including binary) and constraints. The size of the problem explodes due to the usage of big-M formulation, making it inapplicable directly to real scaffolding problem.

C. FIXED OPTIMIZATION PROBLEM

Treating variables t_i from Section 2.2 as fixed allows for significant simplification of the original problem:

1. Eliminates n binary variables and reduces the objective to a linear function, therefore, turning the problem into a LP problem.
2. Reduces the number of optimization constraints by removing constraints

$$\begin{aligned}
h(t, \vec{\xi}, \overleftarrow{\xi}) &= \sum_{i=1, \overline{m}}^{e_{l_i}=0} q_{a_{l_i} b_{l_i}} \left((1 - t_{a_{l_i}}) \min \left(\vec{\xi}_{l_i}, S_{\xi} \right) + t_{a_{l_i}} \min \left(\overleftarrow{\xi}_{l_i}, S_{\xi} \right) \right) \frac{\omega_{l_i}}{2S_{\xi}} + \\
&\quad + \sum_{i=1, \overline{m}}^{e_{l_i}=1} (1 - q_{a_{l_i} b_{l_i}}) \left((1 - t_{a_{l_i}}) \min \left(\vec{\xi}_{l_i}, S_{\xi} \right) + t_{a_{l_i}} \min \left(\overleftarrow{\xi}_{l_i}, S_{\xi} \right) \right) \frac{\omega_{l_i}}{2S_{\xi}} \rightarrow \min
\end{aligned} \tag{4}$$

$$\begin{aligned}
&= \frac{1}{2S_{\xi}} \sum_{p=0}^1 (-1)^p \sum_{i=1, \overline{m}}^{e_{l_i}=p} q_{a_{l_i} b_{l_i}} \left((1 - t_{a_{l_i}}) \min \left(\vec{\xi}_{l_i}, S_{\xi} \right) + t_{a_{l_i}} \min \left(\overleftarrow{\xi}_{l_i}, S_{\xi} \right) \right) \omega_{l_i} + \\
&\quad + \frac{1}{2S_{\xi}} \sum_{i=1, \overline{m}}^{e_{l_i}=1} \left((1 - t_{a_{l_i}}) \min \left(\vec{\xi}_{l_i}, S_{\xi} \right) + t_{a_{l_i}} \min \left(\overleftarrow{\xi}_{l_i}, S_{\xi} \right) \right) \omega_{l_i}
\end{aligned} \tag{5}$$

$$\begin{aligned}
&q_{a_{l_i} b_{l_i}} \left[(1 - t_{a_{l_i}}) \min \left(\vec{\xi}_{l_i}, S_{\xi} \right) + t_{a_{l_i}} \min \left(\overleftarrow{\xi}_{l_i}, S_{\xi} \right) \right] \omega_{l_i} = \\
&= (t_{a_{l_i}} + t_{b_{l_i}} - 2t_{a_{l_i} b_{l_i}}) \left[(1 - t_{a_{l_i}}) \left(\alpha_{l_i} \vec{\xi}_{l_i} + (1 - \alpha_{l_i}) S_{\xi} \right) + t_{a_{l_i}} \left(\alpha_{l_i} \overleftarrow{\xi}_{l_i} + (1 - \alpha_{l_i}) S_{\xi} \right) \right] \omega_{l_i} = \\
&= \omega_{l_i} \left[\alpha_{l_i} t_{a_{l_i}} \overleftarrow{\xi}_{l_i} + t_{a_{l_i}} S_{\xi} - \alpha_{l_i} t_{a_{l_i}} S_{\xi} + \alpha_{l_i} t_{b_{l_i}} \vec{\xi}_{l_i} - \alpha_{l_i} t_{a_{l_i} b_{l_i}} \vec{\xi}_{l_i} + t_{b_{l_i}} S_{\xi} - \alpha_{l_i} t_{b_{l_i}} S_{\xi} + \alpha_{l_i} t_{a_{l_i} b_{l_i}} S_{\xi} - \alpha_{l_i} t_{a_{l_i} b_{l_i}} \overleftarrow{\xi}_{l_i} - \right. \\
&\quad \left. - t_{a_{l_i} b_{l_i}} S_{\xi} + \alpha_{l_i} t_{a_{l_i} b_{l_i}} S_{\xi} \right]
\end{aligned} \tag{6}$$

Box 1: Derivation of distance slack penalty function.

(a) Induced by contig links with unsatisfied relative orientation (i.e. links for which $t_{a_{l_i}} \oplus t_{b_{l_i}} = e_{l_i}$ holds), as they create optimization constraints with slack variables that are not a part of the objective function (only the slack variables corresponding to constraints with satisfied relative orientation are penalized).

(b) That are distance or order constraints for contig pair orientations (canonical/uncanonical) that are not used. Thus either constraints for variables $\vec{\xi}_{l_i}, \vec{\Delta}_{l_i}$ or constraints for variables $\overleftarrow{\xi}_{l_i}, \overleftarrow{\Delta}_{l_i}$ are kept, reducing the number of constraints by $3m$.

The resulting optimization problem takes the following form

$$g = \sum_{i=1, \overline{m}}^{t_{a_{l_i}} \oplus t_{b_{l_i}} \neq e_{l_i}} w_{l_i} \equiv \text{const}$$

$$h(\xi) = \frac{1}{2S_{\xi}} \sum_{i=1, \overline{m}} \min(\xi_{l_i}, S_{\xi}) \omega_{l_i}$$

$$p(\Delta) = \frac{1}{2S_{\Delta}} \sum_{i=1, \overline{m}} \min(\Delta_{l_i}, S_{\Delta}) \omega_{l_i}$$

$$f(\xi, \Delta) = g - h(\xi) - p(\Delta) \rightarrow \max,$$

s.t.

3. Turns part g of the objective function (responsible for contig orientations) into a constant, therefore, leaving linear functions h and p to be optimized.

4. Removes a set of continuous variables through keeping either the forward $\vec{\xi}_{l_i}, \vec{\Delta}_{l_i}$ or the reverse $\overleftarrow{\xi}_{l_i}, \overleftarrow{\Delta}_{l_i}$ slack variables similar to 2 (b). Therefore, reducing the number of variables by $2m$.

5. Removes auxiliary variables and constraints resulting from big-M formulation for products not containing variables α_{l_i} and β_{l_i} and simplifies constraints for products containing these variables.

• For $e_{l_i} = 0 \wedge r_{l_i} = 0$:

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} x_{a_{l_i}} - x_{b_{l_i}} \leq \sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \\ x_{a_{l_i}} - x_{b_{l_i}} \geq -\sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \\ x_{a_{l_i}} - x_{b_{l_i}} \geq -\text{len}(b_{l_i}) \cdot \Delta_{l_i} \end{array} \right. \quad t_{a_{l_i}} = 0 \\ \left\{ \begin{array}{l} x_{b_{l_i}} - x_{a_{l_i}} \leq \sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \geq -\sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \geq -\text{len}(a_{l_i}) \cdot \Delta_{l_i} \end{array} \right. \quad t_{a_{l_i}} = 1 \end{array} \right.$$

- For $e_{l_i} = 0 \wedge r_{l_i} = 1$:

$$\begin{cases} \begin{cases} x_{b_{l_i}} - x_{a_{l_i}} \leq & \sigma_{l_i} \xi_{l_i} + \mu_{l_i} \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & -\sigma_{l_i} \xi_{l_i} + \mu_{l_i}, \quad t_{a_{l_i}} = 0 \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & \text{len}(a_{l_i}) \cdot (1 - \Delta_{l_i}) + \text{len}(b_{l_i}) \end{cases} \\ \begin{cases} x_{a_{l_i}} - x_{b_{l_i}} \leq & \sigma_{l_i} \xi_{l_i} + \mu_{l_i} \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & -\sigma_{l_i} \xi_{l_i} + \mu_{l_i}, \quad t_{a_{l_i}} = 1 \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & \text{len}(a_{l_i}) \cdot (1 - \Delta_{l_i}) + \text{len}(b_{l_i}) \end{cases} \end{cases}$$

- For $e_{l_i} = 1 \wedge r_{l_i} = 0$:

$$\begin{cases} \begin{cases} x_{a_{l_i}} - x_{b_{l_i}} \leq & \sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & -\sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}), \quad t_{a_{l_i}} = 0 \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & \text{len}(b_{l_i}) \cdot (1 - \Delta_{l_i}) \end{cases} \\ \begin{cases} x_{b_{l_i}} - x_{a_{l_i}} \leq & \sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & -\sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(a_{l_i}), \quad t_{a_{l_i}} = 1 \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & \text{len}(b_{l_i}) - \text{len}(a_{l_i}) \Delta_{l_i} \end{cases} \end{cases}$$

- For $e_{l_i} = 1 \wedge r_{l_i} = 1$:

$$\begin{cases} \begin{cases} x_{b_{l_i}} - x_{a_{l_i}} \leq & \sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(b_{l_i}) \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & -\sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(b_{l_i}), \quad t_{a_{l_i}} = 0 \\ x_{b_{l_i}} - x_{a_{l_i}} \geq & \text{len}(a_{l_i}) \cdot (1 - \Delta_{l_i}) \end{cases} \\ \begin{cases} x_{a_{l_i}} - x_{b_{l_i}} \leq & \sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(b_{l_i}) \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & -\sigma_{l_i} \xi_{l_i} + \mu_{l_i} - \text{len}(b_{l_i}), \quad t_{a_{l_i}} = 1 \\ x_{a_{l_i}} - x_{b_{l_i}} \geq & \text{len}(a_{l_i}) - \text{len}(b_{l_i}) \cdot \Delta_{l_i} \end{cases} \end{cases}$$

for every link l_i . It consists of no more than $(n+2m)$ continuous and $2m$ binary optimization variables, and $3m$ optimization constraints (constraints and variables are added iff links have correct relative orientation); and auxiliary variables and constraints used in big-M formulation when $\min(\xi, S_\xi)$ and $\min(\Delta, S_\Delta)$ are expanded.

D. CONTIG LINK BUNDLING

Contig link bundling is a straightforward idea for reducing the number of contig links and hence the number of optimization constraints. The link weight is a part of the contig link description and can directly used for adopting the contig link bundling procedure from (Huson *et al.*, 2002; Gao *et al.*, 2011).

For every (ordered) pair of contigs (a, b) , link bundling consists of adding all links $l_i = (a, b)$ to a list L , and selecting link l_m of median weight and all links from the list which agree with l_m on orientation and order and for which the condition $|\mu_{l_i} - \mu_{l_m}| < C \cdot \sigma_{l_m}$ holds. Here C is a parameter equal to 3 by default (following the original authors' decision). The selected links are then bundled into a single edge and removed from L . This procedure is repeated until L is empty.

The bundled edge inherits orientation and order constraints from the edges l_{τ_i} used to create it. Its weight ω_{l_τ} , distance

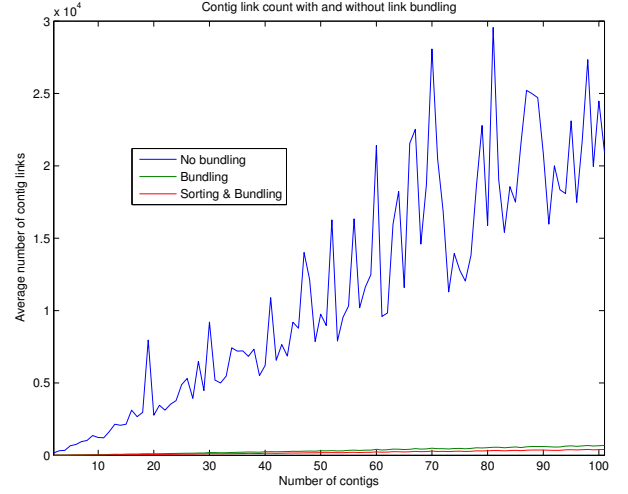


Figure D.1: Comparison of contig link count with and without link bundling. Average counts over 10 experiments are plotted.

μ_{l_τ} and standard deviation σ_{l_τ} are set to

$$\begin{aligned} \omega_{l_\tau} &= \sum_{\tau_i} \omega_{l_{\tau_i}} \\ \mu_{l_\tau} &= \frac{p}{q} \\ \sigma_{l_\tau} &= \frac{1}{\sqrt{q}}, \end{aligned}$$

where p and q are given by

$$\begin{aligned} p &= \sum_{\tau_i} \frac{\mu_{l_{\tau_i}}}{\sigma_{l_{\tau_i}}^2} \\ q &= \sum_{\tau_i} \frac{1}{\sigma_{l_{\tau_i}}^2}. \end{aligned}$$

These formulas come from statistical practices of combining multiple measurements (Huson *et al.*, 2002). Implementation of the bundling procedure has the additional benefit of allowing the bundling of links from different information sources and ambiguous links separately.

D.1 Contig link sorting

Link bundling allows for a major reduction in the number of contig links between a pair of contigs (a, b) . However, contig links define order constraints relative to the first contig in the pair and are, therefore, directed. As a result two consistent contig links $l_i = (a, b)$ and $l_j = (b, a)$ will not be bundled together.

Every contig link $l_i = (a, b)$ can be mirrored as contig link $\bar{l}_i = (b, a)$ such that it induces identical constraints relative to contig b . Only the contig order information of a mirrored link is different from the original and can be calculated as $r_{\bar{l}_i} = r_{l_i} \oplus e_{l_i}$ (sign \oplus denotes exclusive or operation). This makes it possible to sort contig links so that all links originate in contigs with a lower contig ID (used in the implementation to uniquely identify contigs).

Sorting contig links prior to link bundling allows for a fur-

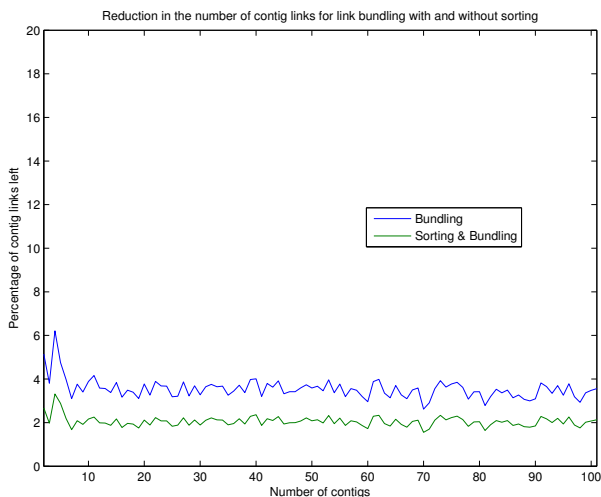


Figure D.2: Percentages of contig links left after bundling. These percentages were calculated for average contig link counts. On average link bundling leaves only 4% of the original links, while bundling performed after link sorting reduces this number to 2%.

ther reduction in the number of contig links. Figure D.1 compares numbers of contig links prior to link bundling to the numbers after (with and without sorting) and Figure D.2 plots the change in the numbers as a percentage. On average, contig link bundling after sorting leaves 408.6 contig links for $n = 101$ which is approximately the number of links for $n = 3$ without link bundling. The data was obtained for Real-large simulated scaffolding problems (see Section 3.3 for description) with 10 problems per problem size.

E. FIXED OPTIMIZATION HEURISTIC

To further evaluate the heuristic used for solving the fixed optimization problem it was compared to the CPLEX optimization suit. CPLEX provides an efficient implementation of the branch-and-cut algorithm for solving mixed-integer linear programming problems. It is based on a branch and bound search through the space of possible integer variable values that solves relaxed LPs with linear constraints derived from the branching at each node of the search tree. To reduce the search space the objective value obtained for a node is used as an upper estimate of the objective value for the complete subtree with the root at this node.

To help CPLEX algorithms cull the search tree, it was provided with the solution obtained by the fixed optimization heuristic as a starting point for the search. Furthermore, binary variables α_{i_i} and β_{i_i} were assigned priorities proportional to slacks ξ_{i_i} and Δ_{i_i} correspondingly. The search algorithm tries to perform branching on integer variables with high priority. Due to high slack of the variables their values should be easily decidable and, therefore, branching them early in the search reduces the search tree size.

The CPLEX suit was used to solve additional scaffolding problems for the Real-split and Real-small simulation setups (described in Section 3.3) for $n = 51, 101, 251, 501, 751, 1001, 1251$

and 1501. The search was allowed to run for 4 hours using 8 cores, after that the objective function values and scaffold correctness of the heuristic and CPLEX solutions were compared. Table E.1 shows that the solutions obtained by the fixed optimization heuristic are very close to the optimal solutions (solution gap is usually within 2% and decreases with problem size). Scaffold correctness and the percent of switch variables (α_{i_i} , β_{i_i}) with different values were examined for the cases when CPLEX obtained better solutions than the heuristic. They show that small objective value differences between CPLEX and the heuristic do not lead to significant changes in scaffold correctness. Although the percentage of switch variables for which CPLEX and the heuristic found different values (Table E.1, column “Difference”) sometimes reached 3%, it always consisted of mostly variables corresponding to contig links that were disabled by the heuristic and reenabled by the CPLEX algorithms. These results verify that the proposed fixed optimization heuristic is robust and provides high quality solutions even for large problems.

F. EM OPTIMIZATION OBJECTIVE DIFFERENCE

The objective function value differences for the EM procedure and the fixed optimization heuristic are small numbers, increasing with the size of the problem, that make more sense when considered individually. Therefore, they are difficult to visualize and are presented in Table F.1. Only the Sim-split setup was considered, as the situations for other experimental setups are similar.

The table shows that in most cases high breakpoint counts coincide with large (relative to the values with no difference in breakpoint counts) positive objective function value differences. Thus it can be concluded that the EM algorithm is capable of finding high quality solutions for the formulated optimization problems. However, these solutions are often different from the ground truth scaffolds. The smaller objective function differences with zero breakpoint count difference are explained by the non-symmetric order constraints in the optimization formulation (described in Section B.2). The non-symmetry is caused by the slack normalization performed by weighing the slack variable Δ_{i_i} with the downstream contig length, as for the opposite contig pair orientation the length of the other contig is used for normalization. Together non-symmetries introduce differences in the objective function values for the two scaffold orientations and make one them more favorable for the optimization.

G. SIMULATION

In order to facilitate algorithm development and testing, small datasets and sample scaffolding problems with known solutions are needed. Data selection and simulation can be used to create the required datasets and problems. Given a complete genome it is possible to select regions of desired size from its chromosomes. These regions can be used to simulate scaffolding information sources or data that maps to the regions can be selected into smaller datasets. The resulting datasets with ground truth reference sequences can be processed into sample scaffolding problem with known solutions.

A tool performing data selection has been developed. First

Table E.1: Comparison between solutions given by the fixed optimization heuristic and CPLEX. The number of breakpoints is reported with the number of scaffolds in brackets. A subset of results for $n = 500, 1000, 1500$ is shown.

Setup	n	Heuristic			CPLEX			Difference
		Value	Gap	Breakpoints	Value	Gap	Breakpoints	
Real-split	501	183349.101	1.14%	65 (4)	184750.719	0.39%	55 (6)	0.76%
		170423.653	1.18%	10 (2)	171934.507	0.31%	12 (2)	0.57%
		182514.416	1.37%	58 (6)	183652.871	0.76%	94 (6)	1.64%
		173837.771	1.70%	72 (4)	175681.327	0.65%	75 (4)	2.55%
		189423.525	0.98%	2 (5)	190786.539	0.27%	3 (7)	1.80%
	1001	337974.067	1.38%	52 (2)	340581.877	0.62%	53 (2)	0.54%
		418681.865	1.02%	178 (10)	418681.865	1.02%	178 (10)	0.00%
		367028.008	1.15%	213 (27)	368030.504	0.88%	203 (28)	2.34%
		361164.754	1.24%	137 (3)	363077.550	0.71%	139 (3)	1.72%
		361081.631	1.28%	164 (29)	363607.603	0.59%	181 (29)	0.93%
	1501	567094.170	1.31%	260 (19)	567094.170	1.31%	260 (19)	0.00%
		498556.854	1.46%	362 (17)	498556.854	1.46%	362 (17)	0.00%
		515458.955	1.35%	259 (10)	515458.955	1.35%	259 (10)	0.00%
		515821.978	1.39%	178 (32)	515821.978	1.39%	178 (32)	0.00%
		515381.712	1.43%	268 (13)	515381.712	1.43%	268 (13)	0.00%
Real-small	501	262173.102	0.63%	73 (2)	263273.603	0.21%	139 (1)	0.61%
		253759.851	0.66%	90 (3)	254195.128	0.49%	48 (4)	2.47%
		261209.751	0.67%	88 (2)	261889.301	0.41%	64 (3)	0.57%
		251719.598	0.95%	166 (2)	252052.455	0.81%	179 (1)	1.76%
		258217.801	0.54%	164 (8)	258720.128	0.35%	168 (7)	0.94%
	1001	515981.787	0.58%	431 (2)	515981.787	0.58%	431 (2)	0.00%
		517552.823	0.66%	311 (2)	517552.823	0.66%	311 (2)	0.00%
		516038.568	0.66%	401 (1)	516038.568	0.66%	401 (1)	0.00%
		512648.724	0.66%	392 (5)	512648.724	0.66%	392 (5)	0.00%
		508650.868	0.63%	391 (3)	508650.868	0.63%	391 (3)	0.00%
	1501	760023.447	0.67%	702 (5)	760023.447	0.67%	702 (5)	0.00%
		761105.363	0.71%	698 (9)	761105.363	0.71%	698 (9)	0.00%
		753861.872	0.72%	625 (6)	753861.872	0.72%	625 (6)	0.00%
		753349.606	0.70%	693 (5)	753349.606	0.70%	692 (5)	0.00%
		761675.547	0.73%	602 (5)	761675.547	0.73%	602 (5)	0.00%

Table F.1: Objective function value (breakpoint count) for the EM solution minus that for the fixed optimization heuristic (Sections 3.4, 3.5) for the Sim-split setup.

#	Problem size									
	101	201	301	401	501	601	701	801	901	1001
1	0.0 (0)	0.0 (0)	9.4 (0)	2.0 (0)	0.4 (0)	205.3 (281)	42.4 (46)	-9.4 (-1)	77.3 (522)	-41.9 (442)
2	-15.7 (1)	1.0 (0)	45.5 (-1)	161.3 (126)	-2.8 (0)	21.1 (61)	-1.4 (312)	-11.1 (0)	3.6 (159)	52.3 (452)
3	0.0 (0)	0.0 (0)	6.0 (0)	0.0 (0)	417.3 (240)	6.0 (0)	5.0 (0)	4.4 (31)	17.8 (268)	8.7 (155)
4	106.8 (6)	0.7 (0)	0.0 (0)	8.0 (0)	1.0 (0)	0.0 (0)	-6.3 (356)	55.7 (46)	0.4 (0)	29.2 (187)
5	-0.1 (0)	0.0 (0)	1.0 (0)	1.6 (0)	3.5 (0)	54.8 (224)	66.3 (75)	98.7 (257)	43.4 (391)	75.8 (183)
6	1.0 (0)	0.0 (0)	-3.9 (13)	0.3 (0)	0.0 (0)	14.7 (0)	45.0 (348)	33.9 (331)	2.2 (368)	-23.9 (121)
7	-1.0 (0)	0.0 (0)	1.2 (-1)	0.0 (0)	0.9 (0)	38.1 (65)	41.4 (187)	-26.7 (358)	6.9 (376)	2.7 (202)
8	0.0 (0)	11.2 (0)	0.3 (2)	0.0 (0)	1.0 (0)	14.0 (253)	27.4 (149)	83.4 (360)	15.8 (50)	-41.7 (372)
9	0.0 (0)	-2.9 (0)	0.0 (0)	1.0 (0)	0.0 (0)	-0.5 (0)	97.8 (185)	1.8 (219)	-1.9 (300)	119.7 (62)
10	0.0 (0)	-2.9 (0)	-1.8 (0)	22.2 (-1)	0.0 (0)	3.1 (200)	4.0 (78)	26.9 (-2)	52.7 (229)	71.1 (400)

it performs selection of precisely defined regions or random selection of the regions based on user specified parameters (number of regions, region length, region chromosome). Region selection makes sure that regions do not intersect. The selected regions are output in FASTA format and selection or simulation of data that maps to the regions is performed. Experiments in this research were limited to scaffolding using HTS data, therefore, only selection and simulation of DNA sequencing reads is covered.

G.1 Sequencing read selection and simulation

Read alignment is necessary in order to perform read selection. Reads are aligned to the original genome and not the chosen regions. This eliminates reads with low mapping quality alignments to the chosen regions, which have better alignments outside the regions. The alignment procedure from Section A was used. Single-end read alignment was performed as paired read alignment can reduce mapping ambiguity. BWA and Novoalign were used with default parameters (i.e. allowing up to 3 mismatches for the available Illumina data, Table 1). Read pairs were selected if at least one of the reads in the pair had an alignment overlapping the chosen regions by any number of nucleotides. This allowed for retaining ambiguity inherent to contig scaffolding problems. Selected read pairs were output in FASTQ format.

Read simulation was used as an alternative to read selection when ground truth read positions in the genome were required. It simulated error-free paired reads for the chosen regions with read length and insert size following specified Gaussian distributions. Read coverage was made uniform by randomly selecting read pair locations in the chosen regions. To produce a read library with coverage depth D a total of $\frac{L \cdot D}{2l}$ read pairs were generated, where L is region length and l is mean read length.

G.2 Problem simulation

Having a subset of data and a ground truth sequence that it corresponds to, it is possible to create sample scaffolding problems. For this the chosen regions have to be cut into fragments, the orientation of resulting fragments and their order must be randomly changed. The resulting fragments should be output as a set of contigs that must be scaffolded. The ground truth scaffold for the simulated problem is then obtained through tracking region cut points and fragment orientation changes. A tool has been developed that performs these operations, however instead of cuts it introduces gaps with size given by a Gaussian distribution (optionally introduces overlaps). These gaps are introduced so that the resulting fragments are at least l bp long (a parameter). Obtained fragments are output in FASTA format.

H. EVALUATION

H.1 Breakpoint count

Optimization formulation proposed in this research describes scaffolds using contig coordinate and orientation, whereas Gao *et al.* (2011) describe scaffolds using contig order, orientation and gap size. Adopting their scaffold comparison procedure requires establishing contig order. This is done by sorting contigs in each scaffold of the set in the increasing order of their downstream end coordinate (i.e.

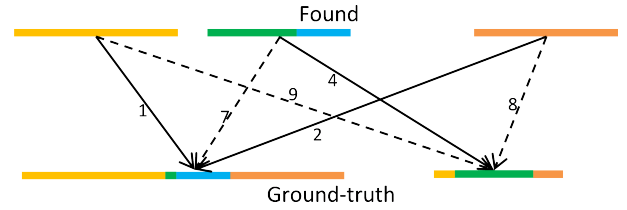


Figure H.1: Breakpoints for the ground-truth (bottom) and the found (top) sets of scaffolds. Breakpoints for all scaffold pairs are computed (shows as numbers next to black arrows) and the minimum for each found scaffold is chosen (solid black arrows). The resulting breakpoint count for the sets is 5.

$x_i - t_i(\text{len}(c_i) - 1)$. The found scaffolds $\{F_i\}_{i=1, n_F}$ are then compared to the ground truth scaffolds $\{G_i\}_{i=1, n_G}$ by finding for every scaffold F_i a ground truth scaffold G_{F_i} with the least number of breakpoints (shown in Figure H.1). Breakpoints are counted (see Figure 4) for two possible orientations of G_{F_i} and the minimum is taken. The sum over F_i of the minimum numbers of breakpoints is then taken as the number of breakpoints between two scaffold sets.

H.2 Orientation mismatch

A simplified procedure is possible when scaffold comparison is only concerned with contig orientation. Orientation mismatch for the found and ground truth scaffolds can be obtained similar to the breakpoint count. For every found scaffold F_i a ground truth scaffold G_{F_i} with the fewest mismatches is found. Orientation mismatches are counted for two possible orientations of G_{F_i} and the minimum is taken. Contig F_i^j is counted as a mismatch if:

1. it does not belong to G_{F_i} ;
2. it has a different orientation in G_{F_i} .

The sum over F_i of the minimal orientation mismatches is then taken as the orientation mismatch between two scaffold sets.

References

- D. Barnett, E. Garrison, G. Marth, and M. Stromberg. BamTools. Online, 2011.
- J. E. Beasley. Heuristic Algorithms for the Unconstrained Binary Quadratic Programming Problem. Technical report, Management School, Imperial College, UK, 1998.
- M. Boetzer, C. V. Henkel, H. J. Jansen, D. Butler, and W. Pirovano. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics/computer Applications in The Biosciences*, 27:578–579, 2011. doi: 10.1093/bioinformatics/btq683.
- L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 5:46–55, 1998. doi: 10.1109/99.660313.

- A. Dayarian, T. P. Michael, and A. M. Sengupta. SOPRA: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, 11, 2010. doi: 10.1186/1471-2105-11-345.
- E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software - Practice and Experience*, 30:1203–1233, 2000. doi: 10.1002/1097-024X(200009)30:11<1203::AID-SPE338>3.3.CO;2-E.
- S. Gao, N. Nagarajan, and W.-K. Sung. Opera: Reconstructing Optimal Genomic Scaffolds with High-Throughput Paired-End Sequences. In V. Bafna and S. Sahinalp, editors, *Research in Computational Molecular Biology*, volume 6577 of *Lecture Notes in Computer Science*, pages 437–451. Springer Berlin / Heidelberg, 2011. URL http://dx.doi.org/10.1007/978-3-642-20036-6_40. doi: 10.1007/978-3-642-20036-6_40.
- D. H. Huson, K. Reinert, and E. W. Myers. The greedy path-merging algorithm for contig scaffolding. *Journal of The ACM*, 49:603–615, 2002. doi: 10.1145/585265.585267.
- K. Katayama, M. Tani, and H. Narihisa. Solving Large Binary Quadratic Programming Problems by Effective Genetic Local Search Algorithm. In *Genetic and Evolutionary Computation Conference*, pages 643–650, 2000.
- W. J. Kent and D. Haussler. Assembly of the Working Draft of the Human Genome with GigAssembler. *Genome Research*, 11:1541–1548, 2001. doi: 10.1101/gr.183201.
- H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics/computer Applications in The Biosciences*, 25:1754–1760, 2009. doi: 10.1093/bioinformatics/btp324.
- H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. R. Abecasis, and R. Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics/computer Applications in The Biosciences*, 25:2078–2079, 2009. doi: 10.1093/bioinformatics/btp352.
- P. Merz and B. Freisleben. Genetic Algorithms for Binary Quadratic Programming. In *Genetic and Evolutionary Computation Conference*, 1999.
- P. Merz and K. Katayama. Memetic algorithms for the unconstrained binary quadratic programming problem. *Biosystems*, 78:99–118, 2004. doi: 10.1016/j.biosystems.2004.08.002.
- J. R. Miller, A. L. Delcher, S. Koren, E. Venter, B. Walenz, A. Brownley, J. Johnson, K. Li, C. M. Mobarry, and G. G. Sutton. Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics/computer Applications in The Biosciences*, 24:2818–2824, 2008. doi: 10.1093/bioinformatics/btn548.
- N. Nagarajan, T. D. Read, and M. Pop. Scaffolding and validation of bacterial genome assemblies using optical restriction maps. *Bioinformatics/computer Applications in The Biosciences*, 24:1229–1235, 2008. doi: 10.1093/bioinformatics/btn102.
- Y. Nesterov. Quality of semidefinite relaxation for non-convex quadratic optimization. CORE Discussion Papers 1997019, Université catholique de Louvain. Center for Operations Research and Econometrics (CORE), March 1997.
- P. M. Pardalos, O. A. Prokopyev, O. V. Shylo, and V. P. Shylo. Global equilibrium search applied to the unconstrained binary quadratic optimization problem. *Optimization Methods Software*, 23:129–140, February 2008. ISSN 1055-6788. doi: 10.1080/10556780701550083. URL <http://portal.acm.org/citation.cfm?id=1451525.1451533>.
- Y. Peng, H. C. M. Leung, S.-M. Yiu, and F. Y. L. Chin. IDBA - A Practical Iterative de Bruijn Graph De Novo Assembler. In *Research in Computational Molecular Biology*, pages 426–440, 2010. doi: 10.1007/978-3-642-12683-3_28.
- M. Pop, D. S. Kosack, and S. L. Salzberg. Hierarchical Scaffolding With Bambus. *Genome Research*, 14:149–159, 2003. doi: 10.1101/gr.1536204.
- J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, and I. Birol. ABySS: A parallel assembler for short read sequence data. *Genome Research*, 19:1117–1123, 2009. doi: 10.1101/gr.089532.108.
- J. Stoye, R. Giegerich, S. Kurtz, J. V. Choudhuri, E. Ohlebusch, and C. Schleiermacher. REPuter: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Research*, 2006.
- IBM ILOG. *IBM ILOG CPLEX V12.1: User's Manual for CPLEX*. IBM, 2009. URL ftp://ftp.software.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancomplex.pdf.
- IBM ILOG. ILOG CPLEX: High-Performance Software for Mathematical Programming and Optimization, 2011. URL <http://www.ilog.com/products/cplex/>.
- D. R. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18:821–829, 2008. doi: 10.1101/gr.074492.107.