GRASS: a generic algorithm for scaffolding next-generation sequencing assemblies

Alexey A. Gritsenko 1,2,3,* Jurgen F. Nijkamp 1,3, Marcel J.T. Reinders 1,2,3 and Dick de Ridder 1,2,3

1 The Delft Bioinformatics Lab, Department of Mediamatics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands.
2 Platform Green Synthetic Biology, P.O. Box 5057, 2600 GA Delft, The Netherlands.
3 Kluvier Centre for Genomics of Industrial Fermentation, P.O. Box 5057, 2600 GA Delft, The Netherlands.

Associate Editor: Dr. Michael Brudno

ABSTRACT

Motivation: The increasing availability of second-generation high-throughput sequencing (HTS) technologies has sparked a growing interest in de novo genome sequencing. This has in turn fueled the need for reliable means of obtaining high-quality draft genomes from short-read sequencing data. The millions of reads usually involved in HTS experiments are first assembled into longer fragments called contigs, which are then scaffolded, i.e. ordered and oriented using additional information, to produce even longer sequences called scaffolds. Most existing scaffolders of HTS genome assemblies are not suited for using information other than paired reads to perform scaffolding. They use this limited information to construct scaffolds, often preferring scaffold length over accuracy, when faced with the tradeoff.

Results: We present GRASS (GeneRic ASsembly Scaffolder) - a novel algorithm for scaffolding second-generation sequencing assemblies capable of using diverse information sources. GRASS offers a mixed-integer programming formulation of the contig scaffolding problem, which combines contig order, distance and orientation in a single optimization objective. The resulting optimization problem is solved using an Expectation-Maximization (EM) procedure and an unconstrained binary quadratic programming approximation of the original problem. We compared GRASS to existing HTS scaffolders using Illumina paired reads of three bacterial genomes. Our algorithm constructs a comparable number of scaffolds, but makes fewer errors. This result is further improved when additional data, in the form of related genome sequences, are used.


Contact: a.gritsenko@tudelft.nl

Supplementary information: Supplementary material is available at Bioinformatics online.

1 INTRODUCTION

High-throughput sequencing (HTS) technologies, such as Illumina (Illumina, Inc., San Diego, CA), 454 (Roche Applied Science, Penzberg, Germany) and SOLiD and IonTorrent (Life Technologies, Carlsbad, CA) produce millions of short DNA reads of typical lengths of 36-500 bp at low cost, making them attractive for de novo sequencing applications. With the aid of assembly algorithms (Zerbino and Birney, 2008; Miller et al., 2008; Peng et al., 2010), short reads can be joined together into longer sequences called contigs. However, contigs are typically shorter than the sequenced DNA molecules, as genomic repeat regions longer than the read length cannot be unambiguously assembled using the read sequences alone. Scaffolding, the process of using additional data to place contigs in the right order, orientation and at the right distance in longer (gapped) supercontigs called scaffolds, is a crucial step in obtaining high quality draft genome sequences.

Paired reads (mate pair or paired end reads, depending on the sequencing protocol), i.e. reads of known relative orientation, order and approximate physical distance, are often used for scaffolding. Additional information, including reference sequences of related organisms, restriction maps (Nagarajan et al., 2008) and RNA-seq data, can be used to derive more accurate contig placement (Kent and Haussler, 2010; Pop et al., 2004), thereby reducing the cost of finishing experiments and allowing for more reliable downstream analyses. However, most existing scaffolding algorithms are not able to utilize such information for scaffolding. To our knowledge, only Bambus (Pop et al., 2004) and SOPRA (Dayarian et al., 2010) can make use of additional data sources, although the latter was not originally designed for this purpose.

Generally, the Contig Scaffolding Problem (CSP) is finding a linear ordering and orientation of contigs that minimizes the number of unsatisfied scaffolding constraints. These constraints are derived from the available data through translation of the inherent distance, order and orientation constraints onto the contigs. The derived constraints can be mutually exclusive, which makes the problem of minimizing the number of unsatisfied constraints NP-hard (Kececioglu and Myers, 1995; Huson et al., 2002). Consequently, practical scaffolding algorithms only approximately solve this problem: Bambus (Pop et al., 2004) separately finds contig orientation and order and uses greedy heuristics to remove inconsistent constraints; SSPACE (Boetzer et al., 2011) greedily extends scaffolds using a heuristic stopping criterion; and SOPRA (Dayarian et al., 2010) uses an iterative procedure to identify a subset of contigs with consistent scaffolding constraints. Notable exceptions are OPERA (Gao et al., 2011) and the MIP Scaffolder (Salmela et al., 2011), which simplify the CSP by dropping types of constraints. OPERA implements an algorithm for finding an exact CSP solution without minimum contig distance constraints;

* to whom correspondence should be addressed
the MIP Scaffold (Salmela et al., 2011) couples a Mixed-Integer Programming (MIP) formulation of the contig scaffolding problem that does not enforce order constraints with an algorithm heuristically dividing the original problem into subproblems to be solved exactly.

We propose a novel GeneRic ASembly Scaffolding (GRASS) algorithm that can be applied to any type of scaffolding information. Our work is similar to Salmela et al. (2011), as we propose a MIP formulation of the scaffolding problem. However, we combine contig orientation, order and distance in a single quadratic optimization objective. Similar to Dayarian et al. (2010), we employ an iterative procedure to select a consistent subset of contigs. However, we apply an expectation-maximization strategy to maximize the objective function that identifies inconsistent constraints rather than contigs, thereby retaining more scaffolding information.

We implemented the algorithm in C++ and tested it on de novo assemblies of paired read data for the bacteria Escherichia coli, Pseudoxanthomonas suwonensis, and Pseudomonas syringae and compared it to the SSPACE, OPERA and MIP scaffolders. GRASS produces a competitive number of scaffolds with fewer scaffolding errors, particularly when combining various sources of scaffolding information.

2 METHODS

2.1 Data representation

Scaffolding constraints on contig distance, order and orientation are derived from the data in a manner depending on the data type. For example, the known relative orientation, relative order and approximate distance of paired reads that map to different contigs can be translated into relative contig orientation, order and approximate contig distance by taking mapping orientations and positions into account; similarly, physical distance, relative order and orientation of two contigs mapping to the same reference sequence can be translated into corresponding constraints. However, different data types eventually define the same type of pair-wise contig constraints, which can be conveniently represented as arcs (i.e., directed edges) \( l_j \) \((a_{l_j}, b_{l_j})\) \(\in E\) of weight \(\omega_{l_j}\) in a digraph \(G = (V,E)\) defined over the set of contigs \(V\) (Huson et al., 2002; Pop et al., 2004; Gao et al., 2011). The weight can be chosen to reflect information source importance and consistency. A relative order \(r_{l_j}\), relative orientation \(e_{l_j}\), and approximate distance suggested by the pair-wise constraints are then associated with every arc \(l_j\). The approximate distance is recorded as mean \(\mu_{l_j}\) and its standard deviation \(\sigma_{l_j}\). This form is a natural choice for capturing variation in contig distances derived from the paired read insert size. It is also suitable for scaffolding constraints without (reliable) distance estimates, for example constraints derived from paired RNA-seq data of an organism with abundant intron splicing, or by mapping contigs to genome of a distant relative. Such constraints can use a large \(\sigma_{l_j}\) to reflect the uncertainty in the data source. We refer to \(l_j\), its importance weight \(\omega_{l_j}\), and the corresponding pair-wise constraints as a contig link, and to \(G\) as the contig link graph. For succinct notation, for every contig link constraints are recorded as

\[
e_{l_j} = \begin{cases} 0 & \text{if } a_{l_j} \text{ and } b_{l_j} \text{ are from different strands} \\ 1 & \text{if } a_{l_j} \text{ and } b_{l_j} \text{ are from the same strand} \end{cases}
\]

\[
r_{l_j} = \begin{cases} 0 & \text{if } a_{l_j} \text{ follows } b_{l_j} \\ 1 & \text{if } b_{l_j} \text{ follows } a_{l_j} \text{ given that } a_{l_j} \text{ has forward orientation.} \end{cases}
\]

This abstract definition is illustrated in Figure 1. It allows capturing any combination of contig order, distance and orientation, including constraints derived from paired end reads, mate pair reads, and contig mapping.

2.2 Contig link bundling and erosion

We create a single contig link for every available piece of evidence (e.g., pair of reads) and by default set its importance weight to one (a parameter adjustable per information source). For high coverage HTS data this procedure creates a large number of links. Contig link bundling is used to reduce the number of links, and thereby the complexity of the problem. For every ordered pair of contigs \((u,v)\), arcs \((u,v)\) \(\in E\) that agree on contig distance, order and orientation are combined into one or more contig links as in Huson et al. (2002). The weight of a link after bundling is equal to the sum of weights of links bundled together to create it. Our definition of contig links permits having links that agree on all constraints, yet cannot be bundled together because they are oppositely directed in \(G\). To enable bundling of such links, we re-set \(r_{l_j}\) relative to the end points of \(l_j\) to make sure that all links connecting a pair of contigs have the same directionality. Finally, contig links with importance weight smaller than a predefined erosion threshold \(\epsilon\) are removed from the graph. This assumes that erroneous links are rare.

2.3 Optimization formulation

We present a mixed-integer quadratic programming (MIQP) formulation of the contig scaffolding problem. Our formulation is equivalent to the traditional one (minimize the number of unsatisfied constraints, Huson et al. (2002)), but uses slack variables as continuous measures of the extent to which each order and orientation constraint is satisfied. This allows for uncertain data, yielding less trustworthy constraints, to be accurately exploited in the scaffolding process. A number of optimization variables are associated with every contig and contig link. We maximize an objective function \(f\) of these variables subject to scaffolding constraints expressed as linear optimization constraints. The function reaches its maximum value when all distance, order and orientation constraints are satisfied. Each valid collection of the optimization variable values forms a solution to the optimization problem. These values are sufficient to puzzle contigs into scaffolds. For every contig \(c_i\), \(i = 1, \ldots, n\) the following variables are defined, as illustrated in Figure 2:

\[
elen(a_{l_j}), x_{a_{l_j}} - x_{b_{l_j}}, t_{a_{l_j}}, t_{b_{l_j}} \text{ associated with contigs. Example for } e_{l_j} = 0 \land r_{l_j} = 0.
\]
• $t_i = \begin{cases} 0, & c_i \text{ comes from the forward strand of the scaffold} \\ 1, & c_i \text{ comes from the reverse strand of the scaffold} \end{cases}$ is used to define contig orientation in the scaffold.

• $x_i \in \mathbb{R}^+$ corresponds to the 5' position of $c_i$ in the scaffold (when input contigs and the constructed scaffold are viewed as having a 5' to 3' orientation).

Naturally $x_i$ should be an integer variable, but it is relaxed to simplify the optimization problem and is rounded to the nearest integer when the solution is converted into scaffold nucleotide sequences. Additionally, with every link $l_j, j = 1, \ldots, m$ the following variables are associated:

• **Slack variables** for distance constraints, $\xi_j = \{\xi_j^a, \xi_j^r\} \in \mathbb{R}^+ \times \mathbb{R}^+$, and order constraints, $\Delta l_j = \{\Delta l_j^a, \Delta l_j^r\} \in \mathbb{R}^+ \times \mathbb{R}^+$ for forward ($t_{a_{lj}} = 0$) and reverse ($t_{a_{lj}} = 1$) orientations of the contig pair respectively. By design these variables reflect the degree to which the corresponding constraints are violated. They are penalized in the optimization objective $f$.

• **Switch variables** for distance constraints, $a_{lj} \in \{0, 1\}$, and order constraints, $b_{lj} \in \{0, 1\}$ (0, constraint is disabled; 1, enabled) used for disabling contig link constraints with high penalties.

As distance and order constraints are influenced by the orientation, different slack variables are required for both orientations. We omit orientation arrows above slacks $\xi_j$ and $\Delta l_j$, when the contig pair orientation is not important, or is clear from the context.

Contig links impose scaffolding constraints, which can be modeled as MIQP optimization constraints. We demonstrate here how such constraints can be derived from paired read data; the same type of constraints can be derived in a similar way from other sources of scaffolding information (for example, see section 3.2).

Distance constraints are expressed as:

$$|d(a_{lj}, b_{lj}) - \mu_{lj}| \leq \xi_j,$$  \hspace{1cm} (1)

where $d(a_{lj}, b_{lj})$ is the distance between contigs $a_{lj}$ and $b_{lj}$, and $\xi_j$ is a distance slack variable. This inequality captures uncertainty in the distance by measuring the difference with the mean in standard deviations. We derive contig distance $d(a_{lj}, b_{lj})$ from the paired read insert size as the gap size plus the contig lengths. The calculation then depends on the order and orientation of contigs connected by $l_j$. It can be fixed by assuming that the contigs have relative orientation and order suggested by $l_j$. For example, for the case of ($e_{lj} = 0$ and $r_{lj} = 0$) shown in Figure 2, the distance expression depends on contig pair orientation through $t_{a_{lj}}$:

$$\begin{align*}
d(a_{lj}, b_{lj}) &= x_{a_{lj}} - x_{b_{lj}} + \text{len}(a_{lj}) + \text{len}(b_{lj}), \quad t_{a_{lj}} = 0 \\
d(a_{lj}, b_{lj}) &= x_{b_{lj}} - x_{a_{lj}} + \text{len}(a_{lj}) + \text{len}(b_{lj}), \quad t_{a_{lj}} = 1.
\end{align*}$$

Combined with (1) the following constraints are obtained:

$$\begin{align*}
x_{a_{lj}} - x_{b_{lj}} &\leq \sigma_{lj}^a \xi_j^a + \mu_{lj} - \text{len}(a_{lj}) - \text{len}(b_{lj}) \\
x_{a_{lj}} - x_{b_{lj}} &\geq -\sigma_{lj}^a \xi_j^a + \mu_{lj} - \text{len}(a_{lj}) - \text{len}(b_{lj}) \\
x_{b_{lj}} - x_{a_{lj}} &\leq \sigma_{lj}^r \xi_j^r + \mu_{lj} - \text{len}(a_{lj}) - \text{len}(b_{lj}) \\
x_{b_{lj}} - x_{a_{lj}} &\geq -\sigma_{lj}^r \xi_j^r + \mu_{lj} - \text{len}(a_{lj}) - \text{len}(b_{lj})
\end{align*}$$  \hspace{1cm} (2)

where different slack variables are used for the two contig pair orientations.

The expressions for other combinations of $e_{lj}$ and $r_{lj}$ are derived similarly.

**Order constraints** are derived from read order constraints (i.e. if $c_j$ follows $c_i$, then they should not overlap and $c_j$ must be upstream of $c_i$), which additionally can be relaxed. The relaxation is necessary because (a) assembled contigs may overlap (Pop et al., 2004); (b) in some cases the order constraints on data are not valid when extended to contigs, as illustrated in Figure 3. Translating order constraints into optimization constraints as formulas shown for $e_{lj} = 0$ and $r_{lj} = 0$ discourages overlaps while still allowing the order constraint to be violated when $\Delta l_j > 1$. These slack variables are weighted by the length of the downstream contig to allow measuring them on a single scale. As for the distance optimization constraints, it is assumed that the relative contig orientation is correct.

**Orientation constraints** are modeled in the optimization objective function, which is designed to attain larger values when more orientation constraints are satisfied. The function is given by a polynomial

$$g(t) = \sum_{j=1}^{l} q_{a_{lj}, b_{lj}} \omega_{lj} + \sum_{j=1}^{l} (1 - q_{a_{lj}, b_{lj}})\omega_{lj},$$

where $q_{a_{lj}, b_{lj}} = t_a + t_b - 2t_{a_{lj}} \equiv \begin{cases} 0, & a \text{ and } b \text{ are equally oriented} \\ 1, & \text{otherwise} \end{cases}$.

It is equal to the sum of weights of contig links with satisfied orientation and serves as a basis for the optimization objective that is further penalized proportionally to slack variables.

**Slack penalties.** The distance and order constraints are added to the optimization problem through slack variable penalization. The penalty is proportional to the importance weight of the corresponding contig link and to the value of the slack variable. To avoid situations when a low-weight violated constraint results in a large penalty, a maximum penalty of half of the importance weight is enforced, after which the constraint is considered disabled. Doing this has the additional benefit of equalizing the influence of order and distance constraints. To this end we penalize as follows

$$\omega_{lj} \min(\xi_j^a, S_{\xi}),$$  \hspace{1cm} (4)

where $\xi_j$ is chosen as $\xi_j^a$ or $\xi_j^r$, according to the contig pair orientation and $S_{\xi}$ is the maximum slack threshold (after which the slack is disabled). Because the expression $\min(\xi_j^a, S_{\xi})$ is not suitable for use in a MIP, it is unraveled using the switch variables as $[a_{lj}, S_{\xi}] + [1 - a_{lj}, S_{\xi}]$. Similar penalties with variables $\Delta l_j$ and $b_{lj}$, and maximum slack threshold $S_{\Delta}$ are used for the order constraints. We set $S_{\xi} = 6$ (i.e. six standard deviations), as in Li and Durbin (2009); Gao et al. (2011); and $S_{\Delta} = 1$, as at this value of slack the physical order constraint is not satisfied anymore. Further, only the slacks for the appropriate contig pair orientation have to be penalized. This is achieved by penalizing $(1 - t_{a_{lj}}) \xi_j^a + t_{a_{lj}} \xi_j^r$ in place of $\xi_j$ in (4). This expression “chooses” which slack variable to penalize depending on the contig pair orientation. Finally, the constraints have to be penalized only when they are meaningful (i.e. the relative contig orientation $e_{lj}$ is assumed to be satisfied). The resulting function looks as follows:

$$h(t, \alpha, \xi_j, S_{\xi}) = \sum_{j=1}^{l} \omega_{lj} \xi_j^a \min(1 - t_{a_{lj}}) \xi_j^a + t_{a_{lj}} \xi_j^r + \sum_{j=1}^{l} (1 - q_{a_{lj}, b_{lj}}) \omega_{lj} \min(1 - t_{a_{lj}}) \xi_j^r + t_{a_{lj}} \xi_j^r.$$  \hspace{1cm} (3)

Expansion of this function leads to a fourth degree polynomial, containing only terms that consist purely of binary variables, or one continuous and up

![Figure 3](http://bioinformatics.oxfordjournals.org/content/3/3/159/Fig3.jpg)
to three binary variables. To construct a MIQP formulation, using the big-
M formulation (Nemhauser and Wolsey, 1988), these terms can be replaced
by a single new auxiliary variable each at the expense of introducing new
optimization constraints.

Putting it all together. We maximize
\[ f(x, t, \alpha, \beta, \xi, \Delta) \equiv g(t) - h(t, \alpha, \xi, S_T) - h(t, \beta, \Delta, S_D), \]
s.t. constraints (2) and (3) are satisfied. Here \( g(t) \) is maximized for
orientation, \( h(t, \alpha, \xi, S_T) \) is minimized for orientation and distance, and
\( h(t, \beta, \Delta, S_D) \) is minimized for orientation and order, in a single
optimization objective. Given the NP-hard nature of MIPs and the large
number of binary variables in the proposed formulation, this problem
becomes intractable even for small numbers of contigs.

2.4 Problem splitting
We tackle this intractability with an expectation-maximization (EM) -like
procedure.

The maximization step assumes the contig orientations are known (i.e., \( t_i \)
and \( q_{0,i} \) are fixed). Knowing \( t_i \) allows us to choose the slack variables (\( \xi_{i,j} \)
or \( \xi_{j,i} \), and \( \Delta_{i} \) or \( \Delta_{j} \)) depending on the contig pair orientations, and
to select contig links with satisfied relative orientation before the optimization
problem is constructed, significantly reducing the number of optimization
constraints and the complexity of the optimization problem:
\[ f(x, \alpha, \beta, \xi, \Delta) \equiv g - h(\alpha, \xi, S_T) - h(\beta, \Delta, S_D) \]
\[ g = \sum \xi_{i,j} \equiv \text{const}, \]
\[ h(\alpha, \xi, S_T) = \sum \max(\xi_{i,j}, S_T) - \frac{\alpha}{\max(\alpha, S_T)} \]
This fixed optimization problem, however, is still NP-hard due to the binary
variables \( \alpha_{ij} \) and \( \beta_{ij} \), involved in expansion of the min terms. We can
approximate this solution to the problem by first exactly solving its continuous
relaxation, choosing \( \alpha_{ij} \) and \( \beta_{ij} \) according to the slack values in the
relaxation solution and finally, re-solving the problem with these values
fixed. The relaxation is obtained by replacing \( h(\alpha, \xi, S_T) \) by \( h(\xi, S_T) = \frac{1}{\max(\xi, S_T)} \sum \xi_{i,j} \xi_{j,i} \) in (5). This eliminates all binary variables, allowing the use
of efficient optimization algorithms (Dantzig, 1988). The solution for the
relaxed problem gives us optimal values for slack \( \xi_{i,j} \) and \( \Delta_{i} \), which are
used to choose \( \alpha_{ij} \) and \( \beta_{ij} \) as
\[ \alpha_{ij} = \begin{cases} 0, & \xi_{i,j} > S_T \\ 1, & \xi_{i,j} \leq S_T \end{cases}, \]
\[ \beta_{ij} = \begin{cases} 0, & \Delta_{i} > S_D \\ 1, & \Delta_{i} \leq S_D \end{cases}, \]
and allows us to re-solve problem (5). The rationale behind is that, since the
majority of link information is assumed to be correct, large slack values will
be associated with incorrect constraints that have to be disabled. The total
penalty for \( l_j \) is memorized (initially set to zero) for use in the expectation
step as
\[ \Theta_{ij} \leftarrow \min(\xi_{i,j}, S_T) - \frac{\alpha}{\max(\alpha, S_T)} \]
The expectation step is used to obtain the expected contig orientations
\( t_i \), which maximize the objective function for the previously observed
penalties. Consider the MIQP problem when penalties associated with the
links are known (i.e., \( \Delta_{ij}, \xi_{ij}, \alpha_{ij} \), and \( \beta_{ij} \) are fixed), and the optimal
contig orientation is sought. In this problem, when a contig link is enabled,
its weight is penalized by the associated slack \( \Theta_{ij} \). We can, therefore,
consider an equivalent problem where all slacks are ‘zero’ and link weights
are modified as \( \tilde{\omega}_{ij} \leftarrow \omega_{ij} - \Theta_{ij} \). The problem is then to maximize
\[ f(t) \equiv g(t) = \sum_{j=1}^{m} q_{0,j} b_j \tilde{\omega}_j + \sum_{j=1}^{m} (1 - q_{0,j} b_j) \tilde{\omega}_j \] free of any constraints. This is an unconstrained binary quadratic
programming (UBQP) problem (Beasley, 1998), the problem of maximizing
a function \( c(t) = t^T C t \), where \( x \) is a binary vector of length \( n \) and \( C \) is an \( n \times n \) real matrix. Consider a vector of orientations \( t \in \{0,1\}^m \) and a
matrix \( C \) of size \( n \). Starting from a zero matrix, \( C = (c_{ij}) \) can be obtained
by updating it for every link \( l_j = (a, b) \) as
\[ c_{aa} \leftarrow (-1)^{t_j+1} \tilde{\omega}_j + c_{aa}, \]
\[ c_{bb} \leftarrow (-1)^{t_j+1} \tilde{\omega}_j + c_{bb}, \]
\[ c_{ab} \leftarrow (-1)^{t_j+1} 2\tilde{\omega}_j + c_{ab}. \]
The functions \( f(t) \) and \( c(t) \) will then differ by a constant and, therefore,
reach maxima for the same \( t \). Solving a UBQP is known to be an NP-
hard, but well-studied problem with efficient heuristic algorithms available
(Nesterov, 1997; Merz and Katayama, 2004; Pardalos et al., 2008). Thus, the
UBQP formulation of the problem is preferred over (6) for obtaining values
of \( t_i \).

The EM steps are iterated while contig orientations change. The algorithm
can be viewed as an iterative UBQP approximation of the original MIQP
problem. In practice, it converges to a solution within \( 7 \) iterations.

2.5 Scaffold extraction and post-processing
Repeat contigs in the contig link graph \( G \) are connected by ambiguous
links, hindering a confident positioning in scaffolds. In a pre-processing step,
we detect such contigs using a modification of the A-statistic (Myers
et al., 2000) proposed by Zerbino (2009), and prevent their incorporation in
scaffolds by removing all links from \( G \) incident to them. The connected
components of \( G \) correspond to separate subproblems, which are solved
independently.

After optimization, each solution tuple \((x, t, \alpha, \beta)\) and corresponding
subgraph \( G' \) are converted into one or more scaffolds. First, contig
links with disabled constraints (i.e. \( \alpha_{ij} = 0 \) or \( \beta_{ij} = 0 \)) are removed from
\( G' \) to minimize the chance of incorrectly incorporating contigs in the
same scaffold. Every connected component of the resulting \( G' \) is used to
construct a single nucleotide sequence. Contigs are processed in order of
their downstream end coordinates. The left end of the first contig is put at
the start of the sequence; every new contig is added to the scaffold such that
the gap between two consecutive contigs is preserved. When consecutive
contigs are predicted to overlap (i.e. have a negative gap size), the new contig
is pushed upstream to eliminate the overlap.

Because resolving contig overlaps in this way potentially leads to erroneous
sequence reconstruction, we also explore an optional post-processing approach
that performs global sequence alignment on consecutive contigs to find the best
overlap. Global alignment is performed using a divide-and-conquer version of the
Needleman-Wunsch algorithm (Hirschberg, 1975). Algorithm implementation from the NCBI C++ Toolkit
was used (National Center for Biotechnology Information, 2011). For every
consecutive pair of contigs predicted to have a gap of \( \mu \) bp, all gap sizes of at
most \( d = 100 \) bp away from the predicted value are examined. Negative gap
sizes indicate overlaps. For each gap size \( g \), global alignment of overlapping
contig ends is performed (match score of \( \text{match}_\text{score} = 2 \), mismatch penalty of
\( \mu = -3 \)). The best gap size is then chosen based on the alignment
score \( S \) and proximity to the predicted gap size \( \mu \) by maximizing
\[ S \geq \text{match}_\text{score} \cdot d - \max (|g - \mu|, d) \]

With the (mis)match scores chosen as above, this expression takes values in
\([-1.5, 1] \). Due to computational complexity only overlaps of no more than
1500 bp are considered (gap sizes with longer overlaps are assigned a score of
-1). The decision to join two contigs, to leave a gap between them or to
split the scaffold is then made:

- If none of the considered gap sizes suggest overlaps, the two contigs
  are positioned in a scaffold with a gap of \( \mu \) bp.
- If value of expression (7) for the chosen gap size \( g \) passes a quality
  threshold of 0.8, the contigs are positioned to have an overlap of \( g \) bp.
The overlap is replaced with the alignment consensus sequence, where
mismatches are masked with unknown nucleotides.
• If the chosen gap size does not pass the quality threshold and is shorter than 50 bp, the two contigs are positioned successively one following another with no overlap.

• Finally, if the chosen gap size suggests a longer overlap, the currently constructed scaffold is split into two with a new scaffold starting from a contig that was predicted to lie upstream.

In principle, the proposed post-processing step with scaffold splitting allows for construction of more accurate scaffolds compared to the naïve scaffold extraction. We refer to the combination of GRASS and post-processing as GRASS+.

2.6 Evaluation criteria

Similar to assemblies, scaffolds are evaluated based on accuracy and contiguity. Scaffolding accuracy can be assessed by comparing scaffolds to available reference sequences. We adopted the evaluation criteria from Dayarian et al. (2010); Gao et al. (2011) and counted the number of scaffold breakpoints, i.e. consecutive contig pairs in the scaffold that do not agree with the reference on contig distance, order or orientation. We perform a local alignment of scaffolds to the reference and count the number of breakpoints within each scaffold. Two consecutive alignments are counted as a breakpoint if any of these hold: (a) they align to two different chromosomes in the reference; (b) their relative orientations in the scaffold and in the reference do not match; (c) their relative orders in the scaffold and in the reference are larger than \( \Delta \). We used \( \Delta = 10 \) kbp and \( \Delta = 500 \) bp to assess contig distance correctness at low and high resolution respectively.

MUMmer (Delcher et al., 2002) was used to align scaffolds to references. Best hits for each position in the scaffold were computed. Only hits with at least 90 aligned bases (alignment length \( \times \) alignment identity), were taken into account. In practice, very few alignments do not pass this cutoff. The alignments are also used to calculate the percentage of the scaffold bases and the reference bases that are aligned (Salmela et al., 2011). These numbers capture scaffold accuracy and completeness.

Finally, scaffold completeness and contiguity are captured as in sequence assembly, calculating total length of all scaffolds, number of scaffolds, maximum scaffold length and the N50 statistic.

3 IMPLEMENTATION

GRASS source code is available under the GNU GPL v3 license. It was developed in C++ and tested on Linux. GRASS consists of linker and scaffolder modules. The linker takes contigs and the available information sources as input and produces linking and coverage data, which is then used by the scaffolder module. It filters out repeat contigs and uses the remaining data to produce scaffolds. Scaffolds are output both as lists of contigs with assigned coordinates and orientations, and as linear FASTA sequences with gaps.

3.1 Paired read data processing

To obtain contig links from paired read data, the linker module performs single-end mapping of the reads to contigs. The algorithm used for mapping depends on the data type: BWA (Li and Durbin, 2009) for Illumina reads, NovoAlign (http://www.novocraft.com/) for 454 data. The aligners are set to output all mapping locations, including non-unique hits, as a SAM file (Li et al., 2009), which is then converted to BAM for further processing. This process is applied to each paired read library.

Read alignments are preprocessed to remove read pairs with low-quality and ambiguous alignments. As a rule, only unique hits with no mismatches and minimum read length of 30 bp are kept. The filtered alignments are then scanned for paired reads that align to different contigs. Each such read pair mapping is used to create a single contig link with distance, order and orientation constraints derived from the mapping and the given read pairing method (i.e. paired ends or mate pairs). The BamTools API (Barnett et al., 2011) is used for filtering and processing read alignments.

3.2 Related genome data processing

An available reference sequence, such as the genome of a related organism, can be used for guiding the scaffolding process. For this purpose, contigs are aligned to the reference sequence. For every contig, a position in the reference sequence is obtained from contig tiling constructed from local alignments using MUMmer. Contig links are then created for every pair of consecutive contigs aligning to the same reference sequence, with relative orientation and order derived from the tiling. To capture alignment quality, weights for links \( l_j = (a_{l_j}, b_{l_j}) \) are set to \( I_{a_{l_j}} \times I_{b_{l_j}} \times C_{a_{l_j}} \times C_{b_{l_j}} \times W \), where \( I_{a_{l_j}} \) and \( I_{b_{l_j}} \) are alignment identities, \( C_{a_{l_j}} \) and \( C_{b_{l_j}} \) are alignment coverages reported by MUMmer for the corresponding contigs, and \( W > 0 \) is a weight assigned to the reference sequences as a scaffolding information source. This procedure is applied for each available reference sequence to create links, which are then used together in the optimization.

3.3 Optimization problem solution

The EM procedure proposed for solving the MIQP formulation of the contig scaffolding problem splits it into a continuous linear programming (LP) problem, and an UBQP problem. Although more efficient algorithms for solving UBQPs are available (Nesterov, 1997; Pardalos et al., 2008), a memetic algorithm from Merz and Katayama (2004) was chosen for ease of implementation. Usually, contig link graphs are sparse due to the linear scaffold structure that they encompass. Memetic algorithms improve individual solutions through local search, which in turn is well-suited for smooth search landscapes (as in the case of sparse contig link graphs). Graph sparsity is further exploited by implementing sparse matrix operations as in Merz and Katayama (2004).

We use the C++ Concert API for the CPLEX Optimizer (IBM ILOG, 2011) to solve LPs. CPLEX is freely available for academic use.

4 RESULTS AND DISCUSSION

4.1 Experimental setup

We have evaluated GRASS on de novo HTS assemblies of three bacterial genomes: Escherichia coli K12, substr. MG1655; Pseudoxanthomonas suwonensis 11-1; and Pseudomonas syringae B728a. For these organisms, finished genome sequences and HTS data from resequencing experiments are available. Presence of a finished genome sequence allows for reliably evaluating the algorithm and comparing it to other scaffolders in a de novo setup. This is achieved by using the reference sequence only in scaffold evaluation (thus not as an additional information source in the scaffolding process). The available test data is summarized in Table 1. Insert size and coverage were obtained from paired read mapping using BWA and BEDTools (Quinlan and Hall, 2010).

Velvet (Zerbino and Birney, 2008) was used to assemble reads into contigs. All assemblies had a coverage cutoff of 6 and were not scaffolded by the assembler. Only contigs longer than 150 bp were kept. Repeat resolution was disabled (i.e. no expected coverage was provided). For each organism, the k-mer length was chosen by performing assemblies for various \( k \) and choosing one based on assembly contiguity, length, percentage of mapped single reads, and percentage of properly paired reads (Li and Durbin, 2009) (Supplementary Tables 1, 2 and 3). For E. coli, P. suwonensis and P. syringae, \( k = 31 \), \( k = 59 \) and \( k = 23 \) were chosen respectively. This way of choosing \( k \) reflects real-life de novo assembly scenarios, yielding a realistic algorithm evaluation. Final assemblies are characterized in Tables 2-4.

4.2 Comparison to other scaffolders

We compared GRASS to SSPACE, MIP and OPERA scaffolders. Where required, insert size estimates from Table 1 were used.
to a join of the available read sets, as they have the same insert size.

Each read library separately, and in the case of *P. suwonensis* insert size.

Chosen by adding and subtracting 6 standard deviations to the mean *P. syringae* scaffolds. Maximum and minimum insert sizes were to 100 for the *E. coli* and *P. syringae* (70 $\pm$ 6) chose those which gave the highest N50 value. Settings (36 27, 5 and 5 were used for the *P. suwonensis* dataset; and 11 was used for the *P. syringae* data (values are given in the order of the experiments in Tables 2-4).

SOPRA was applied to assembly graphs produced by Velvet. However, when used with parameters chosen in accordance to the manual provided, SOPRA produced highly fragmented scaffolds compared to results from Salmela et al. (2011). To allow for a fair comparison, its results were not taken into account.

As a scaffold, Velvet was provided with mean insert size and standard deviation for each library. The data was reassembled with repeat resolution (expected coverage estimated automatically) and scaffolding turned on. Its performance was used as a baseline over which all scaffolders improved on *P. syringae* data and only SSPACE and GRASS improved on *E. coli* and *P. suwonensis* data.

Table 2-4 contain the results. Note that the minimum number of breakpoints is one, due to the circular structure of bacterial genome. Breakpoints at $\Delta = 10$ kb and $\Delta = 500$ bp differ only slightly, suggesting that gap lengths are estimated with high precision. SSPACE produced the longest scaffolds for *E. coli*. It also produced the smallest number of scaffolds for *E. coli* and *P. suwonensis*. The longest scaffolds and the smallest number of scaffolds on the *P. syringae* dataset are achieved by the MIP Scaffolder. Similar scaffold and reference coverage percentages were achieved by

Table 1. Available datasets. NCBI/EBI accession numbers are given for reference sequences and read sets. In all cases reads were produced by the Illumina sequencing platform.

<table>
<thead>
<tr>
<th>Genome size</th>
<th>Coverage</th>
<th>Insert size</th>
<th>Read count</th>
<th>Read length</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>E. coli</em></td>
<td>36 bp</td>
<td>216 ± 10</td>
<td>2 × 10, 408, 224</td>
<td>2 × 7, 047, 668</td>
<td>NC_000913.2</td>
</tr>
<tr>
<td><em>P. suwonensis</em></td>
<td>36 bp</td>
<td>160 × 10</td>
<td>2 × 7, 047, 668</td>
<td>2 × 23, 960, 004</td>
<td>CP0002446.1</td>
</tr>
<tr>
<td><em>P. syringae</em></td>
<td>76 bp</td>
<td>488 ± 18</td>
<td>2 × 19, 789, 425</td>
<td>2 × 19, 789, 425</td>
<td>NC_007000.1</td>
</tr>
</tbody>
</table>

(a) *E. coli* data. Running time ranges in (b) *P. suwonensis* data. Running time ranges in (c) *P. syringae* data. Running time ranges in [8 sec; 27 min 49 sec].

Fig. 4. Scaffold accuracy and contiguity tradeoff on the available datasets. Marker size indicates scaffolding running time in minutes, exact numbers are given in Supplementary Table 4. GRASS+ using paired reads and two related genomes is shown in (a) as ‘GRASS+ (all)’.
When combining several information sources, care has been taken to help scaffold the E. coli assembly: DH10B and BW2952. These genomes were used individually, together and in combination with related genomes. The MIP Scaffolder and SSPACE can achieve smaller numbers of scaffolds, but at the cost of (much) larger numbers of breakpoints. Clearly, GRASS and SSPACE represent two possible choices of scaffolding algorithms, with GRASS being more accurate with respect to the number of breakpoints and SSPACE constructing longer scaffolds. This behavior of the two algorithms is consistent over all datasets.

We also measured scaffolding running times, these are depicted in Figure 4 using marker size. Exact numbers, as well as read mapping running times are available in Supplementary Table 4. Like most scaffolders, GRASS spends a majority of its time on read alignment, making running times of different scaffolders comparable and running time of the core scaffolding part of GRASS on the considered datasets negligible. Based on simulation results, we do not expect computation to become a bottleneck for large genomes. Nevertheless, to reduce computational load it is always possible to split the contig graph into graphs of manageable size by increasing the erosion parameter $e$.

### 4.3 Using additional information

To demonstrate the ability of GRASS to utilize various scaffolding information sources, we used two related genomes (see Figure 5) to help scaffold the E. coli assembly: DH10B and BW2952. These genomes were used individually, together and in combination with paired reads. When combining several information sources, care has to be taken in choosing the weights $W$ and the erosion threshold parameter $e$. In individual genome experiments, $W = 100$ and $e = 80$ were chosen to remove links derived from low-quality alignments. In the experiment using only two related genomes (thus no links derived from paired read data) a higher weight was given to the more closely related strain: $e = 70$ and $W_{DH10B} = 80$, $W_{BW2952} = 100$ were used for the DH10B and BW2952 strains correspondingly. For experiments combining a single genome with paired reads, $W = 10$ and $e = 4$ were chosen. Finally, $W_{DH10B} = W_{BW2952} = 3$ and $e = 5$ were used in the experiment combining all data (including the paired read constraints) to emphasize use of links supported by at least two information sources. When used in the experiment, paired read link weights were set to 1. A standard deviation of 3000 bp was used for links derived from related genomes.

Interestingly, using just related genomes GRASS constructs a smaller number of scaffolds than when only paired reads are used. Table 5 shows, however, that this is achieved at the expense of scaffold accuracy: besides having an increased number of breakpoints, scaffolds constructed based on related genomes alone have a high total assembly length and, as a consequence, a low scaffold coverage. The higher than anticipated total assembly length is due to differences in contig distances (i.e. physical distances obtained by aligning contigs to a genome sequence) between the MG1655 strain and the related strains. This is also the reason for the large differences observed between breakpoints at $\Delta = 10$ kbp and $\Delta = 500$ bp: while relative order and orientation have been preserved for large parts of the genomes of the considered strains, the exact physical distances have not. This situation is partially alleviated when information from the two genomes is combined, because (a) consistent links (derived from the two genomes) get higher weights after link bundling, and (b) the more closely related strain BW2952 was given a higher weight. In this case GRASS is able to further reduce the number of scaffolds without introducing new breakpoints.

Combining paired read data with information from individual related genomes allows for construction of a smaller number of scaffolds with fewer breakpoints than when using these data individually. The results vary between repeated runs of the algorithm, due to inconsistencies between linking information provided by paired reads and related genomes, combined with the stochastic nature of the optimization strategy used for solving the MIQP formulation. Depending on the intermediate solutions found, different contig links are disabled in the optimization process, leading to different final solutions and, thereby to different scaffolds. Table 5 hence shows a range of scaffold and breakpoint counts, and other results as averages over five repeated runs. This variability is smaller when all data is combined, since a “voting” approach can be implemented by setting $W$ and $e$ in such a way that all links supported by only a single information source have low weights and are ignored. Using all available information, GRASS reduced the number of scaffolds by 40% compared to just using paired reads, at the expense of introducing a single new breakpoint. The increase in the number of breakpoints is not surprising, as the de novo scaffolding information is augmented with links derived for a different (related) organism. The best result on combined data is shown in Figure 4 (a).

### 5 Conclusion

We presented GRASS, a generic scaffolding algorithm suitable for combining multiple information sources, as well as GRASS+, incorporating a post-processing scaffolding step. Its use was demonstrated by scaffolding genomes based on paired read data and information in related genome sequences, both individually and combined. GRASS achieves the best results when all available scaffolding information is used, as this allows conflicting information from a single source to be ignored when the majority of sources do not support it. Such a mode of operation is supported.
Table 2. Contiguity and accuracy statistics of the initial assembly of *E. coli* and its scaffolds. Results with the smallest number of breakpoints or scaffolds are shown in bold.

<table>
<thead>
<tr>
<th>Scaffold</th>
<th>Breakpoints $\Delta = 10$ kbp</th>
<th>Breakpoints $\Delta = 500$ bp</th>
<th>Number of scaffolds</th>
<th>N50 Maximum length, bp</th>
<th>Total length, bp</th>
<th>Reference covered</th>
<th>Scaffolds covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velvet contigs</td>
<td>1</td>
<td>1</td>
<td>481</td>
<td>19,872</td>
<td>73,062</td>
<td>4,535,181</td>
<td>97.44%</td>
</tr>
<tr>
<td>Velvet scaffolds</td>
<td>9</td>
<td>10</td>
<td>106</td>
<td>171,726</td>
<td>312,219</td>
<td>4,561,490</td>
<td>97.98%</td>
</tr>
<tr>
<td>SSPACE</td>
<td>15</td>
<td>16</td>
<td>63</td>
<td>178,023</td>
<td>374,265</td>
<td>4,547,685</td>
<td>97.79%</td>
</tr>
<tr>
<td>GRASS</td>
<td>6</td>
<td>6</td>
<td>116</td>
<td>117,964</td>
<td>267,989</td>
<td>4,546,975</td>
<td>97.53%</td>
</tr>
<tr>
<td>GRASS+</td>
<td>2</td>
<td>2</td>
<td>120</td>
<td>112,254</td>
<td>268,030</td>
<td>4,546,640</td>
<td>97.53%</td>
</tr>
<tr>
<td>MIP Scaffold</td>
<td>1</td>
<td>12</td>
<td>148</td>
<td>89,070</td>
<td>221,548</td>
<td>4,546,430</td>
<td>97.54%</td>
</tr>
<tr>
<td>OPERA (SRR001665)</td>
<td>36</td>
<td>38</td>
<td>323</td>
<td>32,799</td>
<td>131,842</td>
<td>4,544,447</td>
<td>97.52%</td>
</tr>
<tr>
<td>OPERA (SRR001666)</td>
<td>26</td>
<td>28</td>
<td>262</td>
<td>37,330</td>
<td>126,797</td>
<td>4,556,203</td>
<td>97.52%</td>
</tr>
</tbody>
</table>

Table 3. Contiguity and accuracy statistics of the initial assembly of *P. suwonensis* and its scaffolds.

<table>
<thead>
<tr>
<th>Scaffold</th>
<th>Breakpoints $\Delta = 10$ kbp</th>
<th>Breakpoints $\Delta = 500$ bp</th>
<th>Number of scaffolds</th>
<th>N50 Maximum length, bp</th>
<th>Total length, bp</th>
<th>Reference covered</th>
<th>Scaffolds covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velvet contigs</td>
<td>1</td>
<td>1</td>
<td>303</td>
<td>26,043</td>
<td>90,572</td>
<td>3,394,128</td>
<td>99.01%</td>
</tr>
<tr>
<td>Velvet scaffolds</td>
<td>4</td>
<td>5</td>
<td>134</td>
<td>57,614</td>
<td>153,169</td>
<td>3,395,148</td>
<td>99.03%</td>
</tr>
<tr>
<td>SSPACE</td>
<td>2</td>
<td>2</td>
<td>127</td>
<td>60,526</td>
<td>151,961</td>
<td>3,388,872</td>
<td>99.09%</td>
</tr>
<tr>
<td>GRASS</td>
<td>2</td>
<td>2</td>
<td>138</td>
<td>62,908</td>
<td>152,258</td>
<td>3,394,155</td>
<td>99.02%</td>
</tr>
<tr>
<td>GRASS+</td>
<td>1</td>
<td>1</td>
<td>144</td>
<td>53,211</td>
<td>151,938</td>
<td>3,389,098</td>
<td>99.02%</td>
</tr>
<tr>
<td>MIP Scaffold</td>
<td>31</td>
<td>32</td>
<td>318</td>
<td>52,743</td>
<td>115,278</td>
<td>3,390,104</td>
<td>99.03%</td>
</tr>
<tr>
<td>OPERA (SRR097515)</td>
<td>1</td>
<td>1</td>
<td>302</td>
<td>26,053</td>
<td>90,582</td>
<td>3,397,028</td>
<td>99.01%</td>
</tr>
<tr>
<td>OPERA (SRR191848)</td>
<td>4</td>
<td>4</td>
<td>225</td>
<td>34,214</td>
<td>90,572</td>
<td>3,397,095</td>
<td>99.02%</td>
</tr>
</tbody>
</table>

by the possibility of choosing weights for the individual information sources, combined with the contig link erosion threshold.

We compared GRASS to a number of state-of-the-art scaffolders (SSPACE, MIP and OPERA) on three datasets. GRASS constructs the most accurate scaffolds on all datasets, while keeping the number of scaffolds low. Only SSPACE consistently produces lower numbers of scaffolds, but these are significantly less accurate. The accuracy/contiguity tradeoff displayed by GRASS puts it in a unique niche compared to existing scaffolders.

The current implementation of GRASS supports the use of paired read information and related genomes for scaffolding. However, the algorithm is not limited to any particular set of information sources. We will extend GRASS to allow use of other sources, such as optical restriction maps, RNA-seq and EST data.

**ACKNOWLEDGEMENT**

**Funding:** This work has been supported by the Plant Green Synthetic Biology Consortium and the Kluiver Centre for Genomics of Industrial Fermentation, subsidiaries of the Netherlands Genomics Initiative (NGI).

**Conflict of Interest:** none declared.

**REFERENCES**


Table 4. Contiguity and accuracy statistics of the initial assembly of *P. syringae* and its scaffolds.

<table>
<thead>
<tr>
<th>Scaffolder</th>
<th>Breakpoints Δ = 10 kbp</th>
<th>Breakpoints Δ = 500 bp</th>
<th>Number of scaffolds</th>
<th>N50</th>
<th>Maximum length, bp</th>
<th>Total length, bp</th>
<th>Reference covered</th>
<th>Scaffolds covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velvet contigs</td>
<td>1</td>
<td>1</td>
<td>1,560</td>
<td>8,599</td>
<td>46,055</td>
<td>5,902,217</td>
<td>96.41%</td>
<td>99.78%</td>
</tr>
<tr>
<td>Velvet scaffolds</td>
<td>25</td>
<td>27</td>
<td>200</td>
<td>122,286</td>
<td>683,615</td>
<td>6,012,535</td>
<td>97.78%</td>
<td>99.37%</td>
</tr>
<tr>
<td>SSSPACE</td>
<td>21</td>
<td>26</td>
<td>192</td>
<td>87,996</td>
<td>520,403</td>
<td>5,946,936</td>
<td>96.61%</td>
<td>99.09%</td>
</tr>
<tr>
<td>GRASS</td>
<td>22</td>
<td>25</td>
<td>243</td>
<td>85,493</td>
<td>618,916</td>
<td>5,931,679</td>
<td>95.57%</td>
<td>99.38%</td>
</tr>
<tr>
<td>GRASS+</td>
<td>6</td>
<td>7</td>
<td>266</td>
<td>77,945</td>
<td>460,726</td>
<td>5,945,096</td>
<td>95.36%</td>
<td>99.01%</td>
</tr>
<tr>
<td>MIP Scaffolder</td>
<td>37</td>
<td>47</td>
<td>167</td>
<td>94,327</td>
<td>279,875</td>
<td>5,943,358</td>
<td>95.68%</td>
<td>99.17%</td>
</tr>
<tr>
<td>OPERA</td>
<td>14</td>
<td>14</td>
<td>705</td>
<td>18,108</td>
<td>76,357</td>
<td>5,950,236</td>
<td>95.68%</td>
<td>99.17%</td>
</tr>
</tbody>
</table>

Table 5. Contiguity and accuracy statistics of *E. coli* scaffolds obtained with GRASS+ using additional data. The "≈" sign indicates mean values over 10 repeated runs in cases, when variation was observed.

<table>
<thead>
<tr>
<th>Reads used</th>
<th>DH10B used</th>
<th>BW2952 used</th>
<th>Breakpoints Δ = 10 kbp</th>
<th>Breakpoints Δ = 500 bp</th>
<th>Number of scaffolds</th>
<th>N50</th>
<th>Maximum length, bp</th>
<th>Total length, bp</th>
<th>Reference covered</th>
<th>Scaffolds covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>2</td>
<td>2</td>
<td>120</td>
<td>112,354</td>
<td>268,030</td>
<td>4,546,640</td>
<td>97.53%</td>
<td>99.55%</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>10</td>
<td>66</td>
<td>105</td>
<td>425,724</td>
<td>1,948,314</td>
<td>5,047,825</td>
<td>97.51%</td>
<td>89.65%</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>4</td>
<td>65</td>
<td>90</td>
<td>843,564</td>
<td>1,099,102</td>
<td>4,773,879</td>
<td>97.52%</td>
<td>92.64%</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>6</td>
<td>70</td>
<td>81</td>
<td>612,889</td>
<td>1,315,367</td>
<td>4,763,935</td>
<td>97.52%</td>
<td>94.80%</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>3-6</td>
<td>40-49</td>
<td>72-120</td>
<td>273,503</td>
<td>850,450</td>
<td>4,804,124</td>
<td>97.52%</td>
<td>94.99%</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>2-7</td>
<td>51-55</td>
<td>67-80</td>
<td>497,383</td>
<td>1,077,789</td>
<td>4,569,001</td>
<td>97.53%</td>
<td>99.06%</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>3</td>
<td>44-46</td>
<td>71-73</td>
<td>363,105</td>
<td>988,508</td>
<td>4,583,534</td>
<td>97.53%</td>
<td>98.75%</td>
</tr>
</tbody>
</table>


