# Automatically eliminating errors induced by suboptimal parameter choices in transcript assembly

Dan DeBlasio and Carl Kingsford
Computational Biology Department
Carnegie Mellon University
Pittsburgh, PA USA 15213.
{deblasio,carlk}@cs.cmu.edu

## Abstract

**Motivation:** The computational tools used for genomic analyses are becoming increasingly sophisticated and complex. While these applications provide more accurate results, a new problem is emerging in that these pieces of software have a large number of tunable parameters. The default parameter choices are designed to work well on average across all inputs, but the most interesting experiments are often not "average". Choosing the wrong parameter values for an application can lead to significant results being overlooked, or false results being reported. This problem is exacerbated when these applications are chained together in analysis pipelines where each step introduces errors due to parameter choices.
**Results:** We take some first steps towards generating a truly automated genomic analysis pipeline by developing a method for automatically choosing input-specific parameter values for reference-based transcript assembly. We apply the parameter advising framework, first developed for multiple sequence alignment, to optimize parameter choices for the `Scallop` transcript assembler. In doing so, we provide the first method for finding advisor sets for applications with large numbers of tunable parameters. This procedure can be parallelized, meaning it does not add any additional wall time. By choosing parameter values for each input, the area under the curve is increased by 28.9% over using only the default parameter choices on 1595 RNA-Seq samples in the Sequence Read Archive. This approach is general, and when applied to `StringTie` it increases AUC by 13.1% on a set of 65 RNA-Seq experiments from ENCODE.
**Availability:** A parameter advisor for `Scallop` is available on Github (https://github.com/Kingsford-Group/scallopadvising).

## 1 Introduction

As the field of computational biology has matured, there has been a significant increase in the amount of data that needs to be processed and the reliance of users without computational expertise on the highly complicated programs that perform the analyses. At the same time, the number and sophistication of such tools has also increased. While the accuracy of such applications is constantly improving, a new problem has emerged: the sometimes overwhelming number of tunable parameters that each of these sophisticated pieces of software brings with them. Changing an application's parameter settings can have a large impact on the quality of the results produced. When incorrect or non-ideal parameter choices are used, significant results can be overlooked or false conclusions can be reported. This is exacerbated when these programs are linked together to create analysis pipelines, as errors can propagate and be amplified at each stage.

The default parameter choices that most users rely on for these complex programs are typically optimized by the the algorithm designer to maximize performance on the average case. This can be a problem since the most interesting experiments are often not "average."

Manually tuning the parameter settings of an application often produces more accurate results, but it is very time consuming. The tuning process can be accelerated for users with domain and/or algorithmic knowledge, as these experts can make more informed decisions about the correct direction to proceed when altering parameter values. But tuning the parameter choices to increase accuracy for one input does not imply that the results will be improved for all inputs. This means that, for optimum performance, tuning must be repeated for each new piece of data.

In the case of high-throughput genomic analysis pipelines, this manual procedure is utterly infeasible. For these applications, without some sort of automatic parameter choice system, the defaults must be used. As these pipelines get longer and more sophisticated, the errors being introduced by the use of non-tuned parameter choices are compounded further.

To address the automated parameter choice problem for multiple sequence alignment (MSA), DeBlasio and Kececioglu [2017a; DeBlasio et al., 2012, Kececioglu and DeBlasio, 2013] have defined a framework to automatically select the parameter values for an input. This process, called "parameter advising," has been shown to greatly increase accuracy of MSA without sacrificing wall-clock running time in most cases, and it can readily be applied to new domains. A parameter advisor, depicted in Figure 1, has two components: (1) a set of parameter choices – assignments of a value to each of the tunable parameters for the application, called an "advisor set"; and (2) an assessment criteria – a method to rank the quality of multiple solutions, called an "advisor estimator". The advisor selects the appropriate parameter choice by first running the application on the input using each parameter choice in the set, and selecting the parameter choice that produces the best result according to the accuracy estimator. Parameter advising for a given application is fast in practice. The instantiations of the application being tuned are independent processes that can be executed in parallel. Assuming that the number of processors available is at least matches the number of parameter choices in the advisor set, the only additional wall time is the assessment of the results using the accuracy estimator (which can also be performed in parallel) and the comparison of these values, both of which are negligible compared to the running time of the application in most cases.

Parameter advising is an example of *a posteriori* parameter selection — it examines an application's output to select a parameter setting. In contrast, separate work has been done in other fields on *a priori* selection, where the parameters are chosen in advance by looking at the raw input. This includes work such as `SATZilla` [Xu et al., 2008] for choosing from a collection of SAT solvers, or `ParamILS` [Hutter et al., 2009] which finds optimal settings for the `CPLEX` computational optimization tool. More information is available when performing *a posteriori* assessment since the full final solution can be examined, but *a priori* prediction is necessary in cases when it is not feasible to apply multiple configurations.

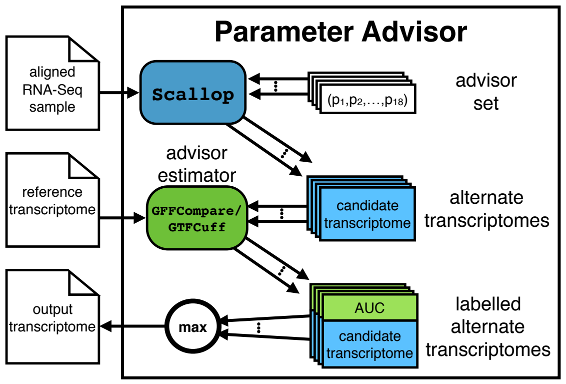

Figure 1: The parameter advising process. An advisor takes as input an RNA-Seq sample that has already been aligned to the reference genome, this mirrors the input to `Scallop`. Using the advisor set a collection of candidate transcriptomes is generated, one for each parameter choice (assignment of a value to each of the tunable parameters of `Scallop`) in the set. An AUC value is assigned to each candidate transcriptome by comparing it to the reference using a a combination of `GFFCompare` and `GTFCuff`. The advisor then returns the transcriptome with the highest AUC value.

We focus this work on improving the performance of reference-based transcriptome assembly by applying parameter advising. Common tools for this task include `Scallop` [Shao and Kingsford, 2017], `Cufflinks` [Trapnell et al., 2010], `StringTie` [Pertea et al., 2015], and `TranscComb` [Liu et al., 2016]. Transcriptome assembly is an essential step in many genomic analysis pipelines. It takes an RNA-Seq sample and reference genome as input and reconstructs the set of transcripts that are present. The assembler first aligns reads to the reference genome using a tool such as `HISAT` [Kim et al., 2015], `STAR` [Dobin et al., 2013], `TopHat` [Kim et al., 2013], or `SpliceMap` [Au et al., 2010]. Using the read splice locations (the positions where a read maps to non-neighboring locations on a genome) the assembler constructs the exons of each transcript. The produced transcriptome consists of a combination of transcripts that can be mapped to ones we already know and transcripts that are unique to the sample that was provided. These transcriptomes are used to perform analyses such as gene quantification [Patro et al., 2017] and differential expression [Love et al., 2014].

Figure 2 shows an example of just how much impact using non-optimal parameter choices can have on a transcriptome assembly. This example shows a region of Chromosome 2 with transcriptomes found using `Scallop` with different parameter choices. If the default parameters had been used, two transcripts at this location alone would not have been identified; both of these transcripts are present in the reference transcriptome and supported by the sequencing reads.

For transcriptome assembly, a natural choice for the estimator is the sensitivity and precision of recovering known transcripts. These measures can be combined in the area under the curve (AUC) metric, which is commonly used to benchmark reference-based transcript assembly tools. AUC measures the area under the receiver operating characteristic (ROC) curve which plots the sensitivity and precision of the computed transcriptome compared to the reference as the minimum transcript coverage threshold is changed. The average coverage of a transcript is the average number of reads that are aligned to each position along the transcript's length. As the threshold on this value increases, the induced subset of transcripts that are above this threshold is reduced in size. These subsets each have an associated sensitivity and precision with respect to the reference transcriptome and represent a point in a two dimensional space, together these points form the ROC curve. Tools such as `GFFCompare` (https://github.com/gpertea/gffcompare) and `GTFCuff` (https://github.com/Kingsford-Group/rnaseqtools) are designed specifically to calculate these measures.

### Contributions

The major contributions of this work are twofold: first, we show for the first time that sets of alternate parameter choices in certain domains can be found using methods other than exhaustive enumeration; and second, we take some of the first steps towards producing a fully automated genomic analysis pipeline by automating sample-specific parameter selection.

There are certain properties of the interaction between parameter choices and accuracy that can be exploited for some applications. When these interactions are generally continuous, iterative optimization techniques can be used to find an advising set. We describe the requirements an application domain must meet in order for these optimization techniques to be used, and show that transcriptome assembly with `Scallop` meets this threshold.

We will show that by then applying the parameter advising framework, we can greatly increase the quality of the transcriptomes produced using the `Scallop` assembly tool. Using our new tool to construct reference-based transcriptomes, the area under the curve shows a median increased of 8.7% over using only the default parameter choice on a set of 10 RNA-Seq experiments contained in the ENCODE database that are commonly used for benchmarking. The median improvement is even larger, 28.9% higher AUC than the default parameter choice, in a simulated high-throughput pipeline that uses over 1500 samples from the Sequence Read Archive.

We go on to confirm that this method can increase AUC for other applications by applying parameter advising to `StringTie`. For a set of 65 examples from the ENCODE database we are able to increase accuracy by 13.1% over using only the default parameter choices.

## 2 Methods

Transcriptome assembly using `Scallop` presents an interesting problem for parameter advising. Even though AUC, the standard metric for transcriptome assembly, is able to be used as the "advisor estimator," finding an advisor set is especially challenging. The set of tunable parameters that need to be set is much larger than in previous applications: 18 compared to the 5 for multiple sequence alignment. This means the previously developed method of enumerating a parameter choice universe then using combinatorial optimization to find an advisor set is infeasible. However, we have an advantage in that in this application almost all 18 of the tunable parameters are continuous rather than discrete values, the remaining parameters are binary and thus have only two possible values.

In this work, we are finding new values for the 18 tunable parameters based on the input, and while we do not interpret the specific meaning of `Scallop` parameters and the optimal values that we will find, it is helpful to have a sense of what parts of the application are impacted. The `Scallop` transcript assembler generates a transcriptome from a set of reads that have been aligned to a reference genome. It first splits the genome into regions of non-overlapping reads, which are called bundles. These bundles can be thought of as genes or groups of overlapping genes. Then, within each bundle a splice graph is constructed based on the split reads that define possible exon boundaries. Paths through the splice graph define potential transcripts, and the final set of transcripts is formed by decomposing the splice graphs

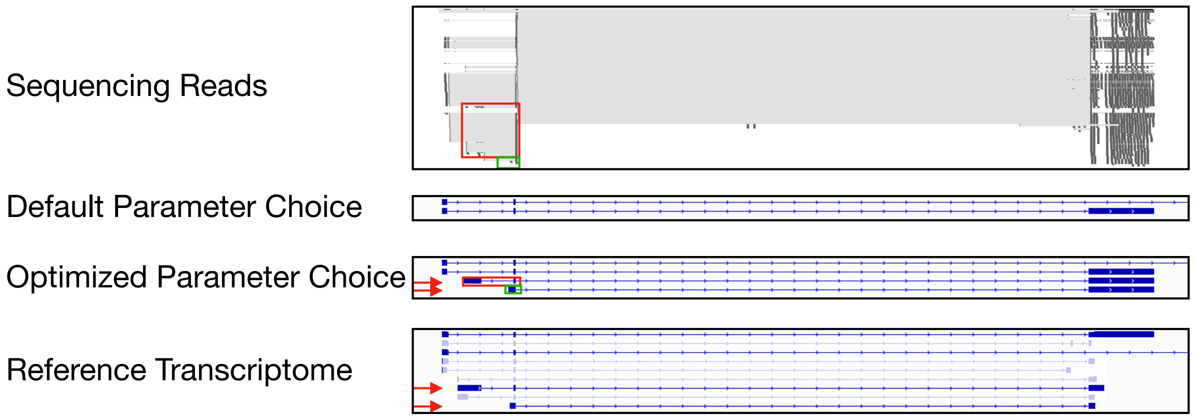

Figure 2: Impact of parameter choice on the produced transcriptome. The four panels show the raw RNA-Seq reads aligned to a region of the human genome, the transcript predictions found using the `Scallop` default parameter choice, the predictions using the parameter choice found using coordinate ascent, and the human reference transcriptome. In the first panel, reads are shown as black boxes, and the gaps within split reads are shown in grey. The input and predicted transcripts are from `SRR543291` which was aligned to the genome using `HISAT` and the region shown are positions 30231125 to 30260786 on Chromosome 2. The optimized parameter choice identifies two transcripts that are both supported by the read data and match with transcripts that are present in the reference. The supporting read information and corresponding exons for the two new transcripts (red arrows) is highlighted in the first and third panels respectively with red and green boxes.

into paths while trying to respect as many of the read mappings as possible. The 18 tunable parameters of `Scallop` govern various stages of this process and are listed below:

- maximum dynamic programming table size (DP)
- maximum edit distance (ED)
- maximum intron contamination coverage (ICC)
- maximum number of exons (NE)
- minimum bundle gap (BG)
- minimum exon length (EL)
- minimum flank length (FL)
- minimum mapping quality (MQ)
- minimum number or hits in a bundle (NH)
- minimum router count (RC)
- minimum splice boundary hits (SBH)
- minimum subregion gap (SG)
- minimum subregion length (SL)
- minimum subregion overlap (SO)
- minimum transcript length base (TLB)
- minimum transcript length increase (TLI)
- uniquely mapped reads only (UM)
- use the secondary alignment (US)

In this work, as opposed to Shao and Kingsford [2017], we choose to not only focus on multi-exon transcripts but include single-exon transcripts in our calculation of AUC as well. In order to produce the AUC value, we first calculate the transcriptome using `Scallop` with the minimum transcript coverage values set to 0 and 20 for multi-exon and single-exon transcripts respectively. The ROC is then calculated by thresholding the computed coverage of the resulting transcripts. In later sections when we use `StringTie`, we perform the same procedure by setting the corresponding minimum coverage set to 0.001 (the smallest value allowed).

One possible solution to finding advisor sets for an application with a large number of tunable parameters such as `Scallop` would be to identify a subset of these parameters then, as with MSA, exhaustively enumerate the universe of all parameter choices. This would necessitate eliminating some tunable parameters, but it is not trivial to identify the parameters that can be ignored without a loss of accuracy. Another option, and the one we will use here, is to intelligently explore the parameter landscape to find a subset of parameter choices that can be used for advising.

## 2.1 Analyzing parameter behavior

Iterative optimization strategies such as gradient ascent [Cauchy, 1847], simulated annealing [Kirkpatrick et al., 1983], and coordinate ascent [Zangwill, 1969, sec. 5.4.3], work by systematically searching high-dimensional spaces based on a specific optimization criteria. They do so by repeatedly examining the local accuracy landscape, either by calculating a gradient or exploring neighboring points, and updating the current location in the direction of maximum increase. But, these methods do not perform well when the space has a large number of similar local optima or when there are discontinuities in the optimization landscape. Therefore, there are several features that differentiate application domains that can use iterative optimization techniques to find more accurate parameter choices from those that require the use of exhaustive enumeration. First, in order to find optimal parameters using iterative optimization, the performance as the parameter is varied should only have a single maxima. Second, exhaustive enumeration must be used for parameters that have tunable parameters that take discrete, non-ordered values.

To determine the behavior of each parameter, we calculate the AUC of the transcriptomes produced using a large list of possible settings of one parameter while keeping the remaining parameters at their default values. Figure 3 shows the effect of varying the "minimum subregion gap" and "minimum transcript length base" parameters. Note that throughout this work, we multiply area under the curve values by $10^4$ for ease of comparison, AUC is a value in the range $[0, 1]$, but generally for transcript assembly the value is very small, typically $< 0.1$.

After examining these curves for several experiments from the ENCODE database, we found that the shape of the curves for all 16 continuous parameters contained only one visible local maximum. This suggests that the parameter space is free from too many local optima.

It is not possible to confirm that the entire landscape of parameter values has a low number of local maxima, but we can go beyond looking at one parameter in isolation. To determine if the full parameter landscape has only one maxima, we examine pairs of parameters in the same manner as above. While not all pairs were visualized, the ones tested all showed single maximal points in two dimensions (an example is shown in Figure 4). These tests show that there is strong evidence that there is likely to be very few local maxima in the high dimensional parameter space.

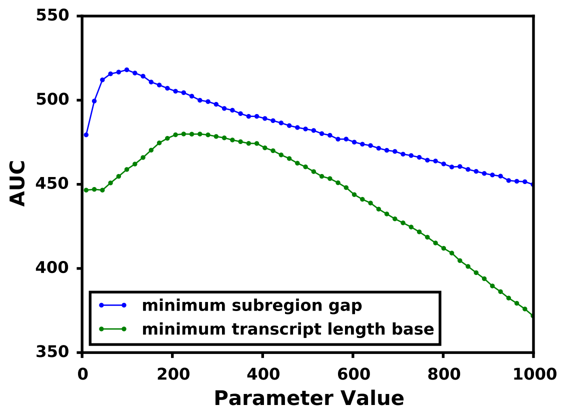

Figure 3: Area under the curve for various values of the "minimum subregion gap" and "minimum transcript length base" parameters. The points in the plot show the area under the curve (vertical axis) for the transcriptome produced by changing the value each of the parameters (horizontal axis) while assigning all other parameters their default values on SRR534291/`HISAT` from `ENCODE10`.

## 2.2 Finding an advisor set using coordinate ascent

Because the parameter landscape does not have a large number of local maxima, parameter choices can be found in practice using iterative optimization. The greedy coordinate-ascent-based procedure we use here starts from the point on the AUC landscape where all of the parameters are set to their default values. Then for one dimension (parameter) at a time we examine the AUC of the points that are one step in both directions. If one of the new values leads to a point with higher AUC, we update the position in that dimension. We continue to move in that dimension until doing so no longer increases AUC. In our procedure, we always choose to move to the maximum AUC point, meaning that there is no probability of taking a step that would allow AUC to stay the same or decrease. The step size for each dimension has a time-versus-granularity tradeoff: the larger the step size, the less time spent in low AUC regions of the landscape; but when the size is too large, the maximum may be repeatedly stepped over without ever being found. Rather than use a computed step size for every coordinate in each iteration like many implementations of coordinate ascent we over-

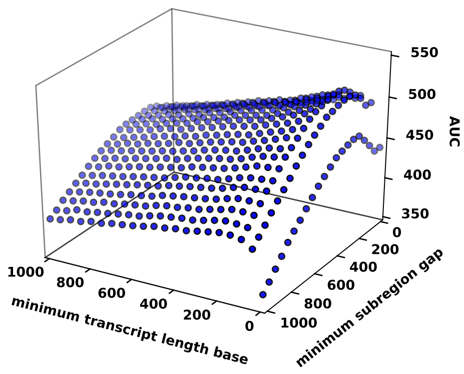

Figure 4: Area under the curve for various values of the "minimum subregion gap" and "minimum transcript length base" parameters. The points in the plot show the area under the curve (z-axis) for the transcriptome produced by changing the value of the "minimum subregion gap" (x-axis) and "minimum transcript length base" (y-axis) parameters while assigning all other parameters their default values on SRR534291/`HISAT` from `ENCODE10`.

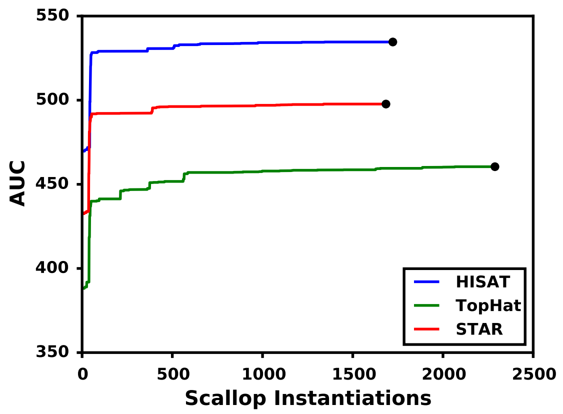

Figure 5: Area under the curve for each step of the coordinate ascent procedure for SRR534291 from `ENCODE10` using all 3 aligners. Each curve in the plot shows the progress of the coordinate ascent landscape exploration for one of `HISAT`, `STAR`, and `TopHat`. Across the horizontal axis is the number of exploratory steps taken in the search and the vertical shows the area under the curve for the current best parameter choice.

come this issue using predetermined, but decreasing, step sizes. We start with large step sizes in each dimension, and any time we interrogate the whole set of parameters without making any change we decrease all of the step sizes by a factor of $\frac{1}{4}$ and repeat the process. This continues until all of the step sizes are small (1 for integer parameters and 0.01 for real numbers) and no more changes to the location are made. For the tunable parameters in `Scallop` that accept only binary input (namely UM and US), the same rules as integer parameters are applied but with an initial step size to 1 and limiting the range to 0 and 1 ('false' and 'true').

Figure 5 shows the trajectory of three such coordinate ascent training sessions. The large increases in AUC in the initial iterations are likely the procedure moving away from the multi-exon optimized parameter choice. As the sessions continue, the increase in AUC becomes smaller because the procedure is narrowing in on the apparent true maximum.

Coordinate ascent will find higher-AUC parameter choices for an input, but it is slow so it is not a viable candidate for this task if the results are needed in a reasonable amount of time. Instead, we can exploit the power of iterative optimization to explore the parameter landscape in order to develop the advisor sets we need. Because the advising set can be computed in advance, the high time commitment of iterative optimization does not impact the running time of advising. To find an advising set, we use a collection of training experiments. For each of these training points, we find their optimal parameter choices using coordinate ascent and use the produced collection as the advising set for other inputs. For the advising set to be generalizable, the set of training examples needs to be representative of the range of possible inputs. In Section 3, we use a carefully constructed, highly diverse set of existing RNA-Seq samples (described next) in order to ensure that our advising set will perform well on new inputs.

## 2.3 Data

We use 3 sets of available data to train and validate our new method:

- `ENCODE10` contains a collection of 10 RNA-seq experiments from the ENCODE database [The ENCODE Project Consortium, 2012] that were used to benchmark `Scallop` and have been extensively used to evaluate transcriptome assembly tools [Pertea et al., 2015, Liu et al., 2016]. Each sample was aligned to the human reference genome (GRCh38) using three tools: `HISAT`, `STAR`, and `TopHat`, producing a set of 30 examples. A subset of these samples was used to manually find the Scallop default parameter choice.

- ENCODE65 contains a collection of 65 RNA-seq experiments, also from ENCODE, that were not included in ENCODE10 and had preexisting alignments in the database. These alignments are produced using an aligner selected by the group that submitted the sample and are mapped to either GRCh37 or GRCh38.
- SRA contains a collection of 1595 RNA-seq experiments from the Sequence Read Archive [Leinonen et al., 2010] that have been filtered for quality. We limit the set to only those samples that contained enough data to be informative — in this case we ensure that there are at least 1GB of raw reads in each selected sample. We also eliminate any sample that contained no or very few reads that mapped to the human genome — we threshold the alignment files that are below 1GB. All of the remaining samples were aligned using STAR to GRCh38.

## 3 Validating the AUC increase using Scallop

To demonstrate the increase in AUC that can be gained from using parameter advising for transcriptome assembly, we first construct an advisor set using ENCODE10 and coordinate ascent. We then use this advisor set to perform advising on samples from ENCODE65 and SRA.

### 3.1 Finding a Scallop advisor set

To find a generalizable advising set for Scallop, we need to start with a highly diverse set of training examples. The ENCODE10 dataset is good for this purpose because it contains samples that are widely accepted as benchmarks and has examples that have been generated using a diverse set of aligners. We then use the coordinate ascent procedure described in Section 2.2 to find improved parameters for each sample. The final settings found for each of these 30 coordinate ascent runs are shown in Table 1 and the improvement in AUC is shown in Table 2. Most of the parameters values deviate quite far from each other, meaning this set of parameter choices is input-specific. The deviation of the parameter choices from the default is not surprising given that in this work we are optimizing AUC on all transcripts, rather than only multi-exon ones as was done previously. This means that the original parameter settings, which worked very well when examining only multi-exon reads, are not ideal for examining all transcripts. In the case when all transcripts are considered a new default parameter setting could be recommended for some single parameters, such as in the case of “minimum mapping quality” where almost all samples used a value of 11 rather then the default of 1, and “minimum transcript length increase” where most samples found improvement by selecting values that are much smaller than the default. In fact, we will show later that a new full default parameter choice that is more accurate on all examples can be found using this method; that is, one of the parameter choices had higher average AUC on all of the training samples than the predefined default parameter choice. Only 15 rather than 18 parameters are shown because we found that changing the other 3 (DP, ED, and RC) never have a positive impact on the AUC. While these parameters were examined in coordinate ascent they are not shown because all instances use the default.

In a server environment, being able to run 31 instances of the Scallop application (30 parameter choices found using coordinate ascent and the default parameter choice) in parallel often is not a problem, and thus parameter advising provides a method that takes only as much wall time as using the default parameter, although with 31 times the CPU cycles. In reduced resource environments, it may be desirable to run fewer parameter settings to keep the number of parallel processes smaller than the number of available threads. We use the oracle set-finding method described by DeBlasio and Kececioglu [2017a,b] to find a subset of parameter choices that maximizes the average AUC for advising. This set is calculated using an integer linear program that has two sets of binary variables: one variable for each parameter choice, and one for each example. Where an example is a parameter choice used to assemble a sample. Constraints are used to ensure that only one example for each sample is chosen, and that the associated parameter setting is also chosen. The objective is then to maximize the sum of the accuracies of the chosen examples while only selecting a predefined number of parameter choices. Using the samples in ENCODE10, we found advising subsets of 1, 2, 4, and 8 parameter vectors. The subsets induce an assembly tool that can be run in reduced resource environments, such as on an individual desktop. The advising subset choices are shown in the right most columns of Table 1. Note that an advising set of size 1 is equivalent to finding a new default parameter choice since it maximizes the average accuracy across the training examples.

Table 1: Optimal parameter choice vectors found by coordinate ascent

| ID | Experiment | Aligner | ICC | NE | BG | EL | FL | MQ | NH | SBH | SG | SL | SO | TLB | TLI | UM | US | 1 | 2 | 4 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SRR545723 | `TopHat` | 2.00 | 1000 | 1359 | 20 | 3 | 11 | 20 | 1 | 39 | 11 | 1.50 | 454 | 8 | false | false | X |  | X | X |
| 2 | SRR307911 | `TopHat` | 2.00 | 1000 | 1032 | 20 | 2 | 11 | 21 | 1 | 52 | 15 | 1.50 | 432 | 19 | false | false |  |  |  |  |
| 3 | SRR534291 | `TopHat` | 0.75 | 51 | 785 | 25 | 3 | 1 | 9 | 1 | 121 | 16 | 1.51 | 503 | 8 | false | false |  |  |  | X |
| 4 | SRR387661 | `TopHat` | 0.22 | 1000 | 999 | 20 | 3 | 11 | 20 | 1 | 53 | 15 | 1.50 | 429 | 19 | false | false |  | X |  |  |
| 5 | SRR315334 | `TopHat` | 1.50 | 1000 | 1425 | 20 | 3 | 11 | 20 | 1 | 50 | 16 | 1.50 | 528 | 1 | false | false |  |  |  |  |
| 6 | SRR307903 | `TopHat` | 1.61 | 250 | 733 | 2 | 0 | 11 | 2 | 1 | 23 | 26 | 0.75 | 521 | 2 | false | false |  |  |  |  |
| 7 | SRR545695 | `TopHat` | 2.00 | 1000 | 852 | 20 | 2 | 2 | 104 | 1 | 5 | 7 | 1.50 | 432 | 19 | false | false |  |  |  |  |
| 8 | SRR534319 | `TopHat` | 2.27 | 1000 | 306 | 20 | 0 | 1 | 20 | 1 | 7 | 11 | 1.50 | 346 | 21 | false | false |  |  |  |  |
| 9 | SRR315323 | `TopHat` | 0.22 | 205 | 945 | 20 | 0 | 2 | 20 | 1 | 39 | 15 | 1.16 | 409 | 8 | true | true |  |  |  |  |
| 10 | SRR534307 | `TopHat` | 2.22 | 1000 | 78 | 32 | 3 | 11 | 16 | 2 | 0 | 32 | 1.50 | 512 | 8 | false | false |  | X |  |  |
| 11 | SRR545723 | `HISAT` | 1.75 | 11000 | 628 | 20 | 3 | 11 | 20 | 2 | 8 | 7 | 1.50 | 408 | 8 | false | false |  |  |  |  |
| 12 | SRR307911 | `HISAT` | 99.66 | 1000 | 1828 | 3 | 5 | 11 | 3 | 1 | 43 | 15 | 0.17 | 392 | 8 | false | false |  |  | X |  |
| 13 | SRR534291 | `HISAT` | 1.00 | 11000 | 851 | 22 | 5 | 11 | 3 | 1 | 120 | 3 | 0.17 | 267 | 30 | false | false |  |  |  |  |
| 14 | SRR387661 | `HISAT` | 1.05 | 454 | 130 | 26 | 4 | 11 | 38 | 1 | 23 | 0 | 0.17 | 880 | 0 | false | false |  |  |  | X |
| 15 | SRR315334 | `HISAT` | 0.00 | 282 | 870 | 20 | 4 | 11 | 1 | 1 | 48 | 8 | 1.28 | 307 | 21 | false | false |  |  |  |  |
| 16 | SRR307903 | `HISAT` | 1.42 | 168 | 745 | 9 | 5 | 11 | 2 | 1 | 23 | 6 | 0.17 | 396 | 0 | false | false |  |  |  |  |
| 17 | SRR545695 | `HISAT` | 5.56 | 1000 | 778 | 26 | 2 | 11 | 25 | 2 | 0 | 0 | 1.50 | 360 | 8 | false | false |  |  |  | X |
| 18 | SRR534319 | `HISAT` | 2.00 | 1000 | 1669 | 20 | 0 | 11 | 81 | 1 | 4 | 7 | 1.50 | 150 | 73 | false | false |  |  |  |  |
| 19 | SRR315323 | `HISAT` | 2.00 | 685 | 782 | 15 | 0 | 11 | 14 | 1 | 40 | 7 | 1.50 | 180 | 50 | false | false |  |  |  |  |
| 20 | SRR534307 | `HISAT` | 2.00 | 146 | 309 | 41 | 8 | 11 | 10 | 3 | 2 | 0 | 1.17 | 370 | 4 | false | false |  |  | X | X |
| 21 | SRR545723 | `STAR` | 2.00 | 11000 | 455 | 22 | 6 | 11 | 3 | 1 | 3 | 21 | 1.26 | 252 | 16 | false | false |  |  |  |  |
| 22 | SRR307911 | `STAR` | 0.22 | 1000 | 490 | 20 | 3 | 11 | 8 | 1 | 53 | 26 | 0.50 | 289 | 24 | false | false |  |  |  |  |
| 23 | SRR534291 | `STAR` | 0.22 | 11000 | 481 | 24 | 0 | 2 | 11 | 1 | 123 | 26 | 0.24 | 352 | 8 | false | false |  |  |  |  |
| 24 | SRR387661 | `STAR` | 2.00 | 1000 | 734 | 20 | 4 | 11 | 35 | 2 | 42 | 29 | 1.50 | 411 | 1 | false | false |  |  |  |  |
| 25 | SRR315334 | `STAR` | 0.50 | 1000 | 1070 | 8 | 0 | 11 | 8 | 1 | 41 | 29 | 0.75 | 267 | 33 | false | false |  |  |  | X |
| 26 | SRR307903 | `STAR` | 2.00 | 1000 | 976 | 8 | 5 | 11 | 20 | 1 | 35 | 35 | 0.17 | 433 | 8 | false | false |  |  |  |  |
| 27 | SRR545695 | `STAR` | 2.00 | 1000 | 388 | 20 | 8 | 11 | 89 | 2 | 20 | 23 | 1.50 | 307 | 19 | false | false |  |  |  |  |
| 28 | SRR534319 | `STAR` | 2.00 | 800 | 1574 | 17 | 4 | 2 | 20 | 1 | 8 | 18 | 1.50 | 307 | 4 | false | false |  |  |  | X |
| 29 | SRR315323 | `STAR` | 2.00 | 11000 | 1043 | 20 | 3 | 7 | 20 | 1 | 33 | 35 | 1.50 | 150 | 50 | true | true |  |  |  |  |
| 30 | SRR534307 | `STAR` | 1.58 | 292 | 54 | 20 | 0 | 11 | 9 | 3 | 0 | 29 | 1.50 | 320 | 8 | false | false |  |  | X | X |
|  | Default |  | 2.00 | 1,000 | 50 | 20 | 3 | 1 | 20 | 1 | 3 | 15 | 1.50 | 150 | 50 | false | false |  |  |  |  |
|  | Initial step size |  | 10.00 | 10,000 | 100 | 100 | 10 | 10 | 100 | 10 | 10 | 20 | 10.00 | 500 | 100 | n/a | n/a |  |  |  |  |

## 3.2 Testing advised Scallop

Using the set of 30 parameter choices provided by coordinate ascent, we can evaluate the improvement in AUC gained by using parameter advising to choose higher AUC parameter choices for the Scallop application. Each of the datasets provides unique insight into why the improved version of Scallop will produce more accurate assemblies in practice.

### 3.2.1 ENCODE10

Table 2 shows the AUC for all of the samples from ENCODE10. Because each sample was aligned using three different aligners, these results cover many of the possible input scenarios that may be used. For each example (a sample combined with an aligner) there are three values shown. The area-under the curve for the transcriptome using the default parameter choice, the trained parameter choice, and the "leave-one-out" (LOO) parameter choice. For the leave-one-out experiment, advising was limited to the 18 parameter choices that were learned on examples produced using the 2 aligners and 9 samples that were different from the example being tested. This test is used to show the robustness of the advising set.

One interesting point is that the most accurate parameter choice for the training samples is not always the one found using coordinate ascent. This occurs once in ENCODE10, for the experiment SRR534319/HISAT the parameter choice found for SRR545695/HISAT has a slightly higher area under the curve, 313.419 versus 315.670, respectively. This is likely due to the fact that while we do not see multiple local maxima when examining one or two parameters, they may exist in the higher dimensional parameter space, so we are not actually finding the optimal parameter choices. In addition to the running time benefits that come from using decreasing step sizes in coordinate ascent, starting with larger steps heuristically attempts to avoid local maxima by first finding regions of high AUC but local maxima cannot be avoided altogether if they exist. The AUC difference here is quite small, so we think the parameter space has small local maxima when examined at a very fine-grained scale. This phenomenon also suggests that some of the parameter settings that were changed may only have very minor impacts on area under the curve, these two parameter choices result in a similar final AUC but several tunable parameters have values that are quite different between them.

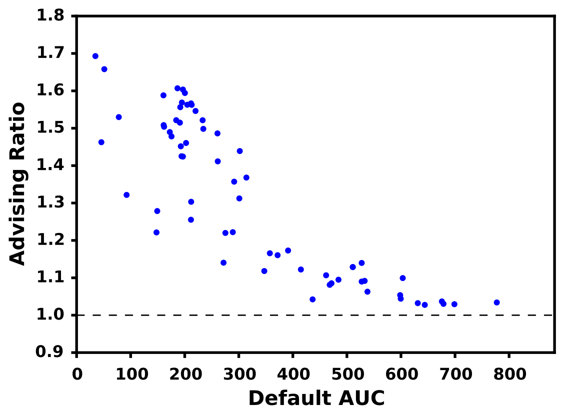

Figure 6: AUC improvement for the ENCODE65 assemblies. Each point in the plot is a single experiment in the data set, its position on the horizontal axis is the area under the curve for the transcriptome produced using the default parameter settings and its point on the vertical is the ratio of the maximum area under the curve for any parameter in the advising set over the default. A value above 1.0 indicates an improvement over the default parameters. For this test the default was excluded from the advising set.

### 3.2.2 ENCODE65

The ENCODE65 dataset is used to show that on a large number of samples from a range of aligners (possibly using non-default parameter settings) advising for Scallop provides a higher AUC transcriptome. Figure 6 shows the increase in AUC for all 65 examples in ENCODE65. The advising ratio shown on the vertical axis is the area under the curve for the advised transcriptome normalized by the AUC for the transcriptome produced using the default parameter choice. A value above 1.0 means the transcriptome produced using advising is more accurate. The figure indicates, as expected, that the higher the default area under the curve, the less room there is for improvement and thus, the smaller the advising ratio. Using advising on this highly diverse set of samples increases the AUC of each transcriptome by a median of 31.2%. When using the resource limited sets, the median increase remains 18.2%, 19.0% and 24.4% for sets of 2, 4, and 8 parameter settings respectively. Even with a small commitment in resources, there is a large increase in AUC.

Figure 7 shows the frequency with which each of the 31, 8, 4, and 2 parameter choices provides the maximum AUC when running parameter advising on the ENCODE65 set. Not all 31 parameter choices are used,

Table 2: Increase in AUC for examples in ENCODE10. Percentages in alternating rows show the increase over the default.

| | HISAT | | | STAR | | | TopHat | | |
|---|---|---|---|---|---|---|---|---|---|
| Experiment | Default | Learned | LOO | Default | Learned | LOO | Default | Learned | LOO |
| SRR307903 | 589.271 | 612.017 | 604.268 | 528.307 | 556.867 | 548.746 | 448.498 | 496.242 | 483.381 |
| | | 3.86% | 2.54% | | 5.4% | 3.86% | | 10.64% | 7.77% |
| SRR307911 | 503.460 | 549.553 | 544.157 | 477.174 | 519.346 | 514.035 | 387.947 | 446.299 | 438.225 |
| | | 9.15% | 8.08% | | 8.83% | 7.72% | | 15.04% | 12.96% |
| SRR315323 | 389.409 | 409.204 | 408.548 | 340.764 | 351.396 | 349.285 | 283.773 | 308.121 | 305.539 |
| | | 5.08% | 4.91% | | 3.12% | 2.5% | | 8.58% | 7.67% |
| SRR315334 | 549.081 | 579.641 | 573.181 | 501.790 | 532.407 | 525.885 | 413.393 | 472.124 | 466.392 |
| | | 5.56% | 4.38% | | 6.1% | 4.8% | | 14.2% | 12.82% |
| SRR387661 | 199.230 | 299.722 | 277.117 | 464.403 | 493.931 | 487.166 | 168.373 | 458.468 | 453.251 |
| | | 50.44% | 39.09% | | 6.35% | 4.9% | | 172.29% | 169.19% |
| SRR534291 | 469.952 | 533.842 | 522.227 | 432.317 | 496.602 | 481.104 | 388.038 | 460.242 | 440.755 |
| | | 13.59% | 11.12% | | 14.86% | 11.28% | | 18.6% | 13.58% |
| SRR534307 | 293.485 | 734.992 | 710.496 | 639.449 | 716.494 | 659.162 | 638.106 | 692.815 | 678.095 |
| | | 150.43% | 142.08% | | 12.04% | 3.08% | | 8.57% | 6.26% |
| SRR534319 | 303.001 | 313.419 | 308.410 | 257.493 | 267.573 | 260.004 | 244.602 | 267.438 | 257.182 |
| | | 3.43% | 1.78% | | 3.91% | 0.97% | | 9.33% | 5.14% |
| SRR545695 | 370.929 | 393.646 | 385.311 | 338.668 | 349.860 | 344.742 | 152.441 | 346.034 | 337.442 |
| | | 6.12% | 3.87% | | 3.3% | 1.79% | | 126.99% | 121.35% |
| SRR545723 | 537.776 | 551.086 | 548.686 | 525.978 | 533.369 | 524.506 | 458.757 | 492.322 | 485.531 |
| | | 2.47% | 2.02% | | 1.4% | -0.27% | | 7.31% | 5.83% |

but more parameter settings are used when they are available. Because the reduced size advisor sets are trained on ENCODE10 and used on ENCODE65 only 4 of the set of 8 are used, as well as only 2 of the 4.

### 3.2.3 SRA

The SRA set gives some insight into the improvement that can be gained in a high-throughput environment. It contains a large number of samples that have all been preprocessed in the same way with respect to the aligner. In this scenario, using advising greatly increases AUC without any manual adjustment. The advising ratio for the 1595 samples in SRA is shown in Figure 8 compared with the area under the curve for the transcriptome produced using the Scallop default parameter settings. Because, in general, the samples in SRA have a smaller initial area under the curve than those in Figure 6 (AUC values of 241.3 and 325.0 respectively), the median improvement is higher at 28.9%. For several samples in the set the AUC increases by more than a factor of 3. These improvements are also seen for the resource-limited advisor sets where the median improvement is 25.6%, 24.1% and 24.3% with 2, 4, and 8 parameter choices, respectively. Notice that the increase in AUC actually goes down slightly when increasing the size from

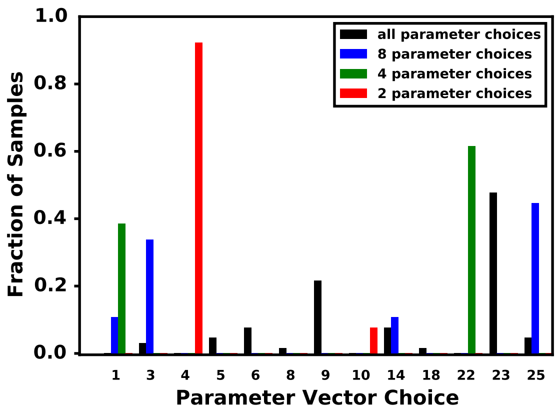

Figure 7: Parameter choice use within the ENCODE65 set. The horizontal axis shows labels of the parameter choices from Table 1 that produced the highest AUC transcriptome for any sample in ENCODE65. The vertical axis is the fraction of samples that have that parameter choice as the maximum. The four groups of bars show the use in the full set of 30 parameter choices and the reduced sets of 2, 4, and 8 parameter choices.

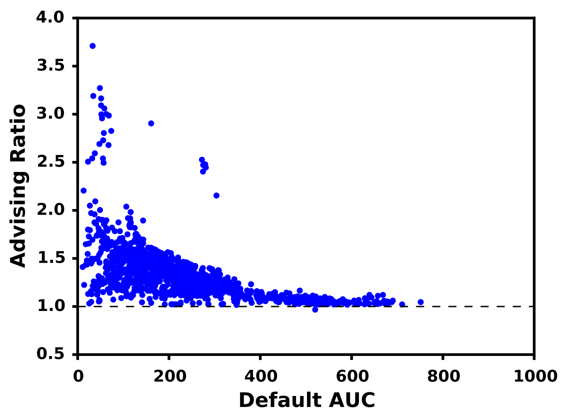

Figure 8: AUC improvement the SRA assemblies. Each point in the plot is a single experiment in the data set, its position on the horizontal axis is the area under the curve for the transcriptome produced using the default parameter settings and its point on the vertical is the ratio of the maximum area under the curve for any parameter in the advising set over the default. A value above 1.0 indicates an improvement over the default parameters. For this test the default was excluded from the advising set, but can be included in practice to ensure the AUC is never reduced.

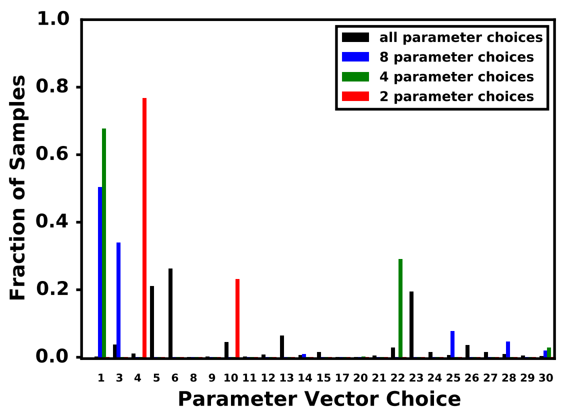

Figure 9: Parameter choice use within the SRA set. The horizontal axis shows labels of the parameter choices from Table 1 that produced the highest AUC transcriptome for any sample in SRA. The vertical axis is the fraction of samples that have that parameter choice as the maximum. The four groups of bars show the use in the full set of 31 parameter choices and the reduced sets of 2, 4, and 8 parameter choices.

2 to 4, this is likely an artifact of the advisor sets not being subsets of each other, this means that the parameter choices and sets may be slightly overfit to the training data.

Figure 9 shows the frequency with which each of the 31, 8, 4, and 2 parameter choices provides the maximum AUC when running parameter advising on the SRA set. More parameter choices are used than were for ENCODE65, which is expected because the set of samples is larger, but because SRA is somewhat homogeneous many of the choices are maximal more frequently. Surprisingly, even-though all of the examples are aligned using STAR, many of the higher-frequency parameter choices were optimized for examples that were aligned using TopHat (IDs 1–10).

## 3.3 Running time

As alluded to earlier, the wall time of running coordinate ascent is much larger than the running time of any single instance of Scallop. For the 30 examples from ENCODE10, running coordinate ascent took between about 40 hours and over 22 days. Since running Scallop using the default parameter choice for the same input takes between about 7 minutes and 1 hour, even if no parallelization was possible parameter advising would be able to run in a fraction of the time of running coordinate ascent.

## 3.4 Advising for other applications

In order to show the generalizability of the method, we also applied it to the StringTie transcript assembler. Just as before, we ran coordinate ascent on the 10 experiments in ENCODE10, then show the utility of using the 30 parameter choices to perform parameter advising on ENCODE65. Since StringTie has only 9 tunable parameters the coordinate ascent time was much shorter, but the increase in accuracy was still measurable. For the 30 coordinate ascent runs, we saw a median increase in AUC of 12.2%. We also saw 10.0% increase in AUC on the similar leave-one-out experiments as those performed above.

Figure 10 shows the advising ratio for the 65 RNA-Seq samples from ENCODE65. For these examples the median gain in AUC is 13.1% over using only the default parameter choices. For the StringTie assembler, samples with lower AUC using the default parameter choices still generally have higher advising ratios but this correlation is not as strong as with Scallop.

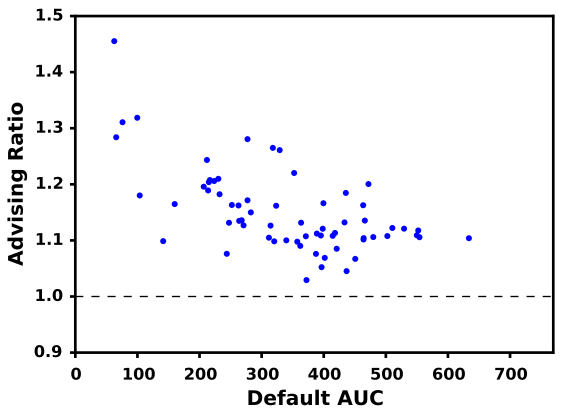

Figure 10: AUC improvement the `ENCODE65` assemblies using `StringTie`. Each point in the plot is a single experiment in the data set, its position on the horizontal axis is the area under the curve for the transcriptome produced using the default parameter settings and its point on the vertical is the ratio of the maximum area under the curve for any parameter in the advising set over the default. A value above 1.0 indicates an improvement over the default parameters. For this test the default was excluded from the advising set.

## 4 Discussion

Our results show that sample-specific parameter choices are essential to developing any strong genomic pipeline that includes transcriptome assembly as a step. In this work, we begin to answer the question of how to produce transcriptome assemblies effectively for any input without sacrificing quality or manpower. This is done using a combination of parameter tuning though exploration using coordinate ascent and the established method of parameter advising. Two key points that made this merger viable and distinguish transcriptome assembly from other domains are: (1) the insight that because the parameter landscape likely has few maxima, finding a suitable parameter set can be achieved by coordinate ascent rather than exhaustive enumeration, and (2) that a small number of training examples is sufficient to provide a large increase in AUC over the default parameter setting.

The coordinate ascent procedure used is a very useful means for finding parameter settings that increase the AUC of predicted transcriptomes. The best results can generally be achieved by simply following this procedure for each new input. Because of the interdependence between multiple parameters, our implementation of coordinate ascent does not allow steps to be taken in multiple directions at once. It is thus difficult to efficiently parallelize this process. In other words, coordinate ascent finds more accurate parameter choices at the cost of large computational time. Instead, we have developed a method, which can be reapplied to any domain that has the same parameter behavior we observed, to find an advising set that is as diverse as the training examples used.

One drawback of using coordinate ascent as a way to find advisor sets is that the individual parameter choices are likely to be overfit to the training examples. We have shown that even with this potential issue, we are able to greatly improve the quality of the transcripts produced according to the AUC measure. One extension to this method that could possibly improve the generalizability would be to perform coordinate ascent simultaneously on more than one sample; we have not explored the parameter landscape in this case but, if the samples are similar enough, it could also have one or a small number of local maxima.

The method we have described automates the task of parameter selection for `Scallop` (and other assemblers) and greatly improves the quality of produced transcriptomes according to the area under the curve measure, which compares the output to the reference transcriptome database. But, in the context of reference-based transcript assembly any metric that only measures accuracy with respect to some reference transcriptome is fundamentally flawed. The unknown transcripts are likely to be the most interesting, and this measure of accuracy penalizes novelty by definition. For instance, an assembler that has the reference transcriptome embedded in the source code and throws away any novel transcript would have very high precision with regard to the reference, but is clearly degrading its result to do so.

In contrast, the metrics used to assess quality for *de novo* transcriptome assembly do not penalize novelty [Li et al., 2014, Bushmanova et al., 2016]. One of the most accurate tools to perform this kind of assessment is `TransRate` [Smith-Unna et al., 2016] which measures the quality of a de novo assembly by mapping the provided sequencing reads to the produced transcriptome and calculating a score that is the sum of four quality metrics: nucleotide differences of the mapped reads, base coverage agreement between positions on a transcript, read-pair mapping agreement, and proportion of bases with no read support. All four of these metrics are still valid when assessing a reference-based transcriptome assembly, but using reference-free assessment tools for this task suffers from the opposite problem: it completely ignores the large swath of knowledge contained in the reference transcriptome.

We emphasize that this is an area of research that is begging to be explored. While we think measuring the accuracy of a transcript assembly by only comparing it to a reference is flawed, it is the current standard. The framework we have described stands on its own as a method to improve the quality of the results produced by `Scallop`, and can be adapted as new, better methods are developed.

## Acknowledgements

The authors would like to thank the members of the Kingsford Group for their helpful comments through out this project, in particular Mingfu Shao, Kwanho Kim, Jonathan King, Heewook Lee, and Guillaume Marçais.

## Funding

This research is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4554 to C.K., by the US National Science Foundation (CCF-1256087 and CCF-1319998) and by the US National Institutes of Health (R01HG007104 and R01GM122935). This work was partially funded by The Shurl and Kay Curci Foundation.

## References

Dan DeBlasio, Travis Wheeler, and John Kececioglu. Estimating the accuracy of multiple alignments and its use in parameter advising. In *Proceedings 16th International Conference Computational Molecular Biology (RECOMB'12)*, volume 7262 of *LNBI*, pages 45–59. Springer-Verlag, 2012.

John Kececioglu and Dan DeBlasio. Accuracy estimation and parameter advising for protein multiple sequence alignment. *J. of Comp. Bio.*, 20(4): 259–279, April 2013.

Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. `SATzilla`: portfolio-based algorithm selection for SAT. *J. Artif. Intel. Res.*, 32: 565–606, 2008.

Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. `ParamILS`: an automatic algorithm configuration framework. *J. Artif. Intel. Res.*, 36(1):267–306, September 2009.

Mingfu Shao and Carl Kingsford. Accurate assembly of transcripts through phase-preserving graph decomposition. *Nature Biotechnology*, 35:1167–1169, 2017.

C. Trapnell, B.A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M.J. Van Baren, S.L. Salzberg, B.J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Natue Biotechnology*, 28(5):511–515, 2010.

Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. `StringTie` enables improved reconstruction of a transcriptome from RNA-seq reads. *Nature Biotechnology*, 33:290–295, February 2015.

Juntao Liu, Ting Yu, Tao Jiang, and Guojun Li. `TransComb`: genome-guided transcriptome assembly via combing junctions in splicing graphs. *Genome Biology*, 17(1):213, October 2016.

Daehwan Kim, Ben Langmead, and Steven L Salzberg. `HISAT`: a fast spliced aligner with low memory requirements. *Nature Methods*, 12:357–360, March 2015.

Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R. Gingeras. `STAR`: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L. Salzberg. `TopHat2`: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, 14(4):R36, April 2013.

K.F. Au, H. Jiang, L. Lin, Y. Xing, and W.H. Wong. Detection of splice junctions from paired-end RNA-seq data by `splicemap`. *Nucleic Acids Research*, 38 (14):4570–4578, 2010.

Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. `Salmon` provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 14(4):417–419, 2017.

Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with `deseq2`. *Genome Biology*, 15(12):550, 2014.

Augustin Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes Rendus Hebdomadaire des Séances de l'Academie des Sciences*, 25(536):536–538, 1847.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, May 1983.

Willard I. Zangwill. *Nonlinear Programming: A Unified Approach.* Prentice-Hall International, Englewood Cliffs, N.J, 1969.

The ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489:57–74, August 2012.

Rasko Leinonen, Hideaki Sugawara, and Shumway, M. on behalf of the International Nucleotide Sequence Database Collaboration. The sequence read archive. *Nucleic Acids Research*, 39:D19–D21, November 2010.

Dan DeBlasio and John Kececioglu. *Parameter advising for multiple sequence alignment.* Springer International Publishing, Cham, Switzerland, 2017a. ISBN 978-3-319-64917-7.

Dan DeBlasio and John Kececioglu. Learning parameter-advising sets for multiple sequence alignment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(5):1028–1041, 2017b.

Bo Li, Nathanael Fillmore, Yongsheng Bai, Mike Collins, James A. Thomson, Ron Stewart, and Colin N. Dewey. Evaluation of de novo transcriptome assemblies from RNA-Seq data. *Genome Biology*, 15(12):553, December 2014.

Elena Bushmanova, Dmitry Antipov, Alla Lapidus, Vladimir Suvorov, and Andrey D Prjibelski. `rnaQUAST`: a quality assessment tool for de novo transcriptome assemblies. *Bioinformatics*, 32(14): 2210–2212, 2016.

Richard Smith-Unna, Chris Boursnell, Rob Patro, Julian M Hibberd, and Steven Kelly. `TransRate`: reference-free quality assessment of de novo transcriptome assemblies. *Genome Research*, 26(8): 1134–1144, August 2016.