Framework and Frontend for Operon and Transcriptional Unit prediction

Dissertation

zur Erlangung des Doktorgrades

der Naturwissenschaften

vorgelegt beim Fachbereich Biowissenschaften (FB15)

der Johann Wolfgang-Goethe-Universität

in Frankfurt am Main

von

Jannik Berz

aus Frankfurt am Main

Frankfurt am Main 2020

(D 30)

Vom Fachbereich Biowissenschaften (FB15)

der Johann Wolfgang-Goethe-Universität als Dissertation angenommen.

Dekan: Prof. Dr. Sven Klimpel

Gutachter: Prof. Dr. Enrico Schleiff, Dr. Kathi Zarnack

Datum der Disputation: 11.11.2020

TABLE OF CONTENTS

TABLE OF CONTENTS I
INDEX OF FIGURES III
INDEX OF TABLES V
INDEX OF SUPPLEMENTAL MATERIAL VI
Index of supplemental figures VI
Index of supplemental tables VI
ABBREVEATIONS AND UNITS VIII
ZUSAMMENFASSUNG IX
ABSTRACT XIV
1. INTRODUCTION 1
1.1. The transcriptional regulation program in prokaryotes 1
1.2. Current methods for identification and prediction of operons 4
1.3. Biological applications for machine learning approaches 6
1.4. Putative influence of operon structures to improve biotechnical applications for prokaryotes 10
1.5. Objectives of this study 11
2. MATERIALS & METHODS 13
2.1. Database sources 13
2.2. Tools and libraries 14
2.3. Organism sources and data samples 22
2.4. Hardware 24
2.5. Statistical measures and implemented calculations 24
2.6. Implementation and benchmark procedures 25
3. RESULTS 33
3.1. Development of operon prediction pipeline prototype based on expression and general genome data 33
3.2. Mapping and counting of RNA-Seq data module of OpPipe 36
3.3. Filter creation based on expression data for operon detection with OpPipe 40
3.4. Intergenic distance is not universal usable for all prokaryotic genomes 45
3.5. Combination of operon filters into plain prediction model of OpPipe 49
3.6. High confidence of OpPipe plain predictor on labeled datasets 52
3.7. Operon prediction via *in-silico* methods leads to different results than experimental TSS data 59
3.8. Machine learning approaches on filter data for labeled literature operons lead to enhanced operon prediction confidence 64
4. DISCUSSION 74
4.1. Intergenic distance is the major criteria for operon prediction and strongly dependent on the derived species 74
4.2. *In-silico* methods demonstrate improved ability of operon identification compared to genome wide TSS identification 75

4.3. Increased ability of operon prediction through generalized filter models based on expression and intergenic distance....77
4.4. Identification of alternative TUs through operon prediction filters....82
5. FUTURE PERSPECTIVES....86
5.1. Validation of predicted operons via OpPipe....86
5.2. Improvements and adaptions of OpPipe....86
6. REFERENCES....89
7. SUPPLEMENTS....106
DANKSAGUNG....134

INDEX OF FIGURES

Figure 1: Schematic operon and TU structure (adapted from [4] [20] [21]). 2

Figure 2: Detection of operons and TUs in Escherichia coli based on read coverage. 4

Figure 3: Classification models support vector machine (A), random forest (B), perceptron (C) and neural network (D). 9

Figure 4: GUIs of OpPipe. 34

Figure 5: Visual output of OpPipe. 35

Figure 6: Benchmark of Aligner. 37

Figure 7: Distance of proper read pairs and non-proper read pairs in Anabaena sp. PCC 7120.. 38

Figure 8: Counted reads of HTSeq-count and OpPipe-count. 39

Figure 9: Implementation of EP filter. 40

Figure 10: Implementation of GG filter. 43

Figure 11: Average distance of adjacent genes of the Watson strand within different genomes.. 45

Figure 12: Schematic structure of the plain predictor model and weightings applied to the filter inputs. 51

Figure 13: Comparison of operons predicted by OpPipe with collected ones from RegulonDB [147] in *Escherichia* coli K-12. 52

Figure 14: Venn diagram of the predicted operon set of four predictors in Escherichia coli K-12. 53

Figure 15: Comparison of operons predicted by three different predictors in Anabaena sp. PCC 7120. 54

Figure 16: Venn diagram of the predicted operon set of four predictors in Anabaena sp. PCC 7120. 55

Figure 17: Representation of the fraC [165] (A), pec [161] (B) and nir operon [159] (C) in Anabaena sp. PCC 7120 and their prediction by different prediction tools. 58

Figure 18: Distribution of transcription start site within the genome of Anabaena sp. PCC 7120. 59

Figure 19: Distribution of TSS (%) divided into number of reads per TSS in Anabaena sp. PCC 7120. 60

Figure 20: TSS distribution based on the operon sets of different predictors from Anabaena sp. PCC 7120. 61

Figure 21: TSS distribution of exclusively found operons in Anabaena sp. PCC 7120. 62

Figure 22: TSS distribution of the literature operon set of Anabaena sp. PCC 7120. 63

Figure 23: Feature importance of RF classifier on filter data. 65

Figure 24: Composition and grid search result for neural network.. 68

Figure 25: Training performance of neural net for 50 epochs 68

Figure 26: Comparison of different operon prediction models based on *Anabaena* sp. PCC 7120. 70

Figure 27: Venn diagram of the predicted operon set of three predictors and OpPipe conserved set in *Anabaena* sp. PCC 7120. 71

Figure 28: TSS for the fraC operon [157]. 76

Figure 29: Visual operon prediction of the *rbcLXS* [166] operon of *Anabaena* sp. PCC 7120 by OpPipe (under control, -Nit and -Fe conditions). 81

Figure 30: Visual operon prediction for a section of the PKS gene cluster [178] of Anabaena sp. PCC 7120 by OpPipe (under control, -Nit and -Fe conditions) and ProOpDB [69], DOOR [65], Rockhopper [36] (control conditions). 84

Figure 31: Schematic creation of the RC filter. 88

INDEX OF TABLES

Table 1: Used parameters for BBMap. 15
Table 2: Used parameters for Bowtie2. 15
Table 3: Used parameters for HTSeq. 17
Table 4: Used parameters for Mason. 19
Table 5: Used parameters for NGM. 20
Table 6: Statistical measures for the evaluation of the different filters and prediction models 24
Table 7: Implemented formulas within the prediction framework. 25
Table 8: Comparison of SGS and TGS data for *Escherichia coli*. 43
Table 9: DGD filter creation for five different species. 47
Table 10: Different statistical measures for the DGD filter. 48
Table 11: Prediction of literature operons from *Anabaena* sp. PCC 7120 by OpPipe, ProOpDB, DOOR and Rockhopper.. 57
Table 12: Performance measure of the training of the random forest classifier for the filter data. 65
Table 13: Performance measure of the training of the support vector machine classifier for the filter data. 67
Table 14: Performance measure of the training of the neural net classifier (filter data). 69
Table 15: Average length of predicted operons from *Anabaena* sp. PCC 7120. 72
Table 16: Number of annotated genes (%) of the *Anabaena* sp. PCC 7120 genome that have been classified into operons.. 73
Table 17: Average length of predicted operons from *Escherichia coli.*. 79
Table 18: Number of annotated genes (%) of the Escherichia coli genome that have been classified into operons. 80
Table 19: *Pec* operon from *Anabaena* sp. PCC 7120 under different EP stringencies and different stress conditions. 82

INDEX OF SUPPLEMENTAL MATERIAL

Index of supplemental figures

Figure S1: Schematic overview of OpPipe packages.....115

Figure S2: GUI of the visualization tool of OpPipe. 118

Figure S3: Decay of the *nir* operon [159] (A) and *pec* operon [161] (B) from *Anabaena* sp. PCC 7120 under different stringencies of the EP filter under control conditions.....120

Figure S4: Evaluation of GG- and DGD filter basing on adjacent gene pairs of *Anabaena* sp. PCC 7120.....121

Figure S5: Average distance of adjacent genes of Watson and Crick strand within different genomes.. 124

Figure S6: AUC for random forest (RF), support vector machine (SVM) and neural net (NN) for 5-fold cross validation on filter data. 128

Figure S7: Visual operon prediction for the PKS gene cluster [178] of *Anabaena* sp. PCC 7120 by OpPipe (under control, -Nit and -Fe conditions) DOOR [65], ProOpDB [69], Rockhopper [36] (under control conditions). 133

Index of supplemental tables

Table S1: Different operon predictors. Table adapted from Brouwer (2008, [27]) and Tjaden (2019, [26]). 106

Table S2: Base composition for each 1L of dropout medium. 110

Table S3: Composition of solutions. 110

Table S4: Collected datasets for different species.. 110

Table S5: Operons identified within the literature for six species. 113

Table S6: Occurrence of SAM flags within the SAM file of *Anabaena* sp. PCC 7120.....116

Table S7: Cut-off calculation for EP filter. 117

Table S8: Literature set operons from *Anabaena* sp. PCC 7120 compositions under different stringencies of the EP filter.....119

Figure S4: Evaluation of GG- and DGD filter basing on adjacent gene pairs of *Anabaena* sp. PCC 7120.....121

Table S9: Overlaps for GG filter and buckets for RC filter distribution for *pec* operon. 122

Table S10: Overlaps for GG filter of *nir*, *fraC* and *pec* operon under different conditions. ...123

Table S11: Different statistical measures for the DGD filter. 125

Table S12: Schematically data for gene pairs basing on EP, GG and DGD filter.. 126

Table S13: Prediction of literature operons from *Escherichia coli* by OpPipe, ProOpDB, DOOR and Rockhopper. 127

Table S14: Scoring of operon prediction of different predictor basing on *Anabaena* sp. PCC 7120 (AB) and *Escherichia coli* (EC). 128

Table S15: Grid Search result for support vector machine classifier (filter data). 129

Table S16: Full hit and not found candidates of predicted operons for different predictors..129

Table S17: Conserved operons that have been predicted by the plain predictor, RF, SVM and NN. 130

ABBREVEATIONS AND UNITS

all	*Anabaena* long left	**NP**	Non operon gene pair
alr	*Anabaena* long right	**nt**	Nucleotides
asl	*Anabaena* short left	**OP**	Operon gene pair
asr	*Anabaena* short right	**RC**	Read coverage
AUC	Area Under the Curve	**RF**	Random forest
Co	Control	**RIN**	RNA integrity number
DGD	Dynamic intergenic distance	**RNA**	Ribonucleic acid
DNA	Deoxyribonucleic acid	**RNA-Seq**	RNA sequencing
e.g.	For example (exempli gratia)	**ROC**	Receiver operator characteristic
EP	Expression pattern	**RT-PCR**	Reverse transcription polymerase chain reaction
Fe	Iron	**SAM**	Sequence Alignment/Map
FN	False negative	**SGD**	Stochastic gradient descent
FP	False positive	**SGS**	Second generation sequencing
GFF	Generic feature format	**SRA**	Sequence read archive
GG	Gene Graph	**ST**	Stringency
GUI	Graphical user interface	**SVM**	Support vector machine
IDE	Integrated development environment	**TF**	Transcription factor
JVM	Java virtual machine	**TFBS**	factor binding sites
ML	Machine learning	**TGS**	Third generation sequencing
mRNA	Messenger RNA	**TN**	True negative
MVP	Minimum viable product	**TP**	True positive
NGS	Next generation sequencing	**TSS**	Transcription start site
Nit	Nitrate	**TU**	Transcription unit
NN	Neural network	**TUC**	Transcription unit cluster

ZUSAMMENFASSUNG

Die Expression von Genen geschieht generell durch den Prozess der Transkription und ermöglicht einem Organismus unter anderem die Reaktion auf eine umweltbedingte Stresssituation (Leake, 2018) (Jacob and Monod, 1961) (Burkhardt *et al.,* 2017). Die Regulation der Transkription erfolgt bei Prokaryoten und Eukaryoten jedoch unterschiedlich (Fan *et al.,* 2017) (Moore and Proudfoot, 2009) (Kornblihtt *et al.,* 2013). Hierbei ist einer der Hauptunterschiede zwischen Prokaryoten und Eukaryoten, dass die Organisation von prokaryotischer DNA in sogenannte transkriptionelle Einheiten („transcription unit", TU) geschieht (Chen *et al.,* 2017) (Cho *et al*., 2013) (Mao *et al.,* 2015). Eine besondere Art von TUs stellen sogenannte Operons dar. Operons sind Einheiten, die aus mehr als einem Gen bestehen und je nach Bedingung an- und abgeschaltet werden können. Operons können darüber hinaus als dynamische Einheiten verstanden werden, sodass verschiedene Operons durchaus überlappen können (Mao *et al.,* 2015). Weiterhin sind innerhalb von verschiedenen Operons mehrere Transkriptionsstartseiten (TSS) bekannt; somit können mehrere TUs, je nach Situation, ausgebildet werden (Mao *et al.,* 2015) (Chen *et al.,* 2017).

Neben experimentellen Ansätzen wie „northern blotting" (Vinnemeier *et al.,* 1998) und „reverse transcription PCR" (RT-PCR) (Gupta, 1999) können Operons auch mithilfe von *in-silico* Methoden vorhergesagt werden (Brouwer *et al.,* 2008) (Moreno-Hagelsieb, 2015). Häufig verwendete genomische Eigenschaften sind hierfür zum Beispiel die intergenische Distanz, funktionale Relation und evolutionäre Stabilität (Brouwer *et al.,* 2008) (Moreno-Hagelsieb, 2015). Beispielsweise tendieren Gene innerhalb eines Operons dazu, eine geringere intergenische Distanz untereinander aufzuweisen als Gene, die sich nicht im gleichen Operon befinden. Weiterhin neigen Gene eines Operons dazu, ähnliche funktionelle Aufgaben zu übernehmen oder sie sind an ähnlichen biologischen Prozessen beteiligt. Darüber hinaus sind Operons evolutionär gesehen stabil und als konservierte Einheiten in vielen Organismen identifiziert worden. Auch die Verwendung von spezifischen Codonen kann eine Rolle spielen, da Gene innerhalb eines Operons häufig eine Ähnlichkeit bei den gewählten Codonen aufweisen. Neben diesen genomischen Eigenschaften wird allerdings mittlerweile auch die genomweite Expression von verschiedenen Genen zur Vorhersage genutzt. Hierbei ist die Hypothese, dass Gene innerhalb einer TU eine ähnliche Expression aufweisen sollten. Im Gegensatz zu den genomischen Eigenschaften, welche vorrangig an Modellorganismen wie *Escherichia coli* oder *Bacillus subtilis* untersucht wurden, sind solche Eigenschaften auch problemlos für Nicht-Modellorganismen wie Cyanobakterien (z.B*. Anabaena* sp. PCC 7120) auffindbar. In diesem Zusammenhang, weiterhin problematisch bei den verschiedenen Vorhersageprogrammen ist hierbei die Spezifität für einen bestimmten Organismus und die Nichtanwendbarkeit auf andere Organismen. Um nun verschiedene Eigenschaften zur

Vorhersage von Operons an Nicht-Modellorganismen zu analysieren, sollten neue Vorhersagemodelle und Filter implementiert und getestet werden.

Insgesamt lässt sich die Arbeit in vier unterschiedliche Abschnitte unterteilen. Der erste Teil beschreibt zunächst den Aufbau der Pipeline und zeigt dabei die Nutzeroberfläche. Der zweite Abschnitt befasst sich mit der Erstellung einer Mapping Pipeline für RNA-Sequencing Daten. Der dritte Teil beschäftigt sich mit der Verarbeitung dieser Daten sowie genomischer Daten, hin zu generalisierten Modellen für die Identifizierung von operonischen Gen-Paaren. Diese Modelle wurden dann in einem ersten, vereinfachten Vorhersagemodell kombiniert. Im vierten und letzten Teil wurde dieses Vorhersagemodell zunächst mit anderen Vorhersagetools und Literaturbeispielen verglichen. Anschließend wurde untersucht, inwiefern die Identifizierung von operonischen und nicht-operonischen Gen-Paaren mithilfe von drei verschiedenen „machine learning“ Modellen die Vorhersage verändert.

Der Prototyp der in dieser Arbeit entwickelten Pipeline „OpPipe“ (Operon Pipeline) ist in der Programmiersprache Java entwickelt worden und inkludiert Vorhersagemodelle, die in der Programmiersprache Python geschrieben sind. Für die Vorhersagemodelle wurden neben einem stark vereinfachten Vorhersagemodell mit vordefinierten Wertungsskalen auch maschinelle Lernansätze wie Random Forest (RF), Support Vector Machine (SVM) und Neuronale Netze (NN) verwendet. Als Eingabe für alle Modelle wurden für jedes mögliche Gen-Paar auf dem gleichen DNA-Strang genomische und Expressionseigenschaften übergeben, um die Gene in Operon- und nicht-Operon Gene zu unterteilen. Da diese Modelle auch über eine Nutzeroberfläche bedienbar sind und ein automatisierter Pipeline-Durchlauf gewährleistet ist, wurde mithilfe der Bibliotheken SpringBoot und Thymeleaf eine Nutzeroberfläche implementiert.

Damit die Pipeline auch Expressionswerte für die Vorhersage verwenden kann, wurde als erster Schritt das korrekte Mapping und die Zuordnung der Reads auf Gene implementiert. Hierbei wurden existierende Aligner wie BBMap, Bowtie2, NextGenMap (NGM) und Rockhopper bezüglich ihrer Laufzeit und Genauigkeit beim Mappen getestet. Darüber hinaus wurde auch bei der Prozedur der Zuordnung von Reads zu Genen ein Vergleich zu einem bestehenden Programm (HTSeq) vorgenommen. Für den Anwendungsfall der Bestimmung von Expressionsprofilen für prokaryotische Genome konnte durch die eigne Mapping-Prozedur eine Verbesserung dieser Zuordnung im Vergleich zu HTSeq festgestellt werden. Der Mapping-Teil der Pipeline ermittelt aus den Expressionsdaten einerseits ein Expressionsprofil, in welchem für jede Position im Genom die Anzahl gemappter Reads gespeichert wird. Hierbei wird davon ausgegangen, dass die unterschiedlichen genomischen Positionen unterschiedlich häufig getroffen werden. Dabei sollte es unter einer bestimmten Umweltbedingung

vergleichbar wenige Positionen geben, die entweder von sehr wenigen reads (herunterregulierte Bereiche) oder von sehr vielen reads (stark hochregulierte Bereiche) getroffen wurden. Die Mehrheit der genomischen Positionen sollte sich um einen gewissen Mittelwert einpendeln (durchgängige Expression). Um hierbei operonische von nicht-operonischen Gen-Paaren zu separieren, wird das zugrundeliegende Expressionsprofil unter verschieden starken Werten beobachtet. Jeder Wert suggeriert dabei, wie viele genomische Positionen von mindestens x reads abgedeckt wurden. Somit ergeben sich verschiedene „Expression patterns“ (EP) je nach Stärke der zugrundeliegenden Werte. Sind zwei Gene unter einer erhöhten Stärke als operonisches Gen-Paar klassifiziert, ist die Wahrscheinlichkeit erhöht, dass sie tatsächlich als solches zu zählen sind. Darüber hinaus speichert der Mapping-Teil der Pipeline auch solche Gen-Paare, die von demselben read getroffen wurden und somit durch diesen verbunden sind. Diese Information wird für einen sogenannten „Gene Graph“ (GG) Filter verwendet. Hierbei wird auch davon ausgegangen, dass operonische Gen-Paare co-exprimiert sind, jedoch erfolgt die Identifizierung über die reads (bzw. Templates) an sich. Existieren reads (bzw. Templates), welche auf mehr als ein Gen mappen, wird dies als Signal aufgefasst, dass diese Gene gleichzeitig abgelesen wurden. Dies ist als starkes Signal zu betrachten, dass zwei Gene als Operon abgelesen wurden. Hierbei wird eine Graph-Struktur aufgebaut, wobei die Kanten, die die unterschiedlichen Gene verbinden, gewichtet sind mit der Anzahl der reads, die beide Gene abgedeckt haben.

Neben diesen Expressionsdaten wurde die Eigenschaft der intergenischen Distanz zur Betrachtung hinzugenommen. Allerdings nicht basierend auf einer bestimmten Spezies, sondern dynamisch gehalten als sogenannte „Dynamic Intergenic Distance“ (DGD). Für die intergenische Distanz konnte bereits gezeigt werden, dass sie zwischen verschiedenen Organismen sehr unterschiedlich sein kann (Price *et al*., 2005) (Rogozin *et al*., 2002). Hierbei wurde ersichtlich, dass das Genom von *Escherichia coli* enger gepackt ist als beispielsweise das Genom von *Bacillus subtilis* oder *Anabaena* sp. PCC 7120. Um hierfür ein allgemeingültiges Modell zu definieren, werden für den DGD zunächst alle intergenischen Distanzen paarweise berechnet und anschließend wird die Distanz gesucht, die von 45-50% der Gen-Paare eingehalten werden.

Basierend auf den drei definierten Operon-Eigenschaften GG, EP und DGD wurden in dieser Studie vier verschiedene Vorhersagemodelle erstellt, wobei eine binäre Kodierung nur zwischen Operon (1) und Nicht-Operon Gen-Paar (0) vorgenommen wurde. Zum Vergleich der Filter und des vereinfachten Vorhersagemodells wurden diese mit verschiedenen Operon Datenbank- und Vorhersagetools sowie einem identifizierten Literaturset verglichen.

Zunächst wurden Operons aus den Datenbanken DOOR und RegulonDB sowie dem Programm Rockhopper basierend auf dem Modellorganismus *Escherichia coli* gegen den eigenen Klassifizierer getestet. Hierbei konnte beobachtet werden, dass der erstellte vereinfachte Klassifizierer eine hohe Überschneidung mit dem Operon Set der Regulon Datenbank hat, die experimentell bestätigte Operons beinhaltet. Weiterhin wies der vereinfachte Klassifizierer eine höhere Überschneidung mit der Regulon-Datenbank auf als die anderen Vorhersageprogramme.

Im Vergleich zu dem Modellorganismus konnten bei dem Cyanobakterium *Anabaena* sp. PCC 7120 sehr starke Unterschiede festgestellt werden. Hierbei wies der vereinfachte Klassifizierer ~38% mehr Operons auf als vergleichbare andere Tools und Datenbanken. Während die verschiedenen Vorhersageprogramme auch untereinander eine niedrige Überschneidung aufweisen, konnte der vereinfachte Klassifizierer ~99% aller Operons der anderen Klassifizierer entweder teilweise oder in der gleichen Art und Weise vorhersagen.

Um einen ersten Eindruck über die Aussagekraft der neuen Vorhersage zu gewinnen, wurden aus der Literatur bestätigte Operons aus beiden Organismen zum Vergleich herangezogen. Auch hierbei konnte gezeigt werden, dass der vereinfachte Klassifizierer mehr als 60% der Operons des Literatursets aus *Escherichia coli* und *Anabaena* sp. PCC 7120 korrekt identifizieren konnte, währen andere Klassifizierungsprogramme ~30-50% korrekt vorhersagen konnten. Im Hinblick auf die Genauigkeit der genomweiten Vorhersage von Operons wurden die Vorhersagen des vereinfachten Modells, der anderen Klassifizierungsprogramme sowie die Operons des Literatursets im Anschluss mit experimentellen Transkriptionsstartseiten (TSS) aus der Literatur verglichen (Mitschke *et al*., 2011). Untereinander zeigten die drei Modelle eine hohe Überschneidung von ~90% und auch verglichen mit dem ursprünglichen, vereinfachten Vorhersagemodell konnten ungefähr 50% der Operons in derselben Zusammensetzung von Genen vorhergesagt werden. Hierbei wurde eine generelle Überschneidung von 567 Operons der jeweils gleichen Struktur, verglichen zwischen den verschiedenen Vorhersagen, gefunden. Zu den Vorhersagetools (bzw. Datenbanken) DOOR, ProOpDB und Rockhopper konnte durch die maschinellen Klassifizierer jedoch kein höherer Überlapp erreicht werden als mit dem vereinfachten Klassifizierer.

Insgesamt zeigte sich, dass die entwickelten Klassifizierer in *Escherichia coli* eine Operon-Länge im Durschnitt von ~3,4 Genen aufwiesen. Somit korrelieren sie im Vergleich zu den anderen Vorhersageprogrammen besser mit dem definierten Literaturset sowie dem Operon Set der Regulon-Datenbank, die eine durchschnittliche Operon-Länge von 3,2 bzw. 4,4 Genen aufwiesen. Für *Anabaena* sp. PCC 7120 wiesen die erstellten Modelle eine durchschnittliche Operon-Länge von ~3,2 Genen auf, während die bekannten Datenbanken und Programme

eine durchschnittliche Länge von ~2,4 aufwiesen. Somit liegt die durchschnittliche Operon-Länge der erstellten Klassifizierer näher an der von der Literatur bestätigten Operon-Länge (4,5 Gene) als die anderen Ansätze. Weiterhin konnte gezeigt werden, dass die erstellten Klassifizierer besser zu der Annahme passen, dass 50-60% aller Gene in Prokaryoten in Operons organisiert sind (Brouwer *et al.,* 2008) (Moreno-Hagelsieb, 2015). In *Escherichia coli* wurden von den eigenen Klassifizierern und der RegulonDB ~60% der Gene in Operons vorhergesagt, während die anderen Datenbanken und Tools ~50% der Gene in Operons klassifizierten. Für *Anabaena* sp. PCC 7120 war für die eigenen Klassifizierer die Anzahl der in Operons klassifizierten Gene mit ~74% zwar leicht erhöht im Vergleich zu den angenommen 50-60%. Jedoch erscheint diese Anzahl plausibler als die nur 27-35% aller Gene, die von den anderen Klassifizierern in Operons vorhergesagt wurden.

Insgesamt konnte durch die verschiedenen Modelle gezeigt werden, dass die wichtigste Eigenschaft für die Operon-Vorhersage das Kriterium der intergenischen Distanz ist. Dieses Kriterium muss jedoch in einer dynamischen Art und Weise angepasst werden, da die Vorhersage sonst sehr spezifisch für einen bestimmten Organismus ist. Weiterhin konnte gezeigt werden, dass eine Identifizierung von TUs durch TSS zu einem unvollständigen Set führen kann und *in-silico* Methoden hierfür eine sehr nützliche Alternative und Erweiterung darstellen. Durch die Kombination der unterschiedlichen Filter, Vorhersagemodelle und einer Visualisierung konnte weiterhin die Nützlichkeit der Pipeline hinsichtlich der Expression von verschiedenen Genclustern und Operons, auch unter verschiedenen Stressbedingungen, gezeigt werden. Die entwickelte Pipeline soll hierfür als Startpunkt dienen und kontinuierlich weiterentwickelt werden. Hierfür sollen beispielsweise die vorhergesagten Operons anschließend mit Informationen über die Funktion der einzelnen Gene sowie Informationen über die evolutionäre Konservierung, sofern beide vorhanden, angereichert werden, um dem Anwender einen besseren Überblick über die vorhergesagten Operons zu verschaffen. Weiterhin können die entwickelten Filter auch verwendet werden, um die Vorhersage von alternativen TUs (auch innerhalb von Operons) umzusetzen und in die Pipeline zu integrieren.

ABSTRACT

Despite parallels in the gene expression between prokaryotes and eukaryotes, several mechanisms differ, especially on the regulatory level. One peculiarity of the prokaryotic system is that it regulates most of the expression via functional transcriptional units (TUs). A specific class of these TUs are the so-called operons, which play a central role during the transcription. Further, the TUs can be adapted via alternative transcription start sites (TSS) to react to external conditions like biotic or abiotic stress conditions or developmental changes. Such operons can be identified via experimental approaches for single operons or assumed for the whole genome. This can be achieved using techniques like next generation sequencing (NGS) to determine similar expression or via transcription start sites (TSS).

Beside the experimental approaches, *in-silico* methods for the prediction of TUs and operons have also been developed based on general genome features as well as the usage of expression data. Even if never stated clearly in the initial operon definition by Jacob and Monod (1961), different *in-silico* tools assume TUs to be non-overlapping single units. However, depending on the external conditions via multiple TSS, even more complex structures have been observed with multiple TUs arising even out of a single operon. This and the availability of large scale proteomic and transcriptomic data under different conditions prove that the assumption of non-overlapping operons held by numerous *in-silico* tools and databases is generally not true.

Overall, several general genomic features and combinations of them are so far used for the prediction of operons. Firstly, the intergenic distance between two genes is one of the most widely used parameters. It is based on the observation in model organisms like *Escherichia coli* where the intergenic distance between genes within an operon typically is less than 100 nucleotides. A second common feature is the functional relation of genes if they belong to an operon, assuming they are part of the same pathway. Further, a third commonly used feature is the conservation, as operons are thought to be evolutionarily conserved. Related to this, due to the availability of whole transcriptome expression data, the co-expression of genes is used as a feature for the prediction of operons. In this context, it is assumed that genes within an operon are transcribed as one unit and share the same expression level.

In comparison to already existing prediction tools and databases, the aim of this study is to generate features for operon prediction which are not based on specific organisms. Out of this, the goal is to develop new and more powerful features based on the NGS expression data instead of simply using the average expression of an annotated gene.
As a second step, these newly developed features are combined into an application by the usage of machine learning methods. These methods assume structures by learning through

the data rather than through explicit programming like a classical algorithm. However, the application of machine learning methods to a given species dataset often leads to a species or dataset bias as the model is only able to infer features of the training species but fails to infer these features on another species.

To address this issue, the pipeline called OpPipe was developed to rely on general genomic and expression data features which should be applicable to a wide range of species. To ensure that the application is not only specifically applicable to a sample species, OpPipe uses a species dynamic intergenic distance as well as expression features. Subsequently, OpPipe includes different classifier methods like a plain predictor model, but also further different machine learning algorithms such as random forest, support vector machine and neural net. The implemented models ensure the identification of operons in a non-overlapping context, but also aim to identify them as multiple alternating TUs. To ensure high applicability and user-friendliness, OpPipe offers a graphical user interface (GUI).

With the use of different *in-silico* predictors of operons, it could be demonstrated that they only have an overlap of 40% to experimental whole transcription start sites (TSS) identification. By this, *in-silico* predictors add additional information to identify operons and TUs. The benchmark of the developed *in-silico* models in this study for two species for 38 known operons from the literature showed a correct prediction for over 60%. This outperforms several available databases and tools, which only reach 30-50%. Further, from the whole genome prediction, OpPipe predicts 60-74% of genes to be present in operons. This fits to the assumptions that at least 50-60% of the genes from a given genome are organized into operons. The other prediction tools, however, are underestimating this assumption by only assigning 27-50% of the genes to operons.

These findings offer practical guidance to improve operon prediction in model and non-model species. Therefore, this study adds a valuable contribution to existing research on operon prediction, by adding species independent features to a genome wide prediction.

1. INTRODUCTION

1.1. The transcriptional regulation program in prokaryotes

The process of transcription, which is mediated by the RNA polymerase, transcribes genes from DNA into RNA and thereby expresses them [1] [2]. The RNA polymerase binds upstream of the gene(s) to be expressed on their referring promoters. Further, the expression of genes during transcription is regulated (activated or inhibited) by transcription factors (TF) [3]. TFs are binding to the DNA at transcription factor binding sites (TFBS), which are located upstream the expressed gene [3] [4]. Alternatively, TFs interact with other parts of the transcriptional machinery and thereby regulate the transcription [5]. However, the transcriptional regulation machinery differs highly in prokaryotes compared to the eukaryotic machinery [6] [7] [8]. While prokaryotes only exhibit one RNA polymerase in charge of gene expression [9], eukaryotes exhibit three RNA-Polymerase for the synthesis of ribosomal RNAs, pre-mRNAs and tRNAs [5] [10]. Further, the eukaryotic genome is structured in highly supercoiled chromatin structures (associated with histones in nucleosomes) [5]. This packaging of the DNA limits the accessibility and thereby influences the transcriptional regulation effect, as some genomic regions are tighter and are therefore less accessible to the transcriptional machinery than loosely coiled (and easily accessible) regions [11]. The expression of genes thereby requires the presence of DNA binding regulatory elements which increase (enhancer) or decrease (repressor) the transcriptional rate [12]. In addition, the deriving mRNA of eukaryotes is monocistronic [13] meaning that in eukaryotes single genes exhibit their own promotor and enhancer elements to turn the expression on or off [5] [12]. The transcription (nuclear) and translation (cytoplasm) is thereby separated in the eukaryotic system and RNAs undergo post-transcriptional adaptions (e.g. capping, polyadenylation and splicing) [1] [13] [14].

Contrasting to the eukaryotic supercoiled structures, prokaryote DNA chromosomes are organized circular [11]. They are often extended by smaller DNA elements like plasmids and, compared to eukaryotic genomes, relatively small [11]. Further, splicing is usually a rare event in prokaryotes and occurs typically in non-coding RNAs [15] [16]. Compared to eukaryotes, the translation and transcription is not locally separated and promoters within the prokaryotic DNA are more easily accessible [11]. Beside the regulation via TFs, the promoter recognition of promoters and the recruiting of the RNA polymerase holoenzyme to the promoter is coordinated by the aid of a single regulatory subunit known as so called sigma factors [2] [17]. These factors are thereby highly discriminatory, as each binds a distinct set of promoter sequences, and therefore plays an important role during the activation of genetic regions under different stress conditions and growth states [2] [18]. This is achieved as they are part of the

holoenzyme but distinct sigma factors compete for the binding to a common pool of RNA polymerases [2].

In contrast to eukaryotes, the prokaryotic genome is organized in so called transcription units (TU), meaning that beside monocistronic mRNA also polycistronic mRNA containing more than one gene gets transcribed [19] [20] [21]. The regulation of these units is facilitated by DNA segments (also called operators) which are acting as activation or repression binding sites for proteins [22]. Thereby for some units, the binding of the activator is necessary for transcription (positive regulation), while for other units the prevention of the repressor protein binding to its target is indispensable (negative regulation) [22]. As intergenic regions in front of genes from defined TUs tend to exhibit an increased number of conserved sequence motifs compared to genes within the TU [21], TUs are likely to each exhibit an own promoter. Continuing, TUs of at least two genes (which are adjacent) harboring a shared genetic regulation signal (e.g. promoter) are called operons (Figure 1A) [21] [23]. These functional units are sets of adjacent genes on the same strand that are getting transcribed together into a polycistronic messenger RNA (Figure 1A) [23] [24] [25] [26]. As it is assumed that 50-60% of all genes in a prokaryotic genome are part of an operon [27] [28], operons regulate gene expression of a prokaryotic system [29]. Therefore, the transcription of an operon is adapted to the cellular needs ensuring that operonic genes are transcriptionally co-regulated in respect to various conditions (e.g. environmental, development or stress conditions) [23] [30]. Despite the fact that operons are defined as transcriptional unit under control of one promoter [20] [21], even more complex transcriptional regulation structures have been reported with operons containing multiple promoters and transcription start sites (TSS) [20] [27]. This leads to the formation of different elements from a given operon (Figure 1B).

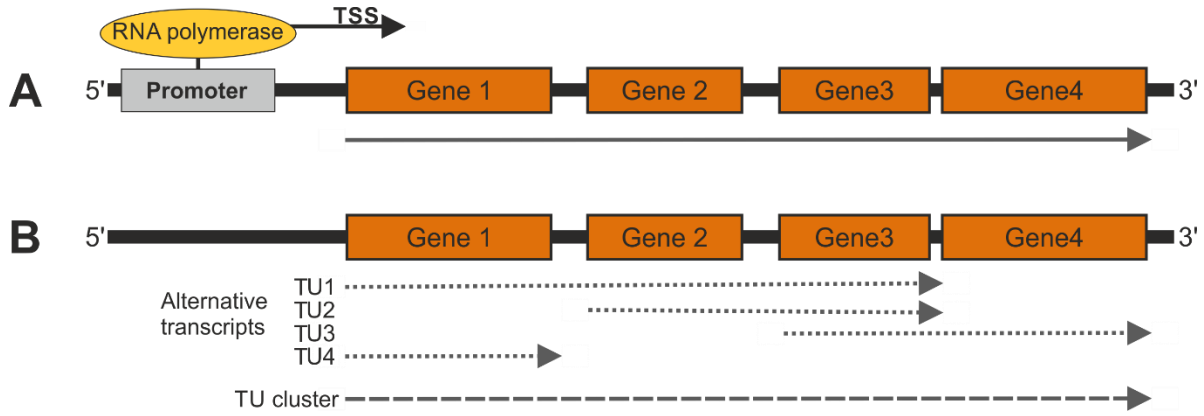


Figure 1: Schematic operon and TU structure (adapted from [4] [20] [21]). (A) Operon structure of bacterial genome including the promoter where the RNA polymerase binds and the TSS downstream the promoter leads to the expression of the adjacent genes 1-4. (B) Formation of different TUs from a given operon structure (TU1-4) and a formation of a TU cluster (TUC) by the overlapping genes of the different TUs.

In this context it has been demonstrated that different sub-sets of genes in an operon may be co-transcribed under different conditions, leading to smaller TUs emerging out of an operon depending on the condition [20] [21] [31]. Consequently, the transcriptional units of an operon do not always have to be unique and multiple transcriptional units (resulting in poly- and monocistronic mRNAs) may arise of one operon [32]. In this context, TUs emerging from a defined operon have shown to possibly overlap each other [33] [34], meaning they are exhibiting shared genes (e.g. Figure 1B). This means they are overlapping each other but are not necessarily a sub-part of each other.

Further, genes within an operon might not only be co-transcribed under different conditions into specific TUs but rather there exist multiple parallel operons and TUs that are overlapping [21]. This observation also leads to the formation of TU clusters (TUC, Figure 1B), where multiple TUs can be grouped into one bigger TUC by their overlapping genes [21]. A key driver for the observation of multiple parallel operons and TUs is the fact that even if operonic genes are co-transcribed and usually exhibit a similar expression profile [27] [35], it has been observed that their co-expression and thereby their expression profile can be different [24] [32]. As the multiple TSS inside an operon can be used in respect to their environmental and/or growth conditions [20], individually expressed units arise depending on the condition [31] [36] [37], but also different TUs arise in respect to the strength of the different promoters [33] [34] (Figure 2).

In the expression profile of the *rplKAJL-rpoBC* [38] operon of *Escherichia coli* K-12 (wildtype; MG1655), a different coverage of expression is revealed when comparing the sub-units rplK-L and rpoBA under different conditions like L-tryptophan supplementation, anaerobic conditions and control. Further, the coverage of the sub-units under anaerobic conditions is altered, as it shows an increased coverage compared to the coverage under normal growth conditions or in the presence of L-Tryptophan (Figure 1A, [38]). The density changes correlate with the presence of promoter and transcriptional attenuator regions [33] [34] [38] [39]. Alike, the spermidine operon (yacC, speED, Figure 1B, [40]) shows an approximately continuous expression profile under anaerobic conditions and in the presence of L-Tryptophan, while under normal conditions yacC appears not to be expressed within the same transcription unit.

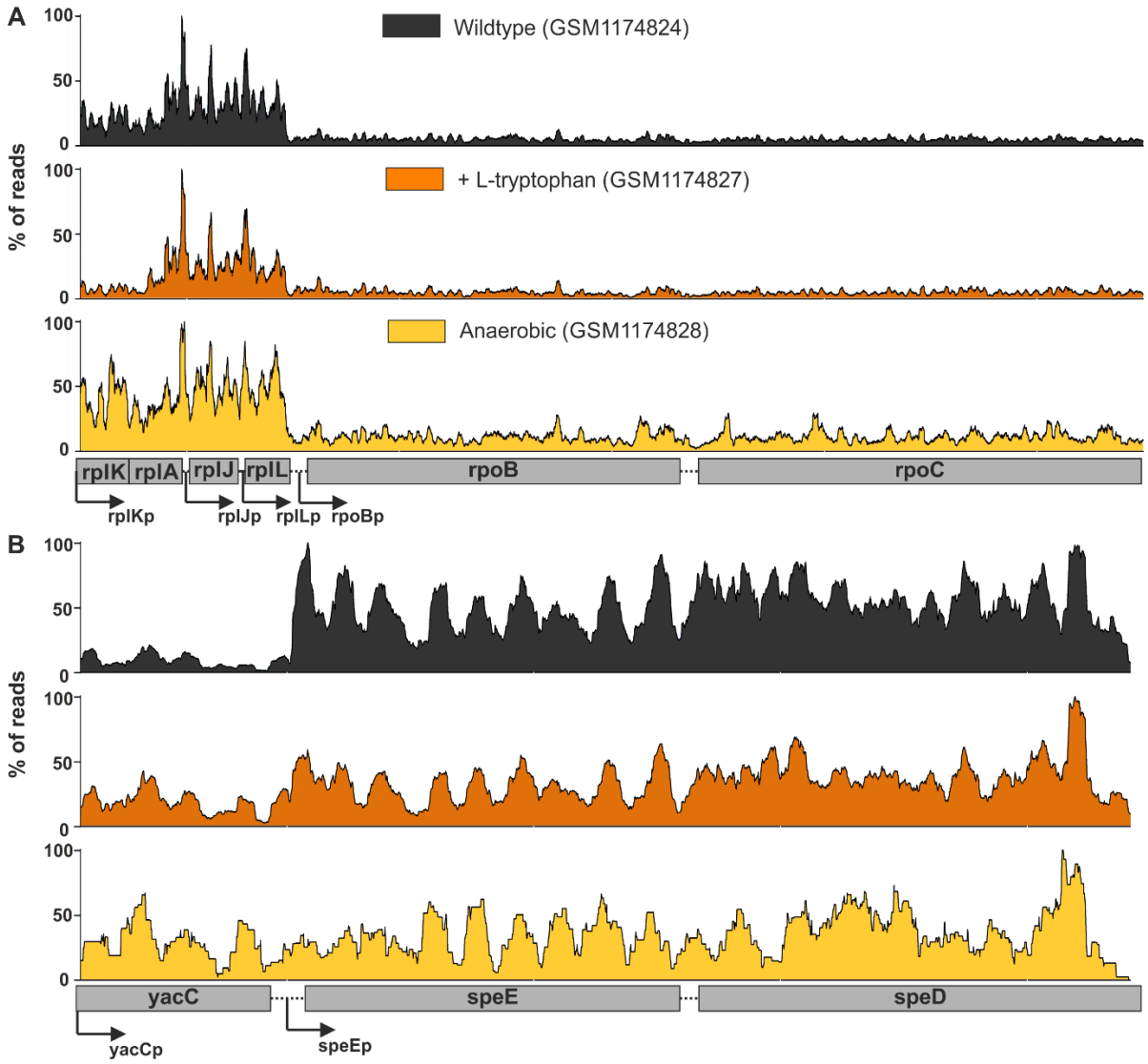


Figure 2: Detection of operons and TUs in *Escherichia coli* based on read coverage. Read coverage of the *rplKAJL-rpoBC* (A, [38]) and *yacC-speED* operon (B, [40]) based RNA-Seq experiments of *Escherichia coli* K-12 grown in normal media (GSM1174824, black), with L-tryptophan supplementation (GSM1174827, orange) and under anaerobic conditions (GSM1174828, yellow) are shown. Solid arrows indicate promoter regions ([33], [34], [38], [39], [40], [41], [42], [43], [44]).

In this case, too, the changes correlate with promotor identified *in-vitro* [40], empowering the assumption that expression data can be used for the prediction of transcriptional units operons [45] and enhances the prediction [36] [46], especially under different conditions. Accordingly, understanding the composition and regulation of operons is a crucial point for understanding the general adaption of species under different conditions.

1.2. Current methods for identification and prediction of operons

Operons can be experimentally identified either by *in-vitro* methods like northern blotting and reverse transcription-PCR (RT-PCR) or by primer extensions, which are subsequently validated with methods like southern and western blotting [47] [48] [49] [50]. For example, northern blotting and primer extension have successfully been used to identify *coxBAC* and *nir*

operons in *Anabaena* sp. PCC 7120 [51] [52] and thereby it could be demonstrated that they are essential for the fructose-dependent growth in the dark (*coxBAC* operon) and are expressed under the presence of nitrate and nitrite (*nir* operon). Further, southern blots and primer extension were successfully used to identify the *coxBAC-III* operon within *Anabaena* sp. PCC 7120 by comparing the resulting blots to *Anabaena variabilis* [53] while a combination of RT-PCR, northern and western blots led to the identification of the *fraC* operon and the included antisense element [54]. Finally, with different RT-PCR experiments, the cluster of the *hyp* genes were demonstrated to be transcribed into one single operon (six *hyp* genes and one ORF), as well as the extension by additionally seven genes (five before the *hyp* genes and two after the *hyp* genes) that are part in the transcription regulation of this operon [55].

However, *in-vitro* methods can be on the one hand time consuming [47] and it is on the other hand impossible to test all genes of a species for being part of an operon. Regarding this, the set of genes to be tested for being part of an operon should be limited previously. Therefore, it is also possible to apply experimental screening methods that were developed based on Next Generation Sequencing (NGS), allowing genome-wide identification verification of single operons by using transcription start sites (TSS) [20] [56] and gene-expression data [27] [57].
Beside experimental identification methods, *in-silico* methods for predicting operon structures were also developed over the last decades (Table S1). Several criteria and features associated with operon structures have been defined and widely used for the identification and prediction of operons such as (i) the intergenic distance between adjacent gene-pairs [27] [58], (ii) the functional relation of gene-pairs [27] [59], (iii) conservation of operons among species [27] [60], (iv) shared sequence based features (e.g. promoter regions [61] [62], transcription start sites [56], codon usage [59] [63] [64]) and (v) co-expression of gene-pairs [27] [46] [64]. One of the simplest methods for the differentiation between operonic gene pairs (OP) and non-operonic gene pairs (NP) is the difference in the intergenic distance which is thought to be smaller for OPs than for NPs, irrespective of different species [35]. It could be shown that a valid intergenic distance measure for *Bacillus subtilis* and *Escherichia coli* is in the range of -50nt to 250nt [65]. As second criterion, it can be assumed that genes of an operon are often functionally related [24]. In this context, their transcription results in products that functionally work together and proteins which are part of the same pathway [24] [65]. Further, from an evolutionary point of view, once evolved, operons in ancestors are likely to be conserved in different species and are thought to be stable over time [24]. In this respect, the term of a so called uber-operon was created, describing operons that can be found in numerous species [37] [66] [67]. As operons are thought to be activated or inactivated during different conditions, genes part of the same operon show similar expression patters and thereby co-expression across different conditions suggests an increasing probability of genes being part of the same operon [26] [27] [35].

However, most of the *in-silico* methods use a combination of the features defining an operon to identify operons genome -wide in a given species (Table S1) [36] [37] [59] [68]. For example, the classifier of the ProOpDB [69] considers the intergenic distance, functional relation and conserved features for the prediction. Further, databases like the DOOR database extend the given features by including phylogenetic relation and DNA-motifs into the prediction model [65]. As it has been shown that gene expression data enhances the operon prediction, methods like Rockhopper [36] and CONDOP [45] make use of RNA-Seq data for their prediction. Further methods also take into consideration the codon usage of the genes within an operon (Table S1) as the codon usage of genes inside an operon showed to be similar [63] and therefore is thought to have a positive impact on the predictive performance [35] [64] [70]. In general, provided features for the identification of operons are combined in a different manner by numerous tools (Table S1). For instance, several methods which consider the intergenic distance and further genomic features for operon prediction are using a plain scoring model. Subsequently, it can be observed that prediction methods making use of experimental datasets (e.g. via RNA-Seq) are using different machine learning approaches like naïve Bayes classifier (Table S1, Rockhopper), SVM (Table S1, SeqTU, CONDOP) and neural networks (Table S1, ProOpDB).

1.3. Biological applications for machine learning approaches

Due to numerous technological advances, the amount of (biological) data increased in the past years in a way that challenges conventional analysis [71]. Advances especially in the field of high throughput sequencing methods made large biological datasets available [72]. Computational analysis offers powerful ways of identifying trends within a vast amount of data [71] [73] [74] [75] [76]. Especially machine learning as a form of artificial intelligence is best suited for the analysis of biological data within the data [71] [77] [78]. Machine learning enables a system to rather learn from the provided data than through explicit programming [78] and discover patterns in the dataset that are hidden or non-obvious [72]. Thereby, machine learning does not suffer from parameter tuning and software re-programming like a conventional algorithm where the developed algorithm has to manually be adapted to another problem than it has been designed for [71].

Common machine learning tasks are classification (assign classes to input data [71]), regression (predict a continuous characteristic of the inputs [71]) and clustering (assign inputs to groups so that points of one group are similar [71]), which can be assigned to the field of supervised and unsupervised methods [79] [80] [81]. Supervised machine learning approaches are used for classification and regression tasks and generally are separated into the training phase, the test and the prediction phase [73] [79] [82]. In the training phase, the machine

learning algorithm uses a sample of labeled data within the dataset to infer structures and relations and to create a model. Following, during the test phase, the trained model is applied onto new data (also labeled data) to evaluate the performance of the model trained on the labeled data. Finally, the trained machine learner can be used to infer relations in a new, unknown dataset and predict these structures. While for supervised approaches, the machine learning algorithm is trained on labeled classes which the output should be mapped onto, unsupervised machine learning tends to observe a trend within the data without any labeled data by themselves. The aim is thereby to group data into clusters based on similarity measures [71]. However, the workflow of machine learning always contains of data cleaning and pre-processing, feature extraction, model fitting and evaluation [73]. For training, validation and verification of the given classification model, the input dataset is usually split into training set (60%, used for training of the model), validation set (10-20%, used for evaluating the hyperparameters) and test set (20-30%, evaluate accuracy of the model [72] [73]. The training process typically consists of many iterations with a large amount of training samples so the model can learn to recognize complex connections between the input features and thus might be able to reliably assign the correct label [83].

Frequently mentioned machine learning applications are for example support vector machines (SVM, Figure 3A), random forest (RF, Figure 3B) and neural networks (NN, Figure 3D) [82]. SVMs are generalized linear classifiers and can be associated with the field of supervised machine learning algorithms [77] [84]. The aim of SVMs is to minimize the expected generalization error while keeping the number of adjustable parameters small [85], which is also equivalent to minimizing the cost function [86]. SVMs therefore try to separate the provided data and to classify them into two groups of features, separated by an optimal hyperplane [87]. The optimal hyperplane separates the two groups in a way that the data points nearest to the hyperplane (support vectors) of the two different classes exhibit the maximum amount of margin [85]. As a hyperplane is a linear separating function, it can be easily applied to linear separable data [85] [87]. SVMs are considered as kernel methods as they use mathematical functions called kernels [77]. These are specific functions for a given problem and act as an interface between the learning system and the data [77] [88]. The so-called “kernel-trick” is applied by an SVM using the kernel functions to map the inputs non-linearly to a high-dimensional space. If used, they apply dimensions to the dataset in order to find possible structures and separations in a higher dimension [89]. Within this higher dimensionality, the input data can then be separated linearly. While the kernel trick allows SVMs to separate non-linear inputs, the kernel function itself can be of a different kind (e.g. a polynomial, linear, radial or sigmoidal) [88] [90]. Although SVMs are robustly designed against overfitting, especially in high-dimensional space [71], native SVMs have been designed as binary classifiers, whereby the adaption to a multiclass problem (e.g. through “one-against-all”,

“one-against-one” or “directed acyclic graph”) results in an increase of computational resources [87] [91] [92] [93].

In contrast to the native SVMs binary separation classifier [87], the classification model of random forest (Figure 2B) has been shown to efficiently handle large sets of training data with many classes of object detection [94]. RFs are an ensemble of decision trees, while the trees of the ensemble are created by using a randomness vector (e.g. random selection of examples from training set, random split selection or random training set selection) [95] [96] [97]. The final prediction is then obtained by aggregating over the ensemble. A striking point of RF is that many relatively uncorrelated models (trees) operating as a committee reduces the overall variance while keeping the low bias for decision trees [71] [96]. The individual trees protect each other from their individual errors (if they don’t constantly all err in the same direction) and thereby RF does not suffer from the overfitting problem while ensuring to be a highly accurate and robust method [98]. However, because of the creation of multiple trees, RF is a complex method and the algorithm needs a lot of computational resources [99].

Another machine learning technique for numerous tasks are neural networks (NN) [100] [101] [102]. This classification model, albeit greatly simplified, is inspired by the neurons in the human brain [78]. A first model of an artificial neuron is the so-called perceptron (Figure 3C), which is a plain model of prediction and has first been described by Frank Rosenblatt [103]. This model basically consists of a single neuron which receives different (weighted) inputs and produces an output on a(n) (adjustable) threshold [104]. Thereby the perceptron separates the outputs in either 0 or 1. In contrast, a sigmoid neuron is able to take output values between 0 and 1 [105]. Cascading numerous neurons leads to a multi-layer perceptron (MLP) [105]. In contrast to a perceptron, a node of an MLP node produces a graded value based on how close the input is to the desired category instead of binary classifying it [73] [78] [106] [107]. The numerous neurons within such a neural network are connected via edges, which are weighted (Figure 3D). Neurons receive a value of activation through these edges from the previous neuron. Each neuron then applies its activation function onto the input and passes the signal with their edges (which are again weighted) to the next neuron.

There are different types of neural networks. A feedforward NN is fed with a static set of data and the neurons of a certain layer will only feed information to a subsequent layer [106]. Further examples are recurrent NNs (which process sequential data, compute data and time steps and create a memory like effect [76] [106]) and convolutional NNs (which apply mathematical operations called convolution on model input data to put them into the form of multidimensional arrays [73] [76] [108]). NNs typically consist of an input and output layer and can harbor numerous layers of multiple neurons. With the possibility of training NNs with more than one

hidden layer, artificial NNs are considered as “deep”, coined the term of deep-learning models [73]. NNs can be used for both, supervised and unsupervised approaches [76]. During training, the NN (or more specifically its adjustable parameters, e.g. the weights and thresholds/bias) are fine-tuned via backpropagation [76] [108] [109]. For this purpose, a gradient vector for the weights is computed by the learning algorithm, which tracks the decrease or increase of the error if the weights are changed by a tiny amount. Graphically, this can be illustrated as a hilly landscape (high dimensional space of weight values), where the minimum has to be found (low output error) by following the negative gradient vector to the steepest descent from the current position (in the optimization landscape). Since calculation of the gradient is often computationally expensive due to the memory parameters, a commonly used technique is stochastic gradient decent (SGD), which computes the gradient using only a randomly sampled subset of parameters. The velocity of traveling downwards the slope of the optimization landscape is expressed via the learning rate. This hyperparameter describes the amount of adjustments (step size) which are done to the weightings of the edges in the model in respect of moving towards a minimum of the loss function. Finding an optimal learning rate is a crucial point, as a learning rate which is too high might lead to a miss of the desired minimum, while a learning rate which is too low might result in a training process that is too slow [76] [100] [101] [102].

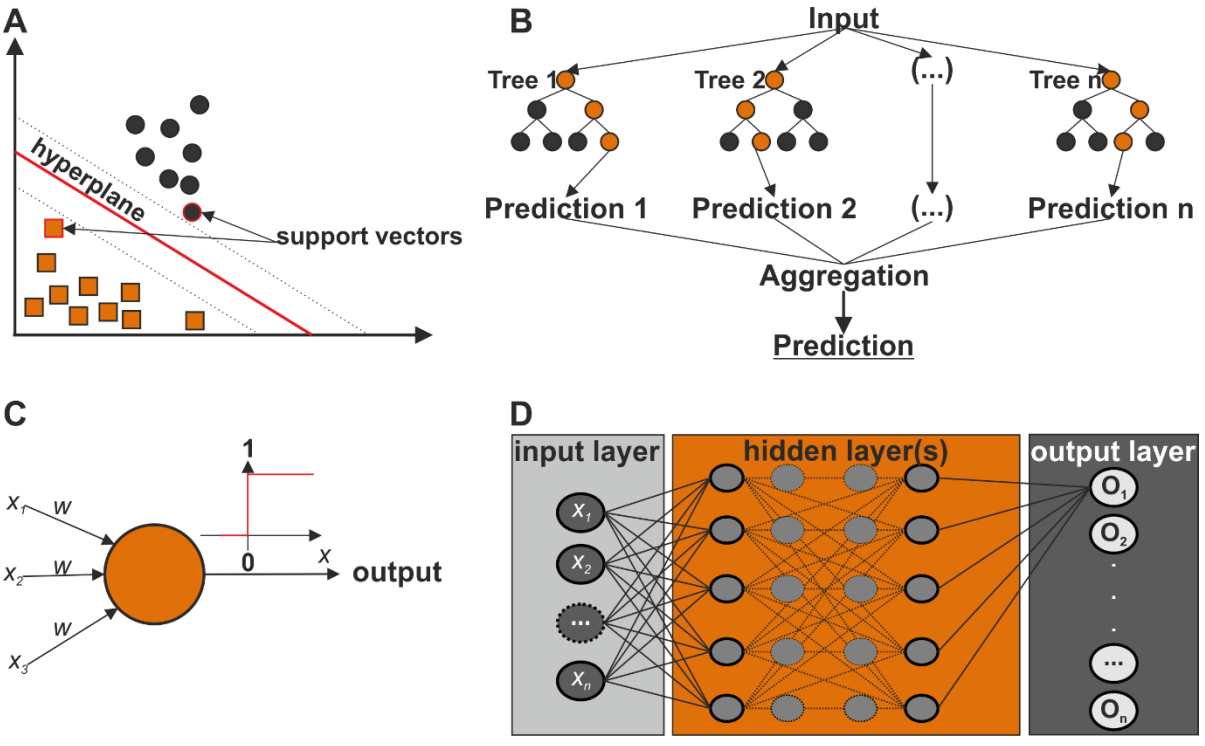


Figure 3: Classification models support vector machine (A), random forest (B), perceptron (C) and neural network (D). While the SVM separates two classes by a hyperplane (A), RF calculates different trees which are then aggregated (B). The plain perceptron (C) receives different inputs and classifies them binary, while the NN can classify multiple classes (D).

Further, the performance of the NN can be influenced by tweaking certain parameters, such as the type of activation function, or by changing/adding biases, which will shift a neuron's activation function in a certain direction along the x-axis [83] [110] [111] [112]. Deep learning methods have been shown to outperform other predictive models on multiple prediction tasks and are the current state-of-the-art for certain domains (e.g. image, audio, and text classification) [73] [76]. However, avoiding overfitting of NNs is a major task and results in their nonlinearity and in many parameters [73] [113]. In general, finding the best model for the prediction always depends on the data and cannot be generalized [73].

In this context, machine learning approaches have successfully applied on biological questions like cancer detection and classification, image classification of biological samples, drug discovery and phenotype prediction [71] [75] [84] [107] [113] [114] [115] [116], as well as the identification of biological structures like TSS, promotors, enhancer, splice sites [82] and operon prediction (Table S1). Additionally, advances in NGS technologies have led to a huge amount of samples, which have the potential to increase our knowledge of the genome dramatically [117]. Especially the use of machine learning applications on these expression data have shown to potentially reveal correlations within this data [117] [118]. Further, the application of these classifiers to expression data (etc. microarray and NGS) has shown to increase the predictive performance of a classifier [46] [119]. Also, for operon prediction, different machine learning methods like RF, SVM and NN (Table S1) have been shown to accurately identify operons by using a combination of different genomic features (e.g. intergenic distance, codon usage and conservation) and expression data approaches [46] [69] [120].

1.4. Putative influence of operon structures to improve biotechnical applications for prokaryotes

As microorganisms regulate their transcriptional program depending on changing conditions [18], it is fundamental to identify and understand operons and TUs for elucidating gene regulatory networks [19] [26] [46]. In times of climate change, there is a major demand for clean energy creation and processes that are able to couple CO_2 fixation with chemical process for energy creation [121] [122]. In this context, the advantage of bacterial systems, in contrast to eukaryotic systems, is the direct coupling of transcription and translation [6] [7] [8]. Together with their faster growth rates compared to plants and the fact that they are easy to manipulate genetically, cyanobacteria are currently in the focus of interest [56], as these photoautotrophic cyanobacteria offer great potential as host organisms for renewable synthesis of chemical bulk products. [123]. Firstly, they are able to directly convert atmospheric CO_2 into hydrocarbons which are suitable for biofuels [124], and secondly, they are able to convert solar energy into biomass [121] [124] [125] [126] [127] [128]. Biofuels, which can be produced through this

biomass, are regarded as one of the central alternative energy sources [129]. Furthermore, compared to other crops, cyanobacteria require a decreased amount of land [129]. Beside their usefulness in terms of biofuels, cyanobacteria were significant contributors to global photosynthetic productivity during evolution [130]. At the same time, cyanobacteria, particularly nitrogen-fixing ones, are beneficiaries of the global climate change [131], which results in enhanced, often harmful, cyanobacterial blooms [132] [133], whereby cyanobacteria secrete compounds with allelopathic properties regulating their ecological community growth [134]. In the environment, cyanobacteria can experience starvation or oversupply of nutrients like metals, phosphate, sulfur or nitrogen [135] [136] [137], which induces regulatory mechanisms to adapt to these environmental changes.

For a detailed understanding of cyanobacterial behavior in ecosystems and communities as well as of their adaption under variable environmental conditions, it is essential to describe underlying molecular adaptations [138] [139]. In addition, it is essential to fully understand their transcriptional regulation and environmental adaptions [56]. For understanding these adaptions, the central elements like operons and alternating TUs are of central interest, as operon identification is an important step in understanding regulatory networks [26] [46]. As bacteria living in biofilms have been shown to be more resistant to antibacterial compounds [140], understanding transcriptional regulation could help to identify genomic regions responsible for the creation of biofilms. However, the formation of operons is still not completely understood [24] [46] [60] [141]. To analyze the transcriptional changes and to conclude on molecular sensors and transducers involved in environmental stresses, systematic approaches including Next-Generation-Sequencing (NGS) methods are used [142] in combination with *in-silico* methods [57].

1.5. Objectives of this study

The transcriptional regulation of prokaryotes is a key element of understanding the gene expression within prokaryotes. In this context, understanding the behavior of prokaryotes when exposed to different stress conditions makes the transcription a central element also in case of an increased productivity of microorganisms for biotechnological approaches under stress conditions. Prokaryotes have shown to exhibit numerous ways of controlling the transcription. Beside known regulations in a classical way of functional non-overlapping controlling regions like operons, further regulation have been discovered even within these controlling regions. This leads to new insights onto transcription unit regulation with overlapping transcription units and multiple TSS even within defined operons. These insights are contrasting to the assumption of non-overlapping operons. In general, regulation features can be discovered via genomic features, which are missing the information of stress dependent expression and serve

perfectly for the traditional way of operon and TU identification. Therefore, the inclusion of expression data via RNA-Seq leads to an increased precision of TU identification in a dynamic way and under different conditions. As numerous available tools do not take expression data into account or are specialized on given model species, there is a need for an expression-based species-free prediction pipeline.

In the first section of this study, the task of creating a prototype of a usable framework for the identification of operons is addressed. For this reason, beside the user intuitive graphical user interface (GUI), data processing steps like the mapping of NGS data should be implemented. In this context, a challenging task is the fast and memory efficient evaluation of these data. For gaining insights into the processed data, different models should be developed that on the one hand guarantee the identification of operons in a traditional way. On the other hand, they should also be able to work without a provided reference while at the same time providing hints on alternative TUs within operons. Beside the evaluation of expression data, the formatting of genome-based features is a crucial part for the TU identification as well. Therefore, a generalized automated way should be developed. Both, expression and genomic features, thereby should be applicable in a generalized manner so that the developed concept does not lack of the drawback of being specialized only for model species.

Consequently, in the second part of this study, the first defined model, which should be a training free approach, will be compared to existing tools and databases as well as to literature defined operon structures based on the model organism *Escherichia coli* and the non-model cyanobacteria *Anabaena* sp. PCC 7120. Following, the predicted structures will be compared to a genome wide experimental approach (TSS identification) in *Anabaena* sp. PCC 7120. Finally, the identification of TUs will be improved by letting machine learning applications learn through the provided data and therefore being free of manual cut-off provision.

2. MATERIALS & METHODS

All databases, tools and datasets have been used, collected and created between March 2017 and March 2020.

2.1. Database sources

2.1.1. CyanoBase

CyanoBase (http://genome.microbedb.jp/mnt.html, [143]) is a database for cyanobacteria and harbors cyanobacterial species information, complete genome sequences, genome-scale experiment data, gene information, gene annotations and mutant information. It was used to access different genomic data of cyanobacteria.

2.1.2. Database of prOkaryotic OpeRons (DOOR)

The DOOR (version 2.0, http://161.117.81.224/DOOR2/index.php, [65]) contains computationally predicted operons of all sequenced prokaryotic genomes. Thereby DOOR contains predicted operons and gene pair lists for 2072 prokaryotic genomes, based on a decision tree trained with data from *Escherichia coli* and *Bacillus subtilis*. DOOR was used to acquire operon labels from the opr files for a list of neighboring gene pairs for *Anabaena* sp. PCC 7120, *Bacillus subtilis*, *Clostridium perfringens*, *Escherichia coli*, *Synechococcus elongatus* and *Synechocystis* PCC 6803.

2.1.3. DNA Data Bank of Japan (DDBJ) Sequence Read Archive

The DDBJ Sequence Read Archive (DRA, https://www.ddbj.nig.ac.jp/dra/index.html, [144]) is an archive database for output data generated by next-generation sequencing machines and offers a close collaboration with NCBI Sequence Read Archive (SRA). It was used to search SRA RNA-Seq samples (http://sra.dbcls.jp/index.html).

2.1.4. Google scholar

Google scholar (http://scholar.google.de) provides the search for scholarly literature and was used to search for different publications.

2.1.5. National Center for Biotechnology Information

The National Center for Biotechnology Information (NCBI, http://ncbi.nlm.nih.gov, [145]) provides biochemical and genomic information. PubMed (http://ncbi.nlm.nih.gov/pubmed) was used to search for publications. Furthermore, the NCBI Genome Database (https://www.ncbi.nlm.nih.gov/genome) organizes information on genomes including sequences, maps, chromosomes, assemblies, and annotations and was used to collect generic feature format files (GFF) and FASTA files of different species. The Sequence Read

Archive (SRA, https://www.ncbi.nlm.nih.gov/sra,[146]) makes biological sequence data available and was used to extract different RNA-Seq data samples.

2.1.6. Prokaryotic Operon DataBase (ProOpDB)

ProOpDB (version 1.0, http://biocomputo2.ibt.unam.mx/OperonPredictor/, [29]) contains operons identified by an operon prediction algorithm and harbors the operon structures of 1200 prokaryotic species. The operons are predicted using a multilayer perceptron neural network with intergenic distance between two subsequent genes as well as STRING-scores as input features. ProOpDB was used to acquire operon labels from the gene pair list files for a list of neighboring gene pairs for *Anabaena* sp. PCC 7120, *Bacillus subtilis, Clostridium perfringens, Escherichia coli, Synechococcus elongatus* and *Synechocystis* PCC 6803.

2.1.7. RegulonDB

RegulonDB (http://regulondb.ccg.unam.mx/, version 10.5, [147]) is a relational database harboring data on the transcriptional regulation in *Escherichia coli* K-12 containing knowledge manually curated from original scientific publications, complemented with high throughput datasets and comprehensive computational predictions. It was used to extract operons for *Escherichia coli* K-12.

2.2. Tools and libraries

2.2.1. Anaconda

Anaconda (https://docs.anaconda.com/anaconda/, version 3 2019.07) is an open-source package and environment manager for python and R for scientific computing (e.g. machine learning and data science).

2.2.2. BBMap

BBMap (https://sourceforge.net/projects/bbmap/, version 38.75, [148]) is a short-read aligner for DNA and RNA-Seq data. It is written in Java and is thereby platform independent and only requires Java being installed on the target system. It was used to map RNA-Seq samples of different species onto a reference genome.

Table 1: Used parameters for BBMap.
The first column shows the used options, the second column shows the used value, and the third column shows their description.

Option	Used	Description
ref=	<file to fasta sequence>	Path to the reference genome.
in=	<r1.fq>	Path to the read file containing mates 1.
in2=	<r2.fq>	Path to the read file containing mates 2.
threads=	16	Number of threads used for candidate search.
ambigous=	best	Set behavior on ambiguously mapped reads (with multiple top-scoring mapping locations).
mappedonly=	true	Write only mapped reads to output file
output=	<outputFile.sam>	Path to output file.

2.2.3. Bowtie2

Bowtie 2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml, version 2.3.5.1, [149]) is a tool for aligning sequencing reads to reference sequences supporting gapped, local, and paired-end alignment modes. Bowtie 2 creates and uses a full-text minute index for each genome. It was used to map RNA-Seq samples of different species onto a reference genome. Further, bowtie2-build was used to create an index file for different FASTA input (bowtie2-build <*inputFile.fa*> <*outputFileIndex*>).

Table 2: Used parameters for Bowtie2.
The first column shows the used options, the second column shows the used value, and the third column shows their description.

Option	Used	Description
-x	<file to fasta sequence>	Path to the reference genome.
-1	<r1.fq>	Path to the read file containing mates 1.
-2	<r2.fq>	Path to the read file containing mates 2.
-t	-t	Print wall-clock time taken by search phases.
-p	16	Number of threads used for candidate search.
--mm	--mm	Use memory-mapped I/O for index; many 'bowtie's can share.
--no-unal	--no-unal	Do not show unaligned reads.
-S	<outputFile.sam>	Path to output file.

2.2.3.1. Jupyter Lab

Jupyter lab (https://jupyter.org/hub, Version 0.35.4) is an open-source web-based user interface for project jupyter. It integrates versioning to create and share documents containing executable code, (interactive) output and descriptive texts. Jupyter lab was used to create notebooks for development and evaluation of machine learning methods and preparation and analysis of data.

2.2.4. CorelDRAW

CorelDRAW (version X6; http://coreldraw.com) is a tool for the creation and modification of images and figures and was used to create and modify figures for this thesis.

2.2.5. Eclipse

Eclipse (version 4.14.0, https://www.eclipse.org/) is an integrated development environment (IDE). It offers the development in different programming languages and is open-source. Eclipse was used to write the framework (in Java) and smaller applications (Java & Python).

2.2.5.1. Git integration for Eclipse (EGit)

EGit (https://www.eclipse.org/egit/, version 5.6.0.201912101111-r) is a provider of the git version control system for Eclipse. It was used to manage versioning of the code.

2.2.5.2. Maven integration for Eclipse (M2Eclipse)

M2Eclipse (https://www.eclipse.org/m2e/, version 1.8.3) enables the integration of Apache Maven into the Eclipse IDE. It was used to manage the developed pipeline in Java with Maven via Eclipse.

2.2.5.3. Python Development Environment for Eclipse (PyDev)

PyDev (https://www.pydev.org/, version 7.4.0) is an open-source integrated Python IDE for Eclipse and enables it to develop Python code.

2.2.5.4. Spring Tools for Eclipse

Spring Tools (https://marketplace.eclipse.org/content/spring-tools-4-spring-boot-aka-spring-tool-suite-4, version 4.14) integrates Spring into the Eclipse IDE and enables it to develop modern Spring Boot applications.

2.2.6. GitHub

GitHub (https://github.com/, version 2.25.0) is a hosting service for software development and version controlling based on git (https://git-scm.com/, version 2.24.1), which is the control system for software project development. It was used to manage the code written for this thesis.

2.2.7. HTSeq

HTSeq ([150] , https://htseq.readthedocs.io/, Version 0.11.1) is a free python package providing functions and data structures to process data from high-throughput sequencing experiments. HTSeq-count was used to analyze genome regions covered by mapped reads.

Table 3: Used parameters for HTSeq.
The first column shows the used options, the second column shows the used value, and the third column shows their description.

Option	Used	Description
path to sam file	<file to SAM>	Path to the SAM file.
path to gff file	<genome.gff>	Path to the reference genome.
-t	gene	Feature type (3rd column in GFF file) to be used.
-s	no?	Specifies if the data is from a strand-specific assay.
-m	union	Mode to handle reads overlapping more than one feature.
-r	pos	For paired-end data, the alignment must be sorted either by read name or by alignment position.
--nonunique	all	Mode to handle reads that align to or are assigned to more than one feature in the overlap <*mode*> of choice (see -m option).
-a	10	Skip all reads with alignment quality lower than the given minimum value.

2.2.8. Java scripts and libraries

Java (https://www.java.com/, https://openjdk.java.net/, version SE 8 LTS) is a class-based, object-oriented general-purpose programming language. It was used for the development of the operon prediction pipeline as well as for the parsing of data out of different files.

2.2.8.1. Deeplearning4J

Deeplearning4J (https://deeplearning4j.org/, version 1.0.0-beta4) is an open-source deep learning library for the java virtual machine (JVM) and was used to import the neural network written in python with TensorFlow into the java-written operon prediction pipeline.

2.2.8.2. JUnit

JUnit (https://junit.org, version 4). is a testing framework for the Java programming language. It was used to create test cases for the different parts of the written framework and evaluate the tests.

2.2.8.3. Kryo

Kryo (https://github.com/EsotericSoftware/kryo, version 5.0.0-RC4) is a binary object graph serialization framework for Java with the focus on speed, efficiency and user-friendly API. It was used to serialize the data structures of the framework.

2.2.8.4. ND4j-native-platform

ND4j (https://nd4j.org, version 1.0.0-beta4, [151]) is an open-source deep learning project for Java including n-dimensional arrays for java. It was used to set up the input data for the prediction models.

2.2.8.5. Simple Logging Facade for Java (SL4J)

SL4J (http://www.slf4j.org/, version 1.7.5) offers different logging frameworks allowing the user to use the desired logging framework at deployment time. It was used to integrate log4J (https://logging.apache.org/log4j/2.x/, version 2.13.0) into the pipeline.

2.2.8.6. Spring framework

The Spring framework (https://spring.io/, version 5.2.2) is an open-source application framework and control container for the Java platform. It can be used for different modern applications like modern cloud-based microservice applications or web-applications. Spring Boot thereby makes it extremely efficient to implement applications and services on top of Java as it is the starting point for all Spring-based applications. From the Spring framework, Spring Boot was used to create the GUI for this thesis.

2.2.8.7. Thymeleaf

Thymeleaf (https://www.thymeleaf.org, version 2.1.6) is a server-side Java template engine for both web and standalone environments, capable of processing HTML, XML, JavaScript, CSS and even plain text. As a Java library it was used to create the GUI of OpPipe and connect the single parts of OpPipe to one pipeline.

2.2.9. Mason

Mason (https://www.seqan.de/apps/mason/, [152], version 2019-03-21) is a read simulator for second generation sequencing (SGS) techniques like Illumina, 454 and Sanger sequencing. It was used to simulate reads out of different genomes to evaluate the mapping part of the framework.

Table 4: Used parameters for Mason.
The first column shows the used options, the second column shows the used value, and the third column shows their description.

Option	Used	Description
sequencing technology	illumina	The first argument specifies the sequencing technology to use.
-mp –mate-pairs	-mp	Enable mate pair simulation.
-n --read-length	100	Length of the reads to simulate.
-s --seed	not used	The seed for the random number generator.
-N --num-reads	100	Number of reads to generate.
-rnp --read-name-prefix	read	Read name prefix.
-f --forward only	-f	Simulate from forward strand only.
-nN –no-N	-nN	If set, no Ns will be introduced in the reads.
-sq --simulate-qualities	-sq	If given, qualities are simulated for the reads and the result is a FASTQ file, is FASTA otherwise.
-o --output-file	random100.fq	Path to the resulting FASTA/FASTQ file.

2.2.10. Maven

Maven (https://maven.apache.org/, version 3.6.3) by Apache is a software management tool and was used to build the different parts of the framework including their dependencies.

2.2.11. Mendeley

Mendeley (https://www.mendeley.com, Version 1.19.5) is a free reference manager to store and manage research and publications. It was used to store different articles together with the Microsoft Word plugin to manage the citations and to create the bibliography and references in this thesis.

2.2.12. Microsoft Office

Microsoft Office (http://office.com, version 365) is an office suite providing different office applications and services. Microsoft Word was used to write this thesis. Microsoft Excel was used to create diverse tables and graphics and Microsoft PowerPoint was used for the creation of different graphics.

2.2.13. Microsoft Windows

Windows 10 (https://www.microsoft.com/de-de/windows, version 10 Home) is an operating system produced by Microsoft and was used as operating system during this thesis.

2.2.14. NCBI Sequence Read Archive (SRA) Toolkit

The SRA Toolkit (https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software, version 2.10.0) allows users to download biological sequence data from SRA. It was used to download different RNA-Seq samples.

2.2.15. NetBeans

NetBeans (http://netbeans.org, version 8.0) is a software development platform and was used to create java scripts for the automatic parsing and for the evaluation of different datasets.

2.2.16. NextGenMap

NextGenMap (NGM, http://cibiv.github.io/NextGenMap/, Version 0.5.5, [153]) is a fast and flexible read mapping program, which is able to handle read data independent of read length and sequencing technology and also to automatically adapt to highly polymorphic regions while keeping runtime low and sensitivity high. NextGenMap was used with default parameters to map reads from RNA-Seq experiments.

Table 5: Used parameters for NGM.
The first column shows the used options, the second column shows the used value, and the third column shows their description.

Option	Used	Description
-r	<file to fasta sequence>	Path to the reference genome.
-1	<r1.fq>	Path to the read file containing mates 1.
-2	<r2.fq>	Path to the read file containing mates 2.
-p	-p	Input data is paired end.
-t	16	Number of threads used for candidate search.
--no-progress	--no-progress	Silent mode.
--strata	--strata	Only output the highest scoring mappings for any given read, up to mappings per read.
--no-unal	--no-unal	Do not show unaligned reads.
-o	<outputFile.sam>	Path to output file.

2.2.17. Oracle VirtualBox

VirtualBox (https://www.virtualbox.org/, version 6.1) is a visualization product for operating systems that supports many guest operating systems (e.g. Windows and Linux).

2.2.18. Picard

Picard (https://broadinstitute.github.io/picard/, version 2.21.6) is a set of command line tools for the work with high-throughput sequencing data and formats (e.g. SAM and BAM). Within Picard, SortSam was used to sort the reads of a SAM (and BAM) file based on their coordinates.

2.2.19. Python scripts and libraries

Python (https://www.python.org/, version 3.6.8) is an interpreted general-purpose programming language. Python was used for the creation of different machine learning models and visualization of these models as well as for parsing information out of different data files.

2.2.19.1. Keras

Keras (https://keras.io/, version 2.3.0) is a high-level API for neural networks written in Python. It has a focus on easy and fast experimentation and prototyping. Keras can run on top of different machine learning libraries. Keras running on top of TensorFlow and Keras without running on top of another library was used for the implementation of the feed-forward neural networks.

2.2.19.2. Matplotlib

Matplotlib (https://matplotlib.org/, version 3.1.0) is a library for data visualization and generating publication quality figures in Python. Matplotlib was used to create visualizations of the machine learning models.

2.2.19.3. NumPy

NumPy (http://www.numpy.org/, version 1.15.4) is a package containing many tools for scientific computing in Python. NumPy was used to handle the data sets and is also a dependency for other used libraries.

2.2.19.4. Pandas

Pandas (https://pandas.pydata.org/, version 0.24.2) is an open-source library for Python providing different data structures and analysis tools, especially for numerical tables and time series. It was used for data analyses in the context of machine learning applications.

2.2.19.5. PyInstaller

PyInstaller (http://www.pyinstaller.org/index.html, version 3.6) bundles a Python application and all its dependencies into a single package. The user can run the packaged app without installing a Python interpreter or any modules. It was used to include the SVM and RF models into a packed app.

2.2.19.6. Scikit-learn

Scikit-learn (https://scikit-learn.org/, version 0.22.1) is an open-source machine learning library written in Python using NumPy, SciPy and matplotlib. It has various tools for data mining and data analysis. Scikit-learn was used for development and performance evaluation of the different machine learning models.

2.2.19.7. Seaborn

Seaborn (https://seaborn.pydata.org/, version 0.9.0) is a high-level interface for statistical data visualization based on matplotlib. Seaborn was used to create visualizations of the machine learning models. It was used to run Ubuntu during this thesis.

2.2.19.8. TensorFlow

TensorFlow (https://www.tensorflow.org/, version 1.13.1) is a library for dataflow programming that was originally developed by the Google Brain Team for internal use but was released as an open-source library in 2015. TensorFlow has a strong support for machine learning and deep learning. TensorFlow was used for the implementation of feed-forward neural networks.

2.2.20. Rockhopper

Rockhopper (https://cs.wellesley.edu/~btjaden/Rockhopper/index.html, [36], version 2) is a tool for the computational analysis of RNA-Seq data including reference based transcript assembly, de novo transcript assembly, normalization of experiments, transcript abundance quantification, differential gene expression and operon prediction. It was used to predict sets of operons for different species to benchmark them against other prediction tools.

2.2.21. SigmaPlot

SigmaPlot (Version 12.5; http://systatsoftware.com/products/sigmaplot) is a tool for the visualization of scientific datasets. It was used to create figures in this thesis.

2.2.22. Ubuntu

Ubuntu (https://ubuntu.com/download, version 18.04 LTS) is a free open-source Linux distribution and was used via virtual box to execute libraries and tools only available for Linux.

2.2.23. WebVector

WebVector (http://cssbox.sourceforge.net/webvector/, version 3.4) converts HTML to SVG, PDF or PNG format. It was used during this thesis to convert the HTML output of the visualizer part into SVG and PNG images.

2.3. Organism sources and data samples

2.3.1. Biological samples for *Anabaena* sp. PCC 7120

For RNA-Seq analysis, different datasets were used from *Anabaena* sp. PCC 7120, which were created and kindly provided by Niclas Wolfgang Fester (PhD student, AK Schleiff). In total, 18 samples with nine different conditions (Control, -Cit, -Co, -Cu, -Fe, -Mn, -Mo, -Nit, -Zn) each with two technical replicates have been created. YBG11_0 was the base media with addition of ammonium chloride as nitrogen source except of the -Nit samples (Tables S2-3). To buffer pH changes due to the ammonium chloride, the media were buffered with TES buffer. For all metal starvation samples, only the corresponding metal was left out. The media have been combined from different premixed and sterile filtered solutions. The autoclaving process was only used for the water to prevent the precipitation of metals. All cultures were grown in 50 ml glass flasks with a metal lid at 30 °C under light conditions (70 µmol photons per m² *

sec). For starvation conditions the *Anabaena* sp. PCC 7120 cultures were pre-starved (in total seven days) to get rid of extant metals that were brought in with the cells. For the pre-starvation, the fresh biomass was washed twice in the corresponding dropout medium before it was inoculated with an OD (optical density) of 0.1. After 3 and 7 days, the cells were washed and reinoculated in fresh dropout medium to reduce intrinsic metal of the bacteria. In the starvation step after 7 days, the cultures were grown until the iron deficient sample reached a chlorophyll a content below (2 µg/µl). As the iron starved cultures reached that point, all samples were precipitated and rapidly frozen with liquid nitrogen.

The total RNA was extracted and purified using the ambion PureLink RNA Mini Kit and the DNA depletion was done on-column (for 15 minutes at ~21 °C). After the extraction, the total RNA was quantified (peqLab NanoDrop 1000) to ensure the purity and concentration needed for the RNA-Seq experiment. After the RNA extraction and in advance of the library preparation, the quality of the total RNA was checked generating a RIN score equivalent with a bioanalyzer and the LabChip GK Software (Version 5.2.2009.0). Subsequently the rRNA depletion was done with a RiboZero Kit. The library preparation was done by GenX Pro and the RNA sequencing was done on an Illumina 1.9 Sequencer. The RNA sequencing yielded about 1 billion read pairs overall. For this thesis, the samples of the control, -Fe and -Nit have been used for operon prediction (Table S4).

2.3.2. Collected data samples from public databases

Beside the provided RNA-Seq datasets for *Anabaena* sp. PCC 7120*,* additional datasets have been collected from different data sources (Table S4). For the species *Anabaena* sp. PCC *7120, Bacillus subtilis, Clostridium perfringens, Corynebacterium xerosis, Escherichia coli, Synechococcus elongatus* and *Synechocystis* PCC 6803, genomic sequence files (GFF, FASTA) were extracted from the CyanoBase (2.1.1) and NCBI genome (2.1.5) (Table S4). Additionally, SGS (second generation sequencing) RNA-Seg data were collected for *Bacillus subtilis, Clostridium perfringens, Corynebacterium xerosis, Escherichia coli, Synechococcus elongatus* and *Synechocystis* PCC 6803 under control conditions using the NCBI SRA. For *Escherichia coli,* additional TGS (third generation sequencing) data was collected, while for *Anabaena* sp. PCC 7120 TSS were collected (TSS selection during nitrogen stress-induced cell differentiation and control conditions, Table S4). For all species, operon annotation was extracted from the databases DOOR (2.1.2), ProOpDB (2.1.6) and RegulonDB (2.1.7) for comparison. Subsequently, experimentally proven operons for *Anabaena* sp. PCC 7120, *Bacillus subtilis, Corynebacterium, Escherichia coli* K-12*, Synechococcus* and *Synechocystis* PCC 6803 were extracted by a literature search (PubMed (2.1.5) and google scholar (2.1.4), Table S5). The literature set resulted in a total of 71 operons, while 18 are designated from

Anabaena sp. PCC 7120, 12 from *Bacillus subtilis,* 3 from *Corynebacterium,* 20 from *Escherichia coli* K-12, 10 from *Synechococcus* and 8 from *Synechocystis* PCC 6803.

2.4. Hardware

All calculations and computations were done on a workstation with an Intel® Core™ i7-4710MQ CPU @ 2.50GHz with 16 GB of RAM, Crucial MX500 SSD (1TB), 2 WD Elements external HDD (4TB and 2TB).

2.5. Statistical measures and implemented calculations

For the measure of the different created filters and the resulting classification models, different statistical measures were used (Table 6). Reference labels to verify true positive (TP), true negative (TN), false positive (FP) and false negative (FN) rates for prediction were based on gene pairs from the experimental proven literature set of known operons (Table S5).

Table 6: Statistical measures for the evaluation of the different filters and prediction models. Indicated are the different formulas for the statistical measures with TP = true positives, TN = true negatives, FP = false positives and FN = false negatives. The F1-score is the harmonic mean of precision and recall.

Statistical measure	Definition
Accuracy	$\frac{TP + TN}{(TP + TN + FP + FN)}$
Precision (Positive Predictive Value)	$\frac{TP}{(TP + FP)}$
Specificity (True Negative Rate)	$\frac{TN}{(TN + FP)}$
Sensitivity (Recall, True Positive Rate)	$\frac{TP}{(TP + FN)}$
F1-score	$\frac{2TP}{(2TP + FP + FN)}$

Subsequently, different formulas have been implemented within OpPipe to allow normalization of mapped reads and intergenic distance between gene pairs (Table 7). For the mapping of reads onto a reference, the template length was considered, which has to be calculated for paired end reads (Table 7, Formula 1). The intergenic distance is calculated by subtracting the start and end positions of two genes (Table 7, Formula 2). The score of one filter within the plain prediction model is reached by multiplying the binary input with a given weight (Table 7, Formula 3). The total score of the plain prediction model is achieved by summing up the scores of the different filters (Table 7, Formula 4). Calculation of a scaling factor for different RNA-Seq samples (Table 7, Formula 5).

Table 7: Implemented formulas within the prediction framework. Indicated are the description of the formula and the formula itself**.** In Formula 1, pNEXT indicates the position of the next template, which is the start of the mate ($read_B$) if the read is properly mapped and the start of the read itself ($read_A$) if not.

	Description	Formula
1.	Calculation of total template length basing on two paired end reads	$templateLength_{\ readA-readB} = POS_{readA} + length_{readA} - pNEXT$
2.	Intergenic distance of two genes	$Intergenic\ distance_{Gene1,Gene2} = Start_{Gene2} - End_{Gene1}$
3.	The score of each sub-filter by multiplying binary encoding with the weighting	$Score_{G1-G2} = I_B * W$
4.	Maximum reachable score for all inputs	$MAX_{Score} = SUM_{EQ_W} + SUM_{GG_W} + SUM_{DGD_W}$
5.	Calculation of scaling factor for library size normalization	$SF = \sum \text{ConditionA} \,/\, \text{MAX}(\sum \text{ConditionA}, \ldots, \sum \text{ConditionN})$

2.6. Implementation and benchmark procedures

2.6.1. Operon prediction and data set creation

Despite the collected operons from the operon databases DOOR (2.1.2) and ProOpDB (2.1.6), Rockhopper (2.2.20) was used to predict operons for *Anabaena* sp. PCC 7120 and *Escherichia coli* using the same RNA-Seq data from control conditions based on the same data as OpPipe. Subsequently, the predicted and collected operon sets were filtered and harmonized to a standard format and saved in a tab separated list. Therefore, all adjacent gene pairs on the same strand of the same operon were extracted as well as information about (i) numeration of genes in operon, whereby genes were numerated basing on their occurrence in the GGF-file of the referring species (ii) genes in operon, (iii) operon number (from 0 to n), (iv) strand of operon (based on GFF file from the referring species).

Overall, from the 71 operons derived out of *Anabaena* sp. PCC 7120*, Bacillus subtilis, Corynebacterium, Escherichia coli* K-12*, Synechococcus* and *Synechocystis* PCC 6803, 280 gene pairs were labelled as operon gene pairs (OP). For the generation of sets of non-operon gene pairs (NP), two approaches were used: (1) The boundary genes of the experimentally known operons from the literature set, meaning the gene before and after the operon, served as NP-set1 resulting in 142 gene pairs, (2) all gene pairs were identified that were not part of an operon in the dataset of Rockhopper, ProOpDB and DOOR and included to a NP set. Out

of this set, 280 gene pairs were randomly drawn which leads to the NP-set2 which is of the same size as the OP set.

2.6.2. Mapping of RNA-Seq via OpPipe

To determine the handling of mapped reads by OpPipe, NGM was used to map the control dataset onto the FASTA reference of *Anabaena* sp. PCC 7120. The arising SAM file was used to evaluate the mappings of the reads per position and per gene by evaluating the SAM flags (Table S6). This led to the adjustment for OpPipe of only considering reads with SAM flags 99-147, 105-149, 153-101 and 155-103 (proper-paired reads and reads which are mapped to a strand but their mate is unmapped) and of discarding reads with 97-145 and reads without a strand information. Consequently, mapped reads, which could be mapped to a strand and whose mates were unmapped but exhibited a quality (SAM position 5, MAPQ, [154]) lower than 10 were also rejected. For defining the counts of mapped reads per genomic position, OpPipe uses (i) the read length of the mapped read for single-end data and for paired end reads where only one pair matched onto the reference, or (ii) the whole template length of the mapped read pairs and therefore always the read ($read_A$) with the negative template length (SAM position 9, tLen, [154]), as it indicates the rightmost position of the whole template [154]. From the start until the end of the read (or the template, respectively) each genomic position is increased by one. For paired end reads with negative tLen, the strand is then inverted, as for the library types fr-unstranded, fr-firststrand and fr-secondstrand reads from the leftmost end of the fragment map to the transcript strand like suggested in Trapnell *et al*. [155]. For the length of $read_A$, the CIGAR string (indicates matches/mismatches, insertions and deletions of a mapped read, [154]) is then evaluated (SAM position 6, CIGAR, [154]). The length of the whole template is then calculated (Table 7, Formula 1) by subtracting the end of $read_A$ (start of $read_A$ + length of $read_A$) from the position of the primary alignment of the next read in the template (SAM position 8, pNEXT [154]). The mapping part of OpPipe thereby needs a provided GFF file and creates different data structures, namely (i) count coverage for each genomic position, (ii) the number of reads mapped onto an annotated gene (based on GFF file) and (ii) the number of reads being mapped onto two annotated genes (based on GFF file) and thereby connecting them. This mapping procedure of OpPipe is done separately for the plus and minus strand of the referring reference. The OpPipe mapping was then applied onto all species.

2.6.3. Mapping and runtime benchmark of OpPipe

For versatility in respect to the mapping part of OpPipe, different aligners were evaluated. Therefore, the runtime and mapping precision of BBMap (2.2.2), Bowtie2 (2.2.3) and NGM (2.2.16) was evaluated. For the runtime analysis of the mapping, the build of an index was

included. For this purpose, FASTA files of five reference species were used to map the RNA-Seq datasets of the control experiments (Table S4).

To further benchmark the mapping of OpPipe, 100 reads were simulated from the genomic sequence of the gene alr2350 from *Anabaena* sp. PCC 7120 using mason (options mentioned in 2.2.8). These reads were mapped onto the genomic reference of *Anabaena* sp. PCC 7120 to generate a SAM file using NGM (2.2.15). For each possible SAM flag of paired end reads, an independent SAM file was created with a different amount of reads and each read only containing this specific SAM flag (and the flag of its mate). Then HTSeq (option compare 2.2.7) and the mapping part of OpPipe were run onto the manually modified SAM files to evaluate which reads were being used by the mapping algorithms. Each of these files contained simulated paired reads with different SAM flags which were created with mason (2.2.10). Subsequently, the mapped reads per gene have been compared between HTSeq-count and the OpPipe-count. Subsequently, HTSeq-count and OpPipe -count were both applied on the fully mapped set of *Anabaena* sp. PCC 7120. In a first step, runtimes of the mapping framework and of HTSeq have been compared. Subsequently, nine different genes were drawn randomly to compare counted reads and to evaluate the SAM flags of these reads.

2.6.4. Filter creation for operon prediction in plain predictor, SVM, NN and RF model

For the creation of input features for the prediction algorithms, genomic information and expression data have been used to create four different so-called filters. The expression data from RNA-Seq have been used to create (i) gene graph filter (GG) and (ii) expression pattern filter (EP). A provided GFF file was used to calculate (iii) the dynamic intergenic distance filter (DGD). As a prerequisite for the expression-based filters, the expression profile was used to calculate coverage per genomic position and the number of reads covering one or more genes. The GG filter constructs a graph with the nodes being the genes of a species with edges connecting them. These are initially weighted with zero. If a pair of genes has been hit by the same read, an edge between them is created and the weight is increased by one. For each following read covering both genes, the weight of their edges is increased by one. This leads to a graph of genes indicating, via the edges, by how many reads both genes have been hit. To identify TUs, the algorithm loops over the gene graph and indicates genes as a start (or end, respectively) of the TU, if their edge-weight is equal or higher (or lower, respectively) than a defined cut-off (co=2 in initial state).

The EP filter determines a cut-off which indicates that at least x% (x can be 0.98, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1 and 0.05) of all genomic positions are covered by n reads. Thereby different kind of the stringency in the range from 0.98 (tolerant) to 0.05 (strict) lead to

12 different EP stringencies. For each stringency between 0.98 and 0.05, the cut-off is determined by first looping over all genomic positions and detecting by how many reads they have been hit (n, Table S7A). Subsequently, it is detected, how often n occurs within the genome (occurrence in genome, o, Table S7B) and all occurrences are summed up (sum of occurrences, S, Table S7). Further, the sum of occurrences (S) is multiplied with the stringency (ST) and subtracting this result from the sum of occurrences (S, Table S7C) leads to a stringency value (ST_{val}, Table S7). The cut-off is then defined by searching the sum of occurrences (S_o, Table S7), which is greater than the stringency value (ST_{val}, Table S7B). For example, for the stringency of 0.98, this would suggest that only 2% of all genomic positions are covered by less than one read, but that at least 98% of all genomic positions are covered by at least one read (Table S7B). Subsequently, it is looped over the expression landscape and if the number of reads hit to a genomic position is higher or equal (or lower, respectively) than the cut-off, this position is marked as the start (or the end of a TU, respectively). Subsequently, all genes of this TU are identified.

From the GFF file, the distance of all adjacent gene pairs is used to create the dynamic gene distance (DGD) filter. Therefore, the intergenic distance (given in nucleotides) of adjacent protein coding genes is grouped into specific distance ranges (smaller than one, 1 to 100, 101 to 200, 201 to 300, 301 to 400, 401 to 500, 501 to 1000, greater than 1000). Consequently, it is evaluated how many percent of all genes are part of each specific distance range. The percentages of each distance range are then summed up (from small to great distance). For defining a cut-off for classifying two adjacent genes in and operon pair (OP) or non-operon pair (NP), it is evaluated at which distance range the sum is greater than a predefined value x. Even though it is assumed that 50-60% of all genes in a prokaryotic genome are part of an operon [27] [28], x was set initially to 45% for the DGD (explaining % of all gene pairs), to minimize the amount of false positives in the initial approach. The upper bound distance range is then set as the cut-off (co). Subsequently, the algorithm loops over each gene and determines for each gene (G_1) the distance to its adjacent pair (G_2). If the distance is within the cut-off, G_1 is marked as the TU start and consequently G_2 is marked as TU end if the distance is greater than the cut-off. In general, each filter is conducted separately for Watson and Crick strand and leads to an independent list of putative TUs.

Subsequently, the filters GG, EP and DGD were grouped together into an output format that served as input for the different prediction algorithms. Each filter indicates for a pair of genes, if they are part of a TU in the given filter (labelled as ‘1’ OP and ‘0’ NP). Thereby, the input is a table with the columns indicating the gene pairs, EP filter (0.98 to 0.05 cut-off), GG filter (GG fulfilled & GG not fulfilled) and DGD filter (DGD fulfilled & DGD not fulfilled). Thereby, the filters

are divided into $EP_{0.98}$, $EP_{0.95}$, $EP_{0.90}$, $EP_{0.80}$, $EP_{0.70}$, $EP_{0.60}$, $EP_{0.50}$, $EP_{0.40}$, $EP_{0.30}$, $EP_{0.20}$, $EP_{0.10}$, $EP_{0.05}$, GG_Y, GG_N, DGD_Y, DGD_N leading to 16 inputs per adjacent gene pair of a specific strand.

2.6.5. Combining filters to a plain prediction model

The defined input format served as starting point for a first operon prediction structure. Therefore, a plain decision tree predictor was built (using the EP, GG and DGD filter). Specific weights were applied to the 12 EP, 2 GG and 2 DGD input vectors while the vectors of each filter sum up to one. The inputs are simply binary coded ('1' adjacent gene pair part of TU within this filter, '0' adjacent gene pair not part of TU within this filter). Afterwards, all achieved scores of all inputs are summed up leading to a total score for each gene pair ($TS_{G1\text{-}G2}$, sum of all scores MAX_{Score}, Table 5, Formula 5). Thereby, MAX_{Score} can be reached if an adjacent gene pair is present in each EP (0.98 to 0.05) filter, present within the GG filter (GG_Y) and present within the DGD filter (DGD_Y). With the initial weightings, this results in MAX_{Score} of 2.7. Subsequently, the $TS_{G1\text{-}G2}$ is divided by MAX_{Score}. If the observed score is greater than a pre-defined cut-off (default 0.6), the two gene pairs are considered as OP.

2.6.6. Filter and operon prediction classifiers benchmark

OpPipe has been used to map RNA-Seq data under control conditions from *Anabaena* sp. PCC 7120, *Bacillus subtilis, Escherichia coli* K-12*, Synechococcus elongatus* and *Synechocystis* sp. PCC 6803, as well as RNA-Seq data under -Fe and -Nit conditions from *Anabaena* sp. PCC 7120 (Table S4). Subsequently, for these datasets all filters were calculated, and operons were identified using the plain predictor model. In a first step, the operons from the literature set (2.6.1) in *Anabaena* sp. PCC 7120 where compared under different EP stringencies. Further, the *nir*, *fraC* and *pec* operon from *Anabaena* sp. PCC 7120 have been modeled under the different EP stringencies and the decay of the TUs has been observed. Further, these three operons were used to monitor differences within the number of (normalized) reads connecting them under the GG filter. In a first attempt, the counts were normalized [156] to one million reads by dividing each counting position with the summed size and then multiplying it with one million. Additionally, library size normalization was conducted using a scaling factor (Table S7, SF). Therefore, the maximum of the summed counts of all used conditions was identified (Table 7, Formula 5). Then, each of the sums was divided with the max leading to a scaling factor. This scaling factor was then multiplied with each count position. Following, the filters were evaluated based on the performance on the literature set (Table S5) using accuracy, precision, sensitivity, specificity and f1-score (Table 4). In a first approach, GG and DGD were evaluated using the OP-set and NP-set1 (2.6.1) and subsequently, DGD was evaluated using OP-set and NP-set2 (2.6.1). In addition, the GG graph was also compared on SGS and TGS data from *Escherichia coli* K-12 (Table S4) basing on the Op-set and the Np-set1 (2.6.1).

Further, an overall comparison was done with operons that can be found in the same composition among different predictors (OpPipe plain predictor, DOOR, ProOpDB, RegulonDB, Rockhopper) leading to an overlap of operons being identified either by one or up to all predictors. In addition, in *Anabaena* sp. PCC 7120, a set of operons has been created which have only been identified by OpPipe (exclusive set) but not by the other predictors (DOOR, ProOpDB, Rockhopper). Subsequently, it has been compared how many of the operons from *Anabaena* sp. PCC 7120 and *Escherichia coli* collected in the literature have been identified by the different predictors (separated into correctly predicted, elongated predicted, partially predicted, not predicted). Further a scoring system was applied to compare the predictors for the predicted operons from *Escherichia coli* and *Anabaena* sp. PCC 7120. Thereby a correctly predicted operon was multiplied with 1, elongated and partially predicted operons with 0.5 and not predicted operons with -1 (basing on OP literature set 2.6.1). The resulting scores were summed up and divided by the total number of operons leading to a predictor-specific score for each species that can be compared.

2.6.7. TSS benchmark of predicted operons

For the identification of TSS within the genome *Anabaena* sp. PCC 7120, a dataset of experimentally identified TSS was used (2.3.2, [56]). Firstly, a whole genome search for these TSS was conducted using different java scripts. It was evaluated where in the genome the TSS are located based on the GFF annotation (before genes, inside genes, between genes). Therefore, if a TSS is in a maximum distance of 200 nucleotides before the start codon (comparable like proposed in [56]), it is counted as “before genes”. If a TSS cannot be assigned to these two groups, it is counted as “between genes”. To sort out low abundant TSS, abundance distribution of TSS regions is calculated over the whole genome leading to the result that a TSS is only counted if it has been hit by at least 50 (or 100, respectively) reads. Subsequently, the predicted operons in *Anabaena* sp. PCC 7120 of DOOR, the OpPipe plain predictor, ProOpDB, Rockhopper, the exclusive set (2.6.6) and the literature set (2.6.1) were used to compare the composition for these sets regarding TSS distribution. For the two cut-offs of 50 and 100, the percentage of operons was evaluated that exhibit a TSS before the operon, within the operon or which do not exhibit a TSS.

2.6.8. Implementation of different classifiers and comparison

For a first inference, the input dataset harboring the filters EP, GG and DGD (2.6.4, output format) were used to create different machine learning models. Therefore, the defined literature set (2.6.1, OP-set, NP-set2) was used for training and testing the predictors. On a first step, a random forest classifier (RF) was created with scikit-learn (2.2.19.6) using a grid search and a five time 5-fold cross validation. The best scoring model was stored for each fold. Within the input features, an importance analysis was conducted and plotted. Further, a

support vector machine (SVM) was created using also scikit-learn (2.2.19.6) with a grid search and a five time 5-fold cross validation. The best scoring model was again stored for each fold. Both the RF and SVM classifier were created using eclipse (2.2.6) in combination with PyDev (2.2.5.3). Subsequently, a neural network (NN) was implemented using TensorFlow (2.2.19.8) and Keras (2.2.19.1). The network was created with Jupyter Lab (2.2.3.1) and is comprised of 16 inputs (EP, GG and DGD filters), two hidden layers with five nodes each, and one binary output. The activation was set to sigmoid. To find the hyperparameters like training epochs, dropout and learning rate, a grid search was performed using scikit-learn grid search. These results were used to train the neural network. With matplotlib (2.2.19.2) and seaborn (2.2.19.7), training accuracy, validation accuracy, training loss and validation loss were plotted. Consequently, the statistical measures (accuracy, precision, recall, f1-score, ROC, AUC) of RF, SVM and NN predictor were compared. PyInstaller (2.2.19.5) was afterwards used to bundle the SVM and RF into an application to make it executable without a python interpreter, while the NN was integrated into OpPipe using Deeplearning4J (2.2.8.1) and ND4J (2.2.8.4). RF, SVM and NN were then used to predict operons based on *Anabaena* sp. PCC 7120 control conditions.

Additionally, an overall comparison was done with operons that can be found in the same composition (i) among different predictors (RF, SVM, NN, DOOR, ProOpDB, Rockhopper), (ii) among different OpPipe predictors (plain predictor, RF, SVM and NN) leading to a conserved overlap set of 556 operons, (iii) conserved overlap set of plain predictor, RF, SVM and RF among different predictors (DOOR, ProOpDB, Rockhopper). For the comparisons, the operons of the RF, SVM and NN training sets were excluded. Finally, the minimum, maximum and average length (number of gene) of all operons, as well as the percentage of operons containing less than five or more than four genes of the predicted operons in *Escherichia coli* K-12 and *Anabaena* sp. PCC 7120 were compared between the different predictors (DOOR, OpPipe plain predictors, ProOpDB, RegulonDB, Rockhopper, RF-filter, SVM and NN). Also, the percentage of genes being assigned into operon in these two species was compared between the different predictors.

2.6.9. Implementation of visual output

To track changes within the composition of predicted operons (e.g. due to environmental changes), the visualizer of operon prediction files was implemented using different java scripts. These scripts parse out the genomic information out of GFF files and the predicted operons of the operon files. Subsequently, a list of adjacent gene pairs is created with a color code indicating if they are part of the same operon. Thereby, on a first step, an HTML file is created providing different genomic information about two adjacent genes like start position, stop position, intergenic distance, and prediction score of the gene pair. If different operon prediction

files are provided, a second HTML output is created, listing the predicted operons along each other (with regard to the provided reference), again with color coding to track the changes within the different predictions. Finally, the HTML outputs are converted to SVG and PNG images using the WebVector library (2.2.23).

2.6.10. Implementation of a pipeline prototype for the operon prediction

The implementation of the OpPipe prototype was conducted in java using different maven projects within Eclipse (2.2.5). The projects were staged with git (2.2.6). First, the functionality of OpPipe was implemented as a single executing framework. For this purpose, different sub packages were created, each with the ability to act independently, to ensure different tasks of OpPipe. In general, OpPipe is separated in eight different modules handling GUI, pipeline, evaluation of genomic feature files as well as mapping of RNA-Seq data and filter creation, prediction, visualization, storage and helper tools part (Figure S1). The package “userInterface” harbors all relevant features for the creation of the different GUIs, namely the GUIs itself and the controlling and data storing parts of the GUIs. Additionally, package “pipeline” manages the execution of OpPipe. The storage part is facilitated by the package “gffTools” which serializes and deserializes provided genomic and experiment files. The serialization is beneficial as the execution time can be decreased drastically, if a genome or experiment was loaded into OpPipe previously. The package “readGFF” evaluates a given GFF file. Once a GFF file is evaluated, it is then serialized using kryo (2.2.8.3) onto a given home repository using the package “gffTools”. For each experiment using this specific GFF, it has not to be evaluated a second time and can be easily deserialized (using kryo). Further, the package “filterCreation” on the one hand evaluates genomic features (intergenic distance). On the other hand, it maps given RNA-Seq experiments onto a given reference (provided in FASTA format) or reads already mapped files (SAM). Subsequently, the filters are created based on the data. Once an experiment is read, the filter relevant features are also serialized and can be reused. The package “predictionModels” harbors different predictive algorithms (plain predictor, neural net classifier, support vector machine, random forest) and applies the given predictor onto the filter data. After the operon prediction, the package “visualization” provides the predicted operons as a visual output also in HTML format. Finally, the package “helperTools” provides different additional methods for e.g. comparing different predicted results against each other. For each functionality, different Junit tests (2.2.8.2) were created and executed. The different GUIs were thereby developed using the SpringFramework (2.2.8.6) and Thymeleaf (2.2.8.7). The different packages and the containing classes and methods thereby should follow in a maximum possible way the principle of single responsibility, understandability and maintainability. In this first step, OpPipe was designed as a prototype of the desired pipeline and can be seen as minimum viable product (MVP).

3. RESULTS

The developed framework for operon prediction called OpPipe during this study combines RNA-Seq data and genomic feature data to increase accuracy of the prediction. OpPipe allows operon prediction even for non-model organisms with only expression or genomic feature data.

3.1. Development of operon prediction pipeline prototype based on expression and general genome data

The prototype of OpPipe is implemented using Java (2.2.8), offering graphical user interface (GUI) written in HTML. In general, the OpPipe is separated into four different GUIs, each with an independent task. While the application is controlled with the initial GUI of OpPipe (Figure 4A), further GUIs exist for the loading of a GFF file (Figure 4B), loading of RNA-Seq experiments (Figure 4C) and for the visualization (Figure S2).

The initial OpPipe GUI is clearly separated into three parts (Figure 4A). The first part (Figure 4A, left control element) consists of the “Genome list” and the “NGS list”. Via the control buttons “Add genome” and “Add experiment”, the user can add a new species or experiment, which is then displayed in the referring dropdown menus above the add buttons. When clicking “Add genome”, the user is redirected to another GUI for loading a GFF file (Figure 4B). For loading a new genome, the user has to provide a name (Figure 4B) and then select the desired GFF file via “Load GFF file”. Subsequently, the GFF file is validated and if needed separated. Finally, the new genome is serialized onto the hard drive and therefore does not have to be read again. The user is re-directed back to the initial OpPipe GUI where the loaded genome is then displayed within the initial GUI in the genome dropdown (Figure 4A) and can be used for every experiment.

The loading procedure of experimental data follows similar steps. For adding a new experiment, the user has to click “Add experiment” and is then redirected to the GUI for loading an experiment (Figure 4C). For loading a new experiment, the user can simply provide a SAM file (Figure 4C). Additionally, it is also possible to start from scratch and provide FASTQ data (single and paired end) and start the mapping with the desired aligner and FASTA genome reference. When the user clicks onto “Submit” the selected choices are then stored (including the paths to the experiment files) internally, and the user is redirected to the initial OpPipe GUI. The experiment is added to the experiment list (dropdown above “Add experiment”). The user is able to delete the loaded GFFs and experiments at any time by clicking “Remove genome” and “Remove experiment”. Further, details for the provided experiment are accessible, providing information about the experiment input type (SAM or FASTQ), library type (single and paired end) and selected aligner.

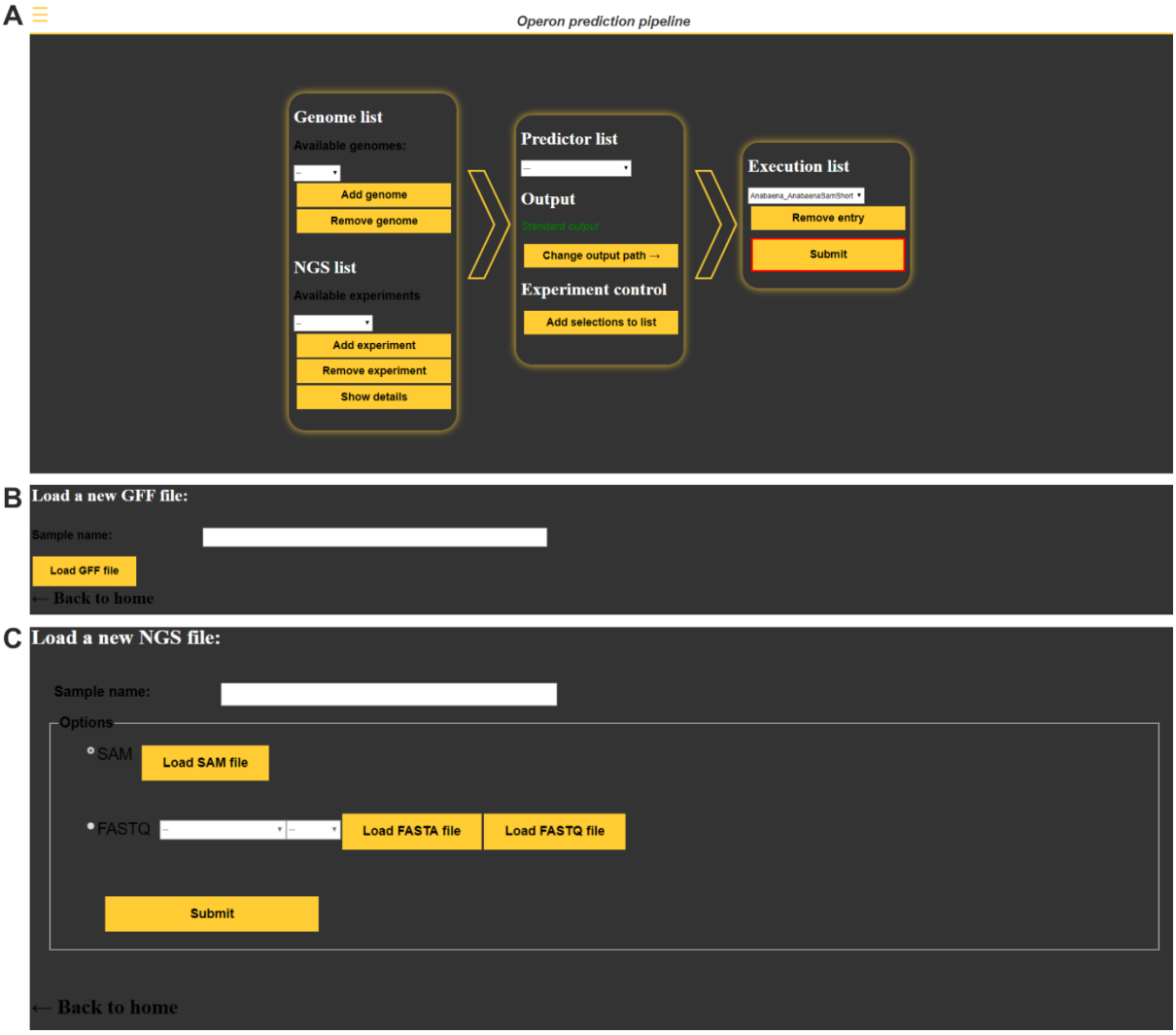


Figure 4: GUIs of OpPipe. (A) The user chooses the species and experiment of interest (left control element), then selects a prediction model (center control element) and starts the pipeline (right control element). (B) GUI for loading a GFF file, with the text field for providing a name and a submit button to start the loading process. (C) GUI for loading an experiment, whereby plain SAM files can be loaded, as well as FASTQ files.

The center control element (Figure 4A), offers the functionality of creating a run. Therefore, first a predictive model has to be chosen from the combo box in the centered control element, where the choice is between a plain predictor model, RF, SVM or NN. Further, a custom output path (for operon results graphically and text format) can be set manually, otherwise the output path will be a standard output path.

For defining a run, the user first has to select a reference genome and an experiment in the left control element, followed by the prediction model. With the option “Add selections to list”, the selected choices are added to the “Execution list” within the right control element. As entry, the “Execution list” contains the chosen species, the chosen experiment and chosen predictor. Internally, the predictor and all other required paths are stored for each entry of the “Execution list”. After adding the experiment, the pipeline is not started, offering the user the possibility of adding numerous experiments to the “Execution list” (e.g. different stress conditions for a

provided genome). If all experiments are added to the “Execution list”, the user clicks “Submit” within the right control element (Figure 4A) and all experiments in the list will be executed. Therefore, the serialized genome data is deserialized for each experiment and the experimental data is applied onto the genome. In different sub-steps, data structures from OpPipe are also getting serialized for following modules. If the run is finished, operon predictions are provided at the specified output path in .txt format.

Beside the plain text output, OpPipe also offers a visualization. The user can access the visual output by moving to the visualizer GUI (Figure S2), which can be achieved by clicking the burger menu in the left upper corner of the initial GUI (Figure 4A) and click onto “OpPipe – visualization”. The visualizer thereby offers the possibility of a “Plain visualization” and a “Compare visualization” (Figure S2). Within the plain variant, the user is able to choose all conducted experiments from the dropdown in the left control element. With clicking “Show visualization”, the graphical output is loaded below. The compare variant offers the user the possibility of comparing different experiments (e.g. different stress conditions). Therefore, different experiments can be added to the compare list by selecting them via the dropdown and clicking “+add to compare list”. For starting the comparison, one experiment has to serve as reference. When clicking “Show visualization”, the comparison is displayed below. The graphical representation of plain and compare visualizer thereby follows the same color code (Figure 5).

Figure 5: Visual output of OpPipe. The visualizer produces an output for each operon file (A) and a comparison of different predictors on the same dataset (B) from left to right.

On the plain visualization, OpPipe provides information of a specific operon (Figure 5A). Listed are the adjacent genes of the operon (green color indicates being part of the same operon)

and different information like genomic position, strand intergenic distance and operon score. For a specific prediction, all predicted operons are visualized. For the comparison of different experiments, the different experiments are placed next to each other, while the reference experiment (e.g. control conditions or a specific predictor) is placed on the left (Figure 5B). As a comparison, the output of other predictors can be used, as well as other environmental conditions (e.g. -Nit, -Fe). All operons from the reference experiment are thereby displayed and the other inputs are compared to the reference. Comparing different prediction runs with each other, red color displays that a gene is not part of an operon in the other condition or predictor, while a green color indicates being part of an operon (Figure 5B). For example, the genes phhB-alr4554 are clustered into operons within the reference prediction and by Predictor 3. However, while Predictor 1 is assigning all genes into one operon, Predictor 3 is separating the gene into two operons (obtainable by different green shades). Contrastingly, Predictor 1, Predictor 2 and Predictor 4 are only assigning a sub-set of the genes into an operon.

3.2. Mapping and counting of RNA-Seq data module of OpPipe

As an expression-based prediction pipeline, OpPipe must be able to deal with provided experimental data. A prerequisite for the prediction of operons and TUs via expression data is the counting of reads per genomic position and thereby the creation of an expression profile for defining start and stop positions of a TU. On the one hand, previously mapped RNA-Seq reads can be provided (e.g. SAM format), while unmapped FASTQ data can be provided by the user, which then must be mapped onto a reference before being evaluated.

To allow good mapping quality, three different available mapping algorithms (NGM 2.2.16; BBMap, 2.2.2; Bowtie2, 2.2.3) were compared to an RNA-Seq mapping including operon predictor (Rockhopper 2.2.20) using different RNA-Seq samples (compare 2.3, Table S4). To obtain different quality and sequencing depth, samples from five different prokaryotes namely *Anabaena* sp. PCC 7120 control conditions (paired, about 88 million reads), *Bacillus subtilis* (control conditions, paired end, about 28 million reads), *Escherichia coli* (control conditions, paired end, about 26 million reads), *Synechococcus elongatus* (control conditions, single end, about 15 million reads), *Synechocystis* PCC 6803 (control conditions, paired end, about 16 million reads) were compared (Figure 6).

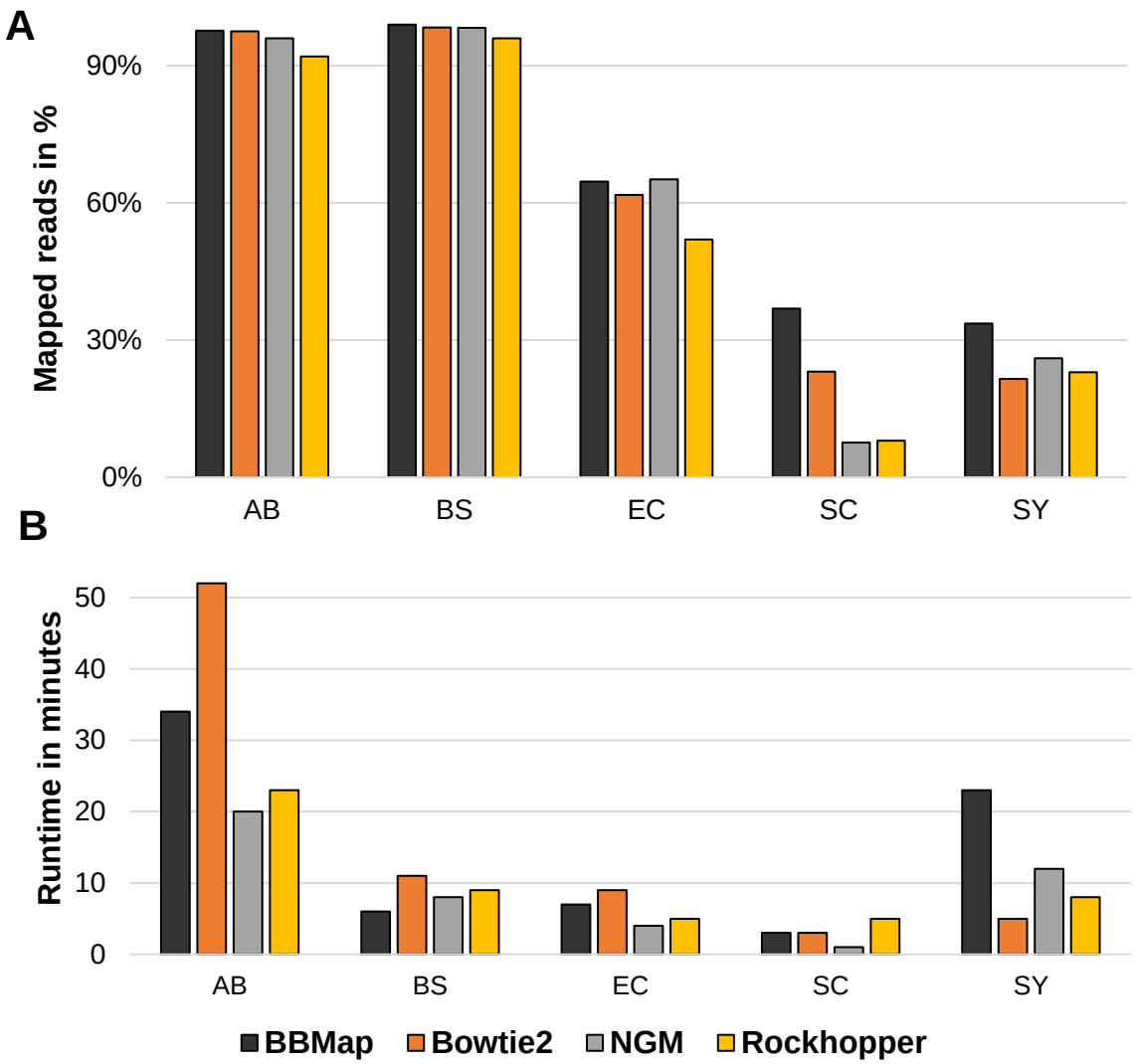


Figure 6: Benchmark of Aligner. Compared are the different aligner BBMap (black), Bowtie2 (orange), NGM (grey) and Rockhopper (yellow) based on the control conditions for different species. For each aligning algorithm *Anabaena* sp. PCC 7120 (AB), *Bacillus subtilis* (BS), *Escherichia coli* (EC), *Synechococcus elongatus* (SC) and *Synechocystis* PCC 6803 (SY) aligned reads as well as the time needed for the mapping is indicated.

In general, NGM exhibits the shortest execution time for all data samples compared to the other mapping algorithms with a maximum of 20 minutes for 88 million reads (Figure 6B, *Anabaena* sp. PCC 7120). While Rockhopper achieved comparable runtimes to NGM, BBMap and Bowtie2 showed a longer execution time for nearly all samples (exception Bowtie2 for *Synechococcus elongatus,* Figure 6B).

For the number of mapped reads, it could be observed that Rockhopper showed the least percentage of mapped reads compared to the other tools in all different datasets. For the species *Anabaena* sp. PCC 7120, *Bacillus subtilis* and *Escherichia coli,* the algorithms BBMap, NGM and Bowtie2 had comparable mapping results (Figure 6A). Interestingly, for *Synechococcus elongatus,* NGM and Rockhopper are only able to align seven to eight percent of the reads onto the reference, while BBMap and Bowtie2 can map ~37% and ~23% onto the

reference. Although BBMap needed six times longer to align the reads of *Synechocystis PCC 6803* than the fastest algorithm (NGM), it resulted in 10% more mapped reads.

Irrespective of an already submitted SAM file or a raw read file in FASTQ format, the next step of OpPipe is the assignment of reads and coverage to the annotated genes in a given GFF file. Therefore, the features of the mapped reads have to be evaluated. In the first step, numerous different SAM flags could be assigned to the different reads (Table S5) and have to be considered. OpPipe thereby only considers (i) reads for which only one mate could be properly assigned to a strand (only the region of this mate was counted), (ii) reads that are mapped as pairs onto different strands. In case of reads where both mates could be mapped on opposite strands, the separation was made into pairs where the mates are considered as mapped in a proper or not in a proper pair. Data from control conditions in *Anabaena* sp. PPC 7120 (mapped by NGM) thereby reveals that only reads mapped in a proper pair should be used for the counting algorithm of OpPipe (Figure 7).

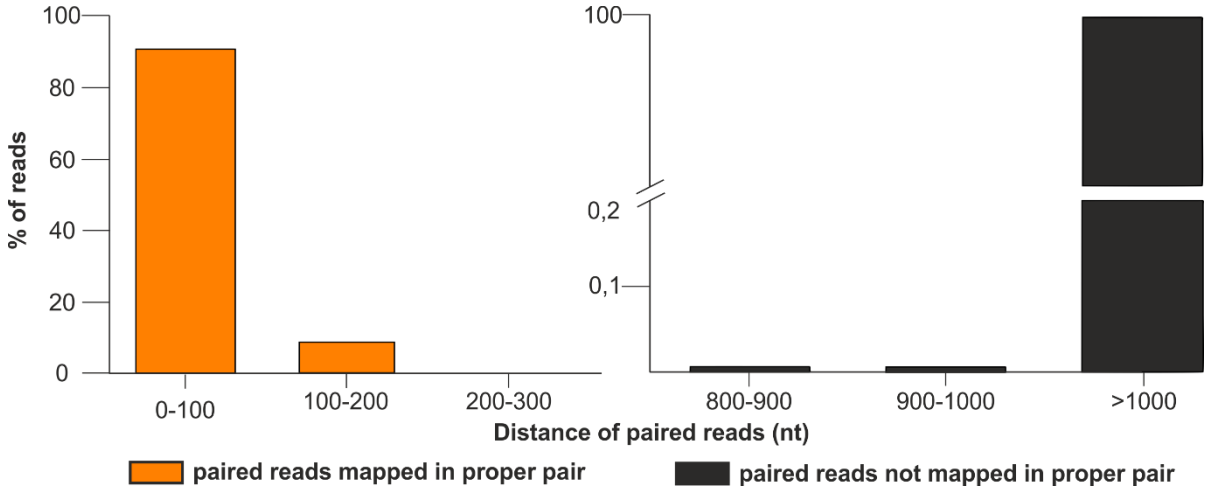


Figure 7: Distance of proper read pairs and non-proper read pairs in *Anabaena* sp. PCC 7120. Indicated are the percentages of all pairwise mapped reads, evaluating the distance of the two read pairs (in nucleotides) for reads mapped in a proper pair (orange) and reads not mapped in a proper pair (dark grey).

It is observable, that reads which are mapped in a proper pair exhibit a maximum distance of 300 nucleotides within the dataset of *Anabaena* sp. PCC 7120 (Figure 7). Further, the amount of these reads is less than 5%. Subsequently, at least 15% of properly mapped read pairs exhibit a nucleotide distance of 100-200nt, while the majority exhibits 0-100 nts (nearly 90%). In contrast, most of the reads not mapped in a proper pair exhibit an inner-mate distance of more than 1000. Even though a few amounts of the reads have an inner distance of 800 to 1000 nucleotides (summed together fewer than 0.1%), more than 99% of the reads in the sample exhibit the distance of 1000 nucleotides. As for prokaryotes splicing is typically not occurring, the distance of paired reads should only be of the size as the inner mate distance (typically 50-300bps). Therefore, only such reads paired properly were considered, while reads

of an improper pair were sorted out. In comparison to a read assignment tool like HTSeq, the choice of used SAM flags of the mapping part of the pipeline (OpPipe-count) revealed differences (Figure 8).

A

	97/145	99/147	105/149	153/101	155/103	number of reads
Reads in file	5+5	82+82	3+3	7	3	100
HTSeq-count	5	82	3	7	3	100
OpPipe-count	0	82	3	7	3	95

B

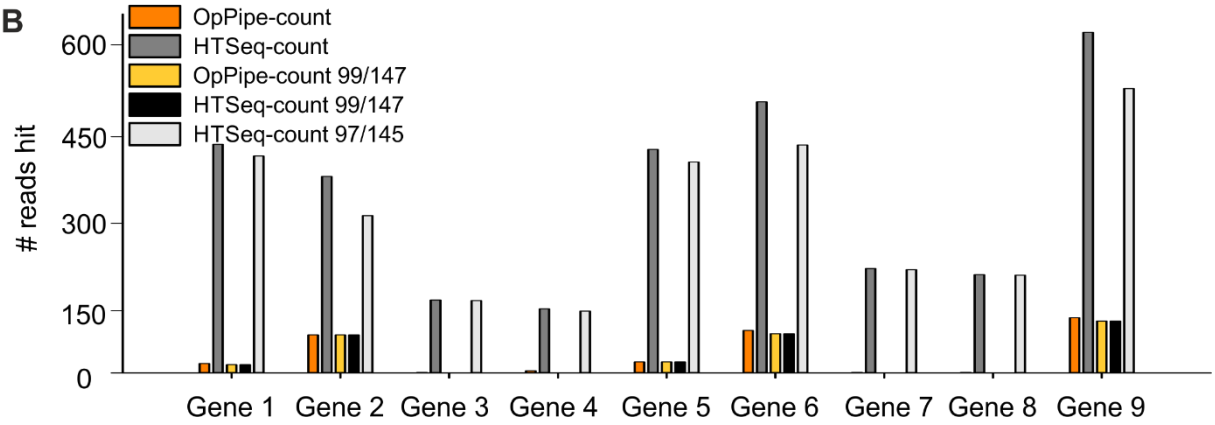


Figure 8: Counted reads of HTSeq-count and OpPipe-count. Indicated are counted reads where both read pairs could be mapped (97/145= mapped in non-proper pair, 99/147 = mapped in proper pair), reads where only one read pair could be mapped (105/149, 153/101, 155/103) and the number of all counted reads (A). Further nine genes from *Anabaena* sp. PCC 7120 and the number of paired reads that are counted (total counts: OpPipe-count = orange, counted by HTSeq-count = dark grey, counted with SAM flag 99/147: OpPipe = yellow, HTSeq-count = black and counted reads with SAM flags 97/145 by HTSeq-count = light grey).

Evaluating 100 read pairs with specific SAM flags (97/145: reads mapped in proper pair, 99/147: reads mapped in non-proper pair, paired reads where only one read pair was mapped) reveals different counting approaches between the counting methods (Figure 8A). Comparing the two approaches, HTSeq-count and OpPipe-count, it is observable that they assign the same number of counted reads to reads being mapped properly (99/147). Further, also for reads where only one read pair could be mapped (Figure 8A, 105/149, 153/101, 15/103), the number of counted reads remains the same for HT-Seq count and OpPipe-count. In addition, OpPipe-count is not counting reads with 97/147 (not mapped in proper pair), while they are counted by HTSeq-count. The genome-wide comparison in *Anabaena* sp. PCC 7120 of OpPipe-count and HTSeq-count based on read counts of nine randomly drawn genes of the control sample from *Anabaena* sp. PCC 7120 show the same behavior (Figure 8B). In general, for all genes, the number of counted reads for HTSeq-count is multiple times higher than for the OpPipe-count. This can be explained with the counting of non-proper paired reads (or ignorance of them, respectively). However, when only counting the proper paired reads (99/147, Figure 8B), the number of counted reads for HTSeq-count and OpPipe-count is equal.

For example, for Gene 1 the number of counted reads for HTSeq-count is about 400, the Framework-count for Gene 1 is lower than 50. However, when comparing the count for the SAM flags 99/147 (reads mapped in a proper pair), it is observable that the counted reads for HTSeq count and OpPipe-count are always identical. Further, when comparing the different counters of HTSeq-count, it is observable that the counts of 99/147 and 97/145 sum up to the total count of HTSeq-count.

3.3. Filter creation based on expression data for operon detection with OpPipe

For the creation of input features for the operon prediction models expression data has been used to create different generalization models called filters. The expression data has been used to create the expression pattern filter (EP, Figure 9) and the gene graph filter (GG, Figure 10). As a prerequisite for the EP filter the counts of mapped reads per genomic position is needed (expression profile), which is delivered by OpPipe-count. Further, for the GG filter a list of pairwise genes is required, with the number of reads indicated that map on both genes and thereby connecting them. As a starting point, genes of a TU (or operon) are assumed to be co-expressed. The EP filter is thereby able to identify TUs without any annotation file information and is only based on the expression landscape of the Watson and Crick strand (Figure 9).

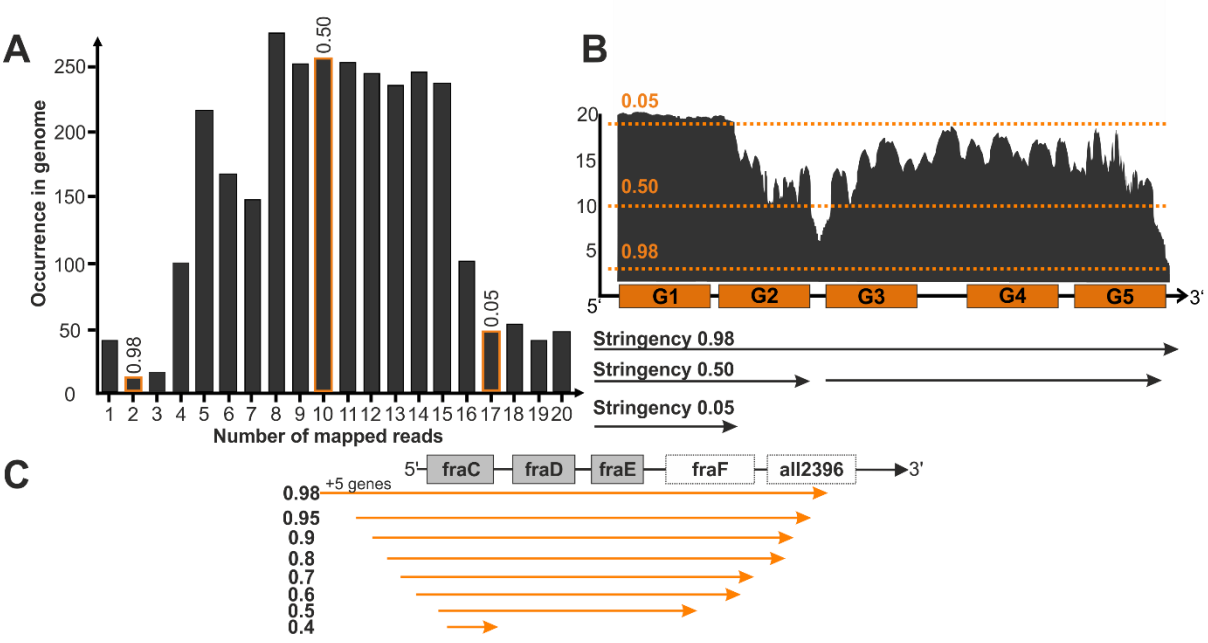


Figure 9: Implementation of EP filter. The different stringencies lead to different cut-off strictness (A), which indicates that at least x% of all genomic positions are covered by at least n reads. The cut-off influences the size and amount of predicted TUs (B). A less stringent cut-off (stringency 0.98) leads to longer TUs while a stricter one (stringency 0.05) leads to shorter TUs. The decay of the *fraC* operon (fraC-fraE, [157], C) of *Anabaena* sp. PCC 7120 for the different stringencies under control conditions can be obtained for different cut-offs while the orange arrows show the length of the TUs. Indicated are the adjacent genes of the operon (grey box) and the genes of the opposite strand (white box).

As a starting point, the EP filter loops over the expression landscape (expression profile of Watson or Crick strand) and thereby determines a cut-off which indicates that at least x% of all genomic positions are covered by a specific number (n) of reads for different stringencies (2.6.4, Figure 9).

In theory, transcribed genomic positions (e.g. operons and TUs) are thought to exhibit a higher read count with an observable increase of counted reads within the expression profile compared to intergenic regions without transcriptional activity. Consequently, it is assumed that the majority of the number of mapped reads onto genomic positions is located around a specific average, while few regions with a high number (highly expressed regions) or a low number (poorly expressed regions) of mapped reads can be obtained (Figure 9A) occur less frequently. Applying cut-offs with a different amount of stringency thereby should be able to identify the boundaries of the transcribed positions within a condition.

In a schematically representation this leads to a sparse amount of genomic positions being hit by one up to four reads. Further, a sparse amount of genomic positions is hit by more than sixteen reads. Many genomic positions are hit by five to fifteen reads (Figure 9A). A moderate cut-off of 0.98 thereby indicates that at least 98% of all genomic positions are covered by at least two reads (Figure 9A). If a genomic position is covered by more than two reads under this stringency, it is considered as a TU start (or TU stop, respectively, if a position is covered by fewer than two reads) which leads then to an elongated TU of five genes (Figure 9B). Applying a cut-off of 0.5 (meaning that 50% of all genomic positions by a specific position, Figure 9A) results to the separation of the TU into two smaller TUs (G1-G2, G3-G4-G5, Figure 9B) compared to the more moderate stringency of 0.98. The expression profile thereby shows a decrease of reads mapped between G2 and G3 which is now below the cut-off defined by stringency 0.50 (Figure 9B). Subsequently, applying a stringency of 0.05 leads to a strict cut-off of 17 reads per genomic position (Figure 9A). The application of this cut-off leads to a small TU which is highly expressed (Figure 9B). This leads to the assumption that if two genes are assigned to the same TU under a strict stringency, the probability of being co-expressed is increased compared to two genes that are only considered as part of the same TU under moderate stringencies.

The EP filter with the aim to indicate the decay of a TU (giving a hint that genes are co-expressed under a strict cut-off) shows that most of the operons from *Anabaena* sp. PCC 7120 (based on literature set, 2.6.1) are elongated under the 0.98 stringency (Table S8), while the number of full hit operons increases with the strictness of the stringency until 0.6. With the start of stringency 0.4, the number of elongated operons starts to decrease, while the number of operons not found increases, leaving only highly expressed operons.

The different sample operons *fraC* operon (Figure 9C), the *nir* operon (Figure S3) and the *pec* operon show different behaviors for the different stringencies of the EP filter under control conditions. The *fraC* operon shows a decay under stricter stringencies (Figure 9C). According to Merino-Puerto *et al.* [157], it consists of the three fra genes (C-E). Downstream the fraE the fraF gene is located on the opposite strand. This gene is known to be part of the operon as antisense transcript [158]. For the most moderate stringency (0.98), the TU consists of the *fraC* operon and five genes upstream (Figure 12C). Furthermore, the TU covers the sequence which, on the opposite strand, has two genes leading to an antisense transcription. However, the TU with 5 genes upstream of the *fraC* operon does not seem to be conserved, as for the next cutoff (0.95) it is not present anymore. Although the TU gets shorter for each stringency, the antisense genes stay a part of the TU for the stringencies of 0.95 to 0.5. For the stringency 0.4, only fraC gene of the operon remains, indicating a higher expression of the fraC compared to the other genes of the operons. For the stricter stringencies, the operon is not abundant anymore under control conditions. A similar behavior can be obtained when comparing the results of the EP filter based on the *fraC* operon to the EP filter based on the *nir* operon. Like mentioned by Frías *et al.* [159] [160], the *nir* operon consists of the adjacent genes nirA, nrtA, nrtB, nrtC, nrtD, narB, followed by the two genes alr0613 and alr0614, which might also be part of this gene cluster. The *nir* operon exhibits for the most moderate stringency (0.98) of all eight genes. This is also the case for the stringencies of 0.95 and 0.9, although the position of the start and stop numbers are changing. With stricter stringency, the operon starts to decay: For the stringency of 0.8, the operon starts to decay into three subunits. The stricter the stringency gets, the smaller the units become, until they are not present anymore after a stringency of 0.3. Contrastingly to the decay of *nir* and *fraC* operon is the behavior of the *pec* operon under control conditions. According to Swanson *et al.* [161], this operon consists of the adjacent genes pecB, A, C and E. The composition of this operon remains the same under each stringency (0.98 to 0.05) although the start and stops of the TUs change (Figure S3). Even though the EP filter might be enough for analyzing TUs in different conditions and allows conclusions of the conservation of operons under specific conditions, the peculiarity of the cut-off detection might lead to a preference of highly expressed genes and a discrimination of lower expressed genes. However, a TU can be identified even without information of gene composition, as a start and end position of a TU can be identified by only considering the expression profile.

For this purpose, the design of the GG filter (Figure 10) tries to take the assumption into account that operonic genes are co-transcribed in one single mRNA. By this, paired end reads, or long reads, should cover parts of adjacent genes in one mRNA.

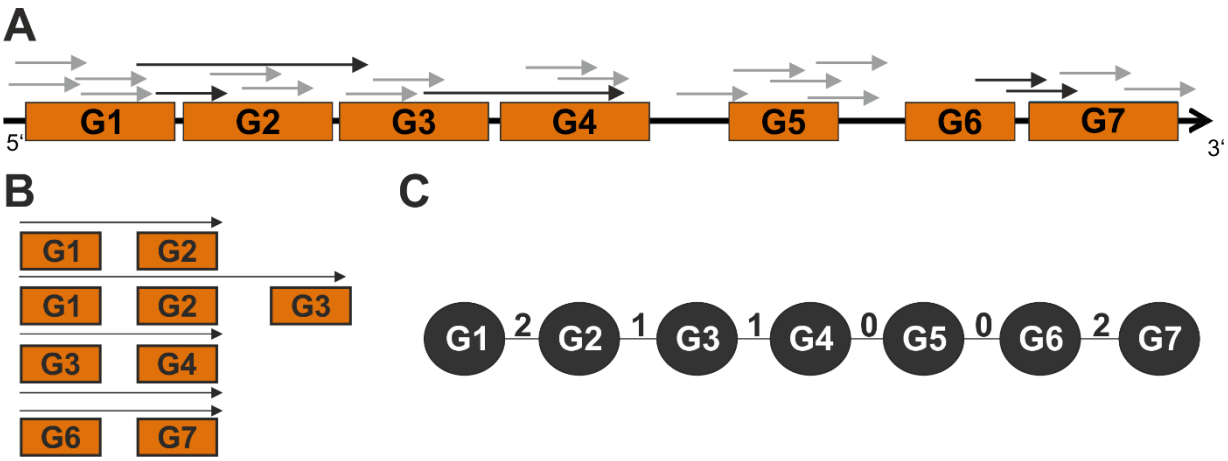


Figure 10: Implementation of GG filter. Differences in the expression (A) leading to reads covering only one specific gene (grey arrow) and genes covering more than one gene (black arrow). Genes covering more than one gene are identified (B) and represented as weighted edges between two gene nodes (C).

To reflect this assumption, the GG filter identifies adjacent genes that have both been hit by the same reads or templates (Figure 10A-B). The GG filter constructs a graph with the nodes being the genes of a species with edges connecting them (Figure 10C), which are initially weighted with zero. Per hit read (or template, respectively), the edge between them is increased by one (Figure 10C). Exemplarily, G1 and G2 are connected by two reads (Figure 10A-B). One read covers G1 and G2, while another one covers G1, G2 and G3, which leads also to the connection of G2 and G3 by one read. While G3 and G4 are again connected by one read, no read is connecting G4 and G5, as well as G5 and G6. Finally, G6 and G7 are connected by two reads. These connections are then assigned to the edges between the referring genes within the created gene graph (Figure 10C).

As the concept of the GG filter relies on reads overlapping more than one gene, it is best suited for TGS data, as TGS produces longer reads that should cover more genes. Therefore, the identification of OPs and NPs through the GG filter based on SGS and TGS from *Escherichia coli* under control conditions (Table S4) has been compared basing on the operons from the literature set (2.6.1) from *Escherichia coli,* showing that TGS data could increase the power of the GG filter (Table 8).

Table 8: Comparison of SGS and TGS data for *Escherichia coli.* Indicated are prediction of OPs and NPs where SGS and TGS lead to the same result and where not. Further, different statistical measures were calculated.

	SGS		TGS	
	gene-pairs	**% of gene-pairs**	gene-pairs	**% of gene-pairs**
True positive	49	**72%**	59	**87%**
False positive	19	**28%**	9	**13%**
True negative	61	**95%**	60	**94%**
False negatives	3	**5%**	4	**6%**
SGS = TGS		**85%**		
SGS! = TGS		**15%**		

Comparing the assigned labels of the gene pairs from *Escherichia coli* from the OP set and NP-set1 (2.6.1) shows that in 85% of the cases, SGS assignments based on the GG filter were identical to the GG filter of TGS data. However, for 15% of the gene pairs, they were not identical. For TGS data, an increased accuracy can be obtained as the TP rate for SGS data is 72%, while for TGS data 87% could be reached. Further, the false positive rate is decreased in TGS (13%) compared to SGS (28%). Regarding the true negative rate (SGS = 95%, TGS = 94%) and the false negative rates (SGS = 5%, TGS = 6%), the two sequencing approaches are comparable.

Evaluating the GG filter on the full OP set and full NP-set1 (2.6.1), it is observable that about 85% are correctly assigned as OP by the GG filter (TP, Figure S4), while about 15% of the OPs are falsely set to NP (FN, Figure S4). However, the rate of falsely predicted OPs is low (FP, Figure S4), as only about 5% of the NPs have been classified as OPs. Consequently, about 95% of all NPs are getting classified correctly as NP (Figure S4). Although the GG filter shows to be able to identify a high number of OPs under control condition, it seems to be easier to identify NPs with this filter. Applying the GG filter onto the control conditions of *Anabaena* sp. PCC 7120, several operons were identified. Among others, the *pec* operon (pecA-E) has been identified by the GG filter. The genes within the *pec* operon (Table S9A) are connected by many reads under control conditions. The surrounding genes of the operon (ORF1 and pecF) do not exhibit any shared reads with the genes of the operon. Comparing the number of mapped reads per each edge (an edge is connecting two genes), differences for the genes of the *pec* operon can be observed (Table S9A). For example, 2,886 reads were found to be matched onto pecA as well as onto pecC and 4,382 reads connecting pecC and pecE. However, the connection between pecB and pecA is approximately nine times more covered (18,823) than the connection between pecA and pecC. In general, through such increases alternative TUs could be identified thorough the expression profile. An increase of mapped reads after the start of an alternative TU within an operon is thereby assumed. Interestingly, two TSS are located upstream the pecB gene of the *pec* operon. This might lead to the expression of the whole operon from one of the TSS and the second giving birth to an alternative TU only harboring pecB and pecA. Consequently, two TSS can be observed within the pecC gene. Consistent with this, the connection between pecC and pecE is increased compared to the connection of pecA to pecC.

In general, the GG filter lacks the ability to identify TUs or operons without provided genomic information (annotation of genes), while the EP filter is independent of annotation of genes. In general, the expression filters of GG and EP are assumed to monitor a specific “operon behavior” (co-expression). Therefore, they are thought to generalize the provided data in a universal valid operon format. Further, the assumed operon behavior tracks only a general

gene connection (GG filter) and changes in expression strength (EP filter). After this generalization, the data is no longer specific to the species but rather operon-specific. Therefore, it is not biased by the derived species and consequently cross-species applicable.

Nevertheless, all expression filters always rely on the derived experiment and the quality of the sequencing. In contrast, genomic features are conserved irrespective of the condition. Assuming the correct annotation of genes within a genome, the inclusion of the feature of the intergenic distance can help increasing the ability of defining OPs.

3.4. Intergenic distance is not universal usable for all prokaryotic genomes

Even though expression-based filters offer several advantages, they are dependent on the quality RNA-Seq datasets and are condition-specific. To overcome this, genomic features should be considered to make prediction of operons based on known transcribed genes on the chromosome and plasmids. One of the most used genomic features is the intergenic distance between two annotated genes, which showed to be different among different prokaryotes (Figure 11, Figure S5).

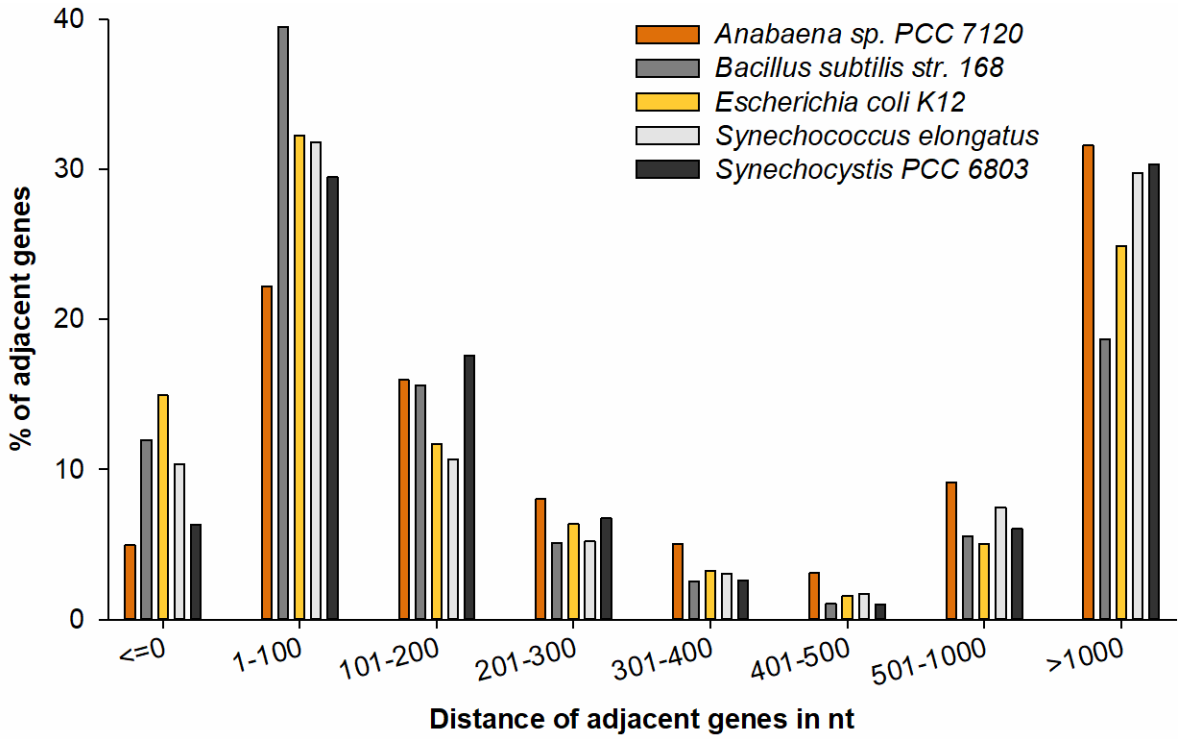


Figure 11: Average distance of adjacent genes of the Watson strand within different genomes. Distribution of the intergenic distance of adjacent genes in *Anabaena* sp. PCC 7120 (orange), *Bacillus subtilis str. 168* (grey), *Escherichia coli* K-12 (yellow), *Synechococcus elongatus* (light grey) and *Synechocystis* PCC 6803 (dark grey) into different distance ranges (in nt).

In general, the adjacent genes from the Watson and Crick strands within one species do not exhibit great differences regarding their intergenic distances (Figure S5). For all five reviewed

species, the adjacent genes assigned to a distance range are approximately of the same size for the Watson and Crick strand (Figure S5). Considering only the Watson strand, for all five species, most of the adjacent genes exhibit either a distance which is lower than 300 nt or greater than 1000 nt (Figure 11). Ignoring the genes with a distance greater than 1000, most of all adjacent genes in every species exhibit a distance of 1 to 100 nucleotides (Figure 11). For *Bacillus subtilis, Escherichia coli* and *Synechococcus elongatus,* more than 30% of all genes are located in this range, while for *Anabaena* sp. PCC 7120 and *Synechocystis* PCC 6803, about 25-28% of all genes exhibit this distance range. However, the genomes of *Bacillus subtilis* and *Escherichia coli* seem to exhibit a smaller distance in general, as more than 50% of all genes exhibit an intergenic distance fewer than 200 nts to their neighbor. For *Escherichia coli* 60% of all genes and for *Bacillus subtilis* 66% of all genes exhibit this distance (Figure 11). In contrast, genes from *Anabaena* sp. PCC 7120 seem to be spaced in greater distances as only 51% of all genes exhibit an intergenic distance which is 300 or fewer. Furthermore, the genes of *Bacillus subtilis* and *Escherichia coli* are overlapping to a greater amount (15% and 12%, respectively) than in *Anabaena* sp. PCC 7120 (5%), which is observable within the range of 0 or fewer nucleotides.

As a result, it is difficult to identify OPs in *Anabaena* sp. PCC 7120 based on the intergenic distance of *Escherichia coli*. A static definition of a cut-off and a training on this definition for the determination of OP and NP based on intergenic distance might lead on the one hand to the discrimination of OPs (cut-off to low) or on the other hand to an enhanced false positive rate (cut-off too high). For this purpose, the intergenic distance feature has to be applicable to a given species in a universal way. Consequently, a generalization of the intergenic distance must be applied, on the one hand leaving the specificity of the distance feature but on the other hand being able to be applied in a sensitive way to every species. To join these genomic features into a generalized filter model, the distance of two genes is used to create the dynamic intergenic gene distance (DGD) filter. Therefore, the intergenic distances of adjacent genes (G_1-G_2, in nucleotides) in different genomes are calculated in a dynamic way (Table 9).

Table 9: DGD filter creation for five different species. The intergenic distances (in nt) of all adjacent genes located on the Watson strand are assigned to different distance ranges (first column) for *Bacillus subtilis* (BS), *Escherichia coli* (EC), and the cyanobacteria *Anabaena* sp. PCC 7120 (AB), *Synechococcus* (SC) and *Synechocystis* (SY). Further, the percentage of genes being part of this range is assigned (second column). The sum of percentages (third column) is marked green for the range where 45% of all genes are reached.

Distance (nt)	*BS*		*EC*		*AB*		*SC*		*SY*	
	%	∑ %	%	∑ %	%	∑ %	%	∑ %	%	∑ %
<=0	12	12	15	15	5	5	10	10	6	6
1-100	39	**51**	32	**47**	22	27	32	42	29	36
101-200	16	67	12	59	16	43	11	**53**	18	**53**
201-300	5	72	6	65	8	**51**	5	58	7	60
301-400	3	75	3	69	5	56	3	61	3	63
401-500	1	76	2	70	3	59	2	63	1	64
501-1000	6	81	5	75	9	68	7	70	6	70
>1000	19	100	25	100	32	100	30	100	30	100

Based on Brouwer *et al.* and in Moreno-Hagelsieb *et al.* [162], around 50-60% of the genes are located in operons. To decrease the false positive rate, the mean intergenic distance for 45% of the genes in each species was searched. The five different species showed the reach the mean intergenic distance of 45% of adjacent genes differently (Table 9). While the 45% mean intergenic distance is reached at a distance of 201-300 for *Anabaena* sp. PCC 7120 (cut-off = 300), the mark is reached for *Bacillus subtilis* and *Escherichia coli* at 1-100 (cut off = 100) and for *Synechococcus* and *Synechocystis* at 101-200 (cut-off) (Table 9). As assumed, this leads to a different cut-off for the species (Table 9).

Applying the defined dynamical distances to a defined literature OP set and NP set1 (2.6.1), it is observable that the DGD filter is able to correctly identify OPs and NPs (Figure S4). From the OP set, about 95% are correctly assigned as OP by the DGD filter (TP), while about 5% of the OPs are falsely set to NP (FN). However, the rate of falsely predicted OPs is low (FP), as only about 2% of the NPs have been classified as OPs. Consequently, about 98% of all NPs are getting classified correctly as NP. In general, the DGD filter proves to be able to identify a high number of OPs under control condition and a higher TP and TN rate compared to the expression-based GG filter. However, it also seems to be easier to identify NPs with this filter. Applying the dynamic distance procedure of a specific species to the literature OP set and NP2 set (2.6.1) of this species thereby shows a high ability of identifying OPs and NPs for the five different species (Table 10). Additionally, applying the dynamic distance of e.g. *Escherichia coli* to the dataset of *Anabaena* sp. PCC 7120 (and vice versa) leads to a decrease of the ability of the DGD filter to correctly identify OPs and NPs (Table 10, Table S11).

Table 10: Different statistical measures for the DGD filter. Calculated are the true positive rate (specificity, spec), true negative rate (sen, sensitivity) rate and accuracy (acc). Letters behind the rates indicate the measurement within a species of *Anabaena* sp. PCC 7120 (AB), *Bacillus subtilis* (BS), *Escherichia coli* (EC), *Synechococcus elongatus* (SC) and *Synechocystis* PCC 6803 (SY) when applying different intergenic distance cut-offs of these species (300,AB = column 2, 100,BS + EC = column 3, 200, SC + SY = column) as well as an average distance cut-off of all cyanobacteria (233, CY = column 5) and average distance over all other distance cut-off (180, All = column 6). Green color indicates the statistical measures from application of the cut-off from one species onto the operon set of the same species.

Stat. measure	300 (AB)	100 (BS + EC)	200 (SC + SY)	CY (233)	All (180)
specAB	**100%**	56%	87%	92%	83%
specBS	100%	**91%**	100%	100%	100%
specEC	100%	**100%**	100%	100%	100%
specSC	100%	14%	**100%**	100%	32%
specSY	100%	75%	**100%**	100%	93%
senAB	**97%**	100%	99%	98%	99%
senBS	60%	**91%**	67%	64%	70%
senEC	72%	**98%**	82%	77%	84%
senSC	83%	97%	**95%**	89%	95%
senSY	86%	99%	**99%**	93%	99%
accAB	**98%**	76%	93%	95%	91%
accBS	80%	**91%**	84%	82%	85%
accEC	86%	**99%**	91%	88%	92%
accSC	92%	26%	**97%**	94%	48%
accSY	93%	87%	**99%**	97%	96%

The dynamic distances were calculated for five different species (Table 10), which leads to three different cut-offs. Thereby, the cut-off of *Anabaena* sp. PCC 7120 is the highest (300), while *Synechococcus* and *Synechocystis* exhibit the same cut-off with 200, as well as *Bacillus subtilis* and *Escherichia coli* (both 100). Subsequently, the average distance cut-off of all cyanobacteria (233) and all five species (180) was calculated. The application of the distance cut-off onto the same species thereby leads to an increased performance compared to the application of a distinct cut-off to a foreign gene pair set. For example, for *Escherichia coli,* the measures when the *Escherichia coli* dynamic distance is applied are increased compared to the application of the *Anabaena* sp. PCC 7120 dynamic distance onto the *Escherichia coli* (Table 10, Table S11).

Further, it is observable, that the application of a stricter dynamic distance to a gene pair set leads to an increase of specificity (true negative rate). For example, the application of the *Anabaena* sp. PCC 7120 distance cut-off (300) onto *Anabaena* sp. PCC 7120 leads to a specificity (specAB) of 97% (Table 10, Table S11), while applying a stricter dynamic distance of 100 *(Escherichia coli* and *Bacillus subtilis),* 200 *(Synechococcus* and *Synechocystis),* 233 (cyanobacterial average) and 180 (average) leads to increased specificity values (100%, 99%, 98%, 99%). However, this can be explained with more genes being rejected (labelled as NP)

by a stricter cut-off. In this context, it is observable that for the stricter cut-offs applied to a gene pair set, the values for the sensitivity (true positive rate) decreases (56%, 87%, 92%, 83%) compared to the application of the species cut-off (100%). Consequently, the opposite effect is observable if a less cut-off is applied to a gene pair list. For example, the application of the cut-off 300 *(Anabaena* sp. PCC 7120), 200 *(Synechococcus* and *Synechocystis),* 233 (cyanobacterial average) and 180 (average) leads to an increase of sensitivity (100%) compared to the application of the *Bacillus subtilis* cut off (100, 91%). Consequently, the specificity decreases from 91% for the distance of 100 *(Escherichia coli* and *Bacillus subtilis* distance) compared to the 300 (60%, *Anabaena* sp. PCC 7120), 200 (67%), 233 (64%), 180 (70%) distance cut-offs.

In general, the measurements show that a species-specific dynamic defined distance is necessary, as otherwise the prediction is inaccurate. Further, congruently to the GG and EP filter, the described filter model addresses an assumed operon behavior. Previously generalizing the distance feature thereby leads to a species-unspecific output that can be included better into a predictive model than a static distance cut-off definition.

3.5. Combination of operon filters into plain prediction model of OpPipe

After the definition of different filters based on expression and genomic data, they have to be combined and used for a predictive model. In a first approach the EP, GG and DGD filters were used. They are grouped together into one output (Table S12). Thereby, each gene pair is listed in a column. The rows for each gene pair are labeled with the referring filters. For example, all stringencies for the EP filters are listed ($EP_{0.98}$-$EP_{0.05}$). For GG and DGD filter, it is listed, whether the gene pair was considered as an OP (GG_Y, DGD_Y) within the filter or as NP (GG_N, DGD_N). For each row it is labeled whether the sub-filters can be applied, whereby “1” is used for labeling the application of the sub-filter and “0” if the filter cannot be applied, leading to a binary encoding for each sub-filter. In a schematic representation gene-par G1-G2 is considered as OP by $EP_{0.98}$, GG and DGD, while gene pair G2-G3 is considered as OP by $EP_{0.98}$-$EP_{0.05}$, GG but considered as NP by DGD (Table S12).

Subsequently, the input for the prediction algorithms is a table with the columns indicating the gene pairs, EP filter (0.98 to 0.05 cut-off), GG filter (GG fulfilled & GG not fulfilled) and DGD filter (DGD fulfilled & DGD not fulfilled). Thereby, the filters are divided into $EP_{0.98}$, $EP_{0.95}$, $EP_{0.90}$, $EP_{0.80}$, $EP_{0.70}$, $EP_{0.60}$, $EP_{0.50}$, $EP_{0.40}$, $EP_{0.30}$, $EP_{0.20}$, $EP_{0.10}$, $EP_{0.05}$, GG_Y, GG_N, DGD_Y, DGD_N, leading to 16 input rows for each gene pair. For a first combination of all developed filters, the filters have been combined to a plain predictor model (Figure 12A). For the inputs of the plain predictor, the filters EP, GG and DGD are used. For each adjacent gene pair, it is provided via

the input data if the sub-referring filters can be applied (label 0 and 1). The input labels are then multiplied by a specific weighting (Figure 12) and are summed up for all filters. This total sum is divided by the maximum reachable score. Subsequently, it is evaluated via the decision, whether the summed score of a gene pair is greater than a defined cut-off. The cut-off is thereby representing 60% of the maximum reachable score. If the reached score of the gene pair is greater than the cut-off, the gene pair is labeled as OP or as NP if the score is below the cut-off (Figure 12A). The weighting of each sub-filter thereby depends on the assumed importance of the sub-filter, while the sum of all sub-filters of one filter always adds up to one. With the current weightings (Figure 12B), the maximum reachable score is 2.7 (all EP stringencies applicable, GG_Y and DGD_Y applied), leading to a cut-off of 1.62 that is the minimum that has to be reached by a provided gene pair to be classified as OP.

The EP filter is weighted based on the different stringencies. A moderate stringency should thereby be weighted with a lower score than a strict score. For this purpose, the $EP_{0.98}$ is weighted with 0.05 while the $EP_{0.05}$ is weighted by 0.13 (Figure 12B). The more EP sub-filters can be applied to two genes, the higher their overall EP score will be. However, the two filters GG and DGD (co-expression, distance) are considered to be the major criteria for the OP/NP classification and therefore the classification of OP and NP by these filters should have a bigger influence than the single sub-filters of EP. As the decision of being an OP (by the plain predictor) is based on the height of the summed score off all sub-filters, the GG_Y and DGD_Y are weighted with a relatively high score of 0.85 each. If an adjacent gene pair is identified as OP by one of these filters, it is strongly promoted to be an OP by the plain predictor, as the overall score is strongly increased. Consequently, if an adjacent gene pair is not considered as an OP by GG and DGD filter (GG_N, DGD_N), a low score is applied (0.15) to promote the label of NP by the decision function of the plain predictor.

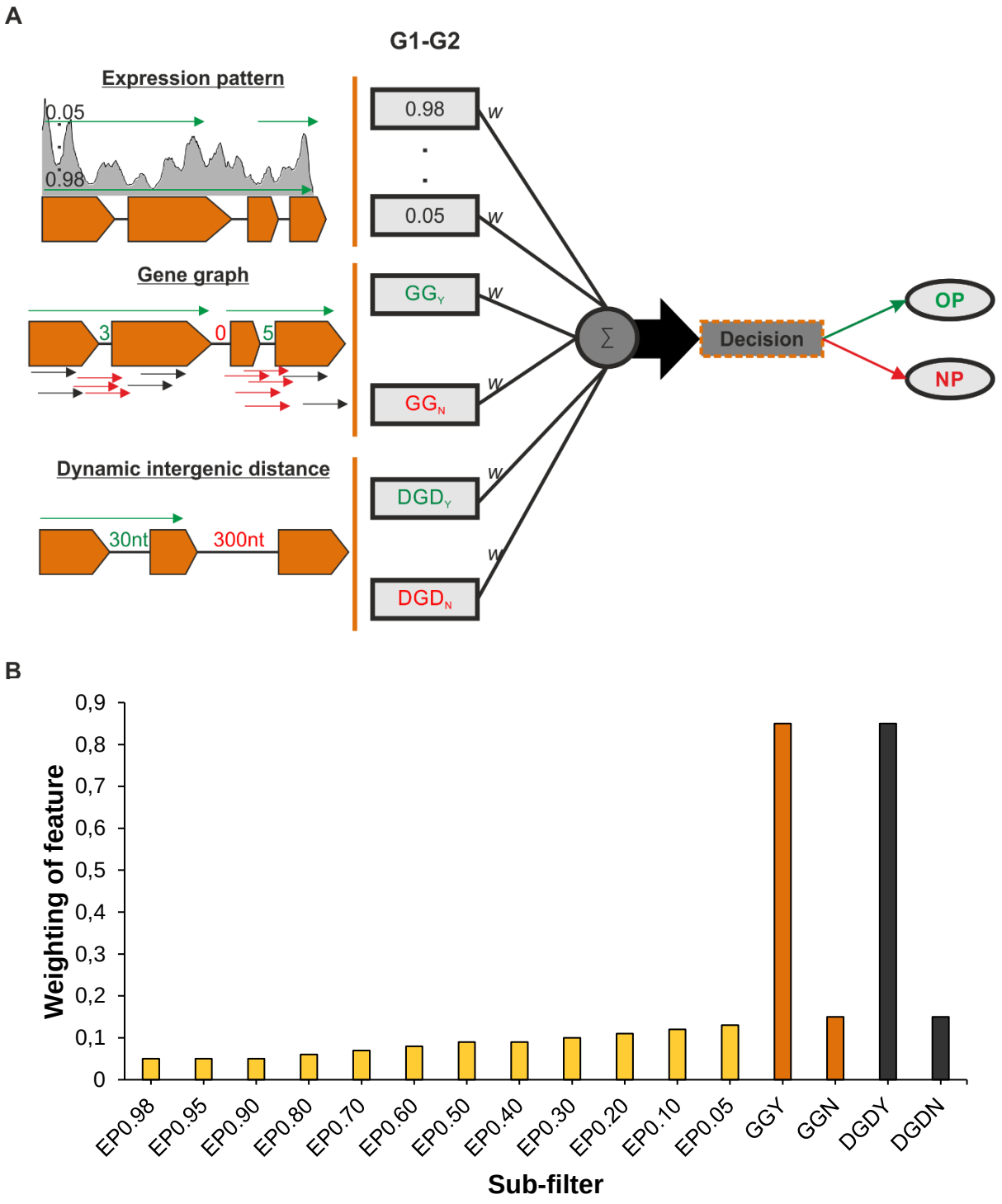


Figure 12: Schematic structure of the plain predictor model and weightings applied to the filter inputs. (A) The model receives the inputs of each filter and multiplies them with the weighting. The summed scored weightings are then passed on to the decision function, which labels two adjacent gene pairs as OP or NP depending on the height of the score. (B) Indicated are the filter names and their filter types (EP, GG, DGD). For each sub-filter a specific weighting score is applied.

In general, a gene pair is considered as OP by the plain predictor if GG and DGD can be applied, leading to a score of 1.7 (which is greater than the cut-off 1.62). However, a gene pair can also be labeled as OP if GG or DGD are applied and $EP_{0.98-0.20}$ can be applied (score of 1.75). In all other scenarios, two genes are considered as NP by the plain predictor. After

determining all OPs and NPs, they are grouped into operons, leading to a list of operons for both Watson and Crick strand. In contrast to typical machine learning approaches, the plain predictor model is not a trained classification model, where training and test data have to be identified for setting up the model. The decision model is based on the assumption that for OPs, a significantly higher score should be reached than for NPs based on the defined filters. Further, a combination of generalized filters into a generalized prediction model gets rid of the typical training set reliance of classical machine learning models, as no specific training set is used to set up the model. In fact, the decision model could best be described with a simple “if-else” decision, as based on summed scores either and OP or NP is declared.

3.6. High confidence of OpPipe plain predictor on labeled datasets

For a first indication whether the assumed generalization caused by the filters and plain predictor are applicable to different species, the predicted operons of the plain predictor of OpPipe were compared to other databases and predictors. Therefore, *Escherichia coli* as a model organism has been used. *Escherichia coli* operon sets identified by the plain predictor have first been compared to the RegulonDB, which contains experimentally and computationally identified operons. Consequently, it has been estimated which operons are part of both predictors and which are not, revealing that the predicted operon sets are comparable between both approaches (Figure 13).

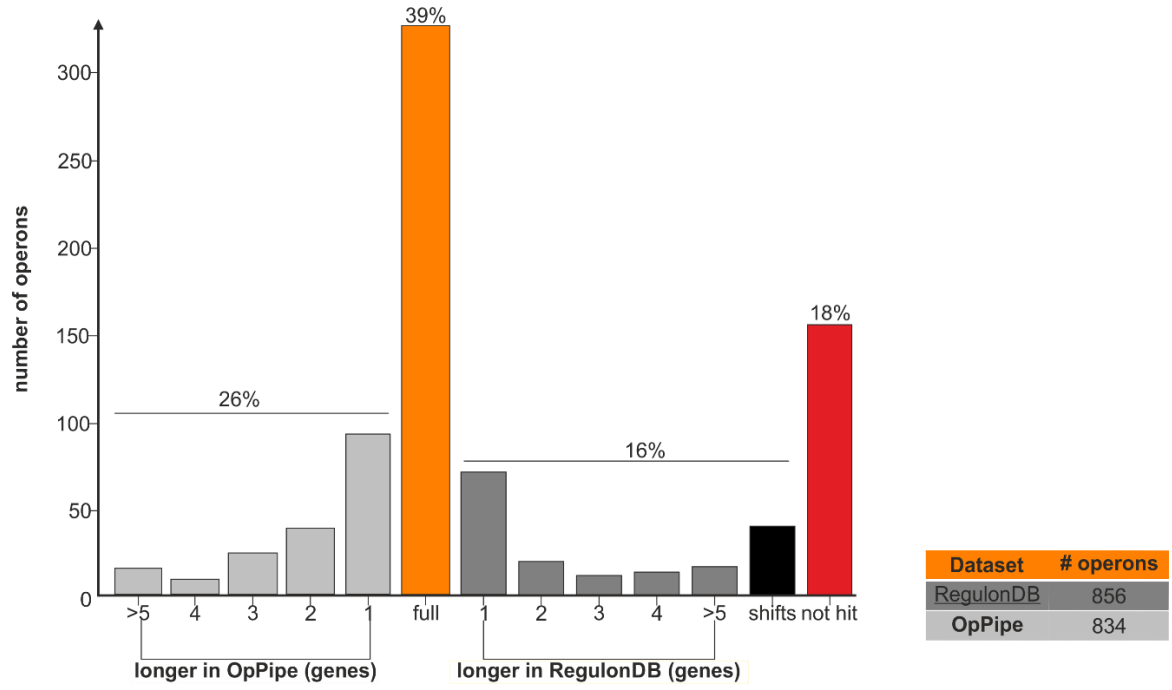
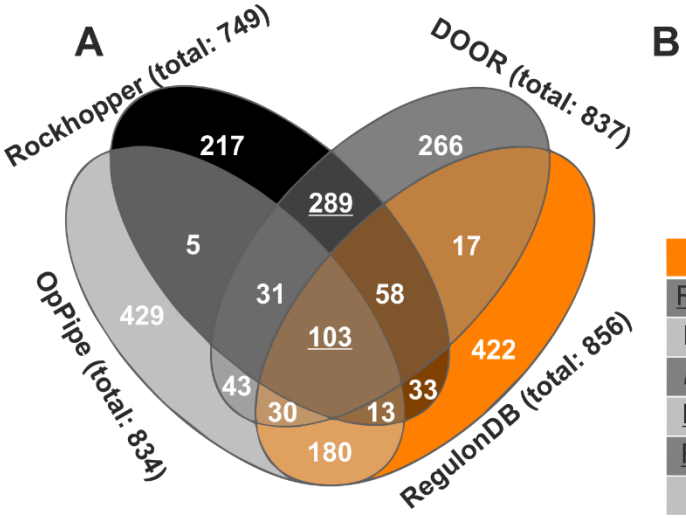


Figure 13: Comparison of operons predicted by OpPipe with collected ones from RegulonDB [147] in *Escherichia coli* K-12. The found operons divide into operons that have been found in the same composition in both sets (full, orange), operons that are found in the same composition in both sets but longer in one of the sets (longer in OpPipe, light grey vs. longer in RegulonDB, moderate grey), operons which share more than one gene, but are not of the same composition (shifted, black) and operons that are exclusively in the OpPipe set (no hit, red).

Starting from the OpPipe set, it has been searched, which operons can be found inside the RegulonDB set (Figure 13). Both sets exhibit a comparable number of operons while the number of operons in the RegulonDB set (856) is slightly increased compared to the OpPipe set (834). In general, 82% of the OpPipe operons can be identified within the RegulonDB set, whereby 39% of the operons could be identified in the same composition in both sets (Figure 13), and 18% are exclusively found in OpPipe (Figure 13). However, comparing the RegulonDB set with the OpPipe set also showed the identification of 81% of the operons while only 19% of the RegulonDB operons could not be identified within the OpPipe set. This leads to a high overlap between the RegulonDB and the OpPipe plain predictor.

Comparing the predicted operon sets of the OpPipe plain predictor, RegulonDB, Rockhopper and DOOR with each other shows that on the one hand OpPipe is the one sharing the most full hit candidates with the RegulonDB, but on the other hand that the predicted operon sets differ between prediction approaches (Figure 14).



B

Comparison	# FH	# NH1	# NH2
Regulon & **OpPipe**	326	168	155
Rockh. & **OpPipe**	152	121	263
DOOR & **OpPipe**	207	145	182
Regulon & Rockh.	207	214	61
Regulon & *DOOR*	208	170	132
Rockh. & *DOOR*	481	50	175

Figure 14: Venn diagram of the predicted operon set of four predictors in *Escherichia coli* K-12. The diagram (A) shows the number of total predicted operons for the predictors Rockhopper [36], DOOR [65], RegulonDB [147] and OpPipe (total number in brackets behind name) and the number of operons which could be found in the overlap (only full hits are counted). The table (B) shows the number of full hits (#FH) of the pairwise comparison and the number of operons not found if the operon set of the first predictor is compared to the second (#NH1) and the number of operons if the operon set of the second predictor is compared to the first (#NH2).

All four predictors exhibit 103 full hit operons which were predicted by all of them. In general, RegulonDB and OpPipe exhibit the most operons (856 and 834), while DOOR (837) and Rockhopper (749) exhibit fewer operons (Figure 14A). Subsequently, Rockhopper and DOOR exhibit 289 operons that are exclusively found within their overlap (Figure 14A). However, 180 have been predicted exclusively by OpPipe and RegulonDB, while OpPipe and DOOR only share 43 exclusively predicted operons (Figure 14A). Interestingly, the two RNA-Seq based approach Rockhopper and OpPipe only share five exclusively predicted operons (Figure 14A).

When compared pairwise, Rockhopper and DOOR exhibit the most full hits (481, Figure 14B, #FH). RegulonDB thereby shares the most full hits with OpPipe (326, Figure 14B, #FH), while it shares only 207 with Rockhopper and 208 with DOOR. Rockhopper and OpPipe only share 152 operons while DOOR and OpPipe sharing 207 full hits (Figure 14B, #FH). When comparing Rockhopper to DOOR, only 50 operons could not be identified (Figure 14B, #NH1), however when comparing DOOR to Rockhopper, 175 operons could not be found (Figure 14B, #NH2). When comparing RegulonDB and OpPipe, 168 operons remain unfound (Figure 14B, #NH1), while 155 (Figure 14B, #NH2) could not be found when comparing OpPipe to RegulonDB. Compared to other predictors, OpPipe is the one being the closest to the operon set of experimentally proven operons provided by the RegulonDB, which suggests a high sensitivity of the OpPipe plain predictor for the dataset of *Escherichia coli.*

Despite the model organism *Escherichia coli,* the prediction for the cyanobacteria *Anabaena* sp. PCC 7120 has also been compared with different predictors (Figure 15, Figure 16), for which the *Escherichia coli* specific RegulonDB was substituted with the ProOpDB.

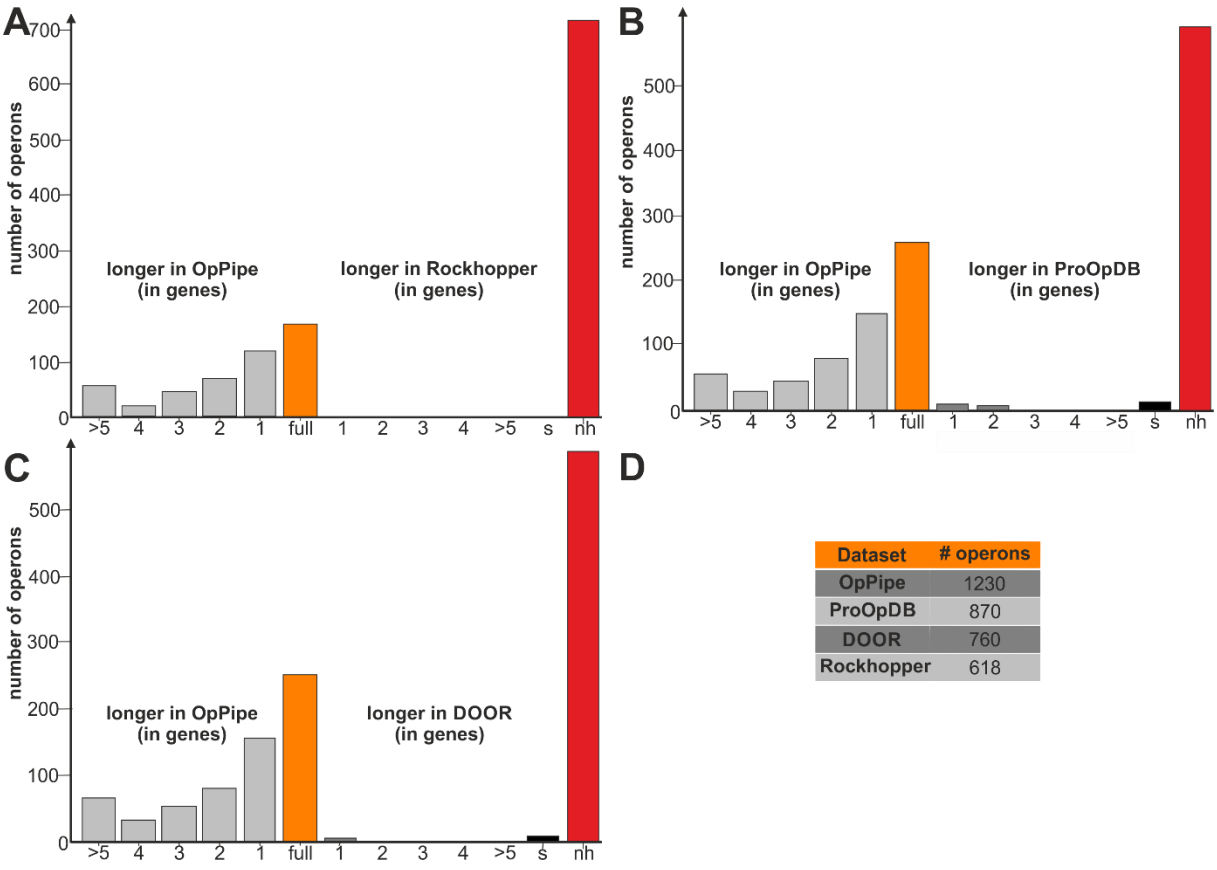
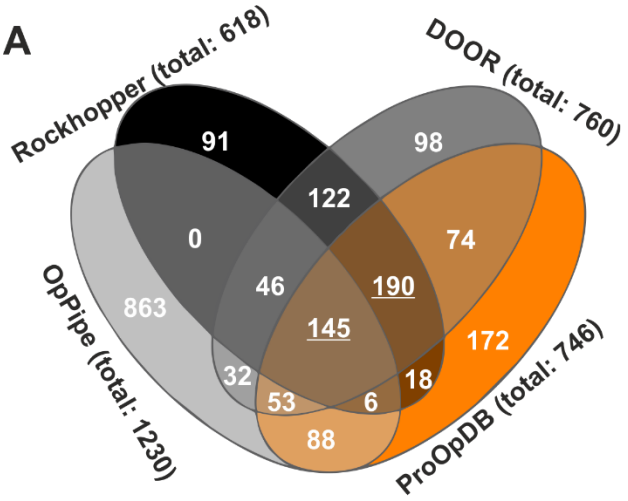


Figure 15: Comparison of operons predicted by three different predictors in *Anabaena* sp. PCC 7120. OpPipe prediction was compared to Rockhopper [36] (A), ProOpDB [69] (B) and DOOR [65] (C) and grouped into operons which are found in the same composition in both sets (full, orange), operons that are found in the same composition in both sets but longer in one of the sets (longer in OpPipe, light grey vs. longer in other predictor, moderate grey) and operons which share more than one gene, but are not of the same composition (shifted, s, black) and OpPipe exclusive ones (no hit, nh, red). Further, the total amount of predicted operons is indicated for all predictors (D).

In contrast to the comparison with the RegulonDB, operons for *Anabaena* sp. PCC 7120 identified by the OpPipe plain predictor show a decreased overlap to other prediction tools (Figure 15). In general, the number of predicted operons differs between the predictors (Figure 15D). While for OpPipe, 1230 operons could be identified (Figure 15D), only about half could also be identified by the RNA-Seq based predictor Rockhopper (618). ProOpDB identified 870 operons and DOOR 760 operon candidates. When comparing the operons from OpPipe to Rockhopper it is interesting that about 700 could not be identified at all in the Rockhopper operon set, while fewer than 200 exhibit the same composition (Figure 15A). Further, numerous operons are elongated by more than five genes within the OpPipe set. While no operon is longer in Rockhopper compared to the OpPipe set (Figure 15A), the prediction of the ProOpDB exhibits few operons that are longer than in the OpPipe prediction (Figure 15B). Here, about 250 operons were fully hit, while about 550 could not be found. Additionally, shifted operons also occur when comparing OpPipe to ProOpDB. Compared to DOOR (Figure 15C) nearly 600 operons from OpPipe could not be identified by DOOR. The predicted operons by OpPipe thereby differ compared to other predictors, as numerous operons from OpPipe were not identified by the other tools. However, congruently to the *Escherichia coli* approach, the different predictors also prove to have a decreased overlap when compared against each other, as few operons can be identified among different prediction sets (Figure 16A).



B

Comparison	# FH	# NH1	# NH2
ProOpDB & **OpPipe**	292	3	578
Rockh. & **OpPipe**	197	1	715
DOOR & **OpPipe**	276	1	585
ProOpDB & Rockh.	359	241	89
ProOpDB & *DOOR*	462	137	130
Rockh. & *DOOR*	503	3	159

Figure 16: Venn diagram of the predicted operon set of four predictors in *Anabaena* sp. PCC 7120. The diagram (A) shows the number of total predicted operons for the predictors Rockhopper [36], DOOR [65], ProOpDB [69] and OpPipe (total number in brackets behind name) and the number of operons which could be found in the overlap (only full hits are counted). The table (B) shows the number of full hits (#FH) of the pairwise comparison and the number of operons not found if the operon set of the first predictor is compared to the second (#NH1) and the number of operons if the operon set of the second predictor is compared to the first (#NH2).

At least 145 operons were identified by all the predictors. However, the biggest amount of full hit operons is between ProOpDB, DOOR and Rockhopper with 190 full hit operons (Figure 16A). With 863 candidates, OpPipe exhibits the highest number of operons which could not be

found as a full hit in another predictor. While for ProOpDB, this is the case for 172 operons, for Rockhopper (91) and DOOR (98), fewer than 100 operons could not be found as a full hit in another predictor. In general, OpPipe exhibits the biggest predicted operon set and, interestingly, approximately twice as many operons compared to Rockhopper. However, there are no operons in the full hit overlap between just OpPipe and Rockhopper. However, they are sharing 197 full hit operons (Figure 16B). Further, it seems like Rockhopper and DOOR have similar prediction routines as they share the most full hit operons (503). Comparing the operons which could not be found when comparing the different approaches, 241 operons from ProOpDB could not be found in any composition within the Rockhopper set, while 89 operons identified by Rockhopper are not part of the ProOpDB set in any composition. Subsequently, 137 operons remain unique when comparing ProOpDB and DOOR (or 130, respectively, when comparing DOOR to ProOpDB). However, when comparing Rockhopper to DOOR, again, only three remain unfound, while 159 operons of DOOR cannot be found in Rockhopper. Remarkably, nearly all operons identified by DOOR, Rockhopper and ProOpDB can be found fully or partially within the operons set of OpPipe. Only three operons could not be identified when comparing ProOpDB to OpPipe, also three when comparing Rockhopper to OpPipe and only one operon when comparing DOOR to OpPipe. The results on the two species suggest that, firstly, different prediction models tend to state different operons within the same species. Secondly, while the individual comparisons in *Anabaena* sp. PCC 7120 leads to a decreased overlap, a combination of the operon sets of ProOpDB, DOOR and Rockhopper (Figure 16B) and a search against the OpPipe set leads to in total 370 full hits, while only three operons remain unfound. The remaining operons are either full hits, partially covered or shifted, leading to a generalized identification by OpPipe as it combines predicted operons of different predictive models within one dataset.

Even though OpPipe exhibited a greater overlap in *Escherichia coli* compared to *Anabaena* sp. PCC 7120, when comparing different predictors, it remains difficult to interpret the values as it is not clear what the ground truth is in *Escherichia coli* as well in *Anabaena* sp. PCC 7120. To address this issue and because the comparison of the predicted operon sets in *Escherichia coli* and *Anabaena* sp. PCC 7120 leads to partially contrasting results, the predictors were evaluated on real operon data. In a first step, 20 operons from the *Escherichia coli* literature set (2.6.1) were used and compared with different predictors (Table S13). Eye-catching is the ability of the RegulonDB to identify all of the operons, while the other predictors do not identify all of the operons (3 not found in OpPipe, 4 not found in ProOpDB, 1 not found in DOOR and 4 not found in Rockhopper). However, all predictors except Rockhopper are able to identify at least 50% of the operons as a full hit.

For a further analysis, 18 operons of the *Anabaena* sp. PCC 7120 literature set (2.6.1) were chosen and compared around the different predictors (Table 11). Thereby, OpPipe and ProOpDB prove their ability to identify the majority of the operons in the stated literature set, while Rockhopper and DOOR identify fewer operons that are part of the literature set.

Table 11: Prediction of literature operons from *Anabaena* sp. PCC 7120 by OpPipe, ProOpDB, DOOR and Rockhopper. The inked boxes indicate if an operon was found (green) by the predictor or not (red). Further, it is indicated if a predicted operon is longer than in the literature set (blue) or if the predicted operon partially hits the literature set operon (yellow).

Operon	OpPipe	ProOpDB*[1]	DOOR*[2]	Rockhopper*[3]	Source
pec					[161]
cpc					[163]
cox1					[164]
cox2					[164]
cox3					[164]
fraC					[165]
nir					[159]
alr2825-alr2831					[166]
alr2835-2841					[166]
devBCA					[166]
all1780-all1781					[166]
cphA-cphB					[166]
nif					[166]
rbcLXS					[166]
natA-natC					[167]
hyp					[55]
all5341-all5350					[166]
cmpA-D					[166], [168]

perfect	partial	elongated	not found

*[1]ProOpDB [69], *[2]DOOR [65], *[3]Rockhopper [36]

The 18 operons stated in the literature set could all be identified by OpPipe (Table 11). However, four of them are elongated, while three of them are only partially found. Likewise, ProOpDB identifies most of the stated operons while failing to identify four of them, just as DOOR. Further, six and eleven operons are only partially covered. No operon is elongated within both approaches while, ProOpDB identifies eight and DOOR three operons within a perfect hit. In contrast, the expression-based predictor Rockhopper is only able to identify two of the operons correctly, while nine of the operons remain unfound. Thereby, seven operons are covered partially. As the overlap between the different prediction tools in Anabaena sp. PCC 7120 and *Escherichia coli* suggests similar prediction models of Rockhopper and DOOR (Figure 14 and Figure 16), DOOR can also only identify three operons, while three remain unfound. However, DOOR seems to be able to identify the operons more precisely in

Anabaena sp. PCC 7120 compared to Rockhopper, as the remaining operons are at least partially covered. The three operons *fraC, pec* and *nir* thereby demonstrate the different predictions (Figure 17).

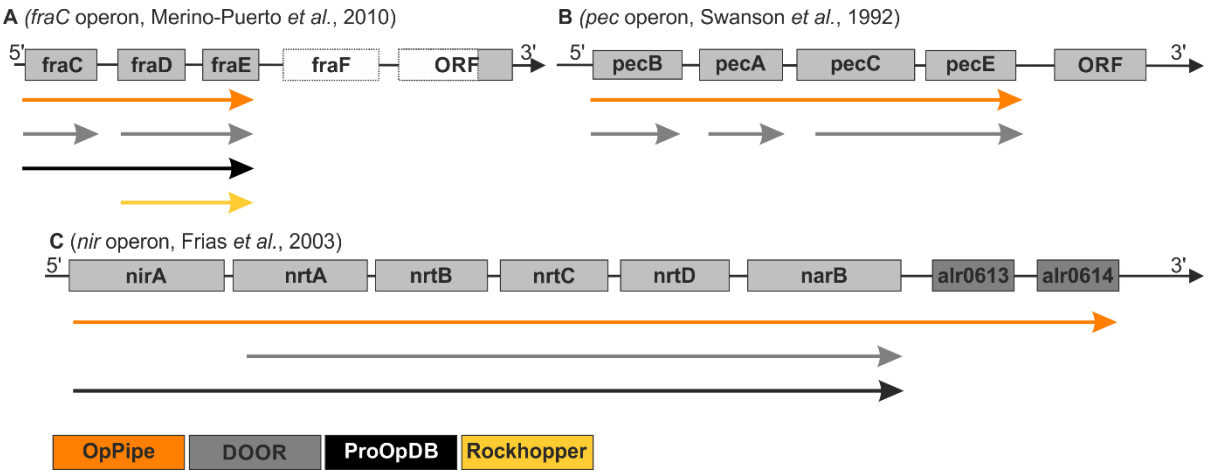


Figure 17: Representation of the *fraC* [165] (A), *pec* [161] (B) and *nir* operon [159] (C) in *Anabaena* sp. PCC 7120 and their prediction by different prediction tools. Shown is the composition of the referring operons with their adjacent genes. Genes part of an operon are inked light grey, genes part of a cluster with the operon are inked dark grey (C) and genes from the opposite strand are marked as dotted white square (A). Indicated are the predicted operons of different predictors as arrows (OpPipe = orange, DOOR [65] = grey, ProOpDB [69] = black, Rockhopper [36] = yellow).

For example, the *fraC* operon (Figure 17A) is only identified by OpPipe and ProOpDB in the correct composition, while the DOOR database splits the operon into two TUs and Rockhopper only predicts fraD and fraE into one operon. Further, only OpPipe identifies the *pec* operon in the correct compositions, while only DOOR identifies parts of the operon and ProOpDB and Rockhopper are not able to identify the operon (Figure 17B). Following, the *nir* operon (Figure 17C) cannot be identified by Rockhopper, while it is shorter in DOOR and longer in OpPipe. The ProOpDB perfectly predicts the operon.

Comparing the scoring models of each predictor for *Anabaena* sp. PCC 7120 and *Escherichia coli* (Table S14), ProOpDB, DOOR and Rockhopper exhibit an increased prediction score of *Escherichia coli* compared to *Anabaena* sp. PCC 7120, while the opposite can be obtained within OpPipe. Subsequently, DOOR and Rockhopper fit best for the model organism *Escherichia coli*, as the operon scores are clearly increased compared to the *Anabaena* sp. PCC 7120 scores. Additionally, DOOR nearly reaches the same scores for *Escherichia coli* compared to the RegulonDB, while exhibiting a poor score for *Anabaena* sp. PCC 7120. Further, ProOpDB seems to be able to offer a generalized prediction, but reaches poorer scores compared to OpPipe. In general, OpPipe shows encouraging abilities of predicting operons in a generalized way that is species independent.

3.7. Operon prediction via *in-silico* methods leads to different results than experimental TSS data

After observation of different *in-silico* predictions of operons, the overlap between an experimental TSS identification approach and the operon prediction was conducted. For this reason, TSS information form the publication of Mitschke *et al.* [56] on *Anabaena* sp. PCC 7120 under control conditions was used. The TSS were thereby identified by using RNA-Seq data and the number of mapped reads per TSS were monitored. Comparable like proposed in the publication, a TSS should be located upstream a genetic arrangement (in at least 200 nts). By this definition, each polycistronic region should contain one strongly expressed TSS for the specific condition. For a first overview, the distribution of all TSS within the genome has been separated into TSS located before an annotated gene (within 200nt), within an annotated gene or between annotated genes (more than 200nt before next annotated gene). Surprisingly, a high number of TSS was found which do not corelate with any gene within the genome of *Anabaena* sp. PCC 7120 (Figure 18).

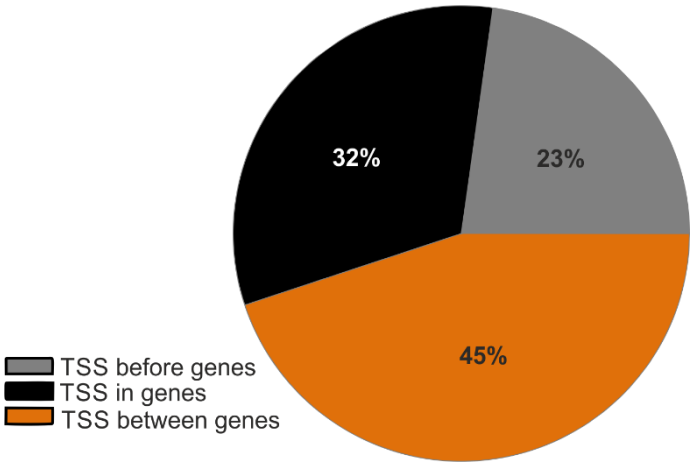


Figure 18: Distribution of transcription start site within the genome of *Anabaena* sp. PCC 7120. Indicated are the percentages of all TSS located before genes (fewer than 200nt before start of annotated gene), TSS within an annotated gene and TSS located between adjacent genes.

Remarkably, only 23% of them are located before genes (within the range of 200nt) and do not lie in an annotated gene position (Figure 18). Further, 32% of the TSS are located within annotated genes. Surprisingly, nearly half of all TSS (45%) are located between adjacent genes, meaning they are not within the distance of 200nt to the next gene and do not lie within an annotated gene. For an evaluation of relevance, the expression coverage of the three TSS groups (before genes, in genes, between genes) has been evaluated (Figure 19).

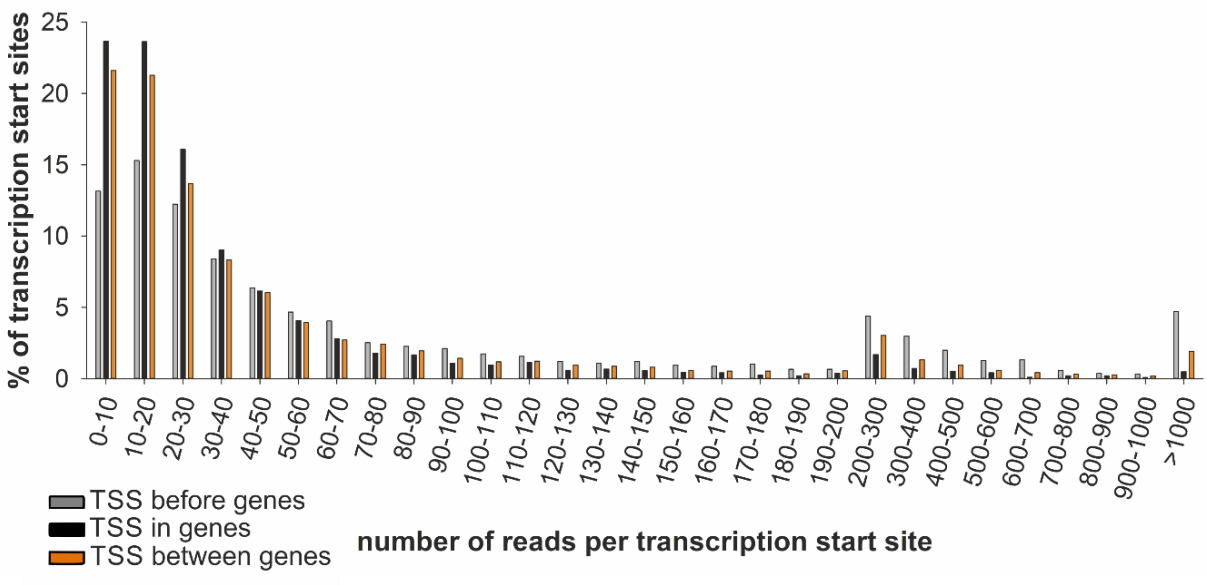


Figure 19: Distribution of TSS (%) divided into number of reads per TSS in *Anabaena* sp. PCC 7120. The TSS are separated into TSS before genes (genes fewer than 200nt before start of annotated gene), TSS within an annotated gene and TSS located between adjacent genes. On the x-axis the TSS are separated into buckets which indicate the number of reads that have been mapped onto the TSS.

In general, the TSS in genes and between genes are higher for the first buckets until bucket 40-50 (Figure 19). TSS located before genes exhibit a higher expression than the other groups firstly observable within the range of 50 reads per TSS (bucket 40-50 and 50-60). However, the values are nearly even for the buckets of 70-80 and 90-100. Starting from the bucket of 90-100, the TSS located before genes are always higher than the other two groups. Therefore, only these TSS were considered which exhibit a number of at least 50 reads and of at least 100 reads per TSS.

After defining a cut-off for considering a TSS to be expressed, the predicted operons for *Anabaena* sp. PCC 7120 from OpPipe, ProOpDB, DOOR and Rockhopper have been compared (Figure 20).

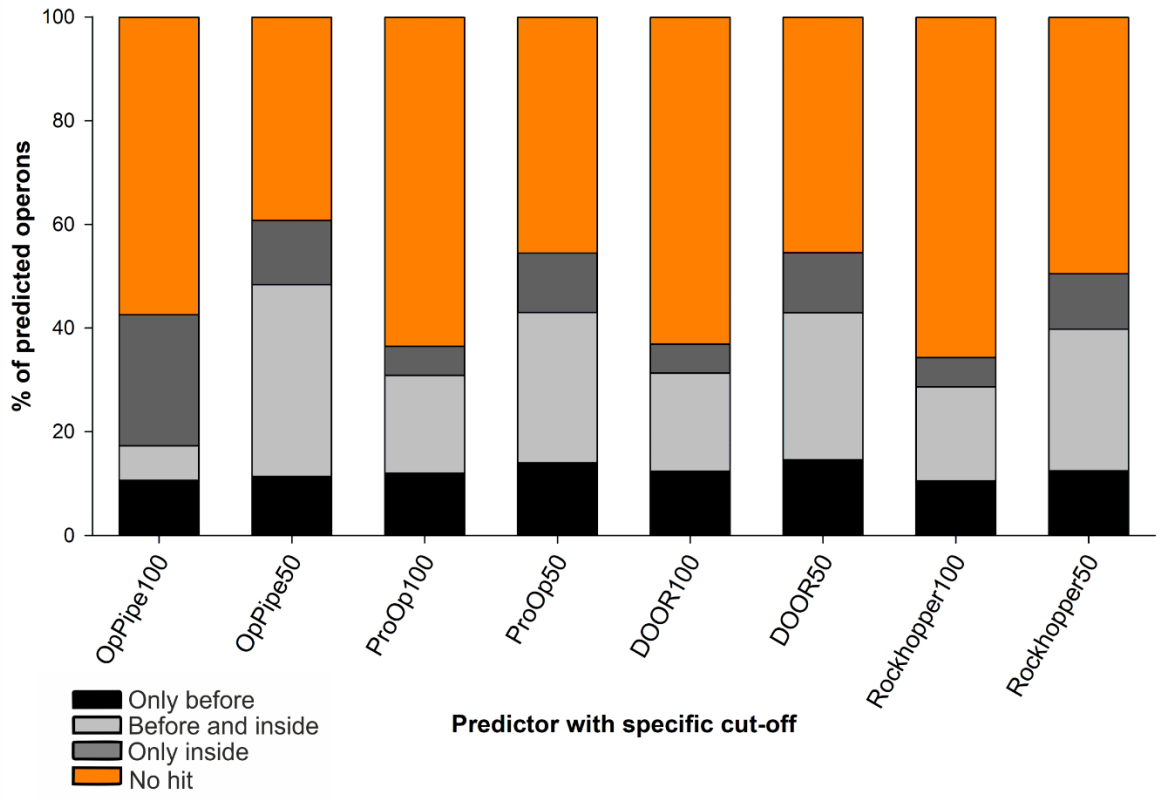


Figure 20: TSS distribution based on the operon sets of different predictors from Anabaena sp. PCC 7120. The predicted operons (y-axis) are separated into different categories, operons that harbor a TSS only before the operon (Only before, black), operons with TSS before the operon and within the operon (Before and inside, light grey), operons harboring TSS only in the inside of the operon (Only inside, moderate grey) and operons that have not been hit by a TSS at all (No hit, orange). The categories are illustrated for all predictors (OpPipe, ProOpDB [69], DOOR [65], Rockhopper [36]) with two different cut-offs of number of reads hit to the TSS (50 and 100).

For both cut-offs, all predictors found between ~40% (cut-off 100) and ~60% (cut-off 50) of operons which could not be supported by a TSS (Figure 20). OpPipe exhibits the fewest number of operons that do not have a TSS at any place of the operon compared to the other operon sets (Figure 20). The number of operons without any TSS was nearly the same for ProOpDB, DOOR and Rockhopper. Interestingly, the number of operons with a TSS located only before an operon remains stable compared to the different cut-offs and predictors.

Comparing the TSS distribution from all predictors to the operon set that has only been identified by OpPipe shows a congruent behavior as only 40-60% of the operons could be validated via TSS (Figure 21).

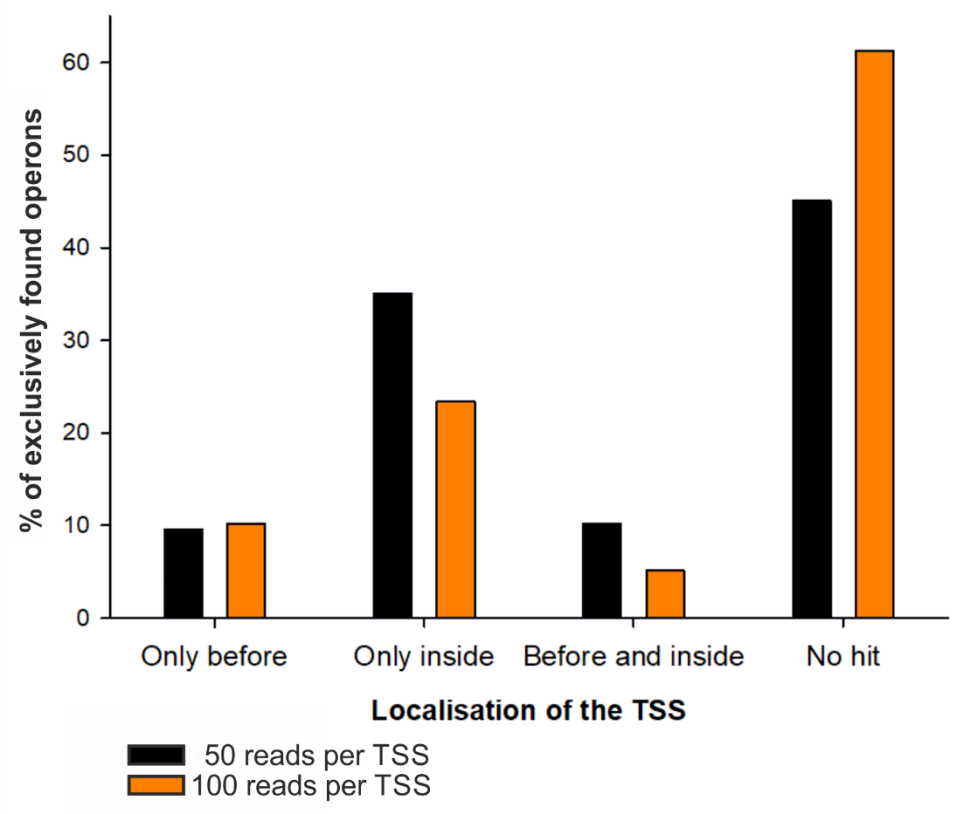


Figure 21: TSS distribution of exclusively found operons in *Anabaena* sp. PCC 7120. The predicted operons (y-axis) are separated into different categories, operons that harbor a TSS only before the operon (Only before, black), operons with TSS before the operon and within the operon (Before and inside, light grey), operons harboring TSS only in the inside of the operon (Only inside, moderate grey) and operons that have not been hit by a TSS at all (No hit, orange). The categories illustrate two different cut-offs of number of reads hit to the TSS (50 = black and 100 = orange).

For both cut-offs, the exclusive set exhibits about 10% of the operons which have a TSS before the operon (Figure 21). However, for both cut-offs, the major number of operons did not have a TSS. To rule out that this observation is misleading due to many not experimentally proven TUs, the OP literature set (2.6.1) of *Anabaena* sp. PCC 7120 was evaluated regarding TSS (Figure 21).

Interestingly, comparing the operon sets of the different predictors (Figure 20) as well as the OpPipe exclusive set (Figure 21) to the literature set (Figure 22), the TSS distribution seems to show a comparable behavior. The number of operons validated via TSS is slightly increased but only 60-65% of the operons could be validated via the TSS localization.

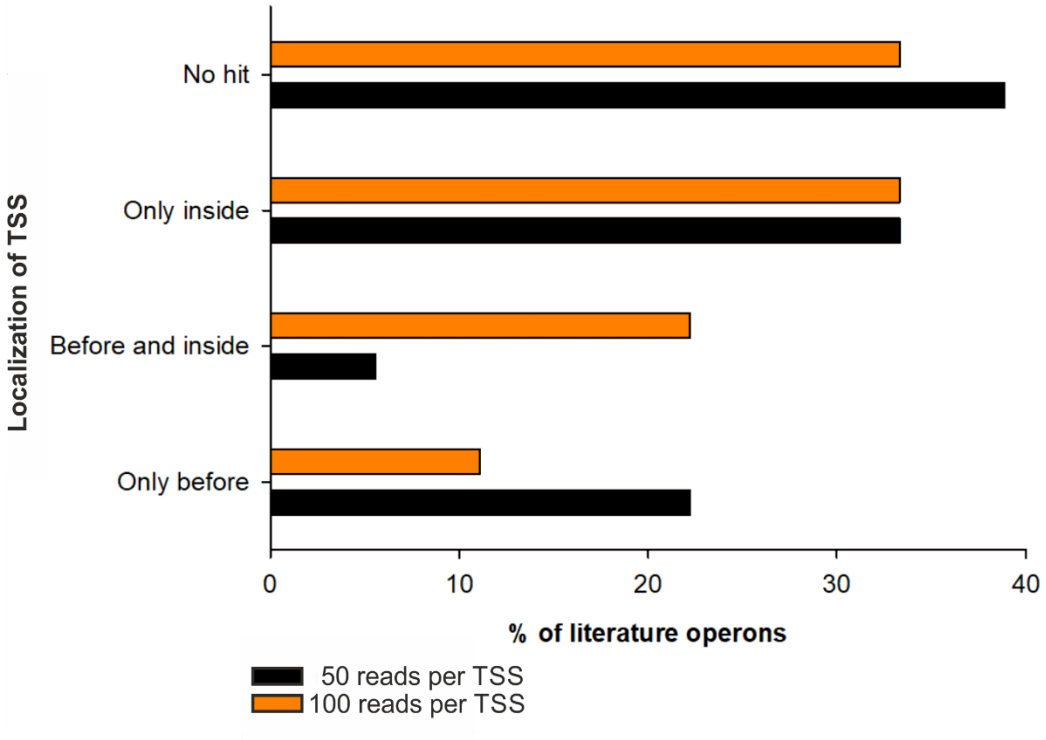


Figure 22: TSS distribution of the literature operon set of *Anabaena* sp. PCC 7120. The operons from the literature set (x-axis) are separated into different categories (y-axis), operons that harbor a TSS only before the operon (Only before), operons with TSS before the operon and within the operon (Before and inside), operons harboring TSS only in the inside of the operon (Only inside) and operons that have not been hit by a TSS at all (No hit). The categories are illustrated for two different cut-offs of number of reads hit to the TSS (50 = orange and 100 = black).

Only ~20% of the operons from the literature set in *Anabaena* sp. PCC 7120 exhibit a TSS solely before the referring first gene of the operon (Figure 22). Interestingly for cut-off 100, more than 20% of the literature operons have a TSS before the first gene, while for the cut-off 50 this is only the case for 10%.

Within the genome of *Anabaena* sp. PCC 7120 nearly half of the TSS (45%) have been identified to be located between adjacent genes with a distance greater than 200 nts. Further, for different *in-silico* methods, the majority of predicted operons (~60%) does not exhibit a TSS (covered by 50-100 reads) that is either located 200 nts before or inside the operon. This observation shows to be congruent when evaluating the operons stated in the literature.

3.8. Machine learning approaches on filter data for labeled literature operons lead to enhanced operon prediction confidence

The comparison of the plain predictor of OpPipe to other available tools and databases (like DOOR, RegulonDB, Rockhopper and ProOpDB) for two different species showed different results. To exclude bias based on the architecture of the plain predictor with defined scores, further models (RF, SVM and NN) were used for the operon prediction. For all three models, the different filters (GG, EP and DGD) were used as inputs.

In contrast to the plain predictor, where the different filter inputs are weighted based on a score that reflects an assumed importance, the different models are evaluated based on different statistical measures for a defined dataset. Nevertheless, through the filters, the data is previously generalized and are assumed to model an “operon-behavior”. Therefore, the input data are assumed to lead to species unspecific training. The input data thereby was a gene pair within the standard input format (Table S12). The input dataset was based on the literature OP set and NP-set 2 (2.6.1) and was split into training and test set (80% training, 20% test) via a cross validation. The validation thereby relied on a 5-fold cross validation split into training and test set of the input dataset.

As a first approach, RF was used to create an operon prediction model. The training set provided through the cross validation was then used within a grid search which also relied on a 5-fold cross validation and the best classification model of each grid search was saved.
The 5-fold cross validation via five-fold grid search revealed at least 92% accuracy for each best RF model of the different folds. Further, the RF models of each fold revealed an AUC of at least 0.99 (Figure S6). The different RF models thereby show differences for the different statistical measure (precision, recall, f1-score) for positive (OP) and negative examples (NP) (Table 12).

Table 12: Performance measure of the training of the random forest classifier for the filter data. For each of the 5-folds, the precision, recall, f1 score is shown. For each fold, the best scoring model of the 5-fold cross validation of the grid search is shown.

	fold 1	fold 2	fold 3	fold 4	fold 5	**Average**	
NP	1,00	1,00	1,00	1,00	0,92	**0,98**	
OP	0,85	1,00	1,00	0,88	1,00	**0,95**	**Precision**
average	0,92	1,00	1,00	0,94	0,96	**0,96**	
NP	0,82	1,00	1,00	0,86	1,00	**0,94**	
OP	1,00	1,00	1,00	1,00	0,91	**0,98**	**Recall**
average	0,91	1,00	1,00	0,93	0,96	**0,96**	
NP	0,90	1,00	1,00	0,92	0,96	**0,96**	
OP	0,92	1,00	1,00	0,93	0,96	**0,96**	**F1-score**
average	0,91	1,00	1,00	0,93	0,96	**0,96**	

Interestingly, fold 1 and 4 lead to an increased amount of gene pairs being classified as OPs. It is observable that the precision for OP (0.85 and 0.88) is decreased compared to the other folds, meaning a higher amount of FP for these classifiers. In the same context, the recall within these folds for NPs is also decreased (0.82 and 0.86) compared to the other folds, meaning an increased amount of FN (NPs that have been predicted as OPs). Albeit not that strong, a similar effect can be obtained for fold 5. An advantage of the RF model is the possibility to extract the feature importance of the single input filter for each model (Figure 23).

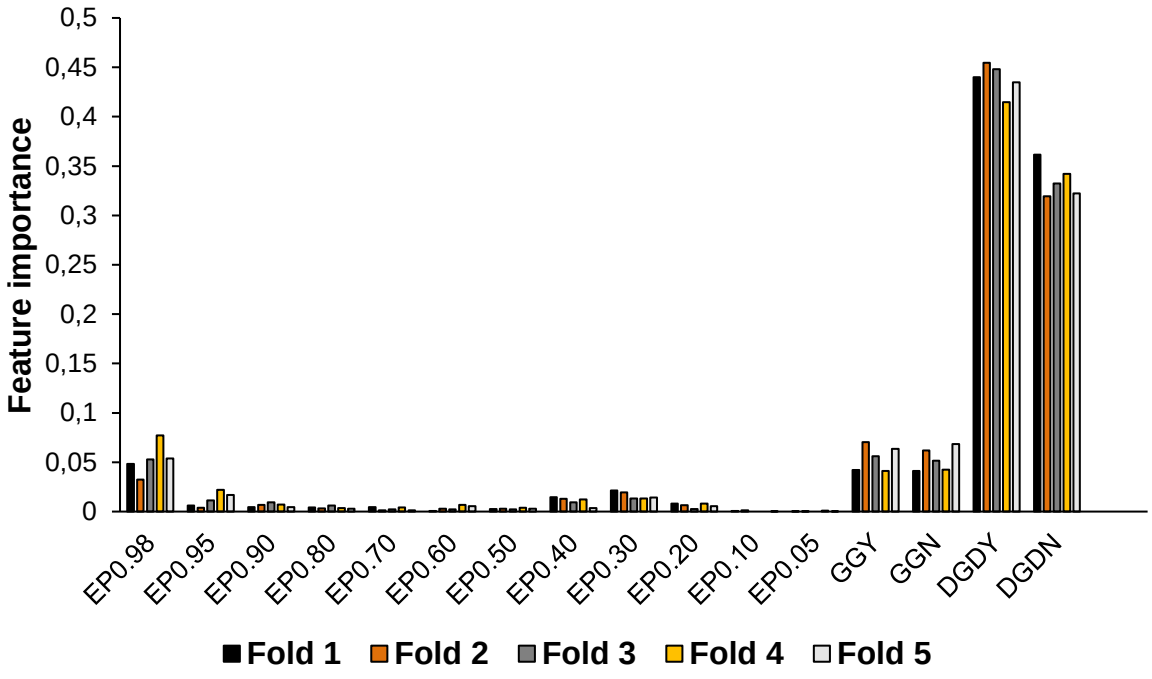


Figure 23: Feature importance of RF classifier on filter data. For each of the filters, the importance of the feature was calculated. The different folds of the cross validation are inked differently. A gene pair served as input.

In general, the DGD is the most important feature in each fold. For all folds, the importance for being a part of the DGD filter is above 0.4. After the DGDY feature, the DGDN is the second

most important feature. The filter reaches an importance of 0.3 to 0.37 for all folds. However, the importance of DGDN is not as high as for DGDY. The importance of DGDY and DGDN seems to be non-related as for example in fold 2, the DGDY has the highest importance within the folds while for DGDN, fold 2 exhibits the lowest importance for all folds. However, the importance analysis suggests that the DGD filter might be the driving factor for the classification of OP and NP. However, when comparing the importance of the filter for the RF classifier, the different provided inputs are differently weighted within the folds. Interestingly, although folds 2 and 3 exhibit the same statistical measures, the importance of each sub-filter is different between these folds. The feature importance of the RF models also shows differences compared to the scoring of the plain predictor. In contrast to the weighting of the EP filter in the plain prediction model, the moderate EP filter stringency of 0.98 seems to be the most important feature within the EP filter. For all folds, this stringency reaches a value of approximately 0.04 to 0.07. However, a decrease of importance can be observed for the following EP stringencies of 0.95 to 0.50 compared to the EP filter with a stringency of 0.98. Following, the GG filter shows a congruently importance between the feature of being part of the GG (GGY) and not (GGN). Interestingly, the importance of the GG filter does not differ dramatically from the EP filter with a stringency of 0.98, which is also in contrast to the plain predictor model, where the GG filter is as important as the DGD filter.

SVM was trained with the same inputs and settings as the RF model. For the SVM, an accuracy of at least 98% could be reached for all folds, which is increased compared to the RF approach (92% accuracy), while the AUC remains comparable between both approaches (Figure S6). With 0.97 (Table 13), the average for the measures of precision, recall and f1-score, are slightly increased compared to the RF model where these measures all reached average values of 0.96 (Table 12).

Table 13: Performance measure of the training of the support vector machine classifier for the filter data. For each of the 5 folds, the precision, recall and f1 score is shown. For each fold, the best scoring model of the 5-fold cross validation of the grid search is shown.

	fold 1	fold 2	fold 3	fold 4	fold 5	**Average**	
NP	0,98	1,00	1,00	0,98	0,92	**0,98**	
OP	0,85	1,00	1,00	0,98	1,00	**0,97**	**Precision**
average	0,91	1,00	1,00	0,98	0,96	**0,97**	
NP	0,82	1,00	1,00	0,98	1,00	**0,96**	
OP	0,98	1,00	1,00	0,98	0,91	**0,97**	**Recall**
average	0,90	1,00	1,00	0,98	0,96	**0,97**	
NP	0,89	1,00	1,00	0,98	0,96	**0,97**	
OP	0,91	1,00	1,00	0,98	0,95	**0,97**	**F1-score**
average	0,90	1,00	1,00	0,98	0,96	**0,97**	

Interestingly, for fold 1, the SVM also shows only 0.85 precision for OPs and consequently a decreased NP value (0.82) within the recall. Likewise, for the RF model, two SVM folds show average statistical measures (precision, recall, f1-score) of 1.0 (Table 13). In contrast to the RF model, the SVM model only exhibits one fold with decreased precision for OP and decreased recall for NP (Table 13, fold 1), while the RF model exhibits two of such folds (Table 12, fold 1 and 4). Equally to the RF model, which showed different feature importance for the different folds (Figure 23), the SVM grid search leads to different parameters of the model (Table S12). While folds 1, 2 and 5 exhibit a rbf kernel, folds 3 and 4 exhibit a polynomial kernel. However, folds 2 and 3 both reach scores of 1.0 for all statistics, although different kernels were chosen.

As a third approach, an NN was created with the architecture of two hidden layers with five nodes each and one output node (Figure 24A), which was chosen via a trial and error approach. As an input of the NN, like for the RF and SVM models, the different filters were used leading to 16 different input nodes (Figure 24A). The activation function was set to sigmoid for both hidden layers. After both hidden layers a dropout (0.1) was added, which was clarified with a grid search (Figure 24B).

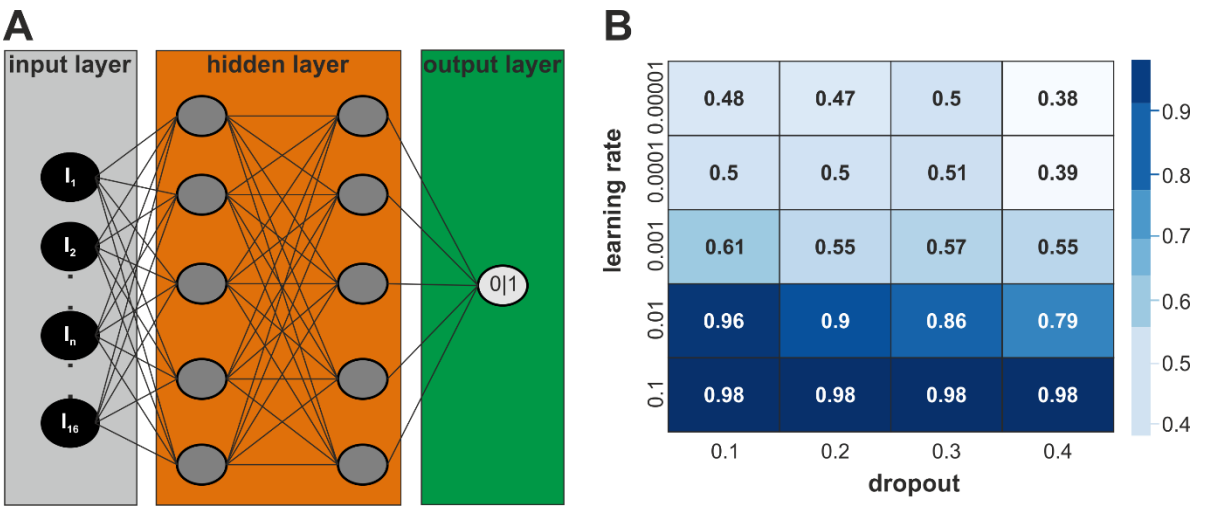


Figure 24: Composition and grid search result for neural network. The different layers for the neural network (A) are displayed in colors. Grid search for different dropouts and learning rates (B) lead to different accuracies. The output of "0" thereby refers to NP, while "1" is assigned to OP.

Even though the accuracies for the dropouts do not show greater differences between a dropout of 0.1 and 0.4, the dropout of 0.1 and learning rate of 0.1 resulted in the best accuracy (Figure 24B). Consequently, also the learning rate was set to 0.1. In contrast to the RF and SVM models, the architecture was not clarified via a grid search leading to a simple 5-fold cross validation. Beside the typical split of 80% test and 20% training data, 20% of the training set were used as validation set. Thereby, the NN model showed a decrease of validation and training loss over the 50 epochs, as well an increase of the training and validation accuracy (Figure 25).

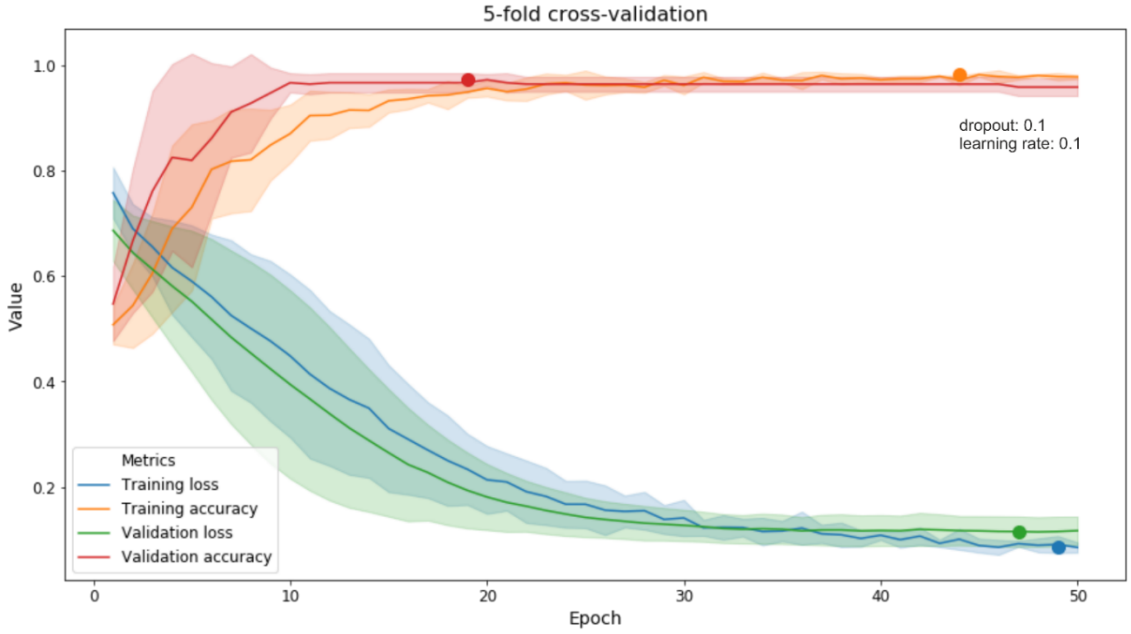


Figure 25: Training performance of neural net for 50 epochs. History of metrics (accuracies of training and validation, loss for training and validation) during training for the 5-fold cross validation. Shaded areas show the standard deviations of the corresponding metric for each epoch across the rounds of the cross-validation. The minimum / maximum values are marked by a large dot.

The validation and training loss reach their minimum at 47 to 49 epochs. Also, the standard deviation decreases (Figure 25). Compared to this, the accuracy for the validation reaches the optimum at 19 epochs, while the accuracy of the training reaches the maximum at about 45 epochs. Compared to the RF and SVM approaches, the NN shows an increased average measure for precision, recall and f1-score with each exhibiting 0.98 (Table 14), while for the SVM, these averages were at 0.97 (Table 13) and for RF at 0.96 (Table 12).

Table 14: Performance measure of the training of the neural net classifier (filter data). For each of the 5 folds the precision, recall and f1 score is shown.

	fold 1	fold 2	fold 3	fold 4	fold 5	**Average**	
NP	0,95	1,00	0,98	0,97	0,98	**0,98**	
OP	1,00	0,97	0,98	1,00	0,98	**0,99**	**Precision**
average	0,97	0,98	0,98	0,98	0,98	**0,98**	
NP	1,00	0,96	0,98	1,00	0,98	**0,98**	
OP	0,95	1,00	0,98	0,96	0,98	**0,97**	**Recall**
average	0,97	0,98	0,98	0,98	0,98	**0,98**	
NP	0,97	0,98	0,98	0,98	0,98	**0,98**	
OP	0,97	0,98	0,98	0,98	0,98	**0,98**	**F1-score**
average	0,97	0,98	0,98	0,98	0,98	**0,98**	

Also comparing the performance of the different folds for the NN model to the SVM and RF models, no fold is observable where the precision for OP and recall for NP is decreased compared to the other folds.

In general, the three machine learning models RF, SVM and NN showed to fit onto the provided dataset and to be able to identify OPs and NPs out of the dataset. Thereby, it could be observed for all three models that they lead to high measures of accuracy, AUC, precision, recall and f1-score. However, the fit onto the provided dataset does not reflect an increased predictive ability of predicting an operon. Therefore, for getting a greater overview of the different prediction models (plain predictor, RF, SVM, NN), operons were predicted in *Anabaena* sp. PCC 7120 with the RF, SVM and NN and compared to the defined operon set of the plain predictor (by leaving out the training set operons, Figure 26).

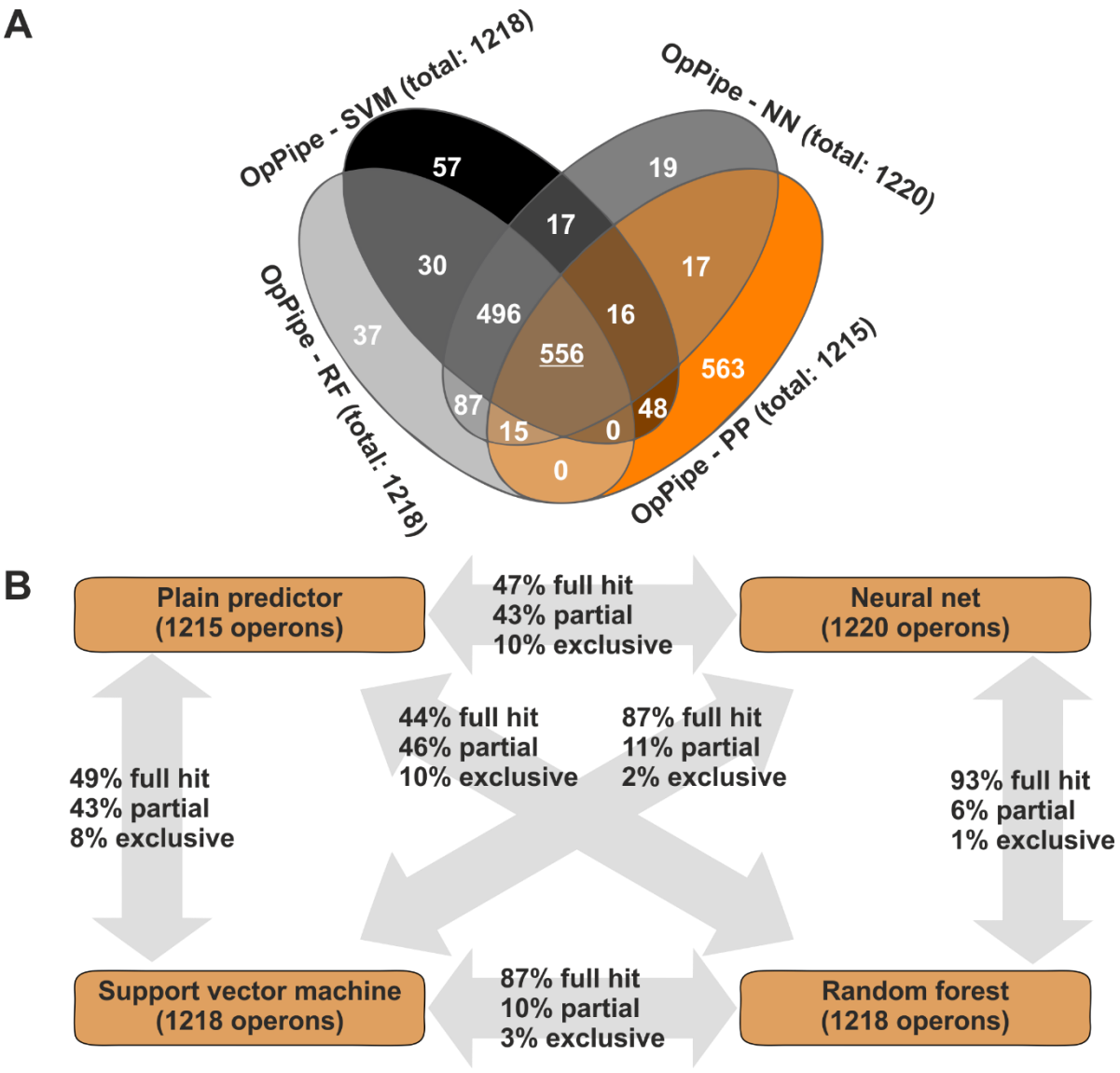


Figure 26: Comparison of different operon prediction models based on *Anabaena* sp. PCC 7120. (A) Number of operons that have been identified as full hit among different classifiers. (B) Indicated are four different classifiers. Grey arrows indicate the pairwise comparison of two classifiers, with the percentages inside the arrows showing the percentages of operons identified by both within the same composition (full hit), operons that overlap ore containing each other (partial) and operons that were only found by one of the classifiers (exclusive).

In general, all four approaches share 556 full hits (Figure 26A, Table S17). These operons seem to be conserved through all four predictors (Figure 26A). In general, the different machine learning algorithms show a high overlap of full hit operons with each other. For the machine learning algorithms, 40-60 operons remain partial or exclusively found, while for the plain predictor around 563 remain partial or exclusively found operons (Figure 26A). Further, the different machine learning algorithms share at least 87% of their predicted operons as full hits. In this context, the NN and RF classifiers share the most full hit operons (93%), while 6% are partially found and only 1% are exclusively found operons (Figure 26B). Comparing the SVM to the NN, 87% of the predicted operons are full hit, 11% are partial and 2% are exclusive operons. In both comparisons, the exclusive operons refer to the RF and SVM models, as all

predicted operons of the NN could be found as full hit or partial hit in the RF and SVM sets. Even though the NN classifier predicted more operons (1220, Figure 26A) than RF and SVM (1218, Figure 26A), the majority of operons when comparing the NN to the RF and SVM are partial hits. This means that the operons of RF and SVM either contain the NN operons or they overlap at the start or end positions. Comparing the RF, SVM and NN to the plain predictor, 9% (SVM) and 10% (RF, NN), respectively, are exclusively found operons. The remaining operons are divided nearly to equal parts into full and partial hits. Using the 556 operons conserved around all four OpPipe predictors (plain predictor, RF, SVM, NN, Table S17), 80 of them could be found in the same composition within DOOR, ProOpDB and Rockhopper (Figure 27).

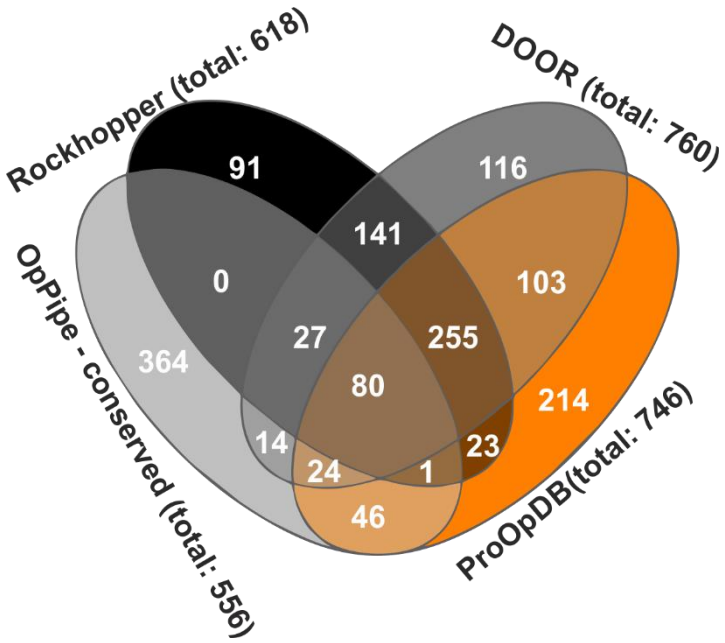


Figure 27: Venn diagram of the predicted operon set of three predictors and OpPipe conserved set in *Anabaena* sp. PCC 7120. The diagram shows the number of total predicted operons (total number in brackets behind name) for the overlap of the OpPipe predictors (plain, RF, SVM, NN) and Rockhopper [36], DOOR [65], ProOpDB [69] (only full hits are counted).

However, around 65% of the different predictor conserved operons could not be found as a full hit (364) in the sets of DOOR, ProOpDB and Rockhopper (Figure 27). In addition, although the machine learning approaches (RF, SVM and NN) showed high statistical measures on the defined train/test dataset, comparing the overlaps between the machine learning models and the operon sets of DOOR, Rockhopper and ProOpDB, the general number of operons that could be found in the same compositions could not be enhanced using machine learning approaches onto the filter models (Table S16). While the plain predictor exhibited 200-300 operons in the overlap to DOOR, ProOpDB and Rockhopper, the different machine learners exhibiting around 140-200 operons in the overlap with these predictors. However, comparing the operons of the four created prediction models being not found in DOOR, ProOpDB and

Rockhopper (Table S16, NH2), the machine learning algorithms exhibit a lower number of operons not found compared to the plain predictor.

Even though the overlap to other prediction tools could not be increased by the use of machine learning, the different OpPipe predictors show an increased correlation to the average operon length (number of genes within an operon) compared to the literature set (Table 15).

Table 15: Average length of predicted operons from *Anabaena* sp. PCC 7120. For the literature set and the different predictors (DOOR [65], ProOpDB [69], Rockhopper [36], Plain predictor, NN predictor, SVM predictor and RF predictor), the average number of genes in an operon of all predicted operons in *Anabaena* sp. PCC 7120 is calculated. The smallest (Min) and largest (Max) operons are displayed as well as the ones with less than five genes and more than four genes.

Predictor	Average operon length	Min operon length	Max operon length	% <5	% >4
Literature set	4.55	2	10	61	39
DOOR	2.41	2	12	97	3
ProOpDB	2.54	2	32	95	5
Rockhopper	2.29	2	9	98	2
Plain predictor	3.22	2	37	84	16
RF predictor	3.23	2	18	85	15
SVM predictor	3.30	2	18	84	16
NN predictor	3.17	2	18	86	14

Comparing different OpPipe classifiers for the operons of *Anabaena* sp. PCC 7120 DOOR, Rockhopper and ProOpDB result in 95-98% of operons with a length smaller than five genes. All OpPipe predictors predict ~85% with less than five genes (Table 15), while about 61% of the operon literature set are smaller than five. The different OpPipe machine learners thereby show again a high correlation for minimal (2) and maximum (18) operon length. While DOOR and Rockhopper show a maximum operon length of ten to twelve genes, ProOpDB and the plain predictor show an increased maximum operon length of 32 to 37 genes within an operon. Interestingly, the operon with the maximum length in the ProOpDB and plain predictor operon set refers to the same gene cluster (in plain predictor all4179-rplC) in *Anabaena* sp. PCC 7120. This gene cluster is separated into smaller operons by the other machine learning algorithms, as well as for DOOR and Rockhopper. It is observable that the NN exhibits the fewest average operon length with 3.17 compared to the plain predictor (3.22), RF (3.23) and SVM (3.30) (Table 15). This leads to an increased number of operons, which are shorter compared to the other models. In general, the operon sets of the OpPipe predictors fit better to the average length of the literature operon sets (4.5) than the operon sets of DOOR (2.41), ProOpDB (2.54) and Rockhopper (2.29) which categorize on average one gene less to an operon than the OpPipe predictors (Table 15).

Regarding the number of all annotated genes of *Anabaena* sp. PCC 7120 being categorized into operons by different predictors, the OpPipe predictors show to categorize a decreased number of genes into operons compared to DOOR, ProOpDB and Rockhopper (Table 16).

Table 16: Number of annotated genes (%) of the *Anabaena* sp. PCC 7120 genome that have been classified into operons. Displayed are the different predictors of DOOR [65], ProOpDB [69], Rockhopper [36], Plain predictor, RF, SVM and NN.

Predictor	Species	% of genes covered
DOOR	*Anabaena* sp. PCC 7120	34%
ProOpDB	*Anabaena* sp. PCC 7120	35%
Rockhopper	*Anabaena* sp. PCC 7120	27%
Plain predictor	*Anabaena* sp. PCC 7120	74%
RF predictor	*Anabaena* sp. PCC 7120	74%
SVM predictor	*Anabaena* sp. PCC 7120	76%
NN predictor	*Anabaena* sp. PCC 7120	73%

In general, from the OpPipe predictors, the SVM predictor classifies the highest number of genes (76%) of *Anabaena* sp. PCC 7120 into operons, while the NN assigns the fewest number (73%) of genes inside an operon (Table 16). However, the predictors of DOOR, ProOpDB and Rockhopper show to only classify about 27-35% of all genes of *Anabaena* sp. PCC 7120 into operons (Table 16).

The different OpPipe predictors showed to identify operons on the provided training/test dataset, while the different machine learners showed a higher overlap between each other compared to the plain predictor. However, 556 operons were found to be conserved in all predictors. Even though the overlap with other prediction tools like DOOR, ProOpDB and Rockhopper could not be increased by the different machine learning algorithms, all OpPipe predictors show an increased correlation with the literature set of *Anabaena* sp. PCC 7120 regarding the average length of the operons. In this context, the four OpPipe predictors show a higher coverage of genes being part of operons compared to DOOR, ProOpDB and Rockhopper in *Anabaena* sp. PCC 7120.

4. DISCUSSION

4.1. Intergenic distance is the major criteria for operon prediction and strongly dependent on the derived species

Nowadays, numerous operon prediction tools and databases exist (Table S1) [26] [27]. Most of them rely on functional features like the intergenic distance, functional relation or evolutionary features. However, it has been demonstrated that a very simple variant using only intergenic distance outperforms several approaches which take into account numerous criteria (e.g. functional and evolutionary relation) [27] [47]. This has for example been proven by Brouwer *et al.* [27], who demonstrated that the operon predictor of Moreno-Hagelsieb *et al.* [169], which relies only on the intergenic distance, outperforms several tools which take into account numerous operon features. In fact, researchers like Brouwer *et al.* [27], Bockhorst *et al.* [70] and De Hoon *et al.* [170] showed that the intergenic distance was, overall, the most accurate feature. Therefore, the intergenic distance may be the driving factor for the prediction of operons [27] [47] [59]. Further, it has been demonstrated that the intergenic assumption can be universally applied to bacterial genomes [47]. In congruence with these assumptions, the plain predictor model was created by assigning an enhanced score to gene pairs if they are considered to be OP by the DGD (Figure 12). Further, the feature importance of the RF model showed an even greater importance of the intergenic distance feature (Figure 23) compared to the scoring of the plain predictor model. Interestingly, also most operon predictors are taking this feature into account (Table S1). A problem with this is that numerous operon prediction tools are trained and tested on sub-sets of operons derived from model organisms like *Escherichia coli* and *Bacillus subtilis* [25] [47] [65] [69]. A driving feature thereby is that for these species, a sustainable number of operons have been proven experimentally [27]. A short coming of this is that applying a pre-trained predictor onto an unknown species may result in misclassification.

This is the fact because it has been revealed that intergenic distances of genes inside an operon vary among species [35] [171]. The same effect was observed when comparing the average intergenic distances between five different species within this study (Figure S5, Figure 11). In general, the genome of *Escherichia coli* seemed to be packed more densely compared to other species like *Bacillus subtilis* or different cyanobacteria (Figure 11). Therefore, using the intergenic distance of *Escherichia coli* can be less effective when applying it to other species. For example, applying a trained model of *Escherichia coli* to *Bacillus subtilis* could result in an accuracy drop due to the low ability of generalization [65]. This phenomenon was described within this study by applying intergenic distances of different species to other species (Table 10, Table S11). It could be observed that applying an intergenic cut-off of *Escherichia coli* to *Anabaena* sp. PCC 7120 leads to an increased FN rate as this cut-off is too strict (Table

10, Table S11). Swapping sides, the intergenic distance of *Anabaena* sp. PCC 7120 leads to an increased FP rate in *Escherichia coli* as this cut-off is too lose (Table 10, Table S11). Such an effect has also been shown by Dam *et al.* [65] when they tried to apply their length ratio from *Escherichia coli* to *Bacillus subtilis.*

To address this issue, the developed DGD filter (Table 9) poses the central question of how to easily define a dynamic distance cut-off that is applicable across species. However, even the DGD within the different ranges showed satisfying results but could be optimized with leaving out the distance ranges. The cut-off then becomes more dynamic by directly searching the distance between two adjacent genes that are present for more than 45% of all adjacent gene pairs. However, the DGD filter with the distance ranges proved to identify operons proven in the literature with high confidence and also proved the necessity of applying this feature in a dynamic way (Table 10, Table S11).

4.2. *In-silico* methods demonstrate improved ability of operon identification compared to genome wide TSS identification

Beside classical *in-silico* approaches, another way of identifying operons is via TSS. However, the approach showed to have limitations. Like Mitschke *et al.* [56] mentioned, on the one hand, for *Synechocystis sp.* PCC 6803, ~87% of the genome should be coding. On the other hand, Mitschke *et al.* [56] resumed that only one third of the TSS are located upstream an annotated gene. Thereby, it has been mentioned by Mitschke *et al.* [56] and Choe *et al.* [20] that ~64% of all TSS in *Synechocystis* give rise to antisense or non-coding RNAs.

Congruently to this, the data for *Anabaena* sp. PCC 7120 demonstrated that only one third of the TSS are located directly before or within genes, while ~45% give rise to non-coding elements (Figure 18). As for example in *Escherichia coli,* Choe *et al.* [20] mentioned that promoters of operons showed to exhibit multiple TSS. Applying these to *Anabaena* sp. PCC 7120 and screening for TSS before (or inside) operons, at least one of these multiple TSS should be found for each operon. However, for all predictors (OpPipe plain predictor, DOOR, Rockhopper, ProOpDB) used in this study, the most operons (~60%) did not exhibit a TSS directly before or even within the operon (Figure 20). Congruently to this, the defined literature set of *Anabaena* sp. PCC 7120 showed a decreased number of operons exhibiting a TSS before or within the operon (Figure 22). Interestingly, the *pec* operon mentioned by Swanson *et al.* [161] which is part of the literature set showed to be expressed under control conditions through the developed filters (Figure S3, Table S9). Further, this operon could be shown to exhibit TSS (Table S9) that were also identified by the experiment of Mitschke *et al.* [56]. The

same effect can be observed for the *fraC* operon mentioned by Merino-Puerto *et al.* [157], where with the data of Mitschke *et al.* [56], two TSS directly before the operon (within 200 nucleotides) could be identified (Figure 28). Both of these TSS were within the defined cut-off of 50 and 100 reads that have to be matched to them to be counted (Figure 19). Additionally, the *fraC* operon shows two weaker TSS within the operon. In contrast to *pec* and *fraC* operon, the nitrate assimilating *nir* operon mentioned by Frías and Flores [51] shows for both cut offs (50 and 100, Figure 22) no TSS before the operon under control conditions. With the TSS identification approach, this operon remains unfound under control conditions. As with the GG filter an increased expression can be observed for the *nir* operon under -Nit conditions compared to control conditions (Table S10) this might lead to an identification of this operon via the TSS approach under a different condition.

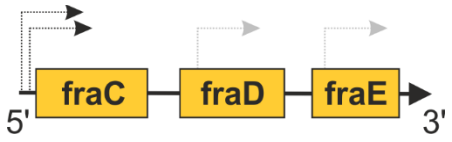


Figure 28: TSS for the *fraC* operon [157]. Black dashed arrows indicate a TSS above 50 and 100 reads, while grey dashed indicates TSS below the cut-off.

The operon TSS analysis within this study shows the disadvantages of an operon prediction only relying on TSS. The identification of operons by a TSS-only based approach is limited to the current used condition because reliance on TSS data only shows active TSS under this specific condition. Therefore, an approach only based on TSS leaves out numerous putative operon classification, as their TSS are not detectable under a specific condition. The inclusion of *in-silico* methods thereby improves the ability to identify operons which are not or only weakly expressed during a specific condition.

4.3. Increased ability of operon prediction through generalized filter models based on expression and intergenic distance

For the prediction of operons, the features of expression and intergenic distance have been used within this study. Comparable operon prediction tools often use phylogenetic inference, functional relations or promoter/terminator site identification for their predictions [47] [69] (Table S1).

OpPipe does not consider functional relations and conservation for the prediction model. On the one hand this is because a functional or evolutionary information is not available in databases for every species. This would lead to a species-specific prediction tool, which was not the aim of OpPipe as it should be applicable onto a broad range of species. On the other hand, operons often harbor genes which can be functional related (e.g. coding for enzymes that catalyze steps in a pathway or genes which are members of a protein complex) and thereby ensure a co-regulation of these genes [141] [172] [173]. However, there are examples where genes of an operon are functional unrelated [24]. In this context, the *rpsU-dnaG-rpoD* operon in *Escherichia coli* encodes the 30S ribosomal protein S21, DNA primase and RNA polymerase and thereby contains genes involved in different pathways [65]. Further, newly formed operons might consist of genes which do not share a similar function, but they may be needed in the same environmental condition.

In the same context, OpPipe does not make use of evolutionary relations. Although functional operons are thought to be stable over time and be present among different prokaryotic species [24], the formation of operons is still not completely understood [24] [46] [60] [141]. The formation of genes into operons is often described as “self-defense” or of operons being “selfish”, as these formatted genes ensure via horizontal gene transfer (HGT) to not be removed from the genome [27] [60] [171]. However, the composition of genes inside an operon can be very diverse compared between species [24] and many operons undergo shuffling events during evolution [174].

This makes it difficult to identify conserved operons, as they exhibit the same genes but in a shuffled order [47]. For this purpose, like functional relations, the evolutionary connections were not used for the operon prediction. In general, enriching the prediction with expression data like DNA microarray or RNA-Seq data showed to improve the prediction for example for the tools of Rockhopper and CONDOP [27] [31] [46] [175]. However, even though numerous operon prediction tools predict operons with a high amount of sensitivity and specificity, they are only specialized and rely on a selected subset of species [27] [59] [65].

To address this issue, OpPipe aims to identify shared features of operons among species and to convert them into uniform data before using them for operon prediction. Within OpPipe this is achieved by assuming a general “operon behavior” previously and trying to monitor these behaviors via filters. By this, adjacent genes of an operon should be co-transcribed which is monitored by the developed EP and GG filters (3.3). Further, operonic genes are assumed to exhibit a specific intergenic distance which is monitored by the DGD filter (3.4).

However, even though a generalized “operon behavior” through the filters is assumed, it is not guaranteed that this does in fact happen. Despite the best fit to a model organism, an operon predictor might still lack the general classification function as the classifier overperforms when applied to another species [65]. Therefore, the dataset was tried to be as broad as possible by including different species (model species *Escherichia coli,* gram positive *Bacillus subtilis* and *Corynebacterium,* cyanobacteria *Anabaena* sp. PCC 7120, *Synechococcus elongatus, Synechocystis* sp. PCC 6803). A problem when comparing the results of a developed predictor with other tools is, on the one hand, the specificity to a given model organism. On the other hand, prediction tools often evaluate their accuracy by comparing their results to experimentally proven operons, resulting in the dead end that the used literature sets often differ between the studies [27]. Further, the comparison of statistical measures often relies on gene pairs, while the setup also differs.

To address this issue, OpPipe was benchmarked against other tools as universal as possible. For this purpose, there was no comparison of statistical measures with other prediction tools but rather, it was attempted to find the overlap of fully predicted operons between the OpPipe predictors, different tools, databases, and literature proven operons (Figure 13, Figure 14, Figure 15, Figure 16, Table S16). On the model organism *Escherichia coli,* the plain predictor showed encouraging results by exhibiting an increased overlap of predicted operons with the Regulon database compared to the other predictors (Figure 13, Figure 14). However, even in this study, a defined literature set was used to gather a possible hint for the correctness of the developed model. In general, the plain predictor showed to outperform the other compared predictors (Table S14) for the literature sets of *Escherichia coli* (Table S13) and *Anabaena* sp. PCC 7120 (Table 11, Figure 17). Although contradicting results on the *Anabaena* sp. PCC 7120 set occurred when comparing it to other predictors, OpPipe showed to perform best on the defined literature operon set for *Anabaena* sp. PCC 7120 compared to the other predictors and also showed to be able to confirm these results on another species *(Escherichia coli).* In general, the comparison of the plain predictor with the other tools and databases based on these literature sets shows encouraging results, as the plain predictor has not been trained onto these data.

However, when focusing on *in-silico* methods, finding the right classification model is always difficult [73]. For this purpose, for this study it was decided to rely on multiple classifiers (plain predictor, NN, RF, SVM). Even though the impression of overprediction could arise when regarding the results of OpPipe in *Anabaena* sp. PCC 7120, during this study, it has been demonstrated that for example the average length of operons of the OpPipe predictors is correlated in a greater way with the defined literature set than the predictors of DOOR, ProOpDB and Rockhopper did (Table 15). In this context, OpPipe also shows an increased correlation regarding the operon length of the defined literature set in *Escherichia coli* and the operon length of Regulon database (Table 17). Comparing the average length of predicted operons in *Escherichia coli* and *Anabaena* sp. PCC 7120 between different predictors (Table 15, Table 17), it is observable that the operons of the plain predictor tend to be elongated compared to the predictors of DOOR, ProOpDB and Rockhopper. Compared to the Regulon database with a length of on average 3.07 genes in an operon in *Escherichia coli,* the plain predictor assigns on average 3.37 genes to an operon. However, DOOR only assigns 3.07 genes to an operon while Rockhopper only assigns 2.86 to an operon. Further, the sets of the OpPipe plain predictor and the Regulon database also correlate to a greater extent to the literature set, where 4.5 genes are part of an operon. Comparing the results of *Escherichia coli* to the operons of *Anabaena* sp. PCC 7120, Rockhopper assigns on average 0.57 times fewer genes and DOOR 0.66 tomes fewer genes to an operon, while the plain predictor assigns on average 0.15 times fewer genes into an operon compared to *Escherichia coli.*

Table 17: Average length of predicted operons from *Escherichia coli.* For the literature set and the different predictors (DOOR [65], RegulonDB *[147]*, Rockhopper [36], Plain predictor), the average number of genes in an operon of all predicted operons in *Escherichia coli* is calculated. The smallest (Min) and largest (Max) operons are displayed as well as the ones with less than five genes and more than four genes.

Predictor	Average operon length	Min operon length	Max operon length	% <5	% >4
Literature set	4.40	2	11	55	45
DOOR	3.07	2	16	85	15
RegulonDB	3.23	2	16	82	12
Rockhopper	2.86	2	12	88	12
Plain predictor	3.37	2	20	80	10

Interestingly, the plain predictor also shows a greater correlation of operons with less than five genes comparing *Anabaena* sp. PCC 7120 and *Escherichia coli* than DOOR and Rockhopper. Salgado *et al.* [58] showed that in *Escherichia coli,* based on operons extracted from the Regulon database, about 80% of all TUs are composed of less than five genes. This observation can be confirmed with the used dataset of the Regulon database (Table 17, 82%). Comparing the results of the different predictors (Table 17), the OpPipe plain predictor (80%) shows a greater correlation with this assumption than DOOR (85%) and Rockhopper (88%). Transferring this assumption to *Anabaena* sp. PCC 7120 (Table 15), the predictors of DOOR

(97%), ProOpDB (95%) and Rockhopper (98%) show an increased number of operons with less than five genes. However, the different OpPipe predictors show to correlate better with the assumption of ~80% of the operons being shorter than five genes, because they predict ~85% of the operons being shorter than five genes.

The two literature sets of *Escherichia coli* and *Anabaena* sp. PCC 7120 thereby seem to contain comparatively long operons as only 55% (Table 17, *Escherichia coli*) and 61% (Table 15, *Anabaena* sp. PCC 7120) consist of less than five genes. However, it is observable, that the length of the operons in the literature sets does not affect the identification of operons via the OpPipe prediction approaches as they exhibit (i) a shorter average length compared to the literature set of *Escherichia coli* (3.37 average length plain predictor and 4.4 average length literature set, Table 17) and *Anabaena* sp. PCC 7120 (3.17-3.3 average length OpPipe predictors and 4.55 average length literature set, Table 15), (ii) correlating with the assumed 80% of operons that exhibit less than five genes (80-85%, Table 15, Table 17).

Subsequently, it is stated in Brouwer *et al.* that 50-60% and in Moreno-Hagelsieb *et al.* [162] that around 60% of genes should be organized in operons. When comparing OpPipe to other predictors regarding the number of annotated genes categorized into operons by them, it is again remarkable that OpPipe tends to outperform the other predictors of DOOR, ProOpDB and Rockhopper (Table 16, Table 18). For *Escherichia coli*, OpPipe shows comparable results to the Regulon database with approximately 60% of the annotated genes being predicted into operons by these two approaches (Table 18). Rockhopper thereby only classifies 47% of the annotated genes into operons (Table 18).

Table 18: Number of annotated genes (%) of the *Escherichia coli* genome that have been classified into operons. Indicated are the different predictors of DOOR [65], RegulonDB [147], Rockhopper [36] and Plain predictor.

Predictor	Species	% of genes covered
DOOR	*Escherichia coli*	56%
RegulonDB	*Escherichia coli*	59%
Rockhopper	*Escherichia coli*	47%
Plain predictor	*Escherichia coli*	61%

Regarding the *Anabaena* sp. PCC 7120 results (Table 16), OpPipe predictors seems to rather fulfill the 50-60% assumption, as with ~74% of covered genes, 14% are classified falsely positive compared to 60% of genes that where expected in operons basing on Moreno-Hagelsieb *et al.* [162]. However, for tools like Rockhopper, 34% of potential operon genes are missing. Interestingly, the machine learning approaches assign nearly the same number of genes into operons, showing the high overlap of these approaches.

In combination with the accurate prediction of OpPipe, the visualization of these results enhances the analysis process. By this, interesting operons can easily be monitored through the visualization tool. Such an interesting operon is for example the *rbcLS* operon in *Synechococcus* sp. PCC 7942. Quintana *et al.* optimized the energy production by metabolic pathway engineering through inclusion of coding sequences for pyruvate decarboxylase and alcohol dehydrogenase II from *Zymomonas mobilis* to produce ethanol [176]. To increase the yield, they were expressed under the control of the *rbcLS* operon promoter. For monitoring the expression of fused operon products like proposed in Liang *et al.* [177] and Quintana *et al.* [176], a tool like OpPipe offers a very useful way of monitoring the expression of this operon under different conditions.

In fact, with the visualizer of OpPipe, it is easy to monitor the high expression of the *rbcLXS* gene cluster under the different conditions of control, -Nit and -Fe (Figure 29). The operon shows to be identified by the OpPipe plain predictor with an elongation of three genes (alr1527-alr1529) showing an increased expression of this gene cluster. The extension of this operon under control conditions thereby shows the importance of operon prediction and the visualization of these results, as maybe not only the core operon is getting expressed but rather further promoters and genes might be part of the regulation of this transcription unit. Additionally, like Picossi *et al.* [168] mentioned, the *rbcLXS* operon in *Anabaena* sp. PCC 7120 encodes for RuBisCo, which facilitates the CO_2 fixation. If the regulator of this operon LysR is knocked down, the mutant is not able to survive under high light. Such knock down experiments can easily be monitored with a tool like OpPipe. Thereby, the usefulness is that if an unknown regulator gets knocked down, it can easily be monitored, which genetic regions and control units this knock down has an effect on. A further useful scope of application of OpPipe and the included visualizer is that especially under different conditions, it is possible to identify highly expressed operons. Through the comparison under different conditions, such operons that are highly expressed under various conditions can be identified and can be exploited to produce other substances.

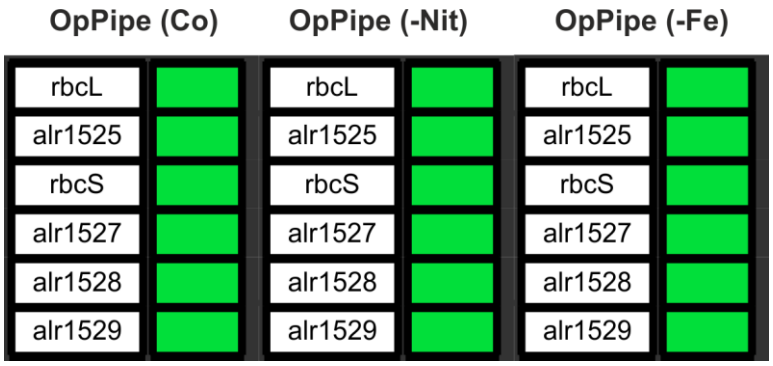


Figure 29: Visual operon prediction of the *rbcLXS* [166] operon of *Anabaena* sp. PCC 7120 by OpPipe (under control, -Nit and -Fe conditions). Green colour indicates an operon.

The results of OpPipe compared to other prediction tools are encouraging and suggest that OpPipe is suitable for the operon prediction and leads to a more generalized model compared to the predictors it was compared to. Further, the defined filters and visualization is a useful tool for monitoring the regulation on the level of transcription of a given species.

4.4. Identification of alternative TUs through operon prediction filters

Despite the plain operon prediction, the search for alternative TUs also remains a challenging task. Alternative TUs thereby are detectable through expression data or TSS. Like Cao *et al.,* Chen *et al.* and Mao *et al*. suggested, operons or not non-overlapping units but rather multiple TUs can arise from an operon through alternative TSS [4] [20] [21]. However, currently available operon prediction tools typically describe operons as non-overlapping elements under a specific condition [21] and do not list alternative TUs [27]. In this context, ~35% of promoters in *Escherichia coli* harbored multiple TSS, which results in multiple transcription units [20]. Therefore, it can be assumed that this number is alike in *Anabaena* sp. PCC 7120. Thereby, prediction tools predicting operons in a non-overlapping way are leaving out this information.

Within OpPipe the monitoring of alternative TUs is ensured by the different expression-based filters that were created. For example, the EP filter evaluates the expression landscape. Therefore, it is, beside the plain operon prediction, best suited for the search for alternative TUs and different regulated TUs under various stress conditions. For example, differences for the *pec* operon under different stress conditions can be monitored using the EP filter (Table 19).

Table 19: *Pec* operon from *Anabaena* sp. PCC 7120 under different EP stringencies and different stress conditions.

EP stringency	Operon composition (control)	Operon composition (-Nit)	Operon composition (-Fe)
0.98	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE
0.95	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE
0.8	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE
0.7	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE
0.6	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE
0.5	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE
0.4	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE
0.3	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA, pecC
0.2	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	pecB-pecA
0.1	pecB-pecA-pecC-pecE	pecB-pecA-pecC-pecE	-
0.05	pecB-pecA-pecC-pecE	pecB-pecA, pecC	-

While the *pec* operon proves to be stable through all stringencies under control conditions, it starts to decrease under -Nit an -Fe conditions.

In this context, the GG filter under different stress conditions also showed to be useful to monitor differences in expression levels of operons (-Nit, -Fe, Table S10). For example, the connection of pecB-pecA is covered by ~15804 normalized reads under control conditions but is downregulated under -Nit as it only exhibits ~4415 normalized reads and under -Fe conditions, where only ~565 normalized reads mapped onto the connection. A similar trend is observable for all genes and connections within the operon for the three conditions. In contrast, the *nir* operon (Table S10) exhibits fewer reads connecting the genes under control conditions compared to the *pec* operon. In contrast to control conditions, the number of reads connecting the genes of the *nir* operon is increased under different stress conditions (Table S10). Under -Fe conditions, the single genes exhibit a highly increased number of reads that cover them compared to control conditions (e.g. nirA ~162 normalized reads to ~1175). However, this effect is even more drastically when comparing control conditions to -Nit conditions (Table S10), where for example nirA is covered by ~48172 normalized reads. Interestingly, it is observable that the number of reads connecting two genes within the operon is heterogenous, which could hint to alternative TSS within this operon. Additionally, the *fraC* operon seems to be constitutively expressed under control conditions but also shows an increase of reads mapped onto the genes under -Nit and -Fe conditions (Table S10). Interestingly, the 3' end of the operon shows an increased expression compared to the 5' end in all three conditions.

For example, in the cyanobacteria *Anabaena* sp. PCC 7120, Simm *et al.* [178] identified several gene clusters. Sustainable differences within the PKS gene cluster can be obtained within different stress conditions, as well between the different predictors (Figure S7, Figure 30). Thereby OpPipe identifies eight different operons out of this gene cluster, while DOOR (4 operons), Rockhopper (2 operons) and ProOpDB (6 operons) identify fewer operons (Figure S7). Under different stress conditions (control, -Nit, -Fe), a similar regulation of the cluster can be observed for several TUs (Figure S7). For example, the section of alr5331-all5347 shows the same characteristics of TUs. However, several differences in the occurrence of TUs within the stress conditions can be observed. Especially the section between alr5348-alr5357 can easily be monitored with the visualizer of OpPipe regarding changes of the TU creation (Figure 30). Comparing different predictors, OpPipe proves to identify four different operons out of this section under control conditions, while DOOR and Rockhopper do not identify any operon within this section (Figure 30A). Moreover, differences between the conditions of control, -Nit and -Fe are monitorable with OpPipe. For example, the sections between asr5349-alr5355 show to be differently regulated under the three observed conditions (Figure 30B).

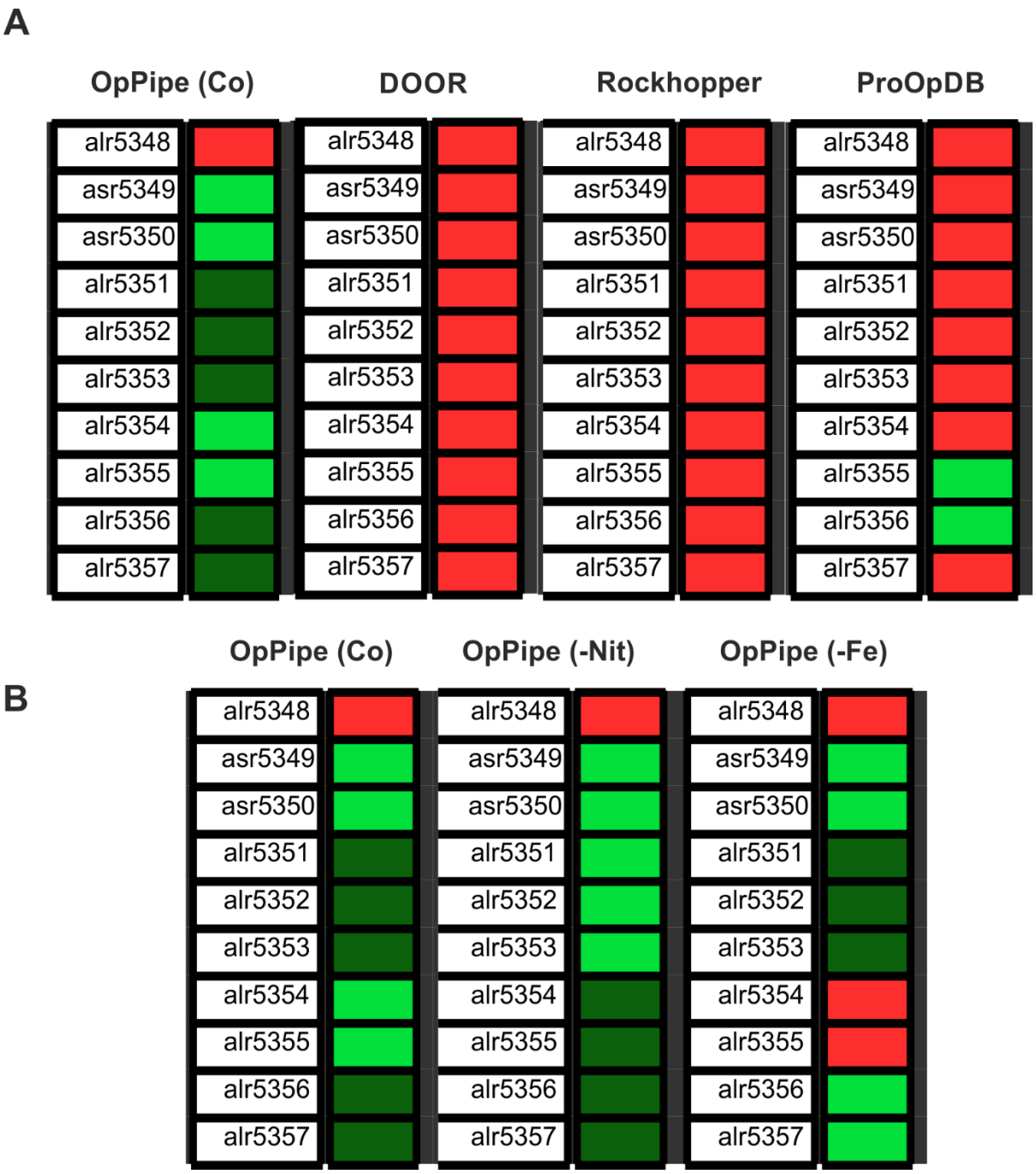


Figure 30: Visual operon prediction for a section of the PKS gene cluster [178] of *Anabaena* sp. PCC 7120 by OpPipe (under control, -Nit and -Fe conditions) and ProOpDB [69], DOOR [65], Rockhopper [36] (control conditions). Red color indicates that no operon has been identified, green color indicates an operon, while a darker green indicates that two operons have been predicted.

To properly describe and understand newly characterized cluster OpPipe is a very helpful tool. Like Simm *et al.* [178] mentioned, the PKS cluster gene is devoted to important cellular processes in cyanobacteria such as iron uptake and nitrogen fixation. With OpPipe the different regulations of this cluster are traceable among different conditions regarding iron and nitrogen uptake. OpPipe can firstly be used to predict regulatory operons out of these clusters and secondly for graphically displaying these predictions.

In general, a main advantage of cyanobacteria for biofuel production is the ability of fixing nitrogen directly from the atmosphere [179] [180]. They are therefore not dependent on a high amount of fertilizer in form of nitrogen like crop plants [181]. However, to make use of such organisms, it is essential to understand gene clusters responsible for the fixation of nitrogen like the PKS cluster identified by Simm *et al.* [178].

Summarizing, the developed pipeline OpPipe proved to identify operons with a high amount of confidence. Thereby, the different developed filters proved to generalize the data of a species in the context of monitoring OP specific features. This transformation of species-specific data to operon-specific data leads to a high affinity of the prediction models for identifying operons in a non-species-specific way. For the prediction of operons, different filter models have been developed during this study, focusing on expression (EP, GG) and genomic features (DGD). However, it has been demonstrated that the intergenic distance is a major criterion for the operon prediction, which can be enhanced by being applied in a dynamic and generalized way. By the inclusion of genomic features, the identification of weakly expressed operons is also possible and demonstrates that OpPipe is not only dependent on the quality and strength of an expression experiment. Thereby, the comparison of different *in-silico* methods to an experimental approach of TSS identification showed the value of operon prediction tools. In this context, the developed four classifiers (plain predictor, RF, SVM, NN) proved to precisely identify operons from different species.

Moreover, with OpPipe changes within the regulation of the transcription due to changing conditions are easily traceable and therefore OpPipe offers the possibility of identifying different regulated TUs. One the one hand, this demonstrates that the inclusion of expression-based features through specific filters can be used for an operon prediction in a non-overlapping context. On the other hand, these filters add valuable information to the prediction regarding the regulation of alternative TUs.

5. FUTURE PERSPECTIVES

5.1. Validation of predicted operons via OpPipe

The developed framework OpPipe to predict operons and alternative TUs relies on functional genomic features (intergenic distance) as well as expression features and can be used for model and non-model organisms. The study showed that the different developed models in form of filters of OpPipe can identify operons and TUs. Further, it has been demonstrated that different features (e.g. intergenic distance) should be generalized previously in order to achieve a universally valid model. Even though the filters and algorithms showed high accuracy for these examples, they showed partially contradicting results for a whole genome approach. A future task for unbiased validation is the creation of broader standard data sets of operons from various species. So far, the validation is based on a small set of experimentally proven operons in few species. This extended dataset can serve as validation for a scoring model like the plain predictor and as training/test set for the machine learning algorithms. Thereby, only experimental proven operons and operons proven by the literature should be extracted from databases like the *Escherichia coli* based Regulon database [147], the *Bacillus subtilis* based DBTBS [182] or the species-independent operon database (ODB, [183]) to generate a set of OPs. Thereby, the identification could follow the approach of NP-set2 (2.6.1) by identifying NPs through the consensus of numerous databases and tools.

Further, it is essential to validate or check the predicted operons in the wet lab. Therefore, different approaches like northern blots can be used. As a starting point, different gene clusters identified in *Anabaena* sp. PCC 7120 by Simm *et al.* [178] could be used, while operons of *Escherichia coli* should also be tested to guarantee the universal application of the pipeline.

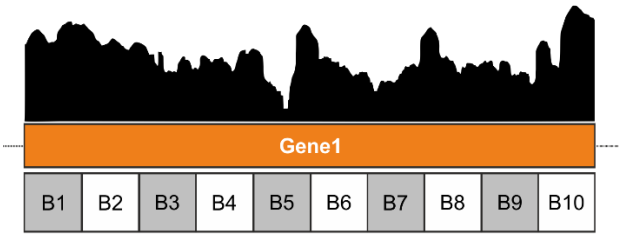
5.2. Improvements and adaptions of OpPipe

In general, the OpPipe is not a dead-end software project, as it is designed to be extended. Different modules can easily be fine-tuned and replaced and therefore offer a great possibility for further work. Thereby, it is clear that the developed GUI and pipeline needs adaptations and extensions. Currently, OpPipe is only available as minimum viable product (MVP). The major choice is if the MVP should serve as a server application or as a local tool, which is both feasible with the chosen architecture. Currently, the different sub-modules of OpPipe (e.g. gffReader, predictionModels) are integrated into one executable project to make it testable as a first MVP. However, for a complete pipeline, it is suggested to split the different packages into sub-projects which are then individually executable projects. This makes adaptions and changes easier. Subsequently, one project should thereby handle the control of the others.

As mentioned, the prediction models of OpPipe currently do not take into account functional and evolutionary functions. Nevertheless, as operons are assumed to be functionally and evolutionarily related, as mentioned for example by Brouwer *et al.* [27], Chen *et al.* [62] and Dam *et al.* [65], such features should not be ignored. However, to stay a generalized model, it is suggested to integrate such information into the predictors' output. The presence of conserved operons among species, which is called uber-operon [37], and functional relations could be used to enhance and validate the predictions given by the different predictors.

Further, the different filters of OpPipe are also in the focus of improvements. Although the applied filters (intergenic distance and expression) proved to be accurate, further features might enhance the operon prediction. A feature that might have a positive impact on the predictive performance is the codon usage, which is assumed to be related for genes inside an operon [35] [63] [64] [70]. This feature was not considered during this study and should be investigated. Further, the generalized expression features have only been created using the control data. As has been demonstrated, the assumption of "local optimal operons" under a given condition is not true [31]. Therefore, the filters have to be validated with regard to stress conditions and maybe the stress data should be included into the training/test data. Subsequently, the identification of alternating transcriptional elements under these conditions is a task. Alternative TUs thereby are detectable through expression data or TSS. Alternative TUs should be visible through the expression profile and it is expected to see an increase of mapped reads after the start of an alternative TU within an operon.

The current filter models proved to be able to identify different TUs under given conditions (Figure 30) but are having the drawback of only being grouped together into binary classifiers. Therefore, an identification of alternative TUs by the increase/decrease of expression could be difficult with the given predictors. For this purpose, a classifier should only rely on expression data. Therefore, the idea of the so-called read coverage (RC) filter could lead to an increased ability of identifying alternative TUs. The RC filter thereby concentrates on a comparison of the same expression pattern between a gene pair in a local view. It considers different length by setting the lengths in an equal amount of 10 buckets (Figure 31) to calculate the mean expression of each bucket.



	B1	B2	…	B10
Start	82	40	26	49
Start +1	98	99	73	23
Start +2	24	81	1	67
…	59	20	15	33
…	64	58	21	27
Start + n	78	49	57	36
End	48	9	1	10
Average	64,71	50,86	27,71	35,00

Figure 31: Schematic creation of the RC filter. Exemplarily shown is a gene that gets divided into ten buckets with each bucket representing a genomic section. For each of these buckets, a mean expression is calculated.

In theory, genes within an operon should exhibit a similar bucket mean expression. However, in case of an alternative TU within an operon, the mean expression within the buckets of two adjacent genes should show an increase. In this context, the RC filter gives hints on the start of a TU (and end, respectively) and with this also on alternative TSS within an operon by the increase and decrease of the mean expression (Figure 31, Table S9B). For example, the genes of the *pec* operon, the different genes (pecB-pecE), exhibit different strengths of expression (Table S9B). As for pecB and pecA, the mean expression of each bucket is above 25k reads. A decrease can be observed for pecC and pecE. Additionally, the number of reads connecting pecB and pecA is also higher compared to the connection of pecA and C as well as for pecC to E. Interestingly, two TSS are located upstream the pecB gene of the *pec* operon (Table S9A). This might lead to the expression of the whole operon from one of the TSS and the second giving birth to an alternative TU only harboring pecB and pecA.

For predicting TUs out of these data, the binary labeling should be extended by labeling the known starts of TUs (e.g. through TSS or operon starts), as well as the end of a TU. Therefore, labeling is demanded with labeling NP (0), simple OP pairs (1), and OP pairs with the start of new TUs (2) and TU stop (3). Instead of labeling the *pec* operon with 0-1-1-1-0, it would then be labelled 0-2-3-1-0. In general, a combination of the GG and RC filter might successfully lead to the identification of alternative TUs because with the combination of increases and decreases of the level of expression, TU starts and stops are easily traceable.

6. REFERENCES

[1] C. M. O'Connor and J. U. Adams, *Essentials of Cell Biology*, MA: NPG Ed. Cambridge, 2010.

[2] A. S. N. Seshasayee, K. Sivaraman, and N. M. Luscombe, "An Overview of Prokaryotic Transcription Factors," in *Hughes T. (eds) A Handbook of Transcription Factors. Subcellular Biochemistry, vol 52. Springer, Dordrecht*, 2011, pp. 7–23.

[3] M. C. Leake, "Transcription factors in eukaryotic cells can functionally regulate gene expression by acting in oligomeric assemblies formed from an intrinsically disordered protein phase transition enabled by molecular crowding," *Transcription*, vol. 9, no. 5, pp. 298–306, 2018.

[4] M. Lis and D. Walther, "The orientation of transcription factor binding site motifs in gene promoter regions: does it matter?," *BMC Genomics*, vol. 17, no. 1, p. 185, 2016.

[5] C. GM, "The Cell: A Molecular Approach. 2nd edition," in *Sunderland (MA): Sinauer Associates*, 2000, p. Regulation of Transcription in Eukaryotes. Availab.

[6] H. Fan *et al.*, "Transcription-Translation coupling: Direct interactions of RNA polymerase with ribosomes and ribosomal subunits," *Nucleic Acids Res.*, vol. 45, no. 19, pp. 11043–11055, 2017.

[7] M. J. Moore and N. J. Proudfoot, "Pre-mRNA Processing Reaches Back toTranscription and Ahead to Translation," *Cell*, vol. 136, no. 4, pp. 688–700, 2009.

[8] A. R. Kornblihtt, I. E. Schor, M. Alló, G. Dujardin, E. Petrillo, and M. J. Muñoz, "Alternative splicing: A pivotal step between eukaryotic transcription and translation," *Nat. Rev. Mol. Cell Biol.*, vol. 14, no. 3, pp. 153–165, 2013.

[9] R. a Young, "RNA POLYMERASE II," *Annu. Rev Biochem.*, vol. 60, pp. 689–715, 1991.

[10] D. Sweetser, M. Nonet, and R. A. Young, "Prokaryotic and eukaryotic RNA polymerases have homologous core subunits," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 84, no. 5, pp. 1192–1196, 1987.

[11] T. Phillips, "Regulation of Transcription and gene expression in eukaryotes," *Nat. Educ.*, vol. 1, no. 2008, pp. 1–4, 2008.

[12] J. Brkljacic and E. Grotewold, "Combinatorial control of plant gene expression," *Biochim. Biophys. Acta - Gene Regul. Mech.*, vol. 1860, no. 1, pp. 31–40, 2017.

[13] H. Lodish, A. Berk, and S. Zipursky, *Molecular Cell Biology*, 4th editio. New York, 2020.

[14] B. Felden and L. Paillard, "When eukaryotes and prokaryotes look alike: the case of regulatory RNAs," *FEMS Microbiol. Rev.*, vol. 41, no. 5, pp. 624–639, 2017.

[15] E. Zhiguo, L. Wang, and J. Zhou, "Splicing and alternative splicing in rice and humans," *BMB Rep.*, vol. 46, no. 9, pp. 439–447, 2013.

[16] B. Reinhold-Hurek and D. A. Shub, “Self-splicing introns in tRNA genes of widely divergent bacteria,” *Nature*, vol. 357, no. 6374, pp. 173–176, 1992.

[17] P. Ortet, G. De Luca, D. E. Whitworth, and M. Barakat, “P2TF: A comprehensive resource for analysis of prokaryotic transcription factors,” *BMC Genomics*, vol. 13, no. 1, 2012.

[18] S. Clancy, “RNA Transcription by RNA Polymerase: Prokaryotes vs Eukaryotes,” *Nat. Educ.*, vol. 1, no. 1, p. 125, 2008.

[19] X. Chen, W.-C. Chou, Q. Ma, Y. Xu, and Y. Xu, “SeqTU: A Web Server for Identification of Bacterial Transcription Units,” *Sci. Rep.*, vol. 7, no. March, p. 43925, 2017.

[20] S. Cho *et al.*, “Current Challenges in Bacterial Transcriptomics,” *Genomics Inform.*, vol. 11, no. 2, p. 76, 2013.

[21] X. Mao, Q. Ma, B. Liu, X. Chen, H. Zhang, and Y. Xu, “Revisiting operons: An analysis of the landscape of transcriptional units in E. coli,” *BMC Bioinformatics*, vol. 16, no. 1, pp. 1–9, 2015.

[22] M. J. Griffiths AJF, Gelbart WM, “The Basics of Prokaryotic Transcriptional Regulation,” in *Modern Genetic Analysis. New York: W. H. Freeman*, vol. The Basics, 1999.

[23] F. Jacob and J. Monod, “Genetic regulatory mechanisms in the synthesis of proteins,” *Journal of Molecular Biology*. 1961.

[24] M. N. Price, A. P. Arkin, and E. J. Alm, “The life-cycle of operons,” *PLoS Genet.*, vol. 2, no. 6, pp. 0859–0873, 2006.

[25] F. Mao, P. Dam, J. Chou, V. Olman, and Y. Xu, “DOOR: A database for prokaryotic operons,” *Nucleic Acids Res.*, vol. 37, no. SUPPL. 1, pp. 459–463, 2009.

[26] B. Tjaden, “A computational system for identifying operons based on RNA-seq data,” *Methods*, no. November 2018, pp. 0–1, 2019.

[27] R. W. W. Brouwer, O. P. Kuipers, and S. A. F. T. Van Hijum, “The relative value of operon predictions,” *Brief. Bioinform.*, vol. 5, pp. 367–375, 2008.

[28] G. Moreno-Hagelsieb, “The power of operon rearrangements for predicting functional associations,” *Comput. Struct. Biotechnol. J.*, vol. 13, pp. 402–406, 2015.

[29] B. Taboada, R. Ciria, C. E. Martinez-Guerrero, and E. Merino, “ProOpDB: Prokaryotic operon database,” *Nucleic Acids Res.*, vol. 40, no. D1, pp. 627–631, 2012.

[30] D. H. Burkhardt, S. Rouskin, Y. Zhang, G. W. Li, J. S. Weissman, and C. A. Gross, “Operon mRNAs are organized into ORF-centric structures that predict translation efficiency,” *Elife*, vol. 6, pp. 1–23, 2017.

[31] V. Fortino, O.-P. Smolander, P. Auvinen, R. Tagliaferri, and D. Greco, “Transcriptome dynamics-based operon prediction in prokaryotes,” *BMC Bioinformatics*, vol. 15, no. 1,

p. 145, 2014.

[32] S. Okuda, S. Kawashima, K. Kobayashi, N. Ogasawara, M. Kanehisa, and S. Goto, “Characterization of relationships between transcriptional units and operon structures in Bacillus subtilis and Escherichia coli,” *BMC Genomics*, vol. 8, pp. 1–12, 2007.

[33] J. C. Ma, A. J. Newman, and R. S. Hayward, “Internal promoters of the rpoBC operon of Escherichia coli,” *MGG Mol. Gen. Genet.*, vol. 184, no. 3, pp. 548–550, 1981.

[34] K. L. Steward and T. Linn, “In vivo analysis of overlapping transcription units in the rplKAJLrpoBC ribosomal protein-RNA polymerase gene cluster of Escherichia coli,” *J. Mol. Biol.*, vol. 218, no. 1, pp. 23–31, 1991.

[35] M. N. Price, K. H. Huang, E. J. Alm, and A. P. Arkin, “A novel method for accurate operon predictions in all sequenced prokaryotes,” *Nucleic Acids Res.*, vol. 33, no. 3, pp. 880–892, 2005.

[36] B. Tjaden, “De novo assembly of bacterial transcriptomes from RNA-seq data,” *Genome Biol.*, vol. 16, no. 1, p. 1, 2015.

[37] H. Cao, Q. Ma, X. Chen, and Y. Xu, “c,” *Brief. Bioinform.*, vol. 20, no. 4, pp. 1568–1575, 2018.

[38] L. E. Post, G. D. Strycharz, M. Nomura, H. Lewis, and P. P. Dennis, “Nucleotide sequence of the ribosomal protein gene cluster adjacent to the gene for RNA polymerase subunit β in Escherichia coli,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 76, no. 4, pp. 1697–1701, 1979.

[39] G. Baughman and M. Nomura, “Translational regulation of the L11 ribosomal protein operon of Escherichia coli: Analysis of the mRNA target site using oligonucleotide-directed mutagenesis,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 81, no. 17 I, pp. 5389–5393, 1984.

[40] Q. W. Xie, C. White Tabor, and H. Tabor, “Spermidine biosynthesis in Escherichia coli: Promoter and termination regions of the speED operon,” *J. Bacteriol.*, vol. 171, no. 8, pp. 4457–4465, 1989.

[41] R. Fukuda and H. Nagasawa Fujimori, “Mechanism of the rifampicin induction of RNA polymerase β and β’ subunit synthesis in Escherichia coli,” *J. Biol. Chem.*, vol. 258, no. 4, pp. 2720–2728, 1983.

[42] G. Ralling and T. Linn, “Relative activities of the transcriptional regulatory sites in the rplKAJLrpoBC gene cluster of Escherichia coli,” *J. Bacteriol.*, vol. 158, no. 1, pp. 279–285, 1984.

[43] S. Gama-Castro *et al.*, “RegulonDB version 9.0: High-level integration of gene regulation, coexpression, motif clustering and beyond,” *Nucleic Acids Res.*, vol. 44, no. D1, pp. D133–D143, 2016.

[44] P. D. Karp *et al.*, “The EcoCyc Database,” *EcoSal Plus*, vol. 6, no. 1, 2014.

[45] V. Fortino, R. Tagliaferri, and D. Greco, “CONDOP: An R package for CONdition-Dependent Operon Predictions,” *Bioinformatics*, vol. 32, no. 20, pp. 3199–3200, 2016.

[46] V. Fortino, O.-P. Smolander, P. Auvinen, R. Tagliaferri, and D. Greco, “Transcriptome dynamics-based operon prediction in prokaryotes.,” *BMC Bioinformatics*, vol. 15, no. 1, p. 145, 2014.

[47] N. H. Bergman, K. D. Passalacqua, P. C. Hanna, and Z. S. Qin, “Operon prediction for sequenced bacterial genomes without experimental information,” *Appl. Environ. Microbiol.*, vol. 73, no. 3, pp. 846–854, 2007.

[48] S. Sáenz-Lahoya *et al.*, “Noncontiguous operon is a genetic organization for coordinating bacterial gene expression,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 116, no. 5, pp. 1733–1738, 2019.

[49] J. Vinnemeier, A. Kunert, and M. Hagemann, “Transcriptional analysis of the isiAB operon in salt-stressed cells of the cyanobacterium Synechocystis sp. PCC 6803,” *FEMS Microbiol. Lett.*, vol. 169, no. 2, pp. 323–330, Dec. 1998.

[50] A. Gupta, “RT-PCR: Characterization of Long Multi-Gene Operons and Multiple Transcript Gene Clusters in Bacteria,” *Biotechniques*, vol. 27, no. 5, pp. 966–972, 1999.

[51] J. E. Frías and E. Flores, “Negative regulation of expression of the nitrate assimilation nirA operon in the heterocyst-forming cyanobacterium Anabaena sp. strain PCC 7120,” *J. Bacteriol.*, vol. 192, no. 11, pp. 2769–2778, 2010.

[52] G. Schmetterer *et al.*, “The coxBAC operon encodes a cytochrome c oxidase required for heterotrophic growth in the cyanobacterium Anabaena variabilis strain ATCC 29413,” *J. Bacteriol.*, 2001.

[53] K. M. Jones and R. Haselkorn, “Newly identified cytochrome c oxidase operon in the nitrogen- fixing cyanobacterium Anabaena sp. strain PCC 7120 specifically induced in heterocysts,” *J. Bacteriol.*, vol. 184, no. 9, pp. 2491–2499, 2002.

[54] V. Merino-Puerto *et al.*, “FraC/FraD-dependent intercellular molecular exchange in the filaments of a heterocyst-forming cyanobacterium, Anabaena sp.,” *Mol. Microbiol.*, 2011.

[55] Å. ̊Agervald, K. Stensjö, M. Holmqvist, and P. Lindblad, “Transcription of the extended hyp-operon in Nostoc sp. strain PCC 7120,” *BMC Microbiol.*, vol. 8, pp. 1–12, 2008.

[56] J. Mitschke *et al.*, “ An experimentally anchored map of transcriptional start sites in the model cyanobacterium Synechocystis sp. PCC6803 ,” *Proc. Natl. Acad. Sci.*, vol. 108, no. 5, pp. 2124–2129, Feb. 2011.

[57] V. Fortino, R. Tagliaferri, and D. Greco, “CONDOP: An R package for CONdition-Dependent Operon Predictions,” *Bioinformatics*, vol. 32, no. 20, pp. 3199–3200, Oct. 2016.

[58] H. Salgado, G. Moreno-Hagelsieb, T. F. Smith, and J. Collado-Vides, “Operons in Escherichia coli: Genomic analyses and predictions,” *Proc. Natl. Acad. Sci.*, vol. 97, no. 12, pp. 6652–6657, 2000.

[59] N. H. Bergman, K. D. Passalacqua, P. C. Hanna, and Z. S. Qin, “Operon prediction for sequenced bacterial genomes without experimental information,” *Appl. Environ. Microbiol.*, vol. 73, no. 3, pp. 846–854, 2007.

[60] J. Lawrence, “Selfish operons: The evolutionary impact of gene clustering in prokaryotes and eukaryotes,” *Curr. Opin. Genet. Dev.*, vol. 9, no. 6, pp. 642–648, 1999.

[61] S. C. Janga, W. F. Lamboy, A. M. Huerta, and G. Moreno-Hagelsieb, “The distinctive signatures of promoter regions and operon junctions across prokaryotes,” *Nucleic Acids Res.*, vol. 34, no. 14, pp. 3980–3987, 2006.

[62] X. Chen, Z. Su, P. Dam, B. Palenik, Y. Xu, and T. Jiang, “Operon prediction by comparative genomics: An application to the Synechococcus sp. WH8102 genome,” *Nucleic Acids Res.*, vol. 32, no. 7, pp. 2147–2157, 2004.

[63] T. J. Goss, H. P. Schweizer, and P. Datta, “Molecular characterization of the tdc operon of Escherichia coli K-12.,” *J. Bacteriol.*, vol. 170, no. 11, pp. 5352–5359, 1988.

[64] J. Bockhorst, Y. Qiu, J. Glasner, M. Liu, F. Blattner, and M. Craven, “Predicting bacterial transcription units using sequence and expression data,” *Bioinformatics*, vol. 19, no. SUPPL. 1, 2003.

[65] P. Dam, V. Olman, K. Harris, Z. Su, and Y. Xu, “Operon prediction using both genome-specific and general genomic information,” *Nucleic Acids Res.*, vol. 35, no. 1, pp. 288–298, 2007.

[66] D. Che, G. Li, F. Mao, H. Wu, and Y. Xu, “Detecting uber-operons in prokaryotic genomes,” *Nucleic Acids Res.*, vol. 34, no. 8, pp. 2418–2427, 2006.

[67] W. C. Lathe, B. Snel, P. Bork, W. C. Lathe, B. Snel, and P. Bork, “Gene context conservation of a higher order than operons.,” *Trends Biochem. Sci.*, vol. 25, no. 10, pp. 474–9, Oct. 2000.

[68] B. Taboada, K. Estrada, R. Ciria, and E. Merino, “Operon-mapper: A web server for precise operon identification in bacterial and archaeal genomes,” *Bioinformatics*, vol. 34, no. 23, pp. 4118–4120, 2018.

[69] B. Taboada, C. Verde, and E. Merino, “High accuracy operon prediction method based on STRING database scores,” *Nucleic Acids Res.*, vol. 38, no. 12, 2010.

[70] J. Bockhorst, M. Craven, D. Page, J. Shavlik, and J. Glasner, “A bayesian network approach to operon prediction,” *Bioinformatics*, vol. 19, no. 10, pp. 1227–1235, 2003.

[71] C. Sommer and D. W. Gerlich, “Machine learning in cell biology – teaching computers to recognize phenotypes,” *J. Cell Sci.*, 2013.

[72] D. Chicco, “Ten quick tips for machine learning in computational biology,” *BioData Mining*. 2017.

[73] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, “Deep learning for computational biology,” *Mol. Syst. Biol.*, vol. 12, no. 7, p. 878, 2016.

[74] B. K. Izabela A. Samborska Warsaw, Leszek Sieczko, Vladimir Aleksandrov, “Artificial neural networks and their application in biological and agricultural research,” *Signpost Open Access J. NanoPhotoBioSciences*, vol. 02, pp. 14–30, 2014.

[75] A. L. Tarca, V. J. Carey, X. Chen, R. Romero, and S. Drăghici, “Machine Learning and Its Applications to Biology,” *PLoS Comput. Biol.*, vol. 3, no. 6, p. e116, 2007.

[76] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[77] M. W. Libbrecht and W. S. Noble, “Machine learning applications in genetics and genomics,” *Nat. Rev. Genet.*, vol. 16, no. 6, pp. 321–332, 2015.

[78] J. Hurwitz and D. Kirsch, *Machine Learning*. 2018.

[79] A. L. Tarca, V. J. Carey, X. wen Chen, R. Romero, and S. Drăghici, “Machine learning and its applications to biology.,” *PLoS computational biology*. 2007.

[80] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans, “Maximum margin clustering,” *Adv. Neural Inf. Process. Syst.*, 2005.

[81] H. T. Nguyen and A. Smeulders, “Active learning using pre-clustering,” *Proceedings, Twenty-First Int. Conf. Mach. Learn. ICML 2004*, pp. 623–630, 2004.

[82] M. W. Libbrecht and W. S. Noble, “Machine learning applications in genetics and genomics,” *Nature Reviews Genetics*. 2015.

[83] G. Dudek, “A Method of Generating Random Weights and Biases in Feedforward Neural Networks with Random Hidden Nodes,” *Inf. Sci.*, vol. 481, pp. 1–12, 2017.

[84] E. Raczko and B. Zagajewski, “Comparison of support vector machine, random forest and neural network classifiers for tree species classification on airborne hyperspectral APEX images,” *Eur. J. Remote Sens.*, vol. 50, no. 1, pp. 144–154, 2017.

[85] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A Training Algorithm for Optimal Margin Classifers,” *ACM Press*, pp. 88–144–152, 1992.

[86] R. Collobert and S. Bengio, “Links between Perceptrons, MLPs and SVMs,” *Proceedings, Twenty-First Int. Conf. Mach. Learn. ICML 2004*, pp. 177–184, 2004.

[87] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Mach. Learn.*, vol. 20, pp. 273–297, 1995.

[88] T. Joachims, N. Cristianini, and J. Shawe-Taylor, “Composite Kernels for Hypertext Categorisation,” *Proc. ICML-01, 18th Int. Conf. Mach. Learn.*, pp. 250–257, 2001.

[89] P. Mahesh and M. F. Giles, “Feature selection for classification of hyperspectral data by SVM,” *IEEE Trans. Geosci. Remote Sens. (IEEE T GEOSCI Remote.*, vol. 48, no.

5, pp. 2297–2307, 2010.

[90] T. Briggs and T. Oates, “Discovering domain-specific composite kernels,” *Proc. Natl. Conf. Artif. Intell.*, vol. 2, pp. 732–738, 2005.

[91] H. Li, F. Qi, and S. Wang, “A comparison of model selection methods for multi-class support vector machines,” *Lect. Notes Comput. Sci.*, vol. 3483, no. IV, pp. 1140–1148, 2005.

[92] T. Zhang, “An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods,” *AI Mag.*, vol. 22, pp. 103–104, 2001.

[93] M. N. Nguyen and C. R. Jagath, “Multi-Class Support Secondary Vector Structure Machines Prediction for Protein Secondary Structure Prediction,” *Genome Informatics*, vol. 14, pp. 218–227, 2003.

[94] J. Gall, N. Razavi, and L. Van Gool, “An introduction to random forests for multi-class object detection,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7474 LNCS, pp. 243–263, 2012.

[95] G. Biau, “Analysis of a random forests model,” *J. Mach. Learn. Res.*, vol. 13, pp. 1063–1095, 2012.

[96] L. Breiman, “Random forest,” *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[97] L. Breiman, “Some infinity theory for predictor ensembles,” *Tech. Rep.*, vol. 577, 2000.

[98] M. Robnik-Sikonja, “Improving random forest,” *Conf. Pap. Lect. Notes Comput. Sci.*, 2004.

[99] S. Ren, X. Cao, Y. Wei, and J. Sun, “Global refinement of random forest,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 723–730, 2015.

[100] L. N. Smith, “Cyclical learning rates for training neural networks,” *Proc. - 2017 IEEE Winter Conf. Appl. Comput. Vision, WACV 2017*, no. April, pp. 464–472, 2017.

[101] I. Loshchilov and F. Hutter, “SGDR: Stochastic gradient descent with warm restarts,” *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, 2017.

[102] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.*, vol. 1, pp. 328–339, 2018.

[103] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[104] S. Haykin, “Rosenblatt ’ s Perceptron,” *Neural Networks Learn. Mach.*, no. 1943, pp. 47–67, 2009.

[105] A. Scherer and A. Scherer, “Das Perzeptron,” *Neuronale Netze*, pp. 65–70, 1997.

[106] A. Ian Goodfellow, Yoshua Bengio and Courville, *Deep Learning*. MIT Press, 2016.

[107] L. Zhang, J. Tan, D. Han, and H. Zhu, “From machine learning to deep learning: Progress in machine intelligence for rational drug discovery,” *Drug Discov. Today*, vol.

00, no. 00, pp. 1–6, 2017.

[108] Y. LeCun and others, “Generalization and network design strategies,” *Connect. Perspect.*, pp. 143–155, 1989.

[109] D. E. Rumelhart, G. E. Hintont, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–535, 1986.

[110] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for Activation Functions,” 2017.

[111] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, “Understanding deep neural networks with rectified linear units,” in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018, pp. 1–17.

[112] S. Eger, P. Youssef, and I. Gurevych, “Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4415–4424.

[113] N. Srivastava, G. Hinton, A. K. I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1929–1958, pp. 520–525, 2014.

[114] P. O. Gislason, J. A. Benediktsson, and J. R. Sveinsson, “Random forests for land cover classification,” *Pattern Recognit. Lett.*, vol. 27, no. 4, pp. 294–300, 2006.

[115] A. Statnikov, L. Wang, and C. F. Aliferis, “A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification,” *BMC Bioinformatics*, vol. 9, pp. 1–10, 2008.

[116] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, “Machine learning applications in cancer prognosis and prediction,” *Comput. Struct. Biotechnol. J.*, 2015.

[117] A. T. Maia, S.-J. Sammut, A. Jacinta-Fernandes, and S.-F. Chin, “Big data in cancer genomics,” *Curr. Opin. Syst. Biol.*, 2017.

[118] Z. D. Stephens *et al.*, “Big data: Astronomical or genomical?,” *PLoS Biol.*, vol. 13, no. 7, pp. 1–11, 2015.

[119] R. McClure *et al.*, “Computational analysis of bacterial RNA-Seq data,” *Nucleic Acids Res.*, vol. 41, no. 14, pp. 1–16, 2013.

[120] S. Charaniya, S. Mehra, W. Lian, K. P. Jayapal, G. Karypis, and W. S. Hu, “Transcriptome dynamics-based operon prediction and verification in Streptomyces coelicolor,” *Nucleic Acids Res.*, vol. 35, no. 21, pp. 7222–7236, 2007.

[121] D. C. Ducat, J. C. Way, and P. A. Silver, “Engineering cyanobacteria to generate high-value products,” *Trends in Biotechnology*. 2011.

[122] D. Cheng and Q. He, “Assessment of Environmental Stresses for Enhanced Microalgal Biofuel Production â€“ An Overview,” *Front. Energy Res.*, vol. 2, no. July, pp. 1–8, 2014.

[123] T. Heidorn *et al.*, *Synthetic Biology in Cyanobacteria*, vol. 497. 2011.

[124] P. Erdrich, H. Knoop, R. Steuer, and S. Klamt, “Cyanobacterial biofuels: new insights and strain design strategies revealed by computational modeling.,” *Microb. Cell Fact.*, vol. 13, no. 1, p. 128, 2014.

[125] C. Beck, H. Knoop, and R. Steuer, “Modules of co-occurrence in the cyanobacterial pan-genome reveal functional associations between groups of ortholog genes,” *PLoS Genet.*, vol. 14, no. 3, pp. 1–20, 2018.

[126] N. S. Lau, M. Matsui, and A. A. A. Abdullah, “Cyanobacteria: Photoautotrophic Microbial Factories for the Sustainable Synthesis of Industrial Products,” *Biomed Res. Int.*, vol. 2015, 2015.

[127] N. E. Nozzi, J. W. K. Oliver, and S. Atsumi, “Cyanobacteria as a Platform for Biofuel Production,” *Front. Bioeng. Biotechnol.*, vol. 1, no. September, pp. 1–6, 2013.

[128] J. Kumar, A. & Singh, “Microalgae and Cyanobacteria Biofuels : A Sustainable Alternate to Crop-based Fuels,” *Microbes Environ. Manag.*, no. December 2015, pp. 1–20, 2016.

[129] J. C. Liao, L. Mi, S. Pontrelli, and S. Luo, “Fuelling the future: microbial engineering for the production of sustainable biofuels,” *Nat. Rev. Microbiol.*, vol. 14, no. 5, pp. 288–304, 2016.

[130] P. G. Falkowski *et al.*, “The evolution of modern eukaryotic phytoplankton,” *Science (80-.).*, vol. 305, no. 5682, pp. 354–360, 2004.

[131] D. A. Hutchins and F. Fu, “Microorganisms and ocean global change,” *Nat. Microbiol.*, vol. 2, no. May, 2017.

[132] D. B. Van De Waal, J. M. H. Verspagen, M. Lürling, E. Van Donk, P. M. Visser, and J. Huisman, “The ecological stoichiometry of toxins produced by harmful cyanobacteria: An experimental test of the carbon-nutrient balance hypothesis,” *Ecol. Lett.*, vol. 12, no. 12, pp. 1326–1335, 2009.

[133] P. M. Visser *et al.*, “How rising CO2 and global warming may stimulate harmful cyanobacterial blooms,” *Harmful Algae*, vol. 54, pp. 145–159, 2016.

[134] P. N. Leão, M. T. S. D. Vasconcelos, and V. M. Vasconcelos, “Allelopathy in freshwater cyanobacteria Allelopathy in freshwater cyanobacteria Pedro N. Leo et al.,” *Crit. Rev. Microbiol.*, vol. 35, no. 4, pp. 271–282, 2009.

[135] M. Rudolf, M. Stevanovic, C. Kranzler, R. Pernil, N. Keren, and E. Schleiff, “Multiplicity and specificity of siderophore uptake in the cyanobacterium Anabaena sp. PCC 7120,” *Plant Mol. Biol.*, vol. 92, no. 1–2, pp. 57–69, 2016.

[136] M. Stevanovic, C. Lehmann, and E. Schleiff, “The response of the TonB-dependent transport network in Anabaena sp. PCC 7120 to cell density and metal availability,” *BioMetals*, vol. 26, no. 4, pp. 549–560, 2013.

[137] S. Simm, R. Pernil, P. Dvorak, and C. Pancrace, “Cyanobacteria: Omics and Manipulation,” no. January, pp. 1–34, 2017.

[138] R. M. Morgan-Kiss, J. C. Priscu, T. Pocock, L. Gudynaite-savitch, and N. P. A. Huner, “Adaptation and Acclimation of Photosynthetic Microorganisms to Permanently Cold Environments,” *Microbiol. Mol. Biol. Rev.*, vol. 70, no. 1, pp. 222–252, 2454.

[139] N. Tandeau de Marsac and J. Houmard, “Adaptation of cyanobacteria to environmental stimuli: new steps towards molecular mechanisms,” *FEMS Microbiol. Lett.*, vol. 104, no. 1–2, pp. 119–189, 1993.

[140] M. Kim and K. sun Kim, “Stress-responsively modulated ymdAB-clsC operon plays a role in biofilm formation and apramycin susceptibility in Escherichia coli,” *FEMS Microbiol. Lett.*, vol. 364, no. 13, pp. 1–9, 2017.

[141] M. N. Price, K. H. Huang, A. P. Arkin, and E. J. Alm, “Operon formation is driven by co-regulation and not by horizontal gene transfer,” *Genome Res.*, vol. 15, no. 6, pp. 809–819, 2005.

[142] D. A. Los, A. Zorina, M. Sinetova, S. Kryazhov, K. Mironov, and V. V. Zinchenko, “Stress sensors and signal transducers in cyanobacteria,” *Sensors*, vol. 10, no. 3, pp. 2386–2415, 2010.

[143] M. Nakao *et al.*, “CyanoBase: The cyanobacteria genome database update 2010,” *Nucleic Acids Res.*, vol. 38, no. SUPPL.1, pp. 2009–2011, 2009.

[144] T. Nakazato, T. Ohta, and H. Bono, “Experimental Design-Based Functional Mining and Characterization of High-Throughput Sequencing Data in the Sequence Read Archive,” *PLoS One*, vol. 8, no. 10, 2013.

[145] N. R. Coordinators, “Database resources of the National Center for Biotechnology Information.,” *Nucleic Acids Res.*, vol. 42, no. Database issue, pp. D7-17, Jan. 2014.

[146] Y. Kodama, M. Shumway, and R. Leinonen, “The sequence read archive: Explosive growth of sequencing data,” *Nucleic Acids Res.*, vol. 40, no. D1, pp. 2011–2013, 2012.

[147] A. Santos-Zavaleta *et al.*, “RegulonDB v 10.5: Tackling challenges to unify classic and high throughput knowledge of gene regulation in E. coli K-12,” *Nucleic Acids Res.*, vol. 47, no. D1, pp. D212–D220, 2019.

[148] B. Bushnell, “BBMap: A Fast, Accurate, Splice-Aware Aligner,” 2014.

[149] B. Langmead and S. L. Salzberg, “Fast gapped-read alignment with Bowtie 2,” *Nat. Methods*, vol. 9, no. 4, pp. 357–359, 2012.

[150] S. Anders, P. T. Pyl, and W. Huber, “HTSeq-A Python framework to work with high-throughput sequencing data,” *Bioinformatics*, vol. 31, no. 2, pp. 166–169, 2015.

[151] N. D. Team, “ND4J: N-dimensional arrays and scientific computing for the JVM,” *Apache Software Foundation License 2.0.**. [Online]. Available: http://nd4j.org.

[152] M. Holtgrewe, “Mason – A Read Simulator for Second Generation Sequencing Data,”

Life Sci., no. October, p. 18, 2010.

[153] F. J. Sedlazeck, P. Rescheneder, and A. Von Haeseler, “NextGenMap : fast and accurate read mapping in highly polymorphic genomes,” vol. 29, no. 21, pp. 2790–2791, 2013.

[154] T. S. F. S. W. Group, “Sequence Alignment/Map Format Specification,” 2019.

[155] C. Trapnell *et al.*, “Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks,” *Nat. Protoc.*, vol. 7, no. 3, pp. 562–578, 2012.

[156] C. Evans, J. Hardin, and D. M. Stoebel, “Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions,” *Brief. Bioinform.*, vol. 19, no. 5, pp. 776–792, 2018.

[157] V. Merino-Puerto, V. Mariscal, C. W. Mullineaux, A. Herrero, and E. Flores, “Fra proteins influencing filament integrity, diazotrophy and localization of septal protein SepJ in the heterocyst-forming cyanobacterium Anabaena sp.,” *Mol. Microbiol.*, vol. 75, no. 5, pp. 1159–1170, 2010.

[158] V. Merino-Puerto, A. Herrero, and E. Flores, “Cluster of genes that encode positive and negative elements influencing filament length in a heterocyst-forming cyanobacterium,” *J. Bacteriol.*, vol. 195, no. 17, pp. 3957–3966, 2013.

[159] J. E. Frias, E. Flores, and A. Herrero, “Nitrate assimilation gene cluster from the heterocyst-forming cyanobacterium Anabaena sp. strain PCC 7120,” *J Bacteriol*, vol. 179, no. 2, pp. 477–486, 1997.

[160] J. E. Frías and E. Flores, “Induction of the nitrate assimilation nirA operon and protein-protein interactions in the maturation of nitrate and nitrite reductases in the cyanobacterium Anabaena sp. strain PCC 7120,” *J. Bacteriol.*, vol. 197, no. 14, pp. 2442–2452, 2015.

[161] R. V. Swanson, R. De Lorimier, and A. N. Glazer, “Genes encoding the phycobilisome rod substructure are clustered on the Anabaena chromosome: Characterization of the phycoerythrocyanin operon,” *J. Bacteriol.*, vol. 174, no. 16, pp. 2640–2647, 1992.

[162] G. Moreno-Hagelsieb, “The power of operon rearrangements for predicting functional associations,” *Comput. Struct. Biotechnol. J.*, vol. 13, pp. 402–406, 2015.

[163] D. A. Bryant, V. L. Stirewalt, M. Glauser, G. Frank, W. Sidler, and H. Zuber, “A small multigene family encodes the rod-core linker polypeptides of Anabaena sp. PCC7120 phycobilisomes,” *Gene*, vol. 107, no. 1, pp. 91–99, Oct. 1991.

[164] A. Valladares, A. Herrero, D. Pils, G. Schmetterer, E. Flores, and A. Vespucio, “Cytochrome c oxidase genes required for nitrogenase activity and diazotrophic growth in Anabaena sp. PCC 7120,” 2003.

[165] V. Merino-Puerto, V. Mariscal, C. W. Mullineaux, A. Herrero, and E. Flores, “Fra proteins influencing filament integrity, diazotrophy and localization of septal protein

SepJ in the heterocyst-forming cyanobacterium Anabaena sp.," *Mol. Microbiol.*, 2010.

[166] E. Flores, S. Picossi, A. Valladares, and A. Herrero, "Mechanisms Transcriptional regulation of development in heterocyst-forming," *BBA - Gene Regul. Mech.*, no. February, pp. 0–1, 2018.

[167] S. Picossi, M. L. Montesinos, R. Pernil, C. Lichtlé, A. Herrero, and E. Flores, "ABC-type neutral amino acid permease N-I is required for optimal diazotrophic growth and is repressed in the heterocysts of Anabaena sp. strain PCC 7120," *Mol. Microbiol.*, vol. 57, no. 6, pp. 1582–1592, 2005.

[168] S. Picossi, E. Flores, and A. Herrero, "The LysR-type transcription factor PacR is a global regulator of photosynthetic carbon assimilation in Anabaena," *Environ. Microbiol.*, vol. 17, no. 9, pp. 3341–3351, 2015.

[169] G. Moreno-Hagelsieb and J. Collado-Vides, "A powerful non-homology method for the prediction of operons in prokaryotes," *Bioinformatics*, vol. 18, no. Suppl 1, pp. S329–S336, 2002.

[170] M. J. L. De Hoon, K. Kobayashi, N. Ogasawara, and S. Miyano, "Predicting the operon structure of Bacillus subtilis using operon length, intergene distance, and gene expression information," *Pacific Symp. Biocomput.*, vol. 9, pp. 276–287, 2004.

[171] I. B. Rogozin, "Congruent evolution of different classes of non-coding DNA in prokaryotic genomes," *Nucleic Acids Res.*, vol. 30, no. 19, pp. 4264–4271, 2002.

[172] Y. Zheng, J. D. Szustakowski, L. Fortnow, R. J. Roberts, and S. Kasif, "Computational identification of operons in microbial genomes," *Genome Res.*, vol. 13, no. 1, pp. 1221–1230, 2003.

[173] P. R. Romero and P. D. Karp, "Using functional and organizational information to improve genome-wide computational prediction of transcription units on pathway-genome databases," *Bioinformatics*, vol. 20, no. 5, pp. 709–717, 2004.

[174] T. Itoh, K. Takemoto, H. Mori, and T. Gojobort, "Evolutionary instability of operon structures disclosed by sequence comparisons of complete microbial genomes," *Mol. Biol. Evol.*, vol. 16, no. 3, pp. 332–346, 1999.

[175] V. Fortino, R. Tagliaferri, and D. Greco, "CONDOP: An R package for CONdition-Dependent Operon Predictions," *Bioinformatics*, vol. 32, no. 20, pp. 3199–3200, 2016.

[176] N. Quintana, F. Van Der Kooy, M. D. Van De Rhee, G. P. Voshol, and R. Verpoorte, "Renewable energy from Cyanobacteria: Energy production optimization by metabolic pathway engineering," *Appl. Microbiol. Biotechnol.*, vol. 91, no. 3, pp. 471–490, 2011.

[177] F. Liang, E. Englund, P. Lindberg, and P. Lindblad, "Engineered cyanobacteria with enhanced growth show increased ethanol production and higher biofuel to biomass ratio," *Metab. Eng.*, vol. 46, no. December 2017, pp. 51–59, 2018.

[178] S. Simm, E. Schleiff, and R. Pernil, "The Cyanobacterial Core Genome: Global and

Specific Features with a Focus on Secondary Metabolites," *Cyanobacteria Omi. Manip.*, no. July, pp. 1–34, 2017.

[179] T. J. Johnson, R. Zhou, and J. G. Johnson, "Outlook on the Potential of Cyanobacteria to Photosynthetically Produce High- Value Chemicals and Biofuels at an Industrial Scale," *Bioenerg. Open Access*, vol. 05, no. 02, 2016.

[180] D. Lea-Smith and C. Howe, "The Use of Cyanobacteria for Biofuel Production," 2017, pp. 143–155.

[181] J. W. Erisman, H. van Grinsven, A. Leip, A. Mosier, and A. Bleeker, "Nitrogen and biofuels; an overview of the current state of knowledge," *Nutr. Cycl. Agroecosystems*, vol. 86, no. 2, pp. 211–223, 2010.

[182] N. Sierro, Y. Makita, M. De hoon, and K. Nakai, "DBTBS: A database of transcriptional regulation in Bacillus subtilis containing upstream intergenic conservation information," *Nucleic Acids Res.*, vol. 36, no. SUPPL. 1, p. 2008, 2008.

[183] S. Okuda, "ODB: a database of operons accumulating known operons across multiple genomes," *Nucleic Acids Res.*, vol. 34, no. 90001, pp. D358–D362, 2006.

[184] T. Yada, M. Nakao, Y. Totoki, and K. Nakai, "Modeling and predicting transcriptional units of Escherichia coli genes using hidden Markov models," *Bioinformatics*, vol. 15, no. 12, pp. 987–993, 1999.

[185] M. D. Ermolaeva, O. White, and S. L. Salzberg, "Prediction of operons in microbial genomes.," *Nucleic Acids Res.*, vol. 29, no. 5, pp. 1216–21, 2001.

[186] C. Sabatti, "Co-expression pattern from DNA microarray experiments as a tool for operon prediction," *Nucleic Acids Res.*, vol. 30, no. 13, pp. 2886–2893, 2002.

[187] B. Tjaden, "Transcriptome analysis of Escherichia coli using high-density oligonucleotide probe arrays," *Nucleic Acids Res.*, vol. 30, no. 17, pp. 3732–3738, 2002.

[188] X. Chen, Z. Su, Y. Xu, and T. Jiang, "Computational prediction of operons in Synechococcus sp. WH8102.," *Genome Inform.*, vol. 15, no. 2, pp. 211–222, 2004.

[189] C. J. Paredes, I. Rigoutsos, and T. Papoutsakis, "Transcriptional organization of the Clostridium acetobutylicum genome," *Nucleic Acids Res.*, vol. 32, no. 6, pp. 1973–1981, 2004.

[190] D. Steinhauser, B. H. Junker, A. Luedemann, J. Selbig, and J. Kopka, "Hypothesis-driven approach to predict transcriptional units from gene expression data," *Bioinformatics*, vol. 20, no. 12, pp. 1928–1939, 2004.

[191] L. Wang, J. D. Trawick, R. Yamamoto, and C. Zamudio, "Genome-wide operon prediction in Staphylococcus aureus," *Nucleic Acids Res.*, vol. 32, no. 12, pp. 3689–3702, 2004.

[192] M. J. L. de Hoon, Y. Makita, K. Nakai, and S. Miyano, "Prediction of Transcriptional

Terminators in Bacillus subtilis and Related Species," *PLoS Comput. Biol.*, vol. 1, no. 3, p. e25, 2005.

[193] B. P. Westover, J. D. Buhler, J. L. Sonnenburg, and J. I. Gordon, "Operon prediction without a training set," *Bioinformatics*, vol. 21, no. 7, pp. 880–888, 2005.

[194] G. qing Zhang, Z. wei Cao, Q. ming Luo, Y. dong Cai, and Y. xue Li, "Operon prediction based on SVM," *Comput. Biol. Chem.*, vol. 30, no. 3, pp. 233–240, 2006.

[195] P. Roback *et al.*, "A predicted operon map for Mycobacterium tuberculosis," *Nucleic Acids Res.*, vol. 35, no. 15, pp. 5085–5095, 2007.

[196] T. T. Tran *et al.*, "Operon prediction in Pyrococcus furiosus," *Nucleic Acids Res.*, vol. 35, no. 1, pp. 11–20, 2007.

[197] E. Laing, K. Sidhu, and S. J. Hubbard, "Predicted transcription factor binding sites as predictors of operons in Escherichia coli and Streptomyces coelicolor," *BMC Genomics*, vol. 9, pp. 1–15, 2008.

[198] A. Valladares, A. Herrero, D. Pils, G. Schmetterer, and E. Flores, "Cytochrome c oxidase genes required for nitrogenase activity and diazotrophic growth in Anabaena sp . PCC 7120," vol. 47, pp. 1239–1249, 2003.

[199] C. D. Carrasco and J. W. Golden, "Two heterocyst-specific DNA rearrangements of nif operons in Anabaena cylindrica and nostoc sp. strain Mac," *Microbiology*, vol. 141, no. 10, pp. 2479–2487, 1995.

[200] S. Picossi, A. Valladares, E. Flores, and A. Herrero, "Nitrogen-regulated genes for the metabolism of cyanophycin, a bacterial nitrogen reserve polymer: Expression and mutational analysis of two cyanophycin synthetase and cyanophycinase gene clusters in the heterocyst-forming cyanobacterium Anabaena sp. PCC 7120," *J. Biol. Chem.*, 2004.

[201] G. Fiedler, A. M. Muro-Pastor, E. Flores, and I. Maldener, "NtcA-dependent expression of the devBCA operon, encoding a heterocyst-specific ATP-binding cassette transporter in Anabaena spp.," *J. Bacteriol.*, vol. 183, no. 12, pp. 3795–3799, 2001.

[202] P. Lindberg, A. Hansel, and P. Lindblad, "hupS and hupL constitute a transcription unit in the cyanobacterium Nostoc sp. PCC 73102," *Arch. Microbiol.*, vol. 174, no. 1–2, pp. 129–133, 2000.

[203] E. Frı, A. Herrero, and E. Flores, "Open Reading Frame all0601 from Anabaena sp . Strain PCC 7120 Represents a Novel Gene , cnaT , Required for Expression of the Nitrate Assimilation nir Operon," vol. 185, no. 17, pp. 5037–5044, 2003.

[204] E. Flores, J. E. Frías, L. M. Rubio, and A. Herrero, "Photosynthetic nitrate assimilation in cyanobacteria," *Photosynth. Res.*, vol. 83, no. 2, pp. 117–133, 2005.

[205] A. Valladares, M. L. Montesinos, A. Herrero, and E. Flores, "An ABC-type, high-affinity

urea permease identified in cyanobacteria," *Mol. Microbiol.*, vol. 43, no. 3, pp. 703–715, 2002.

[206] A. B. and S. K. Apte, "Differential Expression of the Two kdp Operons in the Nitrogen-Fixing Cyanobacterium Anabaena sp. Strain L-31," *Appl. Environ. Microbiol.*, vol. 71, no. 9, pp. 5297–5303, 2005.

[207] S. López-Gomollón, J. A. Hernández, S. Pellicer, V. E. Angarica, M. L. Peleato, and M. F. Fillat, "Cross-talk Between Iron and Nitrogen Regulatory Networks in Anabaena (Nostoc) sp. PCC 7120: Identification of Overlapping Genes in FurA and NtcA Regulons," *J. Mol. Biol.*, vol. 374, no. 1, pp. 267–281, 2007.

[208] M. Ogura and T. Tanaka, "The Bacillus subtilis late competence operon comE is transcriptionally regulated by yutB and under post-transcription initiation control by comN (yrzD)," *J. Bacteriol.*, vol. 191, no. 3, pp. 949–958, Feb. 2009.

[209] C. A. Voigt, D. M. Wolf, and A. P. Arkin, "The Bacillus subtilis sin operon: An evolvable network motif," *Genetics*, vol. 169, no. 3, pp. 1187–1202, Mar. 2005.

[210] J. P. Sarsero, E. Merino, and C. Yanofsky, "A Bacillus subtilis operon containing genes of unknown function senses tRNA Trp charging and regulates expression of the genes of tryptophan biosynthesis," 1999.

[211] B. G. Butcher, Y. P. Lin, and J. D. Helmann, "The yydFGHIJ operon of Bacillus subtilis encodes a peptide that induces the LiaRS two-component system," *J. Bacteriol.*, vol. 189, no. 23, pp. 8616–8625, Dec. 2007.

[212] P. Gollnick, "Regulation of the Bacillus subtilis trp operon by an RNA-binding protein," *Molecular Microbiology*, vol. 11, no. 6. pp. 991–997, 1994.

[213] H. Cruz-Ramos, P. Glaser, L. V. Wray, and S. H. Fisher, "The Bacillus subtilis ureABC operon," *J. Bacteriol.*, vol. 179, no. 10, pp. 3371–3373, 1997.

[214] W. A. Claes, A. Pühler, and J. Kalinowski, "Identification of two prpDBC gene clusters in Corynebacterium glutamicum and their involvement in propionate degradation via the 2-methylcitrate cycle.," *J. Bacteriol.*, vol. 184, no. 10, pp. 2728–39, May 2002.

[215] T. Busche, R. Šilar, M. Pičmanová, M. Pátek, and J. Kalinowski, "Transcriptional regulation of the operon encoding stress-responsive ECF sigma factor SigH and its anti-sigma factor RshA, and control of its regulatory network in Corynebacterium glutamicum," *BMC Genomics*, vol. 13, no. 1, Sep. 2012.

[216] Y. Tanaka, H. Teramoto, and M. Inui, " Regulation of the Expression of De Novo Pyrimidine Biosynthesis Genes in Corynebacterium glutamicum ," *J. Bacteriol.*, vol. 197, no. 20, pp. 3307–3316, Oct. 2015.

[217] B. Yan, M. Boitano, T. Clark, and L. Ettwiller, "SMRT-Cappable-seq reveals complex operon variants in bacteria," *bioRxiv*, p. 262964, Jan. 2018.

[218] S. J. H. Matthew R. Chapman, Lloyd S. Robinson, Jerome S. Pinkner, Robyn Roth,

John Heuser, Mårten Hammar, Staffan Normark, “Role of Escherichia coli Curli Operons in Directing Amyloid Fiber Formation,” *Science (80-.).*, vol. 296, pp. 851–855, 2002.

[219] D. S. Oppenheim and C. Yanofsky, “TRANSLATIONAL COUPLING DURING EXPRESSION OF THE TRYPTOPHAN OPERON OF ESCHERICHIA COLI.”

[220] J. D. Partridge *et al.*, “Characterization of the Escherichia coli K-12 ydhYVWXUT operon: Regulation by FNR, NarL and NarP,” *Microbiology*, vol. 154, no. 2, pp. 608–618, Feb. 2008.

[221] B. E. Wright, A. Longacre, and J. M. Reimers, “Hypermutation in derepressed operons of Escherichia coli K12.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 96, no. 9, pp. 5089–94, Apr. 1999.

[222] T. D. Curran, F. Abacha, S. P. Hibberd, M. D. Rolfe, M. M. Lacey, and J. Green, “Identification of new members of the Escherichia coli K-12 MG1655 SlyA regulon,” *Microbiol. (United Kingdom)*, vol. 163, no. 3, pp. 400–409, Mar. 2017.

[223] B. L. Wanner and B. D. Chang, “The phoBR operon in Escherichia coli K-12.,” *J. Bacteriol.*, vol. 169, no. 12, pp. 5569–5574, 1987.

[224] T. Wang, G. Shen, R. Balasubramanian, L. McIntosh, D. A. Bryant, and J. H. Golbeck, “The sufR Gene (sll0088 in Synechocystis sp. Strain PCC 6803) Functions as a Repressor of the sufBCDS Operon in Iron-Sulfur Cluster Biogenesis in Cyanobacteria,” *J. Bacteriol.*, vol. 186, no. 4, pp. 956–967, Feb. 2004.

[225] R. Webb, K. J. Reddy, and L. A. Sherman, “Regulation and sequence of the Synechococcus sp. strain PCC 7942 groESL operon, encoding a cyanobacterial chaperonin,” *J. Bacteriol.*, vol. 172, no. 9, pp. 5079–5088, 1990.

[226] K. Leonhardt and N. A. Straus, “An iron stress operon involved in photosynthetic electron transport in the marine cyanobacterium Synechococcus sp. PCC 7002,” *J. Gen. Microbiol.*, vol. 138, no. 8, pp. 1613–1621, 1992.

[227] T. Nishimura, Y. Takahashi, O. Yamaguchi, H. Suzuki, S. I. Maeda, and T. Omata, “Mechanism of low CO2-induced activation of the cmp bicarbonate transporter operon by a LysR family protein in the cyanobacterium Synechococcus elongatus strain PCC 7942,” *Mol. Microbiol.*, vol. 68, no. 1, pp. 98–109, 2008.

[228] S. I. Maeda, Y. Kawaguchi, T. A. Ohe, and T. Omata, “Erratum: cis-acting sequences required for NtcB-dependent, nitrite- responsive positive regulation of the nitrate assimilation operon in the cyanobacterium Synechococcus sp. strain PCC 7942 (Journal of Bacteriology (1998) 180:16 (4080-4088)),” *J. Bacteriol.*, vol. 181, no. 16, p. 5134, 1999.

[229] T. H. A. Haverkamp, D. Schouten, M. Doeleman, U. Wollenzien, J. Huisman, and L. J. Stal, “Colorful microdiversity of Synechococcus strains (picocyanobacteria) isolated

from the Baltic Sea," *ISME J.*, vol. 3, no. 4, pp. 397–408, 2009.

[230] W. R. Widger, "The cloning and sequencing of Synechococcus sp. PCC 7002 petCA operon: Implications for the cytochrome c-553 binding domain of cytochrome f," *Photosynth. Res.*, vol. 30, no. 2–3, pp. 71–84, 1991.

[231] C. Lehelsg *et al.*, "A Second groEL-like Gene, Organized in a groESL Operon Is Present in the Genome of Synechocystis sp. PCC 6803*," *J. Biol. Chem.*, vol. 268, no. 3, pp. 1799–1604, 1993.

[232] K. Yamaguchi *et al.*, "A two-component Mn2+-sensing system negatively regulates expression of the mntCAB operon in Synechocystis.," *Plant Cell*, vol. 14, no. 11, pp. 2901–13, Nov. 2002.

[233] R. Jeanjean, E. Zuther, N. Yeremenko, M. Havaux, H. C. P. Matthijs, and M. Hagemann, "A photosystem 1 psaFJ-null mutant of the cyanobacterium Synechocystis PCC 6803 expresses the isiAB operon under iron replete conditions," *FEBS Lett.*, vol. 549, no. 1–3, pp. 52–56, Aug. 2003.

[234] H. Lill and N. Nelson, "The atp1 and atp2 operons of the cyanobacterium Synechocystis sp. PCC 6803," *Plant Mol. Biol.*, vol. 17, no. 4, pp. 641–652, Oct. 1991.

[235] F. Jacob, A. Ullmann, and J. Monod, "Délétions fusionnant l'opéron lactose et un opéron purine chez Escherichia coli," *J. Mol. Biol.*, vol. 13, no. 3, pp. 704–719, Oct. 1965.

[236] P. Gollnick, "Regulation of the Bacillus subtilis trp operon by an RNA-binding protein," *Molecular Microbiology*. 1994.

[237] X. M. Xue *et al.*, "Identification of Steps in the Pathway of Arsenosugar Biosynthesis," *Environ. Sci. Technol.*, vol. 53, no. 2, pp. 634–641, Jan. 2019.

[238] T. Kaneko *et al.*, "Sequence Analysis of the Genome of the Unicellular Cyanobacterium Synechocystis sp. Strain PCC6803. II. Sequence Determination of the Entire Genome and Assignment of Potential Protein-coding Regions," 1996.

7. SUPPLEMENTS

Table S1: Different operon predictors. Table adapted from Brouwer (2008, [27]) and Tjaden (2019, [26]). Indicated are the features of intergenic distance (ID), conservation (Cons.), functional relations, further genome sequence-based features and experimental datasets as well as the underlying scoring model.

Author(s)	Year	ID	Cons.	Functional relations	Genome sequence based	Experimental sets	Scoring method	Source
Yada *et al.*	1999	X			Promoters, transcriptional terminators, ribosome binding sites		Hidden Markov model	[184]
Craven *et al.*	2000	X		Riley's functional classification	Promoters, transcriptional terminators, operon size	39 DNA microarray datasets	Naive Bayes	
Salgado *et al.*	2000	X		Riley's functional classification			Log-likelihood scores	[58]
Ermolaeva *et al.*	2001		X					[185]
Moreno-Hagelsieb *et al.*	2002	X					Log-likelihood scores	[169]
Sabatti *et al.*	2002	X				72 DNA microarray datasets	Bayesian classifier	[186]
Tjaden *et al.*	2002					Genome tilling DNA microarrays		[187]
Zheng *et al.*	2002			Metabolic pathways				
Bockhorst *et al.*	2003	X			Codon usage, promoters, transcriptional terminators, operon length	39 DNA microarray datasets	Bayesian network	[64]
Chen *et al.*	2004	X	X	COG	Transcriptional terminators, conserved promoters		Log-likelihood scores	[62][188]
de Hoon *et al.*	2004	X			Operon length	174 DNA microarray datasets	Bayesian classifier	[170]

Paredes *et al.*	2004	X			Promoters, transcriptional terminators,		Emprical scoring scheme	[189]
Romero *et al.*	2004	X		Riley's functional classification, metabolic pathways, protein complex information, functional classification of upstream genes, similarity in codon usage			Log-likelihood scores	[173]
Steinhauser *et al.*	2004	X				140 DNA microarray datasets	Unweighted average linkage-clustering algorithm	[190]
Wang *et al.*	2004	X	X		Transcriptional terminators		Empirical scoring scheme	[191]
Jacob *et al.*	2005	X	X	Metabolic pathways, protein function			Genetic algorithm	[192]
Price *et al.*	2005	X	X	COG	Codon adaptation index		Naive Bayes approach	[35]
Westover *et al.*	2005	X	X	Functional relatedness			Naïve Bayes approach	[193]
Zhang et al. *et al.*	2006	X	X	Metabolic pathways, interacting protein domains			Support vector machine	[194]
Bergman et *et al.*	2007	X	X				Bayesian hidden markov model	[59]
Charaniya *et al.*	2007	X			Transcriptional terminators	67 DNA microarray datasets	Support vector machine	[120]
Dam *et al.* (DOOR)	2007	X	X	GO	DNA motifs, phylogenetic distance, gene length ratio		11 classifiers from PRTools Mathlab toolbox	[65]

Roback *et al.*	2007	X				474 DNA microarray datasets	Logistic regression predictive model	[195]
Tran *et al.*	2007	X		Metabolic pathways, GO			Neural network incorporating the criteria combined with results from [28, 36, 37]	[196]
Laing *et al.*	2008				Transcription factor binding sites			[197]
Tjaden *et al.* (Rockhopper)	2002, 2015, 2019	X				RNA-Seq	naïve Bayes classifier	[36] [26]
Fortino *et al.* (CONDOP)	2016				x	RNA-Seq, DOOR operons	Neural Networks (NN), Support Vector Machines (SVMs) and Random Forests (RFs)	[45]
Taboada *et al.* (proOpDB)	2010, 2011	X	X	STRING, COG			MLP, artificial neural network	[69]
Okuda *et al.* (ODB)	2006 & 2011		X			Experimental proven and predicted operons		[183]
Santos-Zavaleta *et al.* (RegulonDB)	2019					Experimental proven operons		[147]

Chen *et al.* (SeqTU)	RNA-Seq	SVM	[19]

Table S2: Base composition for each 1L of dropout medium.

S1	10 ml
S2	1 ml
S3	1 ml
S4	1 ml
S5	20 ml
S6	100 ml
S Am	12 ml

Table S3: Composition of solutions.

Solution	Reagents	g/mol	g/L	Dropout
S1	NaNO3	84.99	149.58	
	MgSO4	246.47	7.49	
	Citric Acid	192.13	0.6	
	K2HPO4	174.18	3.05	
	H3BO	61.81	0.28	
S2	FeCl3	270.3	1.62	Not in –Fe
	EDTA NA Salt	372.24	2.23	
S3	CaCl2	147.02	36	
S4	Na2CO3	106	20	
S5	Hepes pH 7.8	238.31	119.14	
S6	MnCl2	197.92	1.81	Not in –Mn
	ZnSO4	287.54	0.222	Not in –Zn
	Na2MoO4	241.96	0.39	Not in –Mo
	CuSO4	249.7	0.079	Not in –Cu
	Co(NO3)2	291.05	0.0494	Not in –Co
	EDTA Na Salt	372.24	0.55	

Table S4: Collected datasets for different species. Indicated are the type of the file (first column) and the species it can be assigned to (second column). The third column indicates the source of the dataset and (if abundant) assigned identifiers.

Type of file	Deriving species	Source (and identifier)
GFF	*Anabaena* sp. PCC 7120	**CyanoBase:** http://genome.kazusa.or.jp/cyanobase/Anabaena/, ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/009/705/GCA_000009705.1_ASM970v1
FASTA	*Anabaena* sp. PCC 7120	**CyanoBase:** http://genome.kazusa.or.jp/cyanobase/Anabaena/, ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/009/705/GCA_000009705.1_ASM970v1
RNA-Seq	*Anabaena* sp. PCC 7120	Provided by Niclas Wolfgang Fester (PhD student, Ak Schleiff, compare 2.3.1), Control, -Fe, -Nit
TSS	*Anabaena* sp. PCC 7120	[56]**:** http://www.cyanolab.de/software_downloads.html

Operons	*Anabaena* sp. PCC 7120	**DOOR:** http://161.117.81.224/DOOR2/displayNCoperon.php?id=898&page=1&nc=NC_003267#tabs-1
Operons	*Anabaena* sp. PCC 7120	**ProOpDB:** http://biocomputo2.ibt.unam.mx/OperonPredictor/
Operons	*Anabaena* sp. PCC 7120	**Literature:** [52], [53], [198], [199], [200], [51], [201], [168], [55], [202], [167], [165], [158], [159], [203], [204], [166], [205], [206], [207], [161], [163]
GFF	*Bacillus subtilis* str. 168	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome?LinkName=nuccore_genome&from_uid=452916715
FASTA	*Bacillus subtilis* str. 168	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome?LinkName=nuccore_genome&from_uid=452916715
RNA-Seq	*Bacillus subtilis*	**SRA:** http://sra.dbcls.jp/details.html?db=sra&accession=SRP018904&_id=SRP018904, SRP018904
Operons	*Bacillus subtilis* str. 168	**DOOR:** http://161.117.81.224/DOOR2/displayNCoperon.php?id=1240&page=1&nc=NC_000964#tabs-1
Operons	*Bacillus subtilis* str. 168	**ProOpDB:** http://biocomputo2.ibt.unam.mx/OperonPredictor/
Operons	*Bacillus subtilis* str. 168	**Literature:** [208], [209], [210], [211], [212], [213]
GFF	*Clostridium perfringens* ATCC 13124	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=txid1502[Organism:noexp]
FASTA	*Clostridium perfringens* ATCC 13124	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=txid1502[Organism:noexp]
RNA-Seq	*Clostridium perfringens*	**SRA:** http://sra.dbcls.jp/details.html?db=sra&accession=SRP095363&_id=SRP095363, SRX5089960, SRX5089961
Operons	*Clostridium perfringens* ATCC 13124	**DOOR:** http://161.117.81.224/DOOR2/displayNCoperon.php?id=1838&page=1&nc=NC_008261#tabs-1
Operons	*Clostridium perfringens* ATCC 13124	**ProOpDB:** http://biocomputo2.ibt.unam.mx/OperonPredictor/

GFF	*Corynebacterium xerosis*	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=txid1725 [Organism:noexp]
FASTA	*Corynebacterium xerosis*	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=txid1725 [Organism:noexp]
RNA-Seq	*Corynebacterium xerosis*	**SRA:** http://sra.dbcls.jp/details.html?db=sra&accession=SRP212221&_id=SRP212221, SRX6372980
Operons	*Corynebacterium*	**Literature:** [214], [215], [216]
GFF	*Escherichia coli* str. K-12	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=Escherichia+coli+enteroinvasive
FASTA	*Escherichia coli* str. K-12	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=Escherichia+coli+enteroinvasive
RNA-Seq	*Escherichia coli* str. K-12	**SRA:** http://sra.dbcls.jp/details.html?db=sra&accession=DRP001474&_id=DRP001474, DRP001474, DRP001475
RNA-Seq	*Escherichia coli* str. K-12	[217]**:** ftp://ftp.neb.com/unpub/yan/
Operons	*Escherichia coli* str. K-12	**RegulonDB:** http://regulondb.ccg.unam.mx/menu/download/datasets/index.jsp
Operons	*Escherichia coli* str. K-12	**DOOR:** http://161.117.81.224/DOOR2/displayNCoperon.php?id=1944&page=1&nc=NC_000913#tabs-1
Operons	*Escherichia coli* str. K-12	**ProOpDB:** http://biocomputo2.ibt.unam.mx/OperonPredictor/
Operons	*Escherichia coli* str. K-12	Literature: [40], [218], [219], [220], [221], [222], [223]
GFF	*Synechococcus elongatus* PCC 6301	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=txid1140 [Organism:noexp]
FASTA	*Synechococcus elongatus* PCC 6301	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=txid1140 [Organism:noexp]
RNA-Seq	*Synechococcus elongatus*	**SRA:** http://sra.dbcls.jp/details.html?db=sra&accession=SRP148555&_id=SRP148555, SRX4105568

RNA-Seq	*Synechococcus*-bacteria coculture system	**SRA:** http://sra.dbcls.jp/details.html?db=sra&accession=SRP199949&_id=SRP199949, SRX5936775
TSS	*Synechococcus elongatus* PCC 6301	[56]**:** http://www.cyanolab.de/software_downloads.html
Operons	*Synechococcus elongatus* PCC 6301	**DOOR:** http://161.117.81.224/DOOR2/displayNCoperon.php?id=3147&page=1&nc=NC_006576#tabs-1
Operons	*Synechococcus elongatus* PCC 6301	**ProOpDB:** http://biocomputo2.ibt.unam.mx/OperonPredictor/
Operons	*Synechococcus*	**Literature:** [204], [224], [225], [226], [227], [228], [229], [230]
GFF	*Synechocystis* sp. PCC 6803	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=ASM972v1
FASTA	*Synechocystis* sp. PCC 6803	**NCBI Genome:** https://www.ncbi.nlm.nih.gov/genome/?term=ASM972v1
RNA-Seq	*Synechocystis* sp. PCC 6803	**SRA:** http://sra.dbcls.jp/details.html?db=sra&accession=SRP029697&_id=SRP029697, SRX347145, SRX347146
Operons	*Synechocystis* sp. PCC 6803	**DOOR:** http://161.117.81.224/DOOR2/displayNCoperon.php?id=1761&page=1&nc=NC_017277#tabs-1
Operons	*Synechocystis* sp. PCC 6803	**ProOpDB:** http://biocomputo2.ibt.unam.mx/OperonPredictor/
Operons	*Synechocystis sp. PCC 6803*	**Literature:** [49], [56], [224], [231], [232], [233], [234]

Table S5: Operons identified within the literature for six species.

Operon	Species	Source
pec	*Anabaena* sp. PCC 7120	[161]
cpc	*Anabaena* sp. PCC 7120	[163]
cox1	*Anabaena* sp. PCC 7120	[164]
cox2	*Anabaena* sp. PCC 7120	[164]
cox3	*Anabaena* sp. PCC 7120	[164]
fraC	*Anabaena* sp. PCC 7120	[165]
nir	*Anabaena* sp. PCC 7120	[159]
alr2825-alr2831	*Anabaena* sp. PCC 7120	[166]
alr2835-2841	*Anabaena* sp. PCC 7120	[166]
devBCA	*Anabaena* sp. PCC 7120	[166]
all1780-all1781	*Anabaena* sp. PCC 7120	[166]
cphA-cphB	*Anabaena* sp. PCC 7120	[166]
nif	*Anabaena* sp. PCC 7120	[166]
rbcLXS	*Anabaena* sp. PCC 7120	[166]
natA-natC	*Anabaena* sp. PCC 7120	[167]

Species	# operons
Anabaena sp.	18
Bacillus subtilis	12
Corynebacterium	3
Escherichia coli K-12	20
Synechococcus	10
Synechocystis PCC 6803	8
Sum	71

hyp	*Anabaena* sp. PCC 7120	[55]
all5341-asr5350	*Anabaena* sp. PCC 7120	[166]
cmpA-D	*Anabaena* sp. PCC 7120	[166], [168]
lacA-Z	*Escherichia coli* K-12	[235]
csgG-D	*Escherichia coli* K-12	[218]
csgB-A	*Escherichia coli* K-12	[218]
trpA-L	*Escherichia coli* K-12	[219]
ydhT-Y	*Escherichia coli* K-12	[220]
leuD-L	*Escherichia coli* K-12	[221], [222]
leuV-Q	*Escherichia coli* K-12	[221], [222]
phoB-R	*Escherichia coli* K-12	[223]
tdcG-A	*Escherichia coli* K-12	[63]
cas2-casB	*Escherichia coli* K-12	[222]
elfA-ycbF	*Escherichia coli* K-12	[222]
agaB-C	*Escherichia coli* K-12	[222]
gspC-D	*Escherichia coli* K-12	[222]
fecE-R	*Escherichia coli* K-12	[222]
ssuB-E	*Escherichia coli* K-12	[222]
yehA –D	*Escherichia coli* K-12	[222]
mngA-B	*Escherichia coli* K-12	[222]
paaA-K	*Escherichia coli* K-12	[222]
crfC-yjcZ	*Escherichia coli* K-12	[222]
sgcC-X	*Escherichia coli* K-12	[222]
comE	*Bacillus subtilis*	[208]
sin	*Bacillus subtilis*	[209]
yczA-ycbK	*Bacillus subtilis*	[210]
yydFGHIJ	*Bacillus subtilis*	[211]
trpA-L	*Bacillus subtilis*	[236]
ureABC	*Bacillus subtilis*	[213]
sdpABC	*Bacillus subtilis*	[211]
liaH-R	*Bacillus subtilis*	[211]
sbo-alb	*Bacillus subtilis*	[211]
sunAT-bdbA-yolJ-dbdB	*Bacillus subtilis*	[211]
aroHBF	*Bacillus subtilis*	[236]
pyr	*Bacillus subtilis*	[231]
sufBCDS	*Synechococcus*	[224]
groESL	*Synechococcus*	[225]
isiAB	*Synechococcus*	[226]
nirA	*Synechococcus*	[228]
cmp	*Synechococcus*	[228]
lux	*Synechococcus*	[228]
petCA	*Synechococcus*	[230]
cpcBA	*Synechococcus*	[229]
psbD-C	*Synechococcus*	[226]
petF	*Synechococcus*	[226]
atp1	*Synechocystis* PCC 6803	[234]
atp2	*Synechocystis* PCC 6803	[234]
slr0303-slr0305	*Synechocystis* PCC 6803	[237]
isiAB	*Synechocystis* PCC 6803	[49]
hox	*Synechocystis* PCC 6803	[56]
sufBCDS	*Synechocystis* PCC 6803	[224]
mntCAB	*Synechocystis* PCC 6803	[232]
groESL	*Synechocystis* PCC 6803	[231]
pyr	*Corynebacterium*	[238]

sigH-rshA	*Corynebacterium*	[215]
prpDBC	*Corynebacterium*	[214]

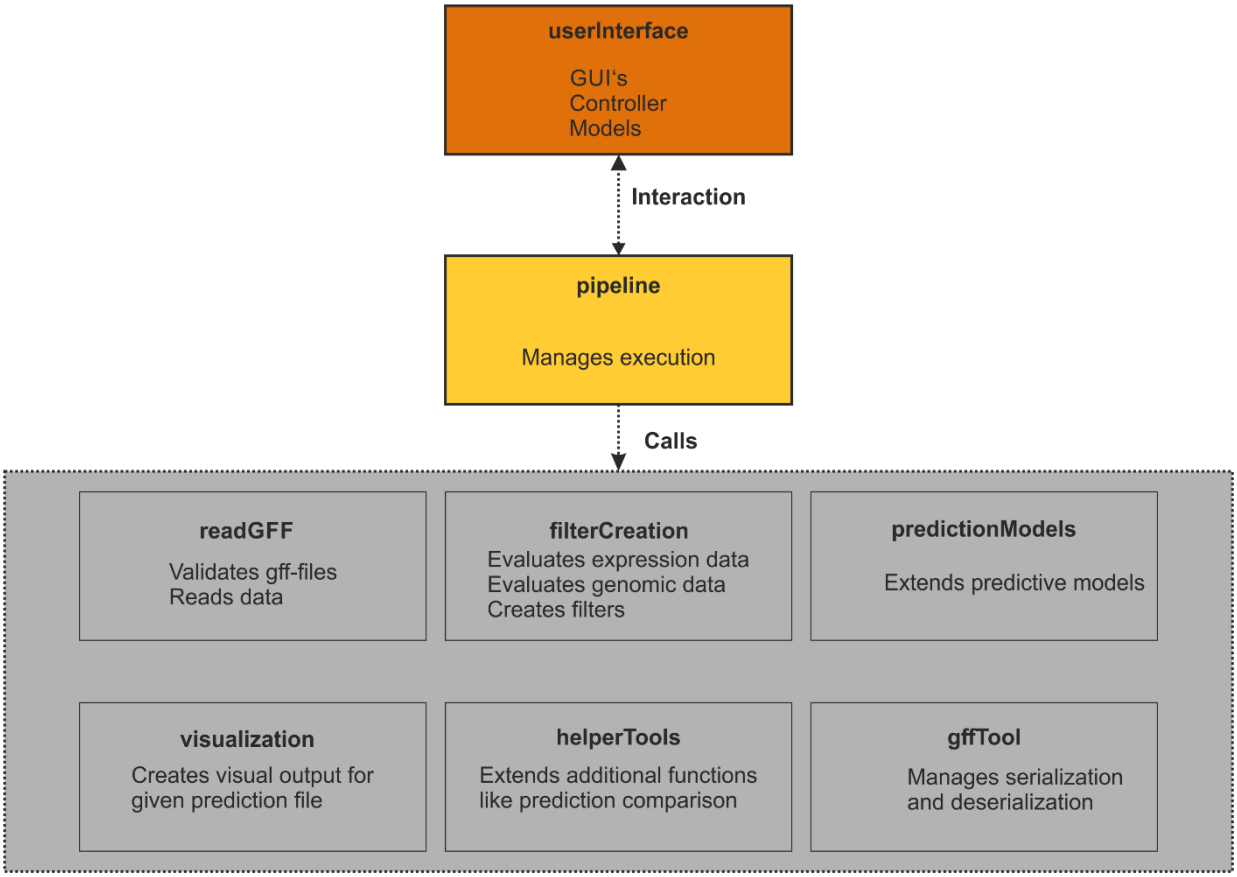


Figure S1: Schematic overview of OpPipe packages. OpPipe separates into the pipeline managing parts (userInterface, pipeline) and the theoretical models (grey dashed box).

Table S6: Occurrence of SAM flags within the SAM file of *Anabaena* sp. PCC 7120. Counted were the occurrences of each SAM flag and indicated are the descriptions of the SAM flags. As NGM was called with the exclusion of non-mapped reads, some of the mates are assigned with value zero.

SAM flag	# occurrence	description
65	24137	read paired, first in pair
129	24137	read paired, second in pair
69	0	
137	23783	read paired, mate unmapped, second in pair
73	123139	read paired, mate unmapped, first in pair
133	0	
81	8394	read paired, read reverse strand, first in pair
161	8394	read paired, mate reverse strand, second in pair
83	20880295	read paired, read mapped in proper pair, read reverse strand, first in pair
163	20880295	read paired, read mapped in proper pair, mate reverse strand, second in pair
89	126802	read paired, mate unmapped, read reverse strand, first in pair
165	0	
97	7605	read paired, mate reverse strand, first in pair
145	7605	read paired, read reverse strand, second in pair
99	20928948	read paired, read mapped in proper pair, mate reverse strand, first in pair
147	20928948	read paired, read mapped in proper pair, read reverse strand, second in pair
101	0	
153	19646	read paired, mate unmapped, read reverse strand, second in pair
113	27774	read paired, read reverse strand, mate reverse strand, first in pair
177	27774	read paired, read reverse strand, mate reverse strand, second in pair

Table S7: Cut-off calculation for EP filter. (A) Number of reads (n) per genomic position is counted. (B) Occurrence of n is counted and summed up. (C) Cut-off is determined basing on given stringency.

A

Number of mapped reads (n)	5	0	1	1	2	5	5	5	3	1
Genomic position	**pos1**	**pos2**	**pos3**	**pos4**	**pos5**	**pos6**	**pos7**	**pos8**	**pos9**	**pos10**

B

Number of mapped reads (n)	**Occurrence in genome (o)**	**Summing occurrences (S_o)**	
1	3	3	cut-off for stringency 0.98, S_O > STval
2	1	4	
3	1	5	cut-off for stringency 0.50, S_O > STval
4	0	5	
5	4	9	cut-off for stringency 0.05, S_O > STval
Sum of occurrences (S)	**9**		

C

Stringency (ST)	**ST_{val} = S - (S * Stringency)**
0,98	0,18
…	…
0,50	4,5
…	…
0,05	8,55

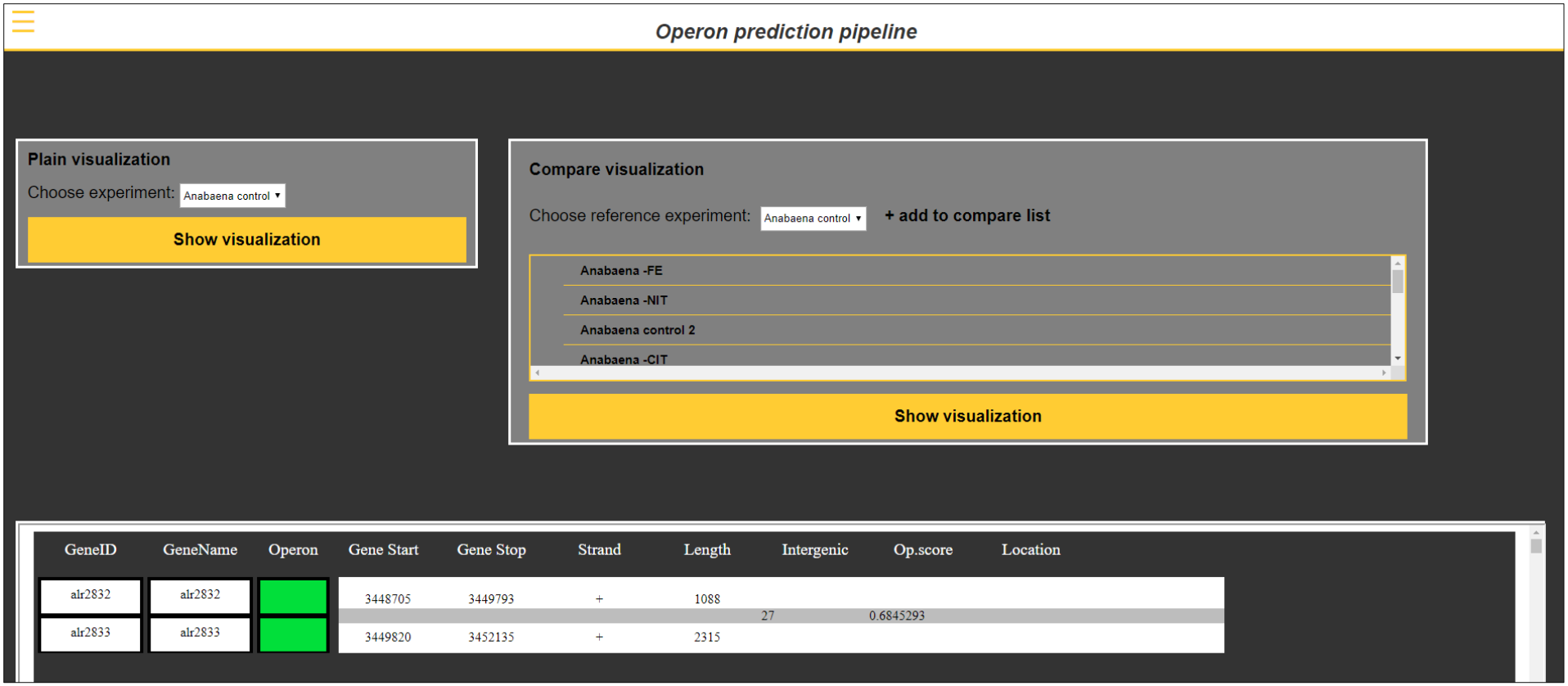


Figure S2: GUI of the visualization tool of OpPipe. In the plain visualizer setting all conducted experiments can be visualized via the dropdown. For the compare visualization the user chooses the desired experiments (dropdown)and adds them to the list. All the experiments in the list will be compared against the reference (combo box).

Table S8: Literature set operons from *Anabaena* sp. PCC 7120 compositions under different stringencies of the EP filter

Stringency	Elongated	Full hit	Partial	No hit
0.98	67%	6%	28%	0%
0.95	39%	22%	39%	0%
0.9	33%	22%	44%	0%
0.8	17%	33%	50%	0%
0.7	6%	39%	56%	0%
0.6	6%	44%	50%	0%
0.5	6%	39%	56%	0%
0.4	6%	28%	56%	11%
0.3	0%	33%	39%	28%
0.2	0%	17%	39%	44%
0.1	0%	11%	33%	56%
0.05	0%	11%	17%	72%

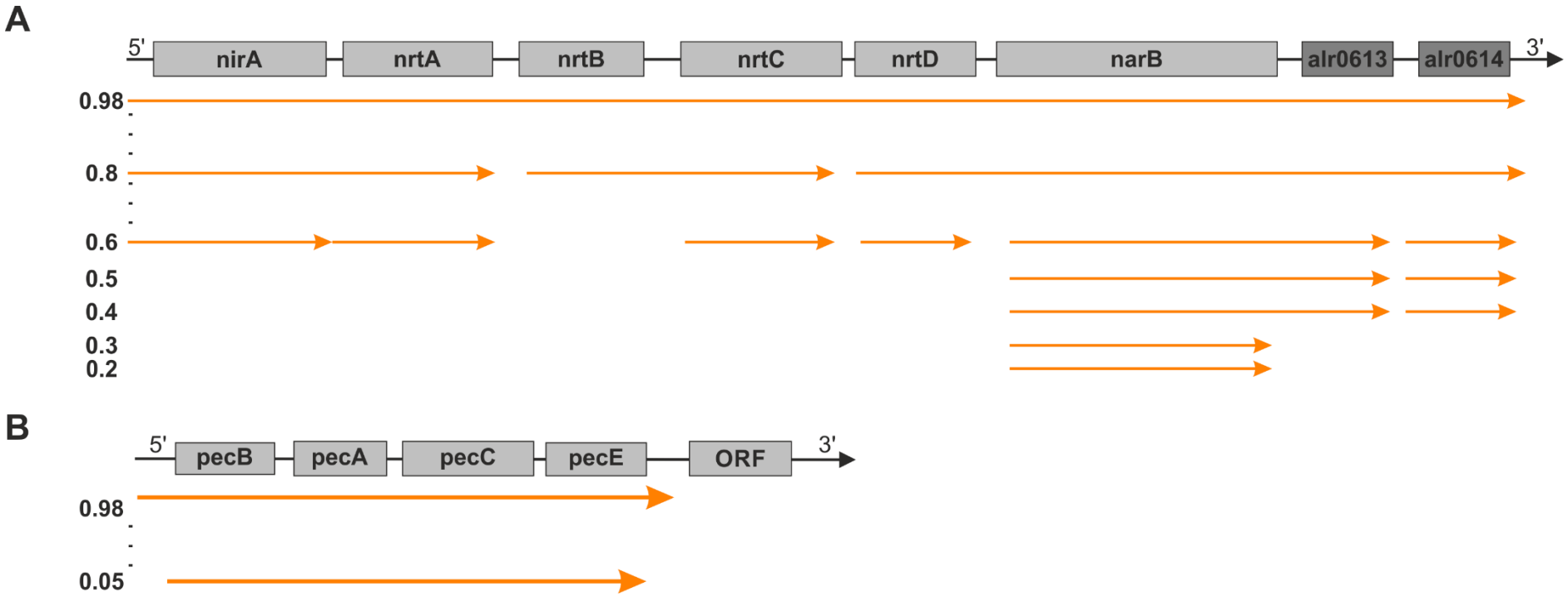


Figure S3: Decay of the *nir* operon [159] (A) and *pec* operon [161] (B) from *Anabaena* sp. PCC 7120 under different stringencies of the EP filter under control conditions. Light grey inked genes are part of the operon, dark grey inked genes might be part of the gene cluster. The orange arrows indicate the length of the referring TU under a specific stringency.

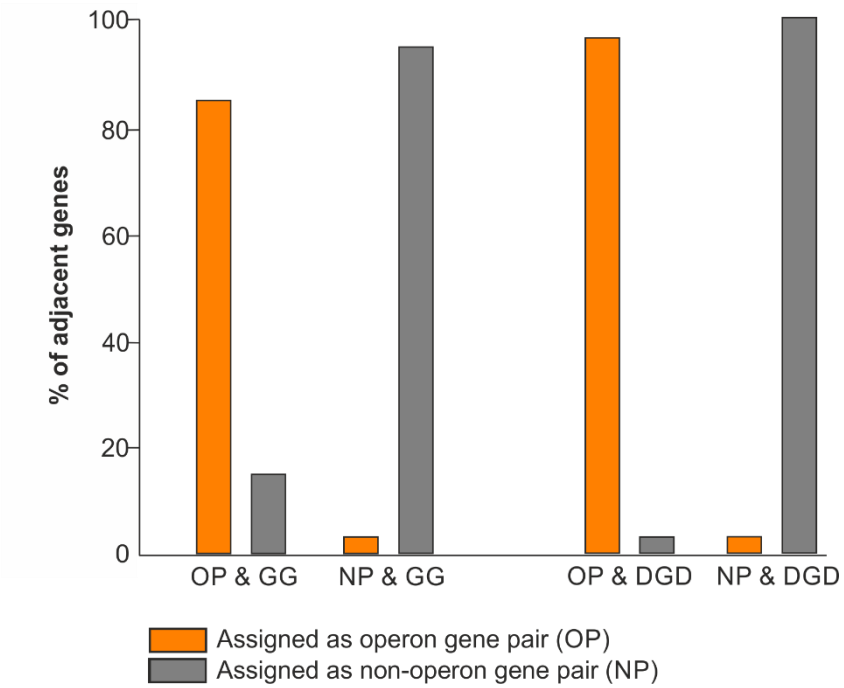
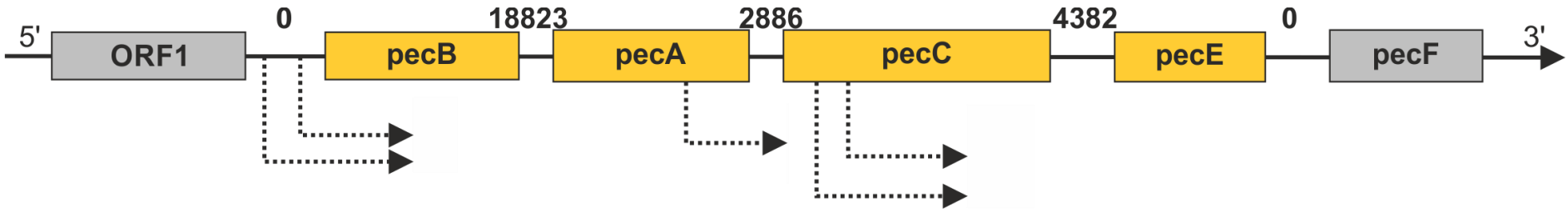


Figure S4: Evaluation of GG- and DGD filter basing on adjacent gene pairs of *Anabaena* sp. PCC 7120. Indicated are all adjacent gene pairs from the literature set (2.6.1) being part of an operon (OP) or not being part of an (NP). It is indicted if adjacent genes have been assigned as OP (orange) or NP (grey) for both filters (GG and DGD) and for the OP and NP set

Table S9: Overlaps for GG filter and buckets for RC filter distribution for *pec* operon. Indicated is the composition of the operon, with the genes belong to the operon (yellow) and the surround genes (grey). The numbers between the genes indicate the number of reads connecting them under CO conditions. Dashed arrows indicate TSS. Each gene of the cluster is divided into Buckets (B1-10) with the numbers indicating the average expression for this bucket under CO condition.

A



B

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
ORF1	225	155	109	97	95	110	145	147	120	80
pecB	**41557**	**41210**	**39478**	**35992**	**31272**	**28439**	**30776**	**30453**	**30867**	**28678**
pecA	**28881**	**36645**	**26623**	**26695**	**26002**	**26591**	**26799**	**26654**	**30867**	**28678**
pecC	**8501**	**9177**	**8736**	**7549**	**6442**	**6790**	**8904**	**9703**	**10501**	**9142**
pecE	**46233**	**3677**	**3065**	**2620**	**3350**	**2143**	**1860**	**1637**	**1149**	**669**
pecF	398	348	331	343	342	449	512	556	521	413

Table S10: Overlaps for GG filter of *nir*, *fraC* and *pec* operon under different conditions. Indicated are the genes and gene pair connections of the *nir*, *fraC* and *pec* operon. For each condition the raw counts are indicated, as well as the normalized values (Norm. PM and Norm. SF). Norm. PM indicates the normalization to one million reads and SF indicates the normalization with a scaling factor (Control scaling factor: 0.84, -Fe scaling factor: 0.79, -Nit scaling factor: 1).

Gene (-pair)	Covered reads (Control)			Covered reads (-Nit)			Covered reads (-Fe)		
	Total count	Norm. PM	**Norm. SF**	Total count	Norm. PM	**Norm. SF**	Total count	Norm. PM	**Norm. SF**
nirA	194	11.96	**162.89**	61056	4004.40	**48172.33**	1175	60.80	**1175.00**
nirA-nrtA	10	0.62	**8.40**	2051	134.52	**1618.21**	49	2.54	**49.00**
nrtA	129	7.95	**108.31**	17724	1162.44	**13983.99**	673	34.83	**673.00**
nrtA-nrtB	14	0.86	**11.75**	1241	81.39	**979.13**	59	3.05	**59.00**
nrtB	78	4.81	**65.49**	2921	191.58	**2304.63**	155	8.02	**155.00**
nrtB-nrtC	28	1.73	**23.51**	381	24.99	**300.60**	39	2.02	**39.00**
nrtC	291	17.93	**244.34**	4036	264.70	**3184.35**	704	36.43	**704.00**
nrtC-nrtD	11	0.68	**9.24**	290	19.02	**228.81**	63	3.26	**63.00**
nrtD	119	7.33	**99.92**	1257	82.44	**991.76**	440	22.77	**440.00**
nrtD-narB	17	1.05	**14.27**	141	9.25	**111.25**	65	3.36	**65.00**
narB	880	54.23	**738.88**	1695	111.17	**1337.33**	2066	106.91	**2066.00**
narB-alr0613	19	1.17	**15.95**	15	0.98	**11.83**	70	3.62	**70.00**
alr0613	268	16.52	**225.02**	392	25.71	**309.28**	773	40.00	**773.00**
alr0613-alr0614	6	0.37	**5.04**	16	1.05	**12.62**	18	0.93	**18.00**
alr0614	222	13.68	**186.40**	242	15.87	**190.93**	485	25.10	**485.00**
fraC	256	15.78	**214.95**	461	30.23	**363.72**	513	26.55	**513.00**
frac-fracD	49	3.02	**41.14**	90	5.90	**71.01**	119	6.16	**119.00**
fracD	437	26.93	**366.92**	743	48.73	**586.22**	841	43.52	**841.00**
fracD-fracE	253	15.59	**212.43**	363	23.81	**286.40**	357	18.47	**357.00**
fracE	931	57.38	**781.71**	1125	73.78	**887.61**	1278	66.13	**1278.00**
pecB	90772	5594.18	**76215.95**	24558	1610.65	**19375.92**	2516	130.19	**2516.00**
pecB-pecA	18823	1160.04	**15804.57**	5597	367.08	**4415.95**	565	29.24	**565.00**
pecA	52099	3210.80	**43744.49**	13814	906.00	**10899.05**	1622	83.93	**1622.00**
pecA-pecC	2886	177.86	**2423.21**	952	62.44	**751.11**	99	5.12	**99.00**
pecC	32008	1972.62	**26875.25**	9156	600.50	**7223.96**	1217	62.98	**1217.00**
pecC-pecE	4382	270.06	**3679.31**	1119	73.39	**882.88**	133	6.88	**133.00**
pecE	6853	422.34	**5754.06**	1774	116.35	**1399.66**	250	12.94	**250.00**

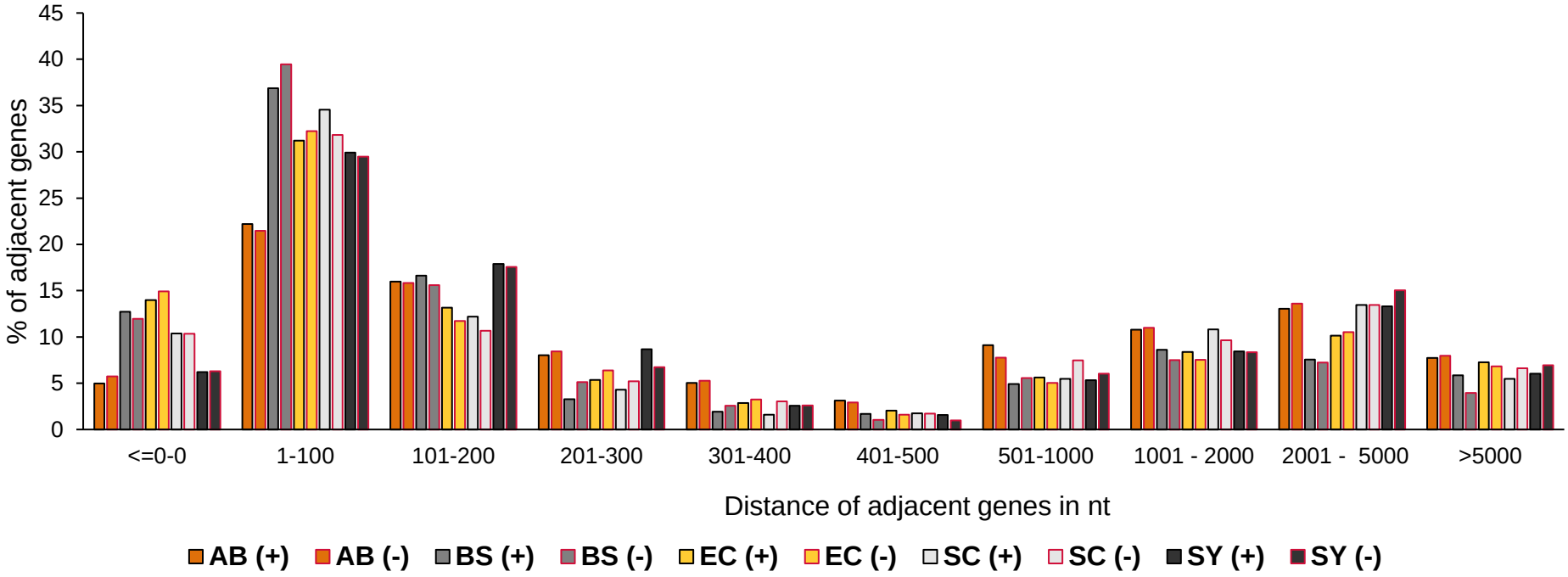
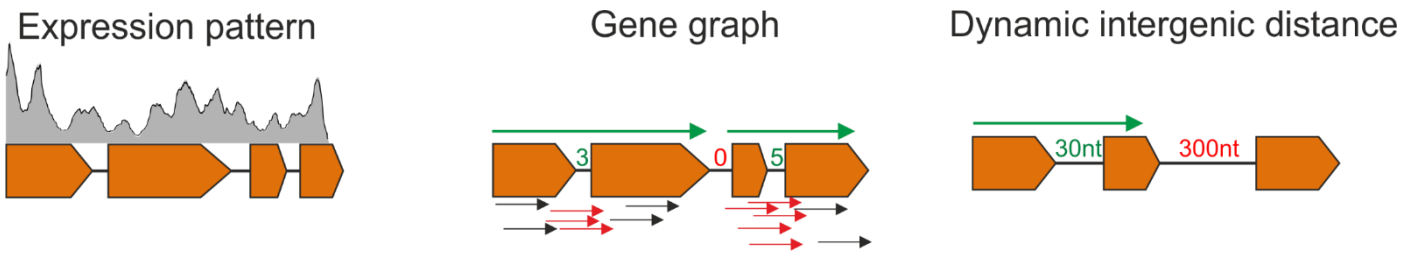


Figure S5: Average distance of adjacent genes of Watson and Crick strand within different genomes. Distribution of the intergenic distance of adjacent genes in *Anabaena* sp. PCC 7120 (orange), *Bacillus subtilis str. 168* (grey), *Escherichia coli K12* (yellow), *Synechococcus elongatus* (light grey) and *Synechocystis* PCC 6803 (dark grey) into different distance ranges (in nt). A black border (first bar of species) indicates the Watson strand and a red border (second bar of species) the Crick strand.

Table S11: Different statistical measures for the DGD filter. Calculated are the sensitivity (sen), specificity (spec), precision (prec), accuracy (acc) and f1-score. Operon sets of *Anabaena* sp. PCC 7120 (AB), *Bacillus subtilis* (BS), *Escherichia coli* (EC), *Synechococcus elongatus* (SC) and *Synechocystis* PCC 6803 (SY) are indicated (last column). Indicated are also the intergenic distance cut-offs of these species (AB = column 2, EC = column 3, BS = column 4, SY = column 5, SC= column 6) as well as an average distance cut-off of all cyanobacteria (CY) and average distance over all other distance cut-off (A). The values of the cut-off are indicated in brackets. The rows two to eight indicate the statistical measures for the operon prediction using the intergenic distance cut-off of a specific species onto a specific operon set. Bold values indicate the application of an intergenic distance of a species to its referring operon set.

Stat. measure	AB (300)	EC (100)	BS (100)	SY (200)	SC (200)	CY (233)	A (180)	
senAB	**100%**	56%	56%	87%	87%	92%	83%	**AB operons**
specAB	**97%**	100%	100%	99%	99%	98%	99%	
precAB	**97%**	100%	100%	99%	99%	98%	99%	
accAB	**98%**	76%	76%	93%	93%	95%	91%	
F1AB	**98%**	72%	72%	92%	92%	95%	90%	
senEC	100%	**100%**	100%	100%	100%	100%	100%	**EC operons**
specEC	72%	**98%**	98%	82%	82%	77%	84%	
precEC	78%	**98%**	98%	84%	84%	81%	86%	
accEC	86%	**99%**	99%	91%	91%	88%	92%	
F1EC	88%	**99%**	99%	92%	92%	90%	92%	
senBS	100%	91%	**91%**	100%	100%	100%	100%	**BS operons**
specBS	60%	91%	**91%**	67%	67%	64%	70%	
precBS	72%	91%	**91%**	75%	75%	74%	77%	
accBS	80%	91%	**91%**	84%	84%	82%	85%	
F1BS	83%	91%	**91%**	86%	86%	85%	87%	
senSY	100%	75%	75%	**100%**	100%	100%	93%	**SY operons**
specSY	86%	99%	99%	**99%**	99%	93%	99%	
precSY	88%	99%	99%	**99%**	99%	94%	99%	
accSY	93%	87%	87%	**99%**	99%	97%	96%	
F1SY	94%	85%	85%	**99%**	99%	97%	96%	
senSC	100%	14%	14%	100%	**100%**	100%	32%	**SC operons**
specSC	83%	97%	97%	95%	**95%**	89%	95%	
precSC	86%	96%	96%	95%	**95%**	90%	95%	
accSC	92%	26%	26%	97%	**97%**	94%	48%	
F1SC	92%	24%	24%	97%	**97%**	95%	47%	

Table S12: Schematically data for gene pairs basing on EP, GG and DGD filter. Shown are schematic views of the different filters. Each gene pair is assigned to a column. The rows are labelled with the different filters. A “1” indicates that the filter can be applied, a “0” indicates that the filter cannot be applied. For the literature set, the gene pairs were labelled (1=OP, 0=NP).



Gene-pair	$EP_{0.98}$	$EP_{...}$	$EP_{0.05}$	GG_Y	GG_N	DGD_Y	DGD_N	Label
G1 - G2	1	…	0	1	0	1	0	1
G2 - G3	1	…	1	1	0	0	1	1
G3 - G4	1	…	0	0	1	0	1	0
G.. - G..	xx	…	xx	xx	xx	x	x	X

Table S13: Prediction of literature operons from *Escherichia coli* by OpPipe, ProOpDB, DOOR and Rockhopper. The inked boxes indicate if an operon was found (green) by the predictor or not (red). Further it is indicated, if a predicted operon is longer as in the literature set (blue) or the predicted operon hits partially the literature set operon (yellow).

Operon	OpPipe	ProOpDB*[1]	DOOR*[2]	Rockhopper*[3]	RegulonDB*[4]	
lacA-Z						[235]
csgG-D						[218]
csgB-A						[218]
trpA-L						[219]
ydhT-Y						[220]
leuD-L						[221], [222]
leuV-Q						[221], [222]
phoB-R						[223]
tdcG-A						[63]
cas2-casB						[222]
elfA-ycbF						[222]
agaB-C						[222]
gspC-D						[222]
fecE-R						[222]
ssuB-E						[222]
yehA –D						[222]
mngA-B						[222]
paaA-K						[222]
crfC-yjcZ						[222]
sgcC-X						[222]

perfect	partial	elongated	not found

*[1]ProOpDB [69], *[2]DOOR [65], *[3]Rockhopper [36], *[4]RegulonDB [147]

Table S14: Scoring of operon prediction of different predictor basing on *Anabaena* sp. PCC 7120 (AB) and *Escherichia coli* (EC). Indicated are operons that have been found as full hit, partial hit, elongated or not found basing on the literature set. Full hits are multiplied by 1, non-hits by -1, elongated and partial with 0.5, leading to a specific score. For OpPipe, the plain predictor is indicated.

	OpPipe		**ProOpDB [69]**		**DOOR [65]**		**Rockhopper [36],**		**RegulonDB [147]**	
	AB	EC	AB	EC	AB	EC	AB	EC	AB	EC
Full Hit	11	12	8	10	3	15	2	9	0	19
Partial	3	5	6	5	11	4	5	7	0	1
Elongated	4	0	0	1	0	0	0	0	0	0
Not found	0	3	4	4	4	1	10	4	0	0
Sum	18	20	18	20	18	20	17	20	0	20
Score	**0,81**	**0,58**	**0,39**	**0,45**	**0,25**	**0,80**	**-0,32**	**0,43**	**0,00**	**0,98**

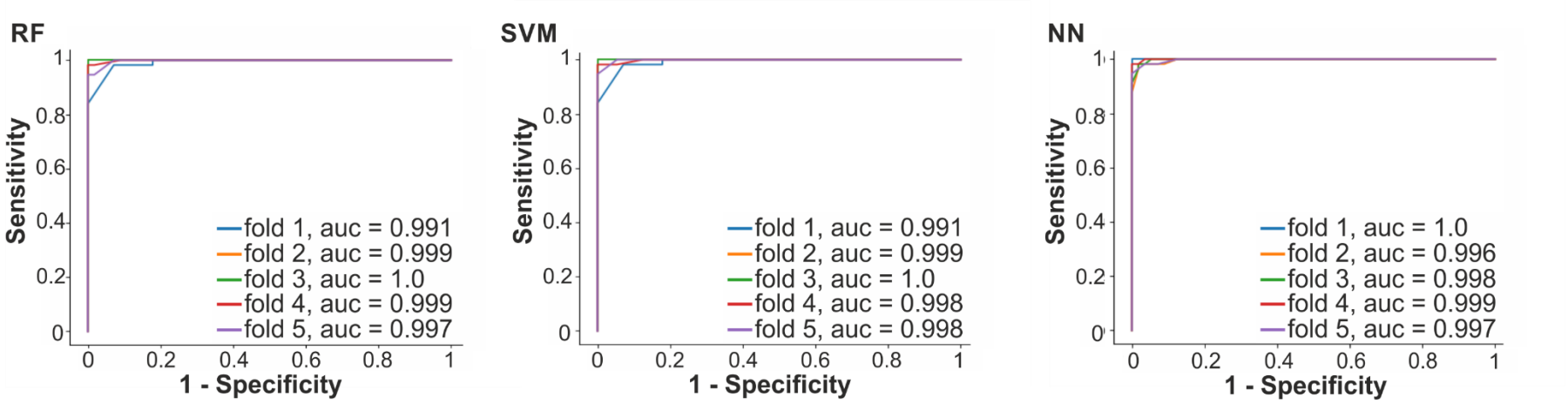


Figure S6: AUC for random forest (RF), support vector machine (SVM) and neural net (NN) for 5-fold cross validation on filter data. For RF and SVM for each fold, the best scoring model of the 5-fold cross validation of the grid search is shown.

Table S15: Grid Search result for support vector machine classifier (filter data). Shown are the calculated parameters fitting best for the classifier. For each fold, the best scoring model of the 5-fold cross validation of the grid search is indicated.

K-Fold	Best Params
RF – Fold 1-5	
SVM - Fold1	{'decision_function_shape': 'ovr', 'max_iter': -1, 'cache_size': 2000, 'kernel': 'rbf', 'degree': 1, 'shrinking': False, 'coef0': 0, 'C': 0.027, 'gamma': 0.464}
SVM - Fold2	{'decision_function_shape': 'ovr', 'max_iter': -1, 'cache_size': 2000, 'kernel': 'rbf', 'degree': 1, 'shrinking': False, 'coef0': 0, 'C': 22.539, 'gamma': 0.01}
SVM - Fold3	{'decision_function_shape': 'ovr', 'max_iter': -1, 'cache_size': 2000, 'kernel': 'poly', 'degree': 1, 'shrinking': False, 'coef0': 0, 'C': 22.539, 'gamma': 0.01}
SVM - Fold4	{'decision_function_shape': 'ovr', 'max_iter': -1, 'cache_size': 2000, 'kernel': 'poly', 'degree': 1, 'shrinking': False, 'coef0': 0, 'C': 6.2e-05, 'gamma': 4641.588}
SVM - Fold5	{'decision_function_shape': 'ovr', 'max_iter': -1, 'cache_size': 2000, 'kernel': 'rbf', 'degree': 1, 'shrinking': False, 'coef0': 0, 'C': 0.050, 'gamma': 1.0}

Table S16: Full hit and not found candidates of predicted operons for different predictors. The first column indicates different predictors (ProOpDB [69], Rockhopper [36], DOOR [65]), while the following columns show the OpPipe plain predictor (PP), the random forest approach (RF), the SVM approach (SVM) and the neural net approach (NN). The first column of each comparison shows the number of full hit operons (FH), the second column (NH1) shows the non-found operons of ProOpDB, Rockhopper and DOOR compared to the OpPipe predictors, while the second comparison column (NH2) indicates the opposite comparison.

	PP			RF			SVM			NN		
	#FH	#NH1	#NH2	#FH	#NH1	#NH2	#FH	#NH1	#NH2	#FH	#NH1	#NH2
ProOpDB	292	3	578	193	37	551	187	38	558	199	40	556
Rockhopper	197	3	715	139	39	702	138	39	709	145	41	707
DOOR	276	1	585	184	46	568	180	45	574	195	49	572

Table S17: Conserved operons that have been predicted by the plain predictor, RF, SVM and NN. Indicated is the start gene and the stop gene of the operon.

Genes of operons				
thrB-ndhD	alr2372-alr2374	alr4132-alr4134	alr3307-asr3309	asr5004-alr5005
alr2306-alr2310	alr2385-asr2389	asl3966-aat	alr1166-alr1167	asr3881-alr3887
alr2014-asr2016	all3256-all3257	alr3364-alr3366	alr1054-alr1055	alr1490-alr1491
hglK-asl0815	alr4153-asr4154	alr0430-asr0431	alr1194-alr1199	alr1285-alr1286
all2753-all2754	asr1945-alr1946	alr1956-alr1957	alr4029-alr4033	all3819-ycf27
all1483-all1484	alr0616-alr0618	alr5134-alr5135	alr4308-alr4311	devB-alr1605
alr4880-alr4882	alr1270-alr1271	aroK-alr1247	alr4745-asr4747	alr1905-asr1907
psbI-alr1278	all2037-all2038	alr3330-alr3331	alr0451-alr0452	alr2708-alr2709
alr0789-alr0790	alr0444-alr0447	asr2474-alr2476	menA-menE	alr3246-alr3247
alr3588-alr3589	psaA-alr5158	radA-alr3825	grpE-dnaK	alr5034-alr5035
all2115-all2116	asr3369-alr3370	tpiA-alr4386	alr5259-asr5261	alr2467-asr2468
alr1332-asr1333	all1487-all1489	all3865-all3866	alr1952-rpsU	asr2666-asr2669
alr3510-alr3514	alr3812-nblB	gidB-alr3183	asr4004-alr4005	alr1713-asr1714
all0844-tatA	alr4094-alr4095	alr0963-alr0965	alr1867-alr1870	alr5360-alr5368
alr4772-alr4773	alr1519-alr1520	alr3361-alr3363	alr4280-alr4282	alr0235-asr0243
alr5329-alr5333	alr2137-alr2138	all0888-all0889	alr1404-asr1406	alr1877-glgA
alr0540-alr0541	alr2184-alr2188	alr1721-alr1722	alr4839-recQ	all0664-all0665
alr3806-alr3807	asr0680-alr0681	prk-alr4124	glyA-alr4808	asr0043-alr0045
alr2304-alr2305	alr2957-asr2959	truB-alr1545	alr3481-alr3486	alr1146-alr1148
alr1652-alr1656	alr1142-alr1144	alr3638-alr3641	thrS-alr0336	alr0810-alr0812
alr2717-alr2719	asr4910-alr4915	asr2602-asr2603	alr4258-alr4259	alr1633-alr1635
alr2790-alr2791	asr5139-alr5143	alr4404-alr4405	alr3583-alr3584	alr3896-alr3897
alr3146-alr3147	mutL-hisD	dmnB-trpD	alr1369-alr1370	alr4114-alr4115
alr4416-alr4417	asr5146-alr5148	all4736-all4737	alr0295-alr0297	alr4878-alr4879
alr3008-alr3009	alr0246-alr0247	alr1232-alr1233	alr2350-radC	asr1309-alr1315
alr3723-alr3725	alr4783-alr4788	asr2953-alr2954	alr3646-alr3649	alr1665-asr1667
alr3930-alr3932	all4256-all4257	alr2426-asr2427	asr3098-alr3102	alr4576-lspA
ndhF-alr0871	alr1627-alr1629	alr0029-glgA	alr3155-alr3156	alr1959-alr1961
asr1817-alr1819	alr2471-alr2472	alr5225-alr5240	alr4973-alr4976	rpmB-all2631
alr0198-alr0199	alr0074-alr0079	alr2857-alr2867	asr1048-pgi	alr4067-alr4069
alr0709-alr0710	dnaJ-asr2994	alr4521-asr4522	all2229-phnC	alr0487-alr0490
alr2738-alr2739	alr5251-alr5254	alr1976-alr1979	all3391-cobW	alr2935-panC
alr1295-alr1302	alr4660-alr4661	alr0730-alr0731	ksgA-alr3231	alr4849-alr4850
alr2256-hstK	asr0365-asr0368	alr3877-asr3878	all4298-all4300	all2622-all2624
asr0148-alr0150	alr1028-alr1031	alr2264-cyaB1	asr0855-alr0857	ileS-alr1074
dnaK-dnaJ	alr2659-alr2660	alr2972-alr2975	alr2482-alr2486	all0955-all0956
alr4504-alr4505	ureE-alr0735	alr4027-alr4028	alr5293-alr5294	alr1531-asr1532
alr0299-alr0301	cobN-alr1690	alr4559-asr4560	aksA-alr1410	alr1094-alr1097
alr4711-alr4712	alr0302-alr0304	alr1200-alr1201	all3735-all3736	alr0819-alr0821
hrcA-alr4046	alr4239-alr4241	bvdR-alr4151	asr0581-alr0587	alr4691-alr4692
alr4695-alr4696	alr4582-alr4589	asr4937-alr4939	groES-alr3663	alr2178-alr2179
alr0642-alr0644	alr2814-alr2816	alr2190-alr2191	alr1674-alr1675	hoxR-psbJ
alr3356-alr3357	alr3095-alr3097	alr3618-alr3623	alr3994-alr3997	alr0739-rpsU

alr2240-alr2241	asr0098-asr0104	alr5286-folE	alr1392-asr1393	alr5180-alr5181
asr4942-alr4944	alr3037-alr3038	alr2558-alr2560	alr4346-alr4347	alr2662-alr2663
asr0798-alr0799	alr1222-alr1224	all2617-all2618	alr3762-asr3763	asr1611-alr1614
alr4009-alr4011	asr5080-alr5081	asr3019-alr3020	asr3042-asr3043	alr3666-ureB
alr2710-alr2711	alr2921-alr2922	asr1451-adx	alr4277-alr4278	alr2111-alr2112
alr1206-alr1209	asr3137-surE	alr2278-alr2280	alr1024-alr1026	alr1004-asr1005
alr0803-alr0806	alr4099-alr4100	alr0668-alr0669	alr4015-alr4017	hetF-alr3548
alr4438-alr4439	alr2131-alr2132	alr0055-alr0058	alr0212-alr0214	alr1808-alr1809
rfbB-alr0039	alr4250-cytA	asr2330-alr2331	alr2741-alr2742	asr1307-alr1308
alr5208-alr5209	asr3598-alr3599	asr5349-asr5350	alr4738-alr4741	alr1821-alr1822
asr5289-alr5290	alr3827-asr3834	asr0682-alr0683	alr2118-asr2120	alr0545-alr0549
alr1128-alr1129	asr5312-asr5313	all2281-all2282	asl3851-all3853	alr2881-exoD
alr4226-alr4230	all5106-all5108	dapF-alr2049	all4396-all4397	uvsE-dnaK
asr0013-alr0014	alr0072-hisB	era-alr0913	alr1229-alr1231	alr1576-alr1579
asr1275-alr1276	alr1085-alr1086	alr3506-alr3508	asr4301-asr4302	alr3590-alr3594
alr1534-alr1535	alr5283-accA	asr3935-alr3937	alr1485-asr1486	alr1492-asr1494
alr2768-alr2769	alr0083-alr0084	all3408-all3410	alr2614-alr2615	asr1661-asr1662
all2955-all2956	alr4064-alr4066	rbpF-alr2312	alr1104-alr1105	alr3187-alr3188
trpE-trpC	alr3910-alr3912	alr5182-asr5183	alr1014-alr1018	alr4222-alr4224
alr4847-alr4848	alr0381-asr0382	hemH-alr3752	alr5317-alr5320	psbX-alr0944
glcD-alr5271	rbpB-all2930	alr3105-alr3106	alr1077-rps1	asl4253-all4254
alr1334-alr1337	alr1726-alr1727	alr2502-alr2505	alr0279-alr0280	orrA-alr3771
alr4637-asr4638	asr2605-asr2607	alr2335-alr2336	alr4512-alr4513	ureC-alr3672
alr1343-asr1344	alr2541-alr2543	alr1923-aroB	alr0515-alr0517	alr2832-alr2833
asr2172-alr2173	alr0840-alr0841	alr2122-alr2123	asr2978-alr2980	asr0179-alr0181
alr0188-alr0191	alr1550-alr1552	asr4594-alr4595	alr1107-alr1108	asr0105-gmk
asr0755-alr0758	all4458-all4459	alr0092-alr0094	ictA-patN	alr4836-alr4837
alr4863-alr4864	alr1917-alr1918	asr4313-asr4314	all4927-all4929	alr3384-asr3390
alr0784-alr0787	alr0518-alr0520	aphA-alr3159	alr2569-alr2570	alr2431-alr2434
alr4447-asr4449	alr2722-alr2723	ubiA-minE	queA-alr1799	alr3085-alr3086
asr3217-alr3219	hepB-alr3701	alr5030-alr5032	alr1113-asr1115	alr4818-alr4819
asr1899-alr1901	alr3561-alr3562	all0166-all0168	alr0114-alr0117	alr1850-alr1855
all4381-all4382	alr0140-alr0142	alr4514-alr4516	alr5242-alr5243	alr1254-alr1255
alr1536-alr1540	ndhD-asr3961	alr3789-alr3790	asr3001-asr3006	coaD-alr4703
alr4995-asr4997	alr2201-alr2207	alr3610-alr3611	all1367-hisH	psbZ-ribH
alr3543-asr3544	pex-alr3980	alr3242-alr3243	alr3863-alr3864	alr0717-alr0720
alr1001-alr1002	alr3471-alr3479	alr2966-alr2967	alr3754-alr3757	alr2535-alr2539
avtA-rimM	alr0288-alr0289	asr3467-alr3469	alr2575-alr2577	alr1376-alr1379
xisC-alr0679	alr0704-asr0705	alr3311-alr3312	asr3657-asnC	alr3376-alr3377
alr3276-alr3277	alr4359-alr4360	asr1399-alr1401	all0807-all0809	alr0552-alr0559
hisG-alr1968	alr1044-alr1045	asr2365-alr2366	alr0972-alr0975	alr4714-crhC
alr2751-alr2752	asr1558-asr1572	alr3052-asr3053	chlH-alr4734	alr5356-hetM
argJ-alr2076	alr5159-alr5162	alr5216-alr5217	alr0998-alr0999	alr3602-alr3603
asr4951-alr4954	alr3296-alr3297	all3740-all3741	alr3707-asr3708	hisF-asr2896
alr0068-rph	asr3605-alr3608	hisC-alr2093	alr4641-alr4642	alr3904-alr3907
murC-alr5068	alr4919-alr4922	alr3955-ndhF	alr1170-alr1171	alr5211-alr5212
alr2780-alr2784	alr3379-moaC	alr5186-alr5190	asl4743-all4744	alr3795-asr3796

ndhF-alr4158	alr1941-alr1942	alr0018-alr0019	alr3175-alr3178	alr1890-alr1892
alr3524-alr3525	asr2937-sodB	psbD-psbC	gatB-alr1398	hglE-alr5353
alr1505-alr1507	alr3351-alr3352	alr2694-alr2698	alr4566-alr4571	alr1920-alr1921
alr4525-alr4537	alr3411-alr3417	asr2135-alr2136	alr3213-alr3216	alr3026-alr3027
alr0946-alr0947	alr2925-alr2927	alr0760-hoxH	asr0837-alr0838	alr0599-alr0600
alr4794-alr4795	alr4454-asr4457	alr2492-alr2496	alr3689-alr3690	ccmK-alr0318
asr3089-alr3091	alr3760-alr3761	alr3280-alr3281	alr3224-alr3225	alr4907-alr4909
alr2478-alr2479	alr3017-alr3018	alr4485-alr4494	alr0576-alr0577	asr0062-rbfA
asr3405-alr3407	alr1372-alr1375	smpB-asr5071	alr2377-asr2378	asr3133-asr3134
alr2294-alr2296	alr4610-asr4612	alr0900-alr0901	alr0428-alr0429	alr2153-asr2155
alr0308-alr0309	alr0970-alr0971	alr3268-alr3269	alr2872-alr2873	alr3803-alr3804
alr1619-alr1624	alr2943-alr2946	icd-asr1828	alr3248-alr3252	alr3240-alr3241
alr5149-alr5152	alr4597-alr4606	alr4469-asr4471	clpP-asr3686	alr4681-alr4684
alr1259-ftsH	alr1997-alr1998	alr1555-alr1556	thrC-asr3294	alr3165-alr3166
alr2411-alr2412	alr1668-alr1669	asr0460-asr0461	pds-pys	alr2081-alr2083
alr3265-asr3266	alr2140-alr2144	alr0986-alr0987	alr3120-ycf44	asr2932-alr2933
alr2522-alr2527	alr2174-alr2176	alr1498-alr1501	alr1449-alr1450	dnaJ-alr2450
asl2370-all2371	alr1212-asr1213	alr3057-lpxD	secA-alr4854	all0333-all0334
glyS-murD	alr4685-alr4686	alr0892-alr0898	asr2041-alr2046	
alr2773-alr2774	alr3077-asr3082	alr2463-alr2465	alr3393-alr3395	
alr3920-glmU	asr0485-alr0486	all3435-all3436	alr5027-alr5028	
alr3816-alr3817	alr4164-alr4167	kaiA-alr2890	all3696-all3697	

Figure S7: Visual operon prediction for the PKS gene cluster [178] of *Anabaena* sp. PCC 7120 by OpPipe (under control, -Nit and -Fe conditions) DOOR [65], ProOpDB [69], Rockhopper [36] (under control conditions). Red color indicates that no operon has been identified, green color indicates an operon, while a darker green indicates that two operons have been predicted.

DANKSAGUNG

Zunächst möchte ich Prof. Dr. Enrico Schleiff für seine Unterstützung und die letzten sechs Jahre danken, in denen ich die Möglichkeit hatte mich von der Bachelor- bis zur Doktorarbeit in verschiedenen Projekten einzubringen. Danke für das spannende und fordernde Forschungsthema sowie den fachlichen Input durch unsere, teilweise zum Glück auch kritischen, Diskussionen. Danke für unser großartiges Bioinformatiker-Domizil im FIAS! Weiterhin möchte ich mich bei Dr. Kathi Zarnack für die Übernahme des Zweitgutachten bedanken.

Mein besonderer Dank gilt hierbei Prof. Dr. Stefan Simm, der mich fachlich bei all meinen Projekten unterstützt hat und immer eine brillante Idee hatte, wenn ich nicht mehr weiterwusste. Weiterhin bin ich sehr stolz auf unsere Freundschaft, die uns nun seit mehr als sechs Jahren verbindet.

Hierbei möchte ich auch meinen liebsten Kritiker Dr. Mario Keller erwähnen, danke für den vielen Input deinerseits und die vielen unvergesslichen Momente. Dem gesamten AK Schleiff danke ich für eine fantastische Zeit und unvergessliche Abende und Sommerschulen. Danke Niclas für die Erweiterung meines musikalischen Spektrums, danke Henning für die interessantesten Gedankenspiele, die man sich vorstellen kann, danke Robin für deine Unterstützung und danke Patrick für die gemeinsame Zeit im AK. Neben dem universitären Leben möchte ich mich beim gesamten Regulated Fund- und Leasing & Factoring Team der Commerzbank AG bedanken, mit denen ich großartige 3 ½ Jahre verbracht habe. Besonderen Dank an Stefan Zantow, Mark Nuttall und Jörg Schreiber für Ihre Unterstützung und Chancen, die mir aufgezeigt wurden. Vor allem dir danke ich Jörg, du warst in den letzten Jahren mein Mentor und hast mir viele Möglichkeiten aufgezeigt und Türen geöffnet. Ich danke dir für deinen Einsatz! Danke an Momo, der mich erst in die Bankenwelt gebracht hat und Martin für viele großartige Erlebnisse.
Auf privater Ebene bedanke ich mich bei Marius und André, an die ich mich immer wenden konnte. Ganz besonders möchte ich aber auf jeden Fall meiner Familie danken, ihr habt mich durch diese Zeit getragen und ohne euch wäre vieles nicht möglich gewesen. Helmut und Christiane danke ich für die Unterstützung und den Rückhalt, den ich schon immer von ihnen erfahren habe, Jenni und Nicky für die Korrektur zahlreicher Texte und ihrer offenen Ohren bei allen Themen dieser Welt. Jenny dir danke ich, dafür dass du mich stark machst, dass du immer für mich da bist und für deine Geduld mit mir in den letzten Jahren!
Der letzte Dank gebührt dir Paula, dafür dass man sich immer auf dich und deine Ratschläge verlassen konnte. Du hast stehst nur das Gute in einem gesehen und unsere gemeinsamen Momente werden mir sehr fehlen.