Heuristic-Based Primer Set Minimization for PCR

Adaptation of Iterated Local Search for a Set Cover Variant

Thys Kok

Supervisor(s): Jasmijn Baaijens, Jasper van Bemmelen

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 28, 2024

Name of the student: Thys Kok
Final project course: CSE3000 Research Project
Thesis committee: Jasmijn Baaijens, Jasper van Bemmelen, Chirag Raman

Abstract

AmpliDiff is an algorithm for studying environmental samples. As the study of those DNA samples is complicated, the runtime of AmpliDiff holds back its usability. One time-consuming part is exactly solving a variant of the Set Cover problem. This paper researches whether this exact solution can be replaced by heuristics to reduce the runtime. The Iterated Local Search framework is applied as heuristic and translated onto our problem. After that, it is tested and compared to the original version of the algorithm based on the found solution size. From the results we can deduce that the heuristic version finds the same solutions as the original. It does so in a faster relative time frame as well. However, due to time constraints, the dataset that was used for testing is rather small and therefore not representative of more complex problems, which are usually solved by AmpliDiff. Thus, for future work we recommend to test the heuristics version on a larger, more representative dataset to check its viability. This paper has laid the groundwork by implementing the heuristic and showing that it effectively works for small datasets, giving reason to also try it out on more complicated problems.

1 Background

Environmental samples are very valuable when it comes to researching genomic data of viruses. Metagenomic profiling [9] directly studies the found samples and attempts to classify them. Those classifications can then be used to study case rates as well as the evolution of viruses present in the samples. In order to be able to classify the metagenomic data it first needs to be multiplied to properly visualise it. While there exist methods that multiply the entirety of the genomic data, such as Whole Genome Sequencing [8], more efficient and cost-beneficent metagenomic profiling can be achieved by only multiplying specific parts of sequences that can differentiate between virus lineages. However finding these parts is not trivial.

AmpliDiff [10] is an algorithm for finding such specific sequences that differentiate well, whilst also finding the necessary DNA strands to be able to initiate DNA multiplication of those sequences. These sequences and strands are respectively called amplicons and primers, which can be used in a multiplying process called PCR [7]. In PCR two primers bind at the edges of the sequence to be multiplied; one forward primer binds on one DNA strand and one reverse primer binds on the other. Via multiple heating cycles primers bind, grow and unbind, thereby exponentially growing the genomic data they are amplifying. Finding the necessary amplicons followed by finding primers to amplify these amplicons is not a trivial process given the large datasets and chemical constraints that come with DNA binding. Hence why some parts of AmpliDiff have an exponential runtime, which cause the algorithm's runtime to increase quite a lot. This research will therefore look into enhancing the runtime of the AmpliDiff algorithm, specifically by improving the third step of the algorithm: the primer minimization step.

In the primer minimization step, an amplicon and DNA sequences, in which the amplicon can be found, are given. Then for each of those sequences primer pairs need to be found to amplify the amplicon. As every input sequence has different DNA surrounding the amplicon or in the edges of the amplicon, a minimization problem arises to find the minimum number of primers needed to amplify the amplicon in all of the input sequences. This problem is similar to the NP-hard Set Cover problem with the sequences being the elements that need to be covered using as few primers as possible. Currently, AmpliDiff uses a mixed-integer linear-programming (MILP) [4] which solves the problem exactly in exponential time. An alternative to the MILP could be heuristics to improve the runtime of AmpliDiff. This research will look into using heuristics for the primer minimization step of the AmpliDiff algorithm to reduce its runtime with the goal of making it more viable for larger datasets, as well as improving its usability in general. It will then compare the found solution to the original version of AmpliDiff to check the effectiveness of the found solution.

2 Methodology

How can the runtime of the primer minimization step of the AmpliDiff algorithm be improved using heuristics? In order to answer this main research question, some assumptions about the problem being solved will be made. Once those are clear, the most suitable heuristic for the problem at hand will be determined. After that the found general heuristic will be translated onto our specific problem.

2.1 Assumptions

Firstly, when it comes to the assumptions made, every amplicon is assumed to be its own problem. This is to prevent the problem from becoming overly complicated, as it would otherwise result in multiplex PCR, which means amplifying multiple different amplicons all in one reaction. If that were the case, a primer added for the first amplicon needs to be kept in mind for all the other amplicons considered after it. The complexity of the problem and solving it efficiently would thereby be increased. Aside from that, AmpliDiff also currently treats every primer as its own problem. While it is therefore interesting to look at multiplex PCR, for the scope of this research we will not look at it.

Secondly, an assumption is made regarding the amplifiability of the amplicons. Amplifiability means the number of genomes in which the found amplicon can actually be amplified with primers. This needs to be done as it is not sure beforehand whether there are actually places in the DNA of each sequence for primers to bind around the amplicon and for primers to be compatible with each other. For this research the amplifiability of the amplicons is assumed to be at 100% to simplify the process of finding

a solution by leaving out a trade-off between amplifiability and the amount of primers needed. Next to that, it is also assumed that the algorithm receives only amplicons for which a primerset exists that can satisfy this 100% amplifiability.

2.2 The Iterated Local Search framework

We apply the Iterated Local Search framework (ILS) as heuristic to use for this project, as it has been proven to work successfully on the Set Cover problem [6]. Given that the primer minimization problem is similar to Set Cover, the ILS framework might also prove successful for our case. Next to this, the ILS framework provides a simple and flexible basis that can be extended to other heuristics in the future. An example of such a heuristic is simulated annealing, which has been proven to be very adaptable to many problems [1]. Furthermore, this implemented version of ILS shares some similarities with the MetaRAPS heuristic [5] when it comes to distance and local search definitions. The MetaRAPS heuristic has also been proven to work well on Set Cover problems.
The ILS framework is based on local search principles, as the name suggests, and works as following: first, an initial solution is found and assigned a score, then its neighbouring solutions are explored and checked to see whether they have better scores. If so, that solution becomes the new solution and its neighbours are explored in the following cycle. After a certain number of cycles, a larger shake-up to the current best solution is introduced, named a perturbation in order to prevent the search from getting stuck in local optima. After a set amount of perturbations have occurred, the search stops and the best-scoring solution is returned as the found solution. ILS can explore the solution space this way to find a solution using heuristics. The ILS framework consists of four core elements:

- The initial solution: how is it constructed?
- The local search
- The perturbation
- The acceptance criteria

In order to use the ILS framework for solving the primer minimization step these four criteria need to be translated such that they work within the primer optimization step and the problem it solves. Therefore the next sections will go into detail about the choices made for each core element regarding the translation to the AmpliDiff algorithm.

ILS: initial solution

For the initial solution a greedy approach was chosen to quickly create a solution that is not wholly random. The greedy initial solution is created as following: The algorithm receives from the previous steps of AmpliDiff an amplicon with its corresponding possible forward and reverse primers and the sequences in which it needs to amplified. First the two sets of primers are both sorted on how many sequences the primer can bind in. After that primers are picked greedily, with the one that can amplify the most sequences picked first. However, before a greedily chosen primer can be added, it first needs to be checked on two constraints: whether it is compatible with the previously-added primers and whether it fits the current temperature range. For the former, it is important to check whether primers do not bind with each other, which would make the PCR process less efficient, as primers bind to each other instead of the sequences that need to be amplified. Secondly, there is a temperature constraint to make sure the binding and unbinding of the primers works [3]. For this to be possible, all primers should have a melting temperature that falls between a given temperature range. Only upon satisfying these two conditions can a primer be added to the solution. Primers are greedily added to the solution until all sequences can be amplified.
However, because of the above-explained restraints on adding primers, it can occur that no feasible solution is found. Simply resetting the creation of the initial solution would not work to solve this issue, as the initial solution is created in a greedy deterministic way. To make sure that the algorithm does not get stuck on an infeasible solution, a banlist is utilised: every time two primers are in conflict with each other for the first time, they both get an added penalty point. Then the probability of adding a primer to the solution becomes one divided by the number of penalty points the primer has. Each primer starts with one penalty point to prevent division by zero. The same concept is used for the temperature constraint as well: whenever a primer cannot be added to the solution, the primer preventing the addition, as well as the current primer, each get a penalty point.

ILS: local search

Then from our initial solution we want to perform local searches to check whether better neighbouring solutions can be found. For this a definition of ”closeness” of solutions needs to be defined to know which solutions to explore in a local search. In our case a solution is a set of primers that we use to amplify the amplicon in every input sequence. Closeness is then defined as the number of primers which are the same between primer sets. Hence, neighbouring solutions are primer sets which differ from the current set by at most X primers with X being an hyperparameter. In order to explore these solutions, a local search consists of a pruning approach: by removing primers from the found solution set, we end up with a partial solution that has at least a notion of closeness equal to the remaining primers to the original solution. Then to come up with a new solution, the problem is once again solved, now taking the partial solution as a starting point. This way the solution space surrounding the initial solution is explored. Solutions can be compared using the primer set size as their score, with as goal to minimise the primer set and thus its size. One last thing to touch upon for the local search is the number of primers to be removed from a

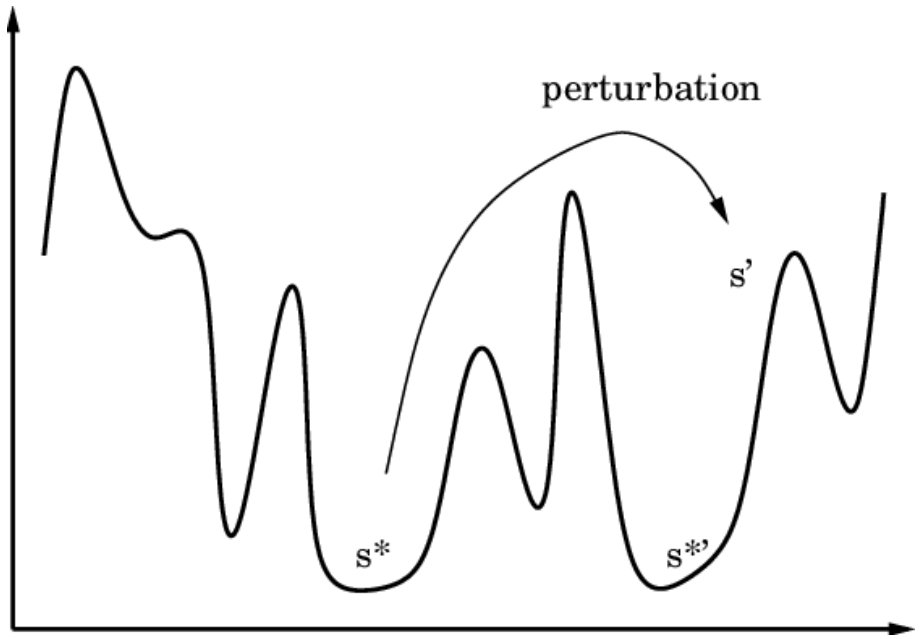

Figure 1: Perturbations can help escape from a local optimum. From "Iterated local search: Framework and applications," by Lourenço, H., Martin, O., & Stützle, T., 2010, *Handbook of Metaheuristics*, p. 6.

found solution: the hyperparameter X. As not all amplicons need the same amount of primers to be fully amplifiable, X is a percentage of the primer set instead of a fixed number. The optimal value of this hyperparameter is to be decided through the tests conducted for this research, as it cannot be decided clearly beforehand.

ILS: perturbations

The approach for a perturbance is similar to that of the local search, except that a larger percentage of the solution is removed than happens with X from local search. We define Y as this percentage, with Y being larger than X. Y is also the second hyperparameter that needs to be optimized for the problem. What needs to be given special care here however is the fact that a perturbance should not be easily undone by the algorithm [6]. If this were the case, the algorithm could often get stuck in the same local optima again. However, removing a large percentage of primers as a perturbance should be able to avoid this trap due to the fact that the primers in the solution set are not disjoint. While a primer might have been the best option during the creation of the initial solution, if subsequently added primers now also cover most of the sequences that primer covers, it might not be the best option to pick it anymore. Instead, as a new combination of sequences now is without coverage, a different primer might be used instead. Thus, the perturbance should avoid the pitfall of quickly reverting to the previous optimal solution.
Another aspect to keep in mind is that two solutions that are both approximately as good as each other can still be very far apart when it comes to the primers they consist of (i.e. two very distinct primer sets yield a similar score). A high percentage for Y is expected to be needed in order to properly explore the entire solution space and avoid getting stuck in local optima. The desired effect of perturbations can be seen in Figure 1: escaping a local optimum and exploring a different part of the solution space.

ILS: acceptance criteria

Lastly, there is the acceptance criteria, that is, when will a new solution be accepted. A solution is better than a previous one when the size of its primerset is smaller.

At the end of the cycles, the best solution found at that point is returned as the final solution. A high-level overview of all the parts of the ILS framework applied to the primer minimization step can be seen in Figure 2.

3 Results

3.1 Experimental Setup

To see whether this ILS version of AmpliDiff works well, it is compared to the original version of AmpliDiff by running both on a dataset and comparing the results. The dataset used for this research is the Example that is part of the AmpliDiff codebase,

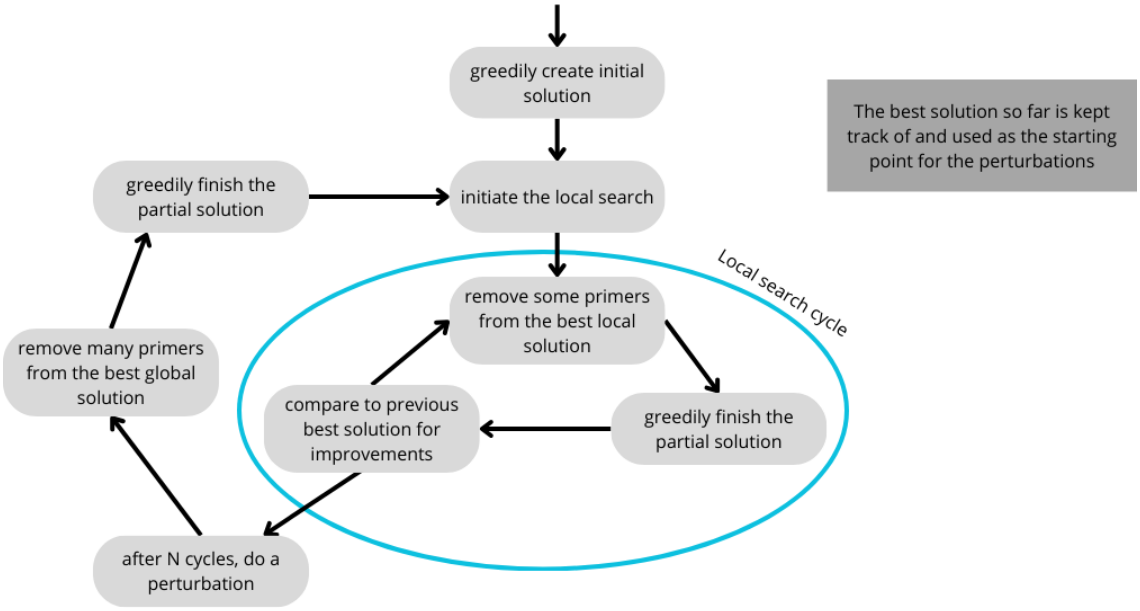

Figure 2: High-level overview of the flow of the heuristic-based primer minimization step.

which can be found on GitHub [1]. It contains 24 SARS-Cov-2 sequences which come from NCBI Virus [2]. The experiment is run on a Windows laptop with 16 GB RAM with a Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz processor. The arguments for both versions of AmpliDiff are the sequences_aligned.fasta containing the pre-aligned input sequences, the metadata.tsv file, which has the lineages for each of the input sequences, and lastly Example_ouput/ as the designated folder for output logs. For the heuristics version of AmpliDiff, the original code is modified such that the primer optimization step works using the ILS framework, as explained above.
There are two hyperparameters for the heuristics version that are tested: one is the amount of primers that is removed in a perturbation and the other is the amount of primers that is removed in a local search. The percentage for the local search removal will be tested with values of 0.10% and 0.25%, while the percentage for the perturbation removal will be tested with values of 0.50% and 0.75%. As the modified algorithm contains randomized elements, such as the choice of which primers are removed during the pruning, the algorithm will be run 10 times for the four combinations of the above-given values for the hyperparameters. The number of perturbances and local search cycles will be set to 5 and 20 respectively for every run, which should be enough to show the change of the found solution over time, given the size of the dataset on which it is run. Based on that, we can compare the heuristic version using the number of cycles it needs to find a satisfactory solution. This should give a better picture than comparing the two versions based on absolute runtime, as the implemented ILS framework most likely still has a lot of room left for optimization, given the relatively short time window during which this project had to be done.

3.2 Results

Four amplicons are needed to cover all the sequences from the Example. Of these four amplicons Figure 3 contains the results of the heuristics version of AmpliDiff for the first one. On the X-axis is the iteration number and on the Y-axis the size of the found solution. Each different colour represents one of the ten different runs. As can be seen in the graph, the algorithm either finds a solution or fails to find any solution at all; there are no iterations where a solution with a different size is found. When the algorithm manages to find a solution, it consists of the same amount of primers found by the original version of AmpliDiff: three primers. For every run, the algorithm only manages to find a feasible solution after 17 runs. After which more are found, because the found one is now used as a starting point for searches and perturbations. Depending on the used hyperparameters the frequency, with which feasible solutions are found, differs. The most notable difference is between runs with 75% perturbation removal versus those with 50%.
For the second amplicon, a feasible solution is immediately found, as shown in Figure 4. Over all iterations that follow, no better solution is found nor is there an infeasible solution found. The small dataset that was used, allowed the greedy selection algorithm to immediately find a solution that was not improved in any run afterwards. Compared to the solution of 8 primers that was found by the original version of AmpliDiff, less primers were found in general. The number of primer pairs found is still the same though.
In Figure 5 the results for the third amplicon can be seen. As was the case with amplicon 2, a solution is found in the first

[1] https://github.com/JaspervB-tud/AmpliDiff/blob/main/Example/README.md

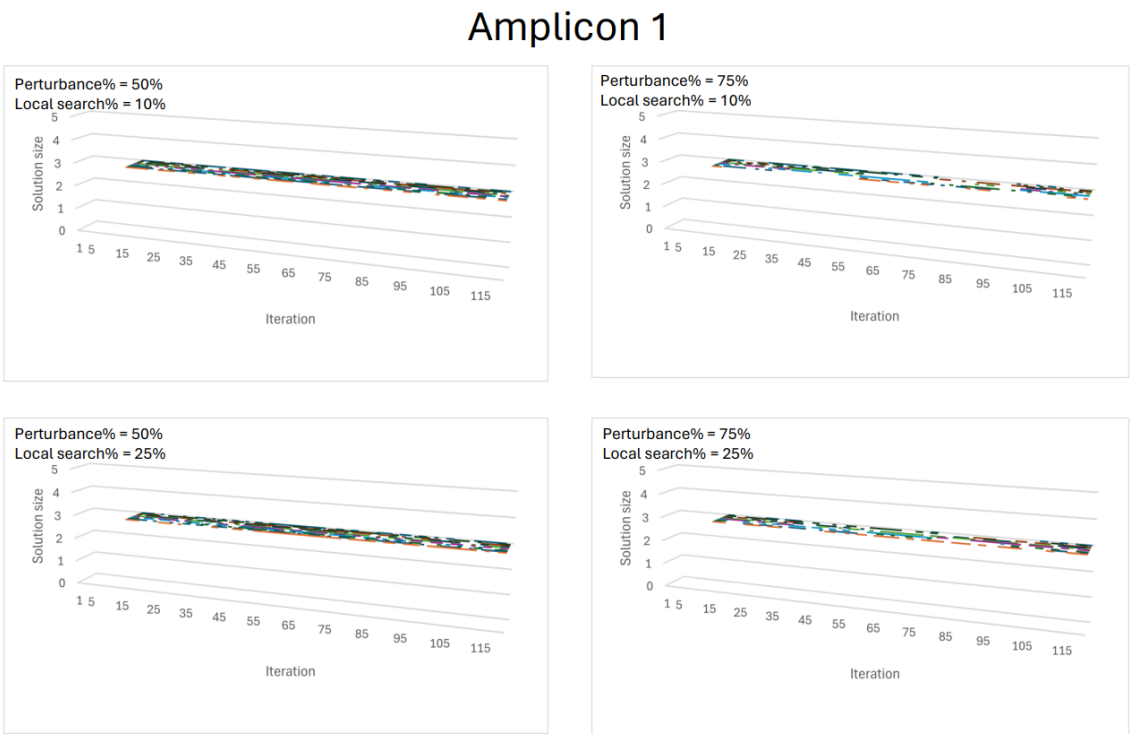

Figure 3: The found solution sizes for the first amplicon per hyperparameter combination. Each colour shows the results of a run for that combination. The solution found by the original version of AmpliDiff contains three primers.

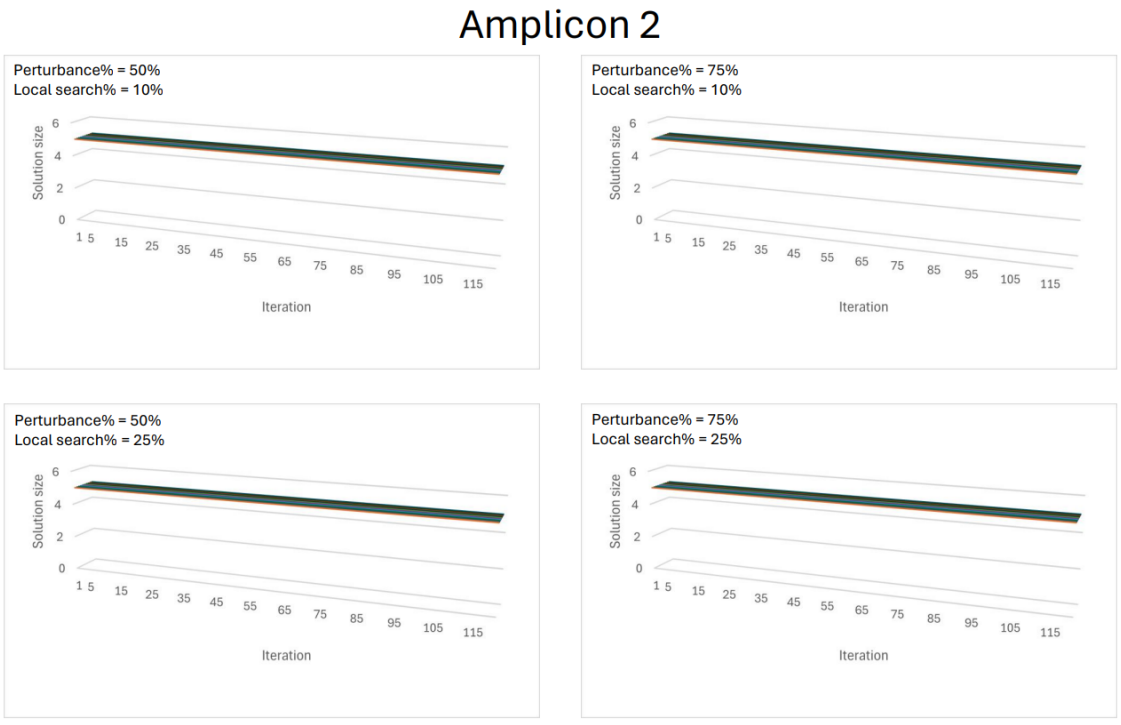

Figure 4: The found solution sizes for the second amplicon. The solution found by the original version of AmpliDiff contains eight primers. However, the amount of primer pairs found is still the same.

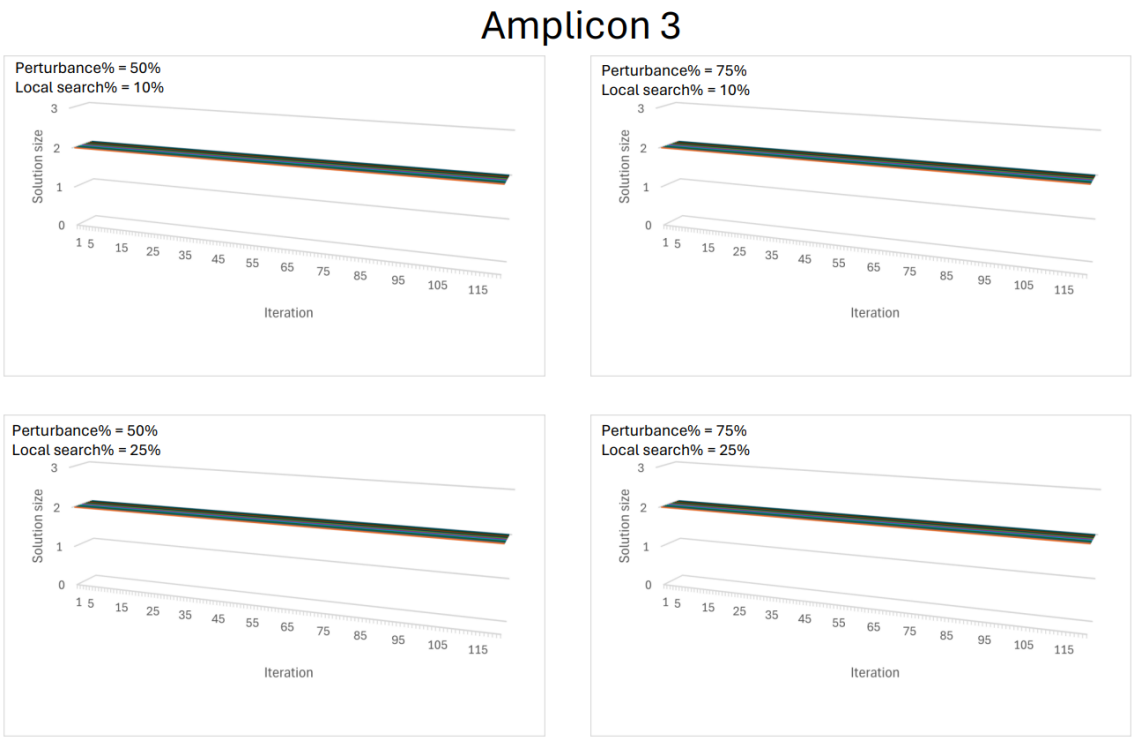

Figure 5: For the third amplicon, a solution of two primers is found, which is the same as found by the initial AmpliDiff algorithm.

iteration and then equally good solutions are found in all the following iterations. Here only two primers are needed to cover all the sequences: one forward primer and one reverse primer. Once again, this same solution was also found by the initial version of AmpliDiff.

For the fourth and last amplicon, the best-found solution is once more achieved during the first iteration, as can be seen in Figure 6. However, when exploring the solution space in subsequent runs, sometimes infeasible solutions are found. As with the first amplicon, depending on the value for the perturbance percentage, the frequency with which infeasible solutions are found changes. The higher percentage of 75% once again causes more infeasible solutions to be found compared to the runs done with a lower perturbation percentage of 50%. Despite that, all runs still returned a solution with equal size to the original AmpliDiff's findings.

4 Responsible Research

These results were achieved in an ethical and appropriate way with regards to the integrity of the data used and the reproducability. For the former, NCBI Virus data was used, as previously mentioned. Given that NCBI is approved and funded by the government of the United States, its data should be of proper quality. When it comes to the type of data we are handling, viral genomes, there are no sensitive aspects involved through which human individuals could be disadvantaged.

For the reproducability of the research, a fork-off of the AmpliDiff GitHub repository[2] contains all the code that was added for the heuristics version of AmpliDiff to work. This fork-off also preserves a snapshot of the state of AmpliDiff at the time of doing the experiments, such that there is always access to a working version, even after the original AmpliDiff is further developed. This way the experiment can always be reproduced, as the Example dataset, on which the experiments were run, is also included. Lastly, the log output files, on which Figures 3 to 6 are based, have also been added to the forked-off repository, such that anyone interested in the results can access them freely.

5 Discussion

The achieved results for the heuristic version are great, based on the graphs shown in the results. All the solutions found by the original version are matched within 20 cycles, and often even immediately. However, the dataset on which we are running, plays a large part in this. Due to time constraints, a small dataset was chosen, which is not very reflective when keeping the broader picture in mind: often many more sequences than 24 need to be amplified. Such larger problems increase the complexity,

[2]https://github.com/DenkeyKong/AmpliDiff/tree/heuristic-based-primer-minimization. The branch named heuristics-based-primer-minimization has the relevant code on it. In the About section, one can see where exactly the changes can be found.

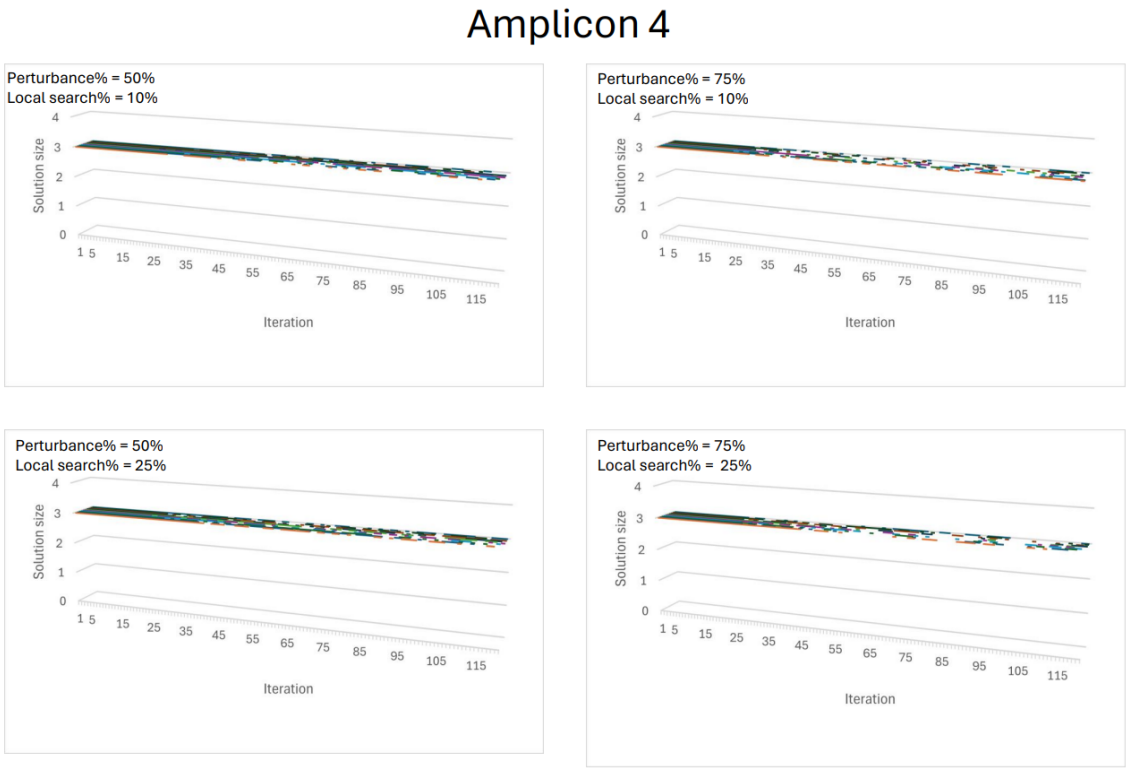

Figure 6: The found solution sizes for the fourth amplicon. The solution found by the original version of AmpliDiff is the same size and also contains three primers.

through which the results of heuristics based AmpliDiff may then be different. As for the result of these experiments, they do show that heuristics are effective for these smaller datasets, which means that they could also very well work for larger datasets. Therefore we recommend testing the heuristics variant on a larger dataset that is more in line with the experiments in [10]. This should give a good image on whether it is worth using it and expanding on it for future AmpliDiff usage.
As for the hyperparameter values that were tested, the local search percentage does not seem to have much influence, whether it is 10% or 25%. This is caused by the chosen dataset, as with solution primer sets of a size of 3, 10% or 25% does not matter much. What does matter, even for these smaller set sizes, is the perturbation percentage: For both amplicon 1 in Figure 3 and amplicon 4 in Figure 6 there is a clear difference in the frequency with which feasible solutions are found between graphs on the left and those on the right. The graphs on the left have 50% of primers removed during perturbations, whereas those on the right have 75%. All graphs initially find a feasible solution, but after that the runs with 75% perturbance removal appear to branch out more. This results in more often finding a solution that is infeasible. However, this is not necessarily a bad thing, as the solution space is explored better. It is difficult to say whether 75% or 50% is preferred, given that these ventures have almost no chance of finding a better solution, as the best solution is so quickly found in the small dataset. Still, based on these results, 75% seems preferred, given that it more thoroughly explores the solution space.

5.1 Future Work

Next to the dataset on which to test the implemented modifications, there are also other future recommendations for improvements to the heuristics version of AmpliDiff. For example, versions of the algorithm that more closely incorporate MetaRAPS or use simulated annealing for the acceptance criterion, can be developed. They can be compared to the current heuristics version to see whether they give better or faster solutions. Given the ease of modification of the ILS framework, such changes can be made efficiently by utilizing the delivered codebase for this project. Next to these alterations to the algorithm, its use-cases can also be improved by relaxing the assumptions made for this research: Sub-100% amplifiability can be implemented, as often the amount of extra primer pairs needed to amplify every single sequence is not worth it. As previously mentioned, multiplex PCR could also be looked into. While it might be a much more complex problem, the version of AmpliDiff developed for this research can provide a starting point for heuristically solving it. Lastly, this research has gone into using heuristics for the primer optimization problem, but there is also the primer feasibility problem for which heuristics can also be used to solve it, although more intense change to the currently-used algorithm might then be needed, as the goal would not be minimization.

6 Conclusion

For more cost-beneficent metagenomic sequencing the AmpliDiff algorithm was developed to find both primers and amplicons in one go for PCR procedures. While the AmpliDiff algorithm led to a more cost-beneficent and efficient alternative compared to existing techniques, such as Whole Genome Sequencing, the algorithm's runtimes could still be improved upon. Its primer optimization step runs in exponential time, which is not good for the runtime and therefore the general usability of the algorithm. Heuristics are an interesting option to explore to improve the runtime, as opposed to the current exact methods. An Iterated Local Search framework was chosen to be implemented for the AmpliDiff algorithm. The heuristics version of AmpliDiff was then compared to the original, based on the size of the primersets they found. Both were run on an Example dataset from the AmpliDiff codebase. The results showed that the heuristics version is able to find solutions of the same quality as the original version, while finding them fast via measurement using the number of cycles the algorithm needed. There is a caveat however in that the used dataset is small and does not have the complexity of the problems that the AmpliDiff algorithm normally solves. Therefore future testing on larger datasets is recommended to make sure the heuristics version provides sufficient results for those problems as well. Because of a lack of time, such tests were sadly not a part of this research. Still, it has shown that the heuristics version at least works on smaller problems, which gives reason to continue testing it for larger ones as well.

References

[1] Emile Aarts, Jan Korst, and Wil Michiels. *Simulated Annealing*, pages 187–210. Springer US, Boston, MA, 2005.

[2] J Rodney Brister, Danso Ako-Adjei, Yiming Bao, and Olga Blinkova. NCBI viral genomes resource. *Nucleic Acids Res*, 43(Database issue):D571–7, November 2014.

[3] CW Dieffenbach, TM Lowe, GS Dveksler, et al. General concepts for pcr primer design. *PCR methods appl*, 3(3):S30–S37, 1993.

[4] J.-M. Gauthier and G. Ribière. Experiments in mixed-integer linear programming using pseudo-costs. *Mathematical Programming*, 12(1):26–47, Dec 1977.

[5] Guanghui Lan, Gail W. DePuy, and Gary E. Whitehouse. An effective and simple heuristic for the set covering problem. *European Journal of Operational Research*, 176(3):1387–1403, 2007.

[6] Helena Lourenço, Olivier Martin, and Thomas Stützle. *Iterated Local Search: Framework and Applications*, volume 146, pages 363–397. 09 2010.

[7] Mike McPherson and Simon Møller. *Pcr*. Taylor & Francis, 2006.

[8] Pauline C. Ng and Ewen F. Kirkness. *Whole Genome Sequencing*, pages 215–226. Humana Press, Totowa, NJ, 2010.

[9] Jonathan L. Sebat, Frederick S. Colwell, and Ronald L. Crawford. Metagenomic profiling: Microarray analysis of an environmental genomic library. *Applied and Environmental Microbiology*, 69(8):4927–4934, Aug 2003.

[10] Jasper van Bemmelen, Davida S. Smyth, and Jasmijn A. Baaijens. Amplidiff: An optimized amplicon sequencing approach to estimating lineage abundances in viral metagenomes. *bioRxiv*, 2023.