A Metaheuristic Approach for Multiple Sequence Alignment

October 2024

By

Peter Moorhouse

Student number 202247615

Word count: 2264

Contents

1. Project background and purpose 3
1.1. Introduction 3
1.2. Objectives 3
1.2.1 Primary Objectives 3
1.2.2 Secondary Objectives 4
1.3. Scope 4
1.4. Deliverables 5
1.4.1 Software Releases 5
1.4.2 Documents 5
1.5. Constraints 6
1.6. Assumptions 6
2. Project rationale and operation 7
2.1. Project benefits 7
2.2. Project operation 7
2.3. Options 8
2.3.1 Programming Languages 8
2.3.2 Algorithm Design 8
2.4. Risk analysis and mitigation 8
2.4.1 Risk Matrix 8
2.4.2 Risk Analysis 9
2.5. Ethical and legal considerations 10
2.6. Commercial considerations 10
3. Project methodology and outcomes 11
3.1. Initial project plan 11
3.1.1. Tasks and milestones 11
3.1.2. Schedule Gantt chart 13
3.2. Project control 14
3.3. Project evaluation 14
4. References 15
5. Appendix a 17
6. Appendix b 18

1. Project background and purpose

1.1. Introduction

In this project, a software tool will be developed to tackle the multi-objective optimization problem of Multiple Sequence Alignment (MSA) - a common analysis task in Bioinformatics. The tool will aim to produce high-quality alignments of biological sequences in a time-efficient manner.

Note: A brief glossary of terms related to the MSA problem has been provided in **Appendix a**.

1.2. Objectives

1.2.1 Primary Objectives

1. Perform Multiple Sequence Alignment in a Time-Efficient Manner

The produced software should be able to align a typical testcase of 6 protein sequences within 10 seconds on a university desktop computer. The resulting alignment of sequences must be a valid solution – conserving the original sequence content and identifiers given as input.

2. Assess the Viability of a Single-State Approach for Iterative Alignment

To address their underrepresentation in recent studies, a single-state metaheuristic algorithm such as 'Simulated Annealing' should be implemented and assessed in its ability to guide an effective optimization process for MSA. A single-state form of the software could be contrasted against a population-based approach or assessed relative to an existing alignment tool.

3. Support Established Bioinformatics File Formats

The alignment tool should be able to read biological sequences from an established file format such as FASTA. Likewise, the tool should support an established file format for outputting sequence alignments, such as FASTA, PHYLIP or NEXUS (EMBOSS, n.d.). The user should be able to specify the input source and output destination as command-line arguments.

4. Consistently Produce High-Quality Alignments of Sequences

As assessed in a case study using structural benchmarking, the alignment tool should demonstrate the ability to consistently produce alignments of a comparable quality to established software packages such as Clustal Omega (Sievers et al., 2011).

For this interdisciplinary project, structural benchmarks such as BAliBASE (Thompson et al., 1999) are a valuable source of domain expertise, this is discussed further in **Section 1.6**.

1.2.2 Secondary Objectives

5. Output a Set of Alignments Offering Different Objective Trade-offs

Keeping step with recent research, the software should leverage multiple objective functions to guide the pareto-optimization process (Ibrahim et al., 2024). As output, the tool should produce a non-dominated set of at least 5 alignments that represent different compromises between the objectives.

6. Support Batch Alignment of Multiple Files

The alignment tool should support the alignment of a series of input files from a specified directory. The user should be able to specify a source and destination directory using command-line arguments. The software should work through each input in sequence and output the resulting alignments to the destination directory.

7. Indicate Progress in Aligning Sequences

As the software may need to process a set of sequences for multiple seconds at a time, the software should display a clear indicator of how much progress has been made on the current alignment. For example, the program could present a progress bar using ASCII characters.

1.3. Scope

This project will entail the development of software that performs the specialist task of Multiple Sequence Alignment (MSA). The alignment tool will be developed using an agile methodology and leverage metaheuristic algorithms to tackle the MSA problem.

While a number of studies have explored metaheuristic approaches for MSA, this project aims to address the underrepresentation of single-state methods in recent research (Calvet et al., 2022). Further, the project offers opportunity to explore novel combinations of objective functions to guide the optimization process.

The project will conclude with a comparative case study, comparing the performance of the developed tool against established alternatives such as MAFFT, Muscle and Clustal Omega (Pais et al., 2014).

Despite being a key feature of some sequence alignment packages such as Clustal X (Larkin et al., 2007), the development of a rich graphical user interface (GUI) lies outside the scope of this project. Instead, emphasis is placed on producing high-quality alignments in a time-efficient manner.

1.4. Deliverables

1.4.1 Software Releases

A series of iterations of a metaheuristic alignment tool 'MAli' will be developed using an iterative methodology. The individual iterations are outlined as follows:

MAli v0.1 (November 12th, 2024)

- Should be able to read sequences from a suitable file format and perform MSA using a simple strategy to produce a valid (possibly low-quality) alignment as output.

MAli v0.2 (November 26th, 2024)

- Should demonstrate iterative alignment using metaheuristic algorithms as a 'proof-of-concept', producing higher quality alignments than those of v0.1.

MAli v1.0 (December 10th, 2024)

- Should present a full implementation of an iterative alignment tool, improving on v0.2 in either solution quality or time-efficiency.

MAli v1.1 (January 14th, 2025)

- Should result from experimentation on the design from v1.0, with the goal of improving solution quality or time-efficiency.

MAli v1.2 (January 28th, 2025)

- Should approximate the 'pareto front' of MSA solutions, producing a set of high-quality alignments as output for a decision maker to choose from.

MAli v1.3 (February 11th, 2025)

- Should fulfil the documented functional and non-function requirements and offer a performance or quality improvement relative to all previous iterations of the software.

1.4.2 Documents

Two key documents will be produced as part of the project:

Project Definition Document (October 14th, 2024)

- A document clarifying the objectives, scope, and deliverables of the project. It should propose a timeline and evidence consideration of risks, mitigations and ethical concerns.

Final Report (April 1st, 2025)

- A comprehensive report which details key elements of the project and documents efforts to meet the objectives outlined. The report should include a literature review and conclude with a critical evaluation of the project.

1.5. Constraints

In order to be a viable choice of alignment tool for bioinformaticians, the software must be compatible with established bioinformatics file formats. This is also necessary to enable effective assessment of the tool as outlined in **Section 3.3**.

With existing alignment tools in mind, a suitable format should be chosen for reading sets of sequences as input, and outputting alignments of sequences respectively. Potential formats to consider for outputting alignments include FASTA, PHYLIP and NEXUS (EMBOSS, n.d.).

1.6. Assumptions

This work is predicated on the assumption that the performance of alignment software as assessed via structural benchmarking is indicative of the tool's real-world performance at Multiple Sequence Alignment (MSA). An assumption of this nature is necessary as none of the project staff are bioinformaticians, and an external review cannot be commissioned due to financial constraints.

As described by Bahr et al. (2001), structural benchmarks are designed to offer a comprehensive evaluation for sequence alignment software. Today, their use is prevalent in the literature. In a review of 45 recent papers, structural benchmarking was found to be the most popular quality measure for MSA (Ibrahim et al., 2024).

2. Project rationale and operation

2.1. Project benefits

If successful, the project will provide an evidenced perspective on the viability of a single-state approach for iterative sequence alignment - a gap in recent research. This could draw attention to single-state methods as candidates for further research. Additionally, the project has scope to contribute to the current understanding of pareto-optimization for MSA, as a new combination of objective functions could be found to be highly effective.

Should the software be shown to produce solutions of sufficiently high quality, the aligner could serve a role alongside other software packages in consensus-based sequence alignment or see direct use as a preference choice by some bioinformaticians. In both these scenarios, the project could serve to improve the accuracy of bioinformatics analysis processes.

2.2. Project operation

To inform the software design and experiments that follow, a literature review will be conducted. Key research tasks will include identifying a suitable way to model the problem, determining which metaheuristic techniques are most applicable to MSA, and collating a selection of objective functions to experiment with.

An agile methodology will be used for the iterative development of the software. Drawing from the SCRUM framework (Schwaber & Sutherland, 2020), development will take place in a series of two-week sprints for which work items are selected from a prioritized backlog.

Each sprint will aim to fulfil a specific goal, such as improving the time-efficiency of the algorithm. The conclusion of each sprint will involve a reflection on progress within the sprint and see the release of a new version of the software – accompanied by details of its performance in testing.

Once the final iteration of the software has been developed, a comparative study will be used to assess the tool relative to established alternatives such as MUSCLE and MAFFT (Pais et al., 2014). This will enable a comprehensive evaluation of the project (discussed in **Section 3.3**).

2.3. Options

2.3.1 Programming Languages

The choice of programming language will likely be informed by the high-level design of the software, after having captured the requirements. A simpler design may be feasible in C++, which may offer performance benefits over other languages. A design with greater complexity, or clear opportunities for unit testing may be a reason to use C# due to its rich ecosystem of supporting tools.

2.3.2 Algorithm Design

In designing the algorithm there are a wide range of metaheuristic strategies to choose from, each with different properties, design choices and parameters. In addition to the choice of a strategy, the configuration of its parameters also presents options to be considered. Further, the literature surrounding MSA offers a selection of objective functions for the problem (Ibrahim et al., 2024).

These many different areas of decision-making present a significant challenge in developing a proficient alignment tool. It is for this reason that an iterative development methodology has been proposed in SCRUM, as each successive sprint of development presents an opportunity to experiment and find ways to improve on the design.

2.4. Risk analysis and mitigation

2.4.1 Risk Matrix

		Likelihood (L)		
		1 - Low	2 - Medium	3 - High
Severity (S)	1 - Very Low	1	2	3
	2 - Low	2	4	6
	3 - Medium	3	6	9
	4 - High	4	8	12
	5 - Very High	5	10	15

Figure 2.4.1 *A risk matrix showing how Risk Impact can be estimated using Likelihood and Severity.*

2.4.2 Risk Analysis

Risk	Raw Risk	Mitigation	L	S	Residual Risk
Hard disk failure	5	Proactive: Make use of cloud storage services for key files relating to the project wherever possible. Maintaining copies on multiple platforms will further reduce the risk of losing work.	1	1	1
Poor project planning	6	Proactive: Break anticipated tasks down until they are shorter than two weeks in duration. Discuss these tasks with the project supervisor and agree on clear milestones to indicate progress.	2	2	4
Final product fails testing due to bugs	10	Reactive: Since an iterative development methodology is in place, select a previous iteration of the software to be used as the final version. Attempt a fix if enough time is available.	2	2	4
Insufficient documentation for use of the software	10	Proactive: All released iterations of the software must include a clear 'README' file explaining software functionality and giving directions for use.	1	3	3
None of the released iterations produce valid solutions	8	Proactive: Ensure that producing valid solutions to the MSA problem is one of the first requirements to be satisfied by a software release, prioritized over performance and solution quality.	1	4	4
Researcher becomes hospitalised due to sudden illness	5	Reactive: Notify the project supervisor of this as soon as possible. Discuss how the project plan can be adapted if necessary and contact student services.	1	4	4
Project supervisor becomes unavailable	4	Reactive: Discuss this circumstance with the module lead if it occurs.	1	2	2
Scope creep results in the project being unfinished	6	Proactive: Ensure that any work taken on aligns with primary or secondary objectives for the project. Prioritize primary objectives.	1	2	2
External datasets for testing become unavailable	3	Reactive: Generate synthetic test data for testing or create a set of simple testcases by hand. Use this data instead and communicate the compromise.	1	2	2

Figure 2.4.2 *Table of Risks and Mitigations*

2.5. Ethical and legal considerations

The project will not involve test subjects or sensitive user data. All data to be used with the software will be either entirely synthetic or sourced from well-documented, reputable datasets – and used in accordance with their license terms. As such, the project has no foreseeable ethical implications and complies with relevant legislation, such as the Data Protection Act (2018).

2.6. Commercial considerations

If undertaken independently from the University of Hull, an estimated cost for this project is £11663.00. A breakdown of this estimate is presented in the table below. The key considerations for expenditure were project staff, software subscriptions and access to journal articles.

Title	Rate	Quantity	Total Cost
Undergraduate Researcher	£16.00/h	400 hours	£6400.00
Project Supervisor - Level 9	£110.00/h	20 hours	£2200.00
GitHub Enterprise	£16.04/mo*	9 months	£144.36
Visual Studio Enterprise	£190.96/mo*	9 months	£1718.64
Literature Access Budget	£1200.00	--	£1200.00
		Total Cost	**£11663.00**

*Costs converted to GBP from United States Dollar (USD)

Table 2.6 *Breakdown of estimated costs for the project – totalling £11663.00.*

While typically not directly monetised, work of this nature may be eligible for charitable funding as it offers potential contribution to bioinformatics and applied soft computing. Alternatively, a project of this kind could attract private investment for an exclusive research tool that may offer a competitive advantage – for example in the pharmaceutical industry.

3. Project methodology and outcomes

3.1. Initial project plan

3.1.1. Tasks and milestones

Category	ID	Task/Milestone Details	Priority	Est. Duration
Planning	1.1	Define a set of aims and objectives to inform the project direction.	High	3 days
Planning	1.2	Consider any causes for ethical concern. Document potential risks and appropriate mitigations.	Medium	1 week
Planning	1.3	Break down and sequence project tasks to produce a Gantt chart of the project schedule.	Medium	1 week
PDD	2.1	Write a section covering the project aims, objectives and scope. Define a set of deliverables.	High	3 days
PDD	2.2	Write a section conveying the benefits of the project and how it will be conducted.	Medium	3 days
PDD	2.3	Ask for feedback on current PDD draft, then make revisions.	Low	1 day
PDD	2.4	Detail an account of how the project will be undertaken in terms of tasks and scheduling.	Medium	3 days
PDD	2.5	Ensure that the document is well-presented and coherent.	Low	3 days
PDD	M-A	**14th Oct, '24** - The Project Definition Document (PDD) has been completed.	Milestone	3 weeks
Report	3.1	Write a section introducing the project including its aims, objectives and context.	Low	1 week
Report	3.2	Conduct a literature search, reading papers and building an overall narrative for the review.	High	2 weeks
Report	3.3	Read relevant articles and write about metaheuristics and their applications.	Medium	3 weeks
Report	3.4	Read relevant articles and write about the history of MSA software and approaches.	Medium	3 weeks
Report	M-B	**25th Oct, '24** - The literature review has been completed.	Milestone	5 weeks
Report	4.1	Capture a discrete set of requirements to direct the design and development of the tool.	High	3 days
Report	4.2	Produce UML diagrams as high-level software design incl. Use Case, Package, Class diagrams.	Medium	3 days

Figure 3.1.1a *– A table of tasks & milestones covering project planning, report writing and software design. (Part 1 of 2)*

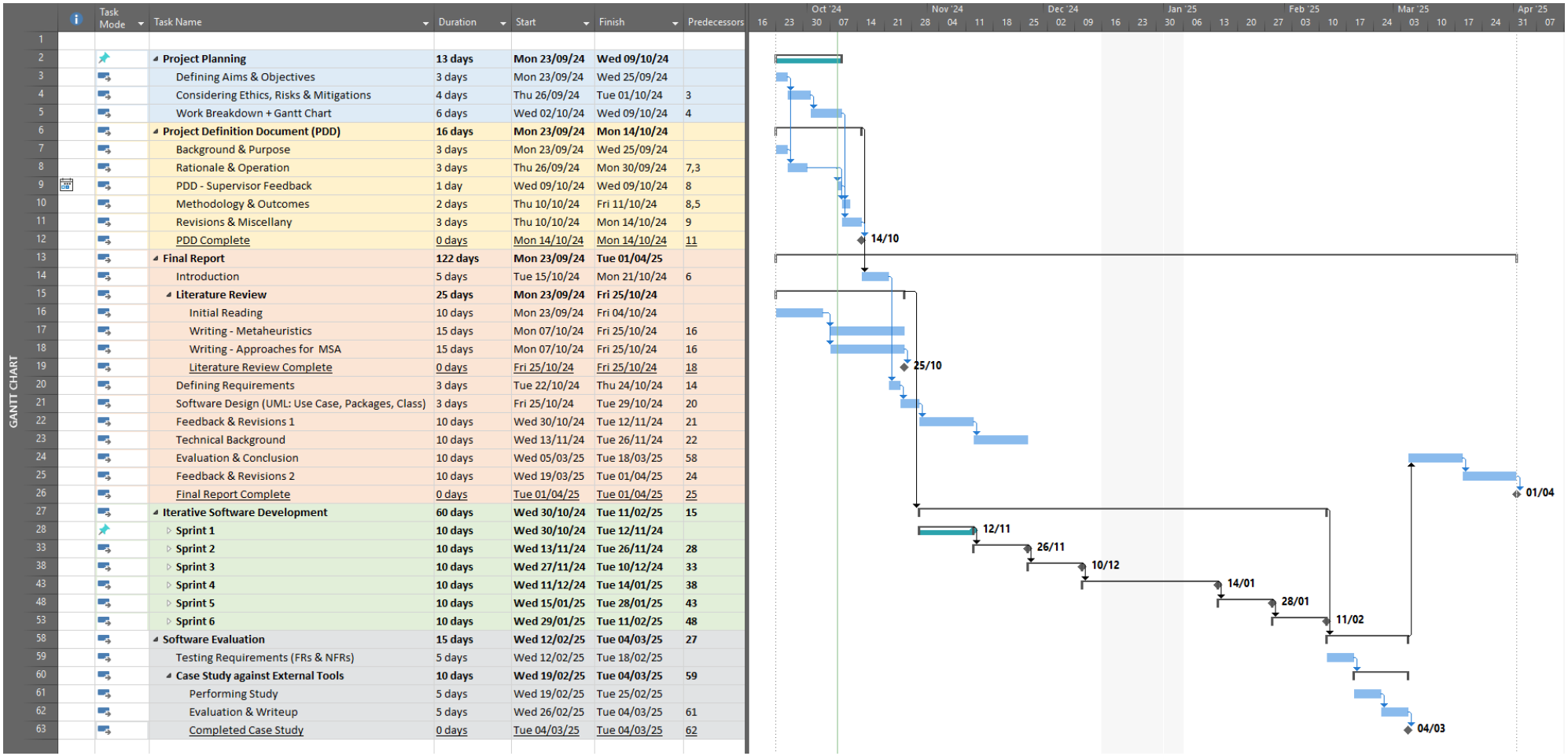
Category	ID	Task/Milestone Details	Priority	Est. Duration
Report	4.3	Ask for feedback on literature review and design sections, then make revisions.	Low	2 weeks
Report	4.4	Write a technical background section on the MSA problem to provide context for readers.	Low	2 weeks
Report	4.5	Evaluate the project as a whole, including comments on development and software testing.	High	2 weeks
Report	4.6	Ask for feedback on technical background and evaluation sections, then make revisions.	Low	2 weeks
Report	M-J	**1st Apr, '25** - The final report has been completed, giving a detailed account of the project.	Milestone	6 months
Soft. Dev.	M-C	**12th Nov, '24** - MAli v0.1 is released - a basic tool that produces low quality, but valid solutions for MSA.	Milestone	2 weeks
Soft. Dev.	M-D	**26th Nov, '24** - MAli v0.2 is released - introducing a metaheuristic algorithm to guide the alignment.	Milestone	2 weeks
Soft. Dev.	M-E	**10th Dec, '24** - MAli v1.0 is released - an improvement on v0.2 and indicative of a full implementation.	Milestone	2 weeks
Soft. Dev.	M-F	**14th Jan, '25** - MAli v1.1 is released - resulting from experimentation on the v1.0 design.	Milestone	2 weeks
Soft. Dev.	M-G	**28th Jan, '25** - MAli v1.2 is released - producing a selection of high-quality solutions as output.	Milestone	2 weeks
Soft. Dev.	M-H	**11th Feb, '25** - MAli v1.3 is released – fulfilling the defined requirements as a capstone of development.	Milestone	2 weeks
Evaluation	5.1	Test the defined functional & non-functional requirements using a university desktop computer.	High	1 week
Evaluation	5.2	Test the tool's performance relative to available alternatives using structural benchmarking.	Medium	1 week
Evaluation	5.3	Analyse the performance and quality of the tools tested in 5.2. Discuss the results.	Medium	1 week
Evaluation	M-I	**4th Mar, '25** - Software requirements have been tested and a case study has been completed.	Milestone	3 weeks

Figure 3.1.1b *– A table of tasks & milestones covering report completion, software releases, and evaluation of the project. (Part 2 of 2)*

A Note on Development Work Items

The development of the software will be undertaken with the agile principles (Beck et al., 2001) in mind. In order to be open to change, a prioritized backlog of work items has been provided in place of definitive tasks (see **Appendix b**). This backlog will likely be revised and expanded over the course of development, to accommodate new priorities identified from research and ultimately produce a more valuable product.

3.1.2. Schedule Gantt chart



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1						
2		**Project Planning**	**13 days**	**Mon 23/09/24**	**Wed 09/10/24**	
3		Defining Aims & Objectives	3 days	Mon 23/09/24	Wed 25/09/24	
4		Considering Ethics, Risks & Mitigations	4 days	Thu 26/09/24	Tue 01/10/24	3
5		Work Breakdown + Gantt Chart	6 days	Wed 02/10/24	Wed 09/10/24	4
6		**Project Definition Document (PDD)**	**16 days**	**Mon 23/09/24**	**Mon 14/10/24**	
7		Background & Purpose	3 days	Mon 23/09/24	Wed 25/09/24	
8		Rationale & Operation	3 days	Thu 26/09/24	Mon 30/09/24	7,3
9		PDD - Supervisor Feedback	1 day	Wed 09/10/24	Wed 09/10/24	8
10		Methodology & Outcomes	2 days	Thu 10/10/24	Fri 11/10/24	8,5
11		Revisions & Miscellany	3 days	Thu 10/10/24	Mon 14/10/24	9
12		PDD Complete	0 days	Mon 14/10/24	Mon 14/10/24	11
13		**Final Report**	**122 days**	**Mon 23/09/24**	**Tue 01/04/25**	
14		Introduction	5 days	Tue 15/10/24	Mon 21/10/24	6
15		**Literature Review**	**25 days**	**Mon 23/09/24**	**Fri 25/10/24**	
16		Initial Reading	10 days	Mon 23/09/24	Fri 04/10/24	
17		Writing - Metaheuristics	15 days	Mon 07/10/24	Fri 25/10/24	16
18		Writing - Approaches for MSA	15 days	Mon 07/10/24	Fri 25/10/24	16
19		Literature Review Complete	0 days	Fri 25/10/24	Fri 25/10/24	18
20		Defining Requirements	3 days	Tue 22/10/24	Thu 24/10/24	14
21		Software Design (UML: Use Case, Packages, Class)	3 days	Fri 25/10/24	Tue 29/10/24	20
22		Feedback & Revisions 1	10 days	Wed 30/10/24	Tue 12/11/24	21
23		Technical Background	10 days	Wed 13/11/24	Tue 26/11/24	22
24		Evaluation & Conclusion	10 days	Wed 05/03/25	Tue 18/03/25	58
25		Feedback & Revisions 2	10 days	Wed 19/03/25	Tue 01/04/25	24
26		Final Report Complete	0 days	Tue 01/04/25	Tue 01/04/25	25
27		**Iterative Software Development**	**60 days**	**Wed 30/10/24**	**Tue 11/02/25**	**15**
28		**Sprint 1**	**10 days**	**Wed 30/10/24**	**Tue 12/11/24**	
33		**Sprint 2**	**10 days**	**Wed 13/11/24**	**Tue 26/11/24**	**28**
38		**Sprint 3**	**10 days**	**Wed 27/11/24**	**Tue 10/12/24**	**33**
43		**Sprint 4**	**10 days**	**Wed 11/12/24**	**Tue 14/01/25**	**38**
48		**Sprint 5**	**10 days**	**Wed 15/01/25**	**Tue 28/01/25**	**43**
53		**Sprint 6**	**10 days**	**Wed 29/01/25**	**Tue 11/02/25**	**48**
58		**Software Evaluation**	**15 days**	**Wed 12/02/25**	**Tue 04/03/25**	**27**
59		Testing Requirements (FRs & NFRs)	5 days	Wed 12/02/25	Tue 18/02/25	
60		**Case Study against External Tools**	**10 days**	**Wed 19/02/25**	**Tue 04/03/25**	**59**
61		Performing Study	5 days	Wed 19/02/25	Tue 25/02/25	
62		Evaluation & Writeup	5 days	Wed 26/02/25	Tue 04/03/25	61
63		Completed Case Study	0 days	Tue 04/03/25	Tue 04/03/25	62

Figure 3.1.2 *– Gantt chart illustrating the planned schedule for the project. Individual development sprints have been collapsed for presentability.*

3.2. Project control

To manage the project effectively, upcoming tasks from the project schedule will be added to an online task board such that their progress can be monitored. Progress on tasks, especially deliverables and milestones will be discussed with the project supervisor, with reference to the project schedule. This will give opportunity to make adaptations if necessary.

Throughout the project, new development work items will be identified and added to the prioritized backlog to be assigned to sprints. For each sprint, a short document will be included in the final report including the sprint goal, development changes and a reflection on progress. As described in **Section 3.3** below, the performance of the software will be assessed as part of each release and in a comparative case study once development has been completed.

A successful project would see the development of a performant alignment tool, fulfilling all functional and non-functional requirements, along with an evidenced account of its performance.

3.3. Project evaluation

For transparency and reproducibility, the methodology of all experiments, release testing and studies will be clearly documented to support their reproduction. This will include details such as initial settings, release versioning, and references to specific test cases where possible.

As stated in **Section 1.6**, the quality of solutions produced by the developed software will be assessed using structural benchmarking. This method of assessment compares alignments produced by MSA software against high-quality references constructed by experts (Thompson et al., 1999).

The solution quality and time-efficiency of the software will be assessed as part of each software release, rotating through partitions of a benchmarking dataset. Once development has concluded, the developer will assess the software (using a university desktop computer) on its fulfilment of the functional and non-functional requirements captured in Task 4.1.

Finally, the tool will be analysed in a structural benchmarking case study alongside established alternatives such as MAFFT, Muscle and Clustal Omega (Pais et al., 2014).

This study will be used to frame the performance of the software in the context of

Following this, the overall success of the project can be evaluated with reference to the original aims and objectives.

4. References

Bahr A., Thompson J.D., Thierry J.-C. & Poch O. (2001) BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Research*, 29(323–326). https://doi.org/10.1093/nar/29.1.323

Beck K., Beedle M., van Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R.C., Mellor S., Schwaber K., Sutherland J. & Thomas D. (2001) *Principles behind the Agile Manifesto.* https://agilemanifesto.org/principles.html [Accessed 11 Oct 24].

Biosynthesis (n.d.) *Bioinformatics Glossary.* https://www.biosyn.com/bioinformatics.aspx [Accessed 11 Oct 24].

Calvet L., Benito S., Juan A.A. & Prados F. (2022) On the role of metaheuristic optimization in bioinformatics. *International Transactions in Operational Research*, 30(6). https://doi.org/10.1111/itor.13164

Data Protection Act (2018) Chapter 12. https://www.legislation.gov.uk/ukpga/2018/12/contents [Accessed 11 Oct 24].

EMBOSS (n.d.) *Sequence Formats*. https://emboss.sourceforge.net/docs/themes/SequenceFormats.html [Accessed 11 Oct 24].

Ibrahim M.K., Yusof U.K., Eisa T.A.E. & Nasser M. (2024) Bioinspired Algorithms for Multiple Sequence Alignment: A Systematic Review and Roadmap. *Applied Sciences*, 14(6). https://doi.org/10.3390/app14062433

Larkin M.A., Blackshields G., Brown N.P., Chenna R., McGettigan P.A., McWilliam H., Valentin F., Wallace I.M., Wilm A., Lopez R., Thompson J.D., Gibson T.J. & Higgins D.G. (2007) Clustal W and Clustal X version 2.0, *Bioinformatics*, 23(2947–2948). https://doi.org/10.1093/bioinformatics/btm404

Pais F.S., Ruy P.C., Oliveira G. & Coimbra R.S. (2014) Assessing the efficiency of multiple sequence alignment programs. *Algorithms for Molecular Biology,* 9(4). https://doi.org/10.1186/1748-7188-9-4

Schwaber, K. & Sutherland, J. (2020) *The Scrum Guide*. https://scrumguides.org/scrum-guide.html [Accessed 11 Oct 24].

Sievers F., Wilm A., Dineen D., Gibson T.J., Karplus K., Li W., Lopez R., McWilliam H., Remmert M., Söding J., Thompson J.D. & Higgins D.G. (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7:539. https://doi.org/10.1038%2Fmsb.2011.75

Thompson J.D., Plewniak F., Poch O. (1999) BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(87–88). https://doi.org/10.1093/bioinformatics/15.1.87

5. Appendix a

Multiple Sequence Alignment – Glossary

Alignment – A matrix of ASCII characters resulting from the comparison of biological sequences to highlight regions of similarity that may represent evolutionary relationships. (Biosynthesis, n.d.).

BAliBASE – A collection of structural reference alignments of protein sequences, for use in structural benchmarking to evaluate MSA software (Thompson et al., 1999).

Biological Sequence – A sequence of residues (nucleotide bases or amino acids) that make up a molecule – generally a protein, DNA or RNA (Calvet et al., 2022).

Multiple Sequence Alignment (MSA) – The process of aligning a set of biological sequences to best highlight potential evolutionary relationships between the sequences. (Biosynthesis, n.d.).

Substitution Matrix – A matrix used to score individual pairs of residues within a column of an alignment. These matrices are a source of domain knowledge, having been constructed based on models of protein evolution, and indicate how likely a pair of residues are to have once shared a position in a common ancestor (Biosynthesis, n.d.).

Structural Benchmarking – The evaluation of an alignment software tool, by comparing alignments produced by the tool against high-quality structural reference alignments of the same sequences (Thompson et al., 1999).

Structural Reference Alignment – An alignment of protein sequences constructed by experts in the field based on the known 3D structure of the molecules, from which certain evolutionary relationships can be inferred (Thompson et al., 1999).

6. Appendix b

Development Backlog

ID	Software Functionality	Priority	Relative Cost
Dev-01	Can load a set of biological sequences from an appropriate bioinformatics file format.	Medium	3
Dev-02	Given sequences to align, produces a valid solution - independent of quality.	High	8
Dev-03	Can output aligned sets of sequences using an appropriate bioinformatics file format.	Medium	3
Dev-04	Employs a metaheuristic algorithm (such as Genetic Algorithm) to guide the alignment process.	Medium	13
Dev-05	Makes use of an established substitution matrix in evaluating alignment states during optimization.	Low	8
Dev-06	Demonstrates MSA using a single-state metaheuristic algorithm - such as Simulated Annealing.	Medium	13
Dev-07	Leverages multiple objective functions to guide the alignment optimization process.	Low	13
Dev-08	Approximates the Pareto Front, outputting a set of solutions that offer different trade-offs.	Low	21
Dev-09	Aligns sets of 6 typical protein sequences within 10 seconds on a university machine.	Low	8
Dev-10	Consistently produces higher quality solutions than random selection of a candidate.	High	3
Dev-11	Consistently produces higher quality solutions than a greedy heuristic.	Medium	5
Dev-12	Produces alignments of a quality comparable to currently available alternatives.	Medium	13
Dev-13	Performs alignment with a time-efficiency comparable to currently available alternatives.	Low	21
Dev-14	Interface displays progress on the current alignment task - in terms of time or iterations.	Low	2
Dev-15	Supports batch alignment of a series of sets of sequences from a directory.	Low	3
Dev-16	Employs a heuristic to estimate a number of iterations needed to align each set of sequences.	Medium	5

Figure 6.0 *– Table of software functionalities that make up the current development backlog – to be assigned to sprints as work items.*