BiMuSA: An implementation for biobjective multiple sequence alignment problems

CISUC Technical Report TR2013/03

Sebastian Schenker[1], Luís Paquete[2]

[1]Zuse Institute Berlin, Germany.
`schenker@zib.de`
[2]CISUC, Department of Informatics Engineering,
University of Coimbra, Portugal.
`paquete@dei.uc.pt`

Abstract

This report introduces BiMuSA, an implementation for solving biobjective multiple sequence alignment problems. It outputs a sequence of optimal alignments, in a multicriteria sense, by solving a sequence of integer linear programs. The formulation of the latter is based on a graph representation. This implementation can be used to solve small to medium sized problems with an arbitrary number of strings and to generate upper bounds for larger problems for benchmarking purposes.

1 Introduction

In this work, we introduce BiMuSA, an implementation for solving biobjective multiple sequence alignment problems. The theoretical background and algorithmic approach concerning the multiple sequence alignment problem is based on [4]. In this work, Althaus et al. describe a branch-and-cut framework for solving an extension of the *gapped trace problem* which can be represented in graph-theoretic terms. The latter leads then to an integer linear program which is solved via branch-and-cut or integer programming, respectively.

There have also been several works extending algorithms for pairwise sequence alignment for a multicriteria setting [3, 5, 8–10]; for an extensive review about bioinformatics problems recast as multicriteria optimization problems see also [7].

The biobjective problem considered in this work consists in minimizing the number of indels or gaps, respectively, and in maximizing the number of matches minus the number of mismatches. The latter objective can easily be extended to a more general substitution score objective (by implementing a given virtual edge weight function). The working principle of the implementation is to solve a sequence of integer linear programming (ILP) problems within a branch-and-cut framework. The implementation outputs the Pareto optimal alignments via incorporating an ϵ-constraint method by taking the number of indels or gaps, respectively, as an additional constraint into the LP formulation. By iterating properly over these indel/gap constraints, it is possible to generate the complete Pareto optimal alignment set. This technique is known as ϵ-constraint in the multicriteria optimization field [6].

2 Branch-and-cut framework

The solution technique to solve the ILP problem is based on the branch-and-cut work of [4]. Each symbol in every sequence is a vertex and vertices are connected by edges and arcs. A selection of these edges and arcs, respectively, forms an alignment. In order to ensure a feasible alignment the subset of edges and arcs needs to fulfill several constraints. These constraints can be written such that the resulting formulation constitutes a linear programming (LP) formulation with, in general, exponentially many constraints. Since, in general, LPs with exponentially many constraints cannot be solved efficiently, the authors tackle this problem by introducing facet-defining inequalities and using them within a branch-and-cut framework. Firstly, the original inequalities are replaced by more general inequalities that are shown to be facet-defining. Moreover, for these facet-defining inequalities efficient separation procedures are proposed that yield the branch-and-cut approach.

The first separation approach is concerned with so-called maximal clique inequalities. Maximal clique inequalities consider pairwise incompatible edges and arcs that are not allowed to be chosen simultaneously in order to achieve a feasible alignment. The structure of maximal clique inequalities can be exploited or separeted, respectively, with the help of a longest path computation in a pairgraph. The second separation approach is concerned with lifted mixed cycle inequalities. A violation of these inequalities also yields an infeasible alignment because a mixed cycle means that a certain symbol of one of the sequences is ambiguously aligned to symbols of another sequence. An efficient separation of lifted mixed cycle inequalities can be achieved by shortest path computations in a directed graph with distances that can be computed by path computations in pairgraphs computed during the maximal clique separation. The last separation approach is concerned with general transitivity inequalities. In order to achieve a feasible alignment the variables in the LP formulation need to fulfill so-called transitivity constraints. These constraints are separated by a maximum flow computations in an extended bipartite graph.

3 Implementation details

BiMuSa is implemented in C++. Each separation procedure is developed as an independent class. For the computations within each separation procedure we use the (freely available) LEMON graph library [2]. For the optimization of the (integer) linear program we use cplex and its concert technology [1].

After reading the sequences from a given input file, the basic data structures (edges and arcs of the underlying graph) are created via the graphMsa class. The (integer) linear program instance is established in the ilpMsa class and incorporates in the first iteration only a basic set of assignment constraints. These assignment equalities constrain each node of the graph, i.e., each letter in each string, to be either assigned to another node or to be spanned by an indel or gap, respectively, with respect to all other strings. For further details of the underlying integer model and the separation approaches see [4].

Each solution of the relaxed integer program is checked for feasibility by checking whether it leads to a violated maximal clique inequality, a violated mixed cycle inequality or a violated general transitivity inequality.

The separation of maximal cliques is done in the maxClique class. For each given pair of strings it checks whether the sum of pairwise incompatible arcs and pairwise incompatible edges exceeds its limit. The latter value is computed by longest path computations in a constructed pairgraph. If we find a violated maximal clique, then the corresponding inequality is incorporated into the linear program and the extended linear program is solved and rechecked.

If we cannot find a violated maximal clique constraint anymore, we check for violated lifted mixed cycle constraints. This done via the mixedCycle class. This separation involves shortest path computations in a corresponding directed graph whose weights stem from maximal sets of pairwise incompatible edges (and can be computed in the previous maximal clique separation). As long as there are no violated maximal clique inequalities we check for violated lifted mixed cycles and incorporate the latter into the linear program.

The separation of generalized transitivity inequalities is done after the separation of maximal cliques and mixed cycles. Violated general transitivity inequalities are computed within the generalTrans class and involve the computation of maximal flows or minimal cuts, respectively.

After finishing the separation process, we check whether the current lp solution for integrality. If the lp solution is integer, we are done. If the solution is fractional, we transform the linear program into a integer program and achieve feasibility by solving the latter.

The implementation can also be used to compute upper bounds by bounding the number of iterations with respect to the separations. This may be relevant for benchmarking purposes, i.e. assessing the performance of heuristic methods.

Acknowledgements

This work was support by the Fundação para a Ciência e Tecnologia, project MOSAL - Multiobjective sequence alignment (PTDC/EIA-CCO/098674/2008) and by FEDER, Programa Operacional Factores de Competitividade do QREN, ref. COMPETE: FCOMP-01-0124-FEDER-010024.

References

[1] Cplex optimization studio. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer`.

[2] Lemon graph library. `http://lemon.cs.elte.hu/trac/lemon`.

[3] M. Abbasi, L. Paquete, A. Liefooghe, M. Pinheiro, and P. Matias. Improvements on bicriteria pairwise sequence alignment: algorithms and applications. *Bioinformatics*, 29(8):996–1003, 2013.

[4] E. Althaus, A. Caprara, H.-P. Lenhof, and K. Reinert. A branch-and-cut algorithm for multiple sequence alignment. *Mathematical Programming, Ser. B*, (105):387–425, 2006.

[5] K.W. DeRonne and G. Karypis. Pareto optimal pairwise sequence alignment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(2):481–493, 2013.

[6] M. Ehrgott. *Multicriteria optimization.* Springer, 2005.

[7] Julia Handl, Douglas B. Kell, and Joshua D. Knowles. Multiobjective optimization in bioinformatics and computational biology. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):279–292, 2007.

[8] M.A. Roytberg, M.N. Semionenkov, and O.I. Tabolina. Pareto-optimal alignment of biological sequences. *Biophysics*, 44(4):565–577, 1999.

[9] T. Schnattinger, U. Schöning, and H. Kestler. Structural rna alignment by multi-objective optimization. *Bioinformatics*, 29(13):1607–1613, 2013.

[10] Akito Taneda. Multi-objective pairwise RNA sequence alignment. *Bioinformatics*, 26(19):2383–2390, 2010.