# Simple Approximation Algorithm for Nonoverlapping Local Alignments

Piotr Berman* Bhaskar DasGupta† S. Muthukrishnan‡

**Problem.** We study the following maximization problem called the *Independent subset of Rectangles* (IR) problem. We are a given a set $S$ of $N$ positively weighted axis parallel rectangles such that, for each axis, the projection of a rectangle on this axis does not enclose that of another. Define two rectangles to be *independent* if for each axis, the projection of one rectangle does not overlap that of another. The goal of the IR problem is to select a subset $S' \subseteq S$ of independent rectangles of total maximum weight.

**Motivation.** A fundamental problem in Computational Molecular Biology is to elucidate similarities between sequences (See [4] for detailed case). A cornerstone result in this area is that given two strings of length $n$ and $m$, there are *local alignment algorithms* that will score pairs of substrings for "similarity" according to various biologically meaningful scoring functions and we can pull out all "similar" or high scoring substring pairs [6]. Having found the high scoring substring pairs, a global description of the similarity between two sequences is obtained by choosing the disjoint subset of these pairs of highest total score. This problem was formulated as the IR problem by Bafna et al. [1] with each output substring pair being represented as a rectangle.[1]

**Previous Results.** Bafna et al. [1] proved this problem to be NP-Hard and presented an $\Omega(N^2)$ time approximation algorithm with a performance ratio of 3.25.[2] The current best approximation algorithm for the IR problem is due to Berman [2] which has a performance ratio of $2.5+\varepsilon$ (for any constant $\varepsilon > 0$), and takes $\Omega(N^4)$ running time. Construct the conflict graph of the input rectangles in which each node is a rectangle, and there is an edge between two rectangles if and only if they are independent. This graph is 5-claw free. IR problem is equivalent to finding the maximum weight independent set (MWIS) of this graph; known results provide greedy improvement based algorithms for finding MWIS.

**Our Result and Significance.** The basic bottleneck of known approximation algorithms for the IR problem is that they are computationally expensive taking time $\Omega(N^2)$. Our main result here is an approximation algorithm for the IR problem that has running time nearly linear in the input size. Our algorithm takes $O(N \log N)$ time and space, and is a factor 3 approximation. It is also exceedingly simple, requiring Phase 1 of a single pass over the data to push a subset of rectangles into a stack and Phase 2 of popping the stack and adding a selection of rectangles into our solution. This algorithm thus uses an approach reminiscent of real-time machine scheduling algorithms [3], and is quite different from the known approaches for solving this problem. The difficult part of the result is the analysis to prove that this algorithm is indeed 3-approximation. Thus, while this algorithm does not produce the best approximation known for this problem, it is simpler, substantially faster, and hence, likely to be more practical.

**Algorithm Description.** Let $R_1, R_2, \ldots, R_N$ be the $N$ input rectangles in our collection, where $R_i = X_i \times Y_i$ for some two intervals $X_i = [d_i, e_i]$ and $Y_i = [f_i, g_i]$. Consider the intervals $X_1, X_2, \ldots, X_N$ formed by projecting the rectangles on one axis and call two intervals $X_i$ and $X_j$ independent if and only if the corresponding rectangles $R_i$ and $R_j$ are independent. The notation $X_i \simeq X_j$ (respectively, $X_i \not\simeq X_j$) is used to denote if two intervals $X_i$ and $X_j$ are independent (respectively, not independent). We can assume the endpoints of the rectangles are numbered using integers in $[1, 2N]$, without loss of generality. Our algorithm consists of the evaluation phase followed by the selection phase, described below. We define a *triplet* $(\alpha, \beta, \gamma)$ to be an ordered sequence of three values $\alpha$, $\beta$ and $\gamma$. Let $\mathbf{L}$ is sequence that contains a triplet $(w(R_i), d_i, e_i)$ for every $R_i = X_i \times Y_i$ with $X_i = [d_i, e_i]$; $\mathbf{L}$ is sorted so the values of $e_i$'s are in non-decreasing order. Let $\mathbf{S}$ is an initially empty stack that stores triplets. TOTAL$(X_j)$ returns the sum of $v$'s of those triplets

*Department of Computer Science, Pennsylvania State University, University Park, PA 16802. Email: berman@cse.psu.edu. Supported in part by NSF grant CCR-9700053, NLM grant LM05110 and DFG grant Bo 56/157-1.

†Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607. Email: dasgupta@cs.uic.edu. Supported in part by NSF Grant CCR-9800086 and a startup fund from UIC.

‡AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932. Email: muthu@research.att.com

[1] For alternate formulation, see the chaining problem [4, Page 326].

[2] If rectangles are unweighted, a polynomial time approximation algorithm with a performance ratio of $2+\varepsilon$ (for any constant $\varepsilon > 0$) is known. [5].

$(v, a, b) \in \mathbf{S}$such that $[a, b] \not\simeq X_j$. The evaluation phase consists of processing each item from $\mathbf{L}$ in turn. Say $(w(R_i), d_i, e_i)$ is being processed. We evaluate $v \leftarrow w(R_i) - \text{TOTAL}([d_i, e_i])$; if $v > 0$, we push $((v, d_i, e_i), \mathbf{S})$ onto the stack and continue. When $\mathbf{L}$ is empty, we being the selection phase. We pop $\mathbf{S}$ until it is empty. Say $(v, d_i, e_i)$ has been popped. If $[d_i, e_i] \simeq X$ for every interval $X$ in our solution, we add $[d_i, e_i]$ to our solution. That completes the algorithm description. The full version of this paper will show how to implement this algorithm in $O(N \log N)$ time.

**Analysis of the Algorithm.** Let $B$ be a solution returned by the algorithm $A$ be any optimal solution. For a rectangle $R \in A$, let $\beta_R$ denote the number of those rectangles in $B$ that were *not* independent of $R$ *and* were examined no earlier than $R$ by the evaluation phase and let $\beta = \max_{R \in A} \beta_R$. We first show that our algorithm has performance ratio $\beta$. Consider the set of intervals $\mathbf{S}$ in the stack at the end of the evaluation phase. Let $W(A) = \sum_{R_i \in A} w(R_i)$ and $V(\mathbf{S}) = \sum_{(v, d_i, e_i) \in \mathbf{S}} v$. The sum of the weights of the rectangles selected during the selection phase is at least $V(\mathbf{S})$. Hence, it suffices to show that $\beta V(\mathbf{S}) \geq W(A)$. Consider a rectangle $R_i = X_i \times Y_i \in A$ and the time when the evaluation phase starts the processing of $X_i = [d_i, e_i]$. Let $\text{TOTAL}'([d_i, e_i])$ and $\text{TOTAL}''([d_i, e_i])$ be the values of $\text{TOTAL}([d_i, e_i])$ before and after the processing of $X_i$, respectively. If $w(R_i) < \text{TOTAL}'([d_i, e_i])$, then $X_i$ is not pushed to the stack and $\text{TOTAL}''([d_i, e_i]) = \text{TOTAL}'([d_i, e_i])$. On the other hand, if $w(R_i) \geq \text{TOTAL}'([d_i, e_i])$, then $X_i$ is pushed to the stack with a value of $w(R_i) - \text{TOTAL}'([d_i, e_i])$, as a result of which $\text{TOTAL}''([d_i, e_i])$ becomes at least $\text{TOTAL}'([d_i, e_i]) + (w(R_i) - \text{TOTAL}'([d_i, e_i])) = w(R_i)$. Hence, in either case $\text{TOTAL}''([d_i, e_i]) \geq w(R_i)$. Summing up over all $R_i$'s,

$$W(A) = \sum_{R_i \in A} w(R_i) \leq \sum_{R_i = [d_i, e_i] \times Y_i \in A} \text{TOTAL}''([d_i, e_i])$$

By definition of $\beta$ and $\text{TOTAL}''([d_i, e_i])$,

$$W(A) \leq \sum_{((v,a,b) \in \mathbf{S}) \wedge (b \leq e_i) \wedge ([a,b] \not\simeq [d_i, e_i]))} v \leq \beta \sum_{(v,a,b) \in \mathbf{S}} v$$

which results in $W(A) \leq \beta V(\mathbf{S})$ completing the proof. We can now prove that $\beta \leq 3$, completing the analysis (omitted here).

**Concluding Remarks.** We have designed a fast, exceedingly simple two-phase algorithm for the IR problem: a stack and an augmented balanced search tree suffice to implement this algorithm. Additionally, this algorithm *incremental*, that is, it can be directly embedded into the dynamic programming for the local alignments problem; thus, intermediate rectangles that will not be stored on the stack need not to be maintained. We expect our algorithm to be practical. In an attempt to improve the approximation, we can run our algorithm two times each on the projections of rectangles on the $x$ and $y$ axes in left-to-right/right-to-left and top-to-bottom/bottom-to-top order to take the best of the four solutions. We can show by an example that this does not improve our approximation ratio. We have used the planar geometry induced by the rectangles for the IR problem to obtain a 3-approximation. Can we exploit the geometry of rectangles more to design simple approximation algorithms with performance ratios 2.5 or better? From the computational biology viewpoint, it is highly desirable to perform similarity analysis on multiple strings, and hence, $d$-dimensional version of this problem is of interest too. Our algorithm gives $2^d - 1$ approximation in a straightforward way. Whether one can design an algorithm with a performance ratio that increases less drastically (e.g., linearly) with $d$ is still open.

## References


[1] V. Bafna, B. Narayanan and R. Ravi. *Nonoverlapping local alignments (Weighted independent sets of axis-parallel rectangles*, Discrete Applied Mathematics, 71, pp. 41-53, 1996.

[2] P. Berman. *A $d/2$ approximation for maximum weight independent set in $d$-claw free graphs*, Proc. of the 7th Scandinavian Workshop on Algorithmic Theory, LNCS 1851, July 2000, pp. 214-219.

[3] P. Berman and B. DasGupta. *Improvements in Throughput Maximization for Real-Time Scheduling*, proceedings of the 32nd Annual ACM Symposium on Theory of Computing, May 2000, pp. 680-687.

[4] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge Univ Press, 1997.

[5] M. M. Halldórsson. *Approximating discrete collections via local improvements*, proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms, January 1995, pp. 160-169.

[6] M. S. Waterman and M. Eggert, *A new algorithm for best subsequence alignments with application to trna-rrna comparisons*, J. Mol. Biol. 197, 1987, pp. 723-728.