A Divide and Conquer Approach to Multiple Alignment

Andreas Dress, Georg Füllen and Sören Perrey

Research Center for Studies on Structure Formation (RCSF)
University of Bielefeld, Postfach 10 01 31
D-33501 Bielefeld, Germany
e-mail: jordan,perrey@mathematik.uni-bielefeld.de

Abstract

We present a report on work in progress on a *divide and conquer* approach to multiple alignment[1].

The algorithm makes use of the costs calculated from applying the standard dynamic programming scheme to all pairs of sequences. The resulting cost matrices for pairwise alignment give rise to *secondary* matrices containing the additional costs imposed by fixing the path through the dynamic programming graph at a particular vertex. Such a constraint corresponds to a division of the problem obtained by slicing both sequences between two particular positions, and aligning the two sequences on the left and the two sequences on the right, charging for gaps introduced at the slicing point.

To obtain an estimate for the additional cost imposed by forcing the multiple alignment through a particular vertex in the whole hypercube, we will take a (weighted) sum of secondary costs over all pairwise projections of the division of the problem, as defined by this vertex, that is, by slicing all sequences at the points suggested by the vertex.

We then use that partition of every single sequence under consideration into two 'halfs' which imposes a minimal (weighted) sum of pairwise additional costs, making sure that one of the sequences is divided somewhere close to its midpoint. Hence, each iteration can cut the problem size in half. As the enumeration of *all* possible partitions may restrict this approach to small-size problems, we eliminate futile partitions, and organize their enumeration in a way that starts with the most promising ones.

Comparing our approach for the case of 3 sequences with a) structurally verified alignments and b) alignments from literature, indicates high quality alignments, with roughly the same num-ber of errors as the "optimal" (in the dynamic programming framework) solution in a), and being as close as the "optimal" to a maximum weight trace done by Kececioglu, using 6 sequences altogether [6].

Introduction.

Multiple sequence alignment is presently a very active area of research, with an abundance of papers and software contributions [3].

For large-size problems, all non-heuristic approaches suffer from high computational costs, while heuristics are often not yielding the desired results, so that the need for good *and* fast heuristics arises naturally. Pairwise alignments (so-called 'projections') for cutting down the number of dynamic programming steps have been used already, e.g. by Altschul et al. (see [2] and [9] in particular), but – according to our knowledge – using pairwise alignments for the calculation of secondary (additional) costs resulting from imposing specific vertices to be used by the path through the dynamic programming matrix is a novel approach, which we believe to have strong advantages.

Description of the Algorithm

For sequences $s_0, s_1, \ldots, s_n$ of length $|s_0|, |s_1|, \ldots, |s_n|$, respectively, and a cost function c defined for pairwise alignments, the main loop of our heuristic multiple alignment algorithm is the search for tuples $(i_0, i_1, \ldots, i_n)$ which minimize the additional cost $C(i_0, i_1, \ldots, i_n)$ imposed by forcing the alignment path through the vertex $(i_0, i_1, \ldots, i_n)$ and defined by

$$C(i_0, i_1, \ldots, i_n) := \sum_{0 \leq p < q \leq n} C_{p,q}[i_p, i_q] \alpha_{pq},$$

where

$$
\begin{aligned}
C_{p,q}[i_p, i_q] := \ & c(s_p(\leq i_p), s_q(\leq i_q)) \\
& + c(s_p(> i_p), s_q(> i_q)) \\
& - c(s_p, s_q))
\end{aligned}
$$

are the additional costs imposed on the pairwise alignment of sequences s_p and s_q and $\alpha_{pq} \geq 0$ are appropriately chosen weight factors reflecting e.g. phylogenetic relationships. Here, the notation $s_p(\leq i_p)$ denotes the subsequence of s_p with indices running from 0 to i_p and $s_p(> i_p)$ denotes the subsequence starting at site $i_p + 1$.

The result of the minimum search is called the "minimal" tuple. Given such a tuple, the algorithm then recursively solves the 2 subproblems of aligning $\big(s_k(\leq i_k)\big)_{k=0,\ldots,n}$ and $\big(s_k(> i_k)\big)_{k=0,\ldots,n}$.

Several techniques can be employed to cut down the number of tuples considered:

1. The index i_0 is fixed to be $\lceil |s_0|/2 \rceil$. This is problematic only if a particular part of the *a priori* optimal alignment has gaps on both sides of the index i_0 in the sequence s_0, because any gap insertion below i_0 will then be charged twice (i.e. in both subproblems). Readjusting the final alignment in the proximity of gaps should at least reduce the problem.

2. While searching for the "minimum" tuple, if some part of the sum $\sum_{0 \leq p < q \leq n} C_{p,q}[i_p, i_q]\alpha_{pq}$, e.g. $C_{0,1}[r, s]\alpha_{pq}$ is larger than the minimum found so far, no tuples $(i_0, i_1, i_2, \ldots, i_n)$ with corresponding indices (e.g. $(r, s, i_2, \ldots, i_n)$) will be considered.

3. The exploration of "tuple space" $\{(i_0, \ldots, i_n) \mid i_p \in \{0, \ldots, |s_p|\}\}$ starts in the middle of the sequences, and gradually traverses back and forth towards the endpoints. This way, it is very easy to cut the exploration process at any time, and be confident that the most promising tuples have been explored, if the *a priori* alignment is "regular", i.e. does not contain too many unequally distributed gaps. We can expect that we obtain inferior results if the sequences are of very different length, or are not homologous.

Table 1 displays the Algorithm Flowchart for the case $n = 3$, and $\alpha_{pq} := 1$.

Implementation Notes

1. The classic dynamic programming similarity matrices are obtained using the *Toplign* package developed by [7]. We use the Dayhoff PAM250 matrix to calculate the amino acid substitution costs, and we charge 13 units for a gap-insertion, and 1 unit for any gap extension.

2. If all of the sequences in a subproblem have length less than *minLength* (typically 3), the recursion is stopped, and an optimal alignment algorithm by Huang [5] is called in a black-box manner. This is merely a convenience, since each call to the *Toplign* package currently involves re-reading the (Dayhoff) matrix via the file system.

First Results

Our alignment algorithm has been subjected to a few test cases, with all weight factors being put to 1. On average, our algorithm needed to explore ca. 1% of all possibilities to find the best second and third index (whereas the first index is always fixed to one half the length of the first sequence). The maximum deviation from the starting point in the middle of the sequence was ca. 15 % during the first step, and ca. 30 % during the second step. As expected, the longer the subsequence, the smaller the percentage of the tuple space that needs to be explored. Of course, to evaluate our method in general, one has to investigate many more examples.

Fig. 1 gives a comparison between the alignment of three cytochromes, cytochrome c_2 from *Rhodospirillum rubrum*, cytochrome c_2 from *Rhodobacter capsulatus*, and cytochrome c from *Oryza sativa l.*

The alignment obtained by our method is labeled "Divide and Conquer", and the structurally verified alignment derived from Lessel and Schomburg (1994) [8] is labeled "Structurally verified". Also, the MSA optimal multiple alignment is given, labeled "MSA" [9]. Finally, the pairwise alignments obtained by using the standard dynamic programming calculation employed by *Toplign* are labeled "Dynamic programming pairwise alignments" (see Implementation Notes, 1). Only the residues marked by "*" are indeed part of the homology. All computed alignments are compared to these homologous residues. Deviations from this standard-of-truth are marked by "!". Finally, the "Divide and Conquer"-alignment is preceded by a line "X:" indicating depth and location of the recursive division. For example, the division of the two subproblems during the second step takes place on the right of the positions marked by arrows in Figure 1.

The number of "mistakes" made by our approach is of the same order as that of the "optimal" alignment. They may have resulted from incorrect similarity assessment caused by the PAM250 matrix (i.e. incorrect with respect to the structurally verified alignment), or they may be due to a misdirection by the secondary costs at a fairly high recursion level, i.e. during the treatment of the subproblem to the right at recursion level 1.

Fig. 2 features the alignment of three blue copper proteins, viz. pseudoazurin from *Alcaligenes faecalis*, plastocyanin green alga (*Enterom. prolifera*), and cucumber basic protein. As for Fig. 1, the structurally verified alignment is derived from [8].

In this case, our algorithm makes almost the same errors as the MSA multiple alignment obtained directly by dynamic programming. Shifts by one single residue (which we do not consider a mistake because the process of structural alignment typically yields results which may deviate by one position) are marked by "+". Again, misalignments made by our method

Table 1. Algorithm Flowchart for the case $n = 3$, and $\alpha_{pq} := 1$.

Alisub(a_0, a_1, a_2)
If any sequence length is smaller than a fixed constant $minLength$,
then $(\bar{a}_0, \bar{a}_1, \bar{a}_2) :=$ conventional Alignment(a_0, a_1, a_2)
else Calculate secondary cost matrices $C_{p,q}$ using dynamic progr. matrices
$\quad i := \lceil |a_0|/2 \rceil; currMin := \infty$
$\quad$ For $j := \lceil |a_1|/2 \rceil, \lceil |a_1|/2 - 1 \rceil, \lceil |a_1|/2 + 1 \rceil, \ldots,$ do
$\quad\quad$ If $C_{0,1}[i, j] < currMin$
$\quad\quad\quad$ For $k := \lceil |a_2|/2 \rceil, \lceil |a_2|/2 - 1 \rceil, \lceil |a_2|/2 + 1 \rceil, \ldots,$ do
$\quad\quad\quad\quad$ If $C_{0,1}[i, j] + C_{0,2}[i, k] + C_{1,2}[j, k] < currMin$
$\quad\quad\quad\quad\quad$ Update currMin and save (i, j, k)
$(a'_0, a'_1, a'_2) := \text{Alisub}\Big(a_0(\leq i), a_1(\leq j), a_2(\leq k)\Big)$
$(a''_0, a''_1, a''_2) := \text{Alisub}\Big(a_0(> i), a_1(> j), a_2(> k)\Big)$
$(\bar{a}_0, \bar{a}_1, \bar{a}_2) := (concatenate(a'_0, a''_0), concatenate(a'_1, a''_1), concatenate(a'_2, a''_2))$
Return $(\bar{a}_0, \bar{a}_1, \bar{a}_2)$

can already be found in the pairwise alignments ("Dynamic programming pairwise alignments") upon which our calculations are based.

Fig. 3 compares our approach to results obtained using "Maximum weight trace" [6]. In this case, the trace published by Kececioglu is taken as the "standard of truth"; only the first, third, and last sequence are considered. Using these 3 sequences, our alignment (Fig. 3) makes ca. 70 "mistakes", all in the neighborhood of gaps. However, the "optimal" approach, labeled again "MSA", differs from Kececioglu's result in about as many positions.

Open Problems, and Suggestions for Further Research

The following section details some ideas for further investigation.

Many more examples need to be investigated.

A windowing approach may be used to correct the obtained alignments in the proximity of gaps and/or division points.

In case a heuristic is applied to reduce the number of tuples to be investigated, the influence of the input order of the sequences needs to be investigated.

The pairwise projections of our alignment should be compared to the pairwise alignments provided by "Toplign", since the former alignments are based indirectly on the "Toplign" pairwise alignment cost matrices. Moreover, substitution matrices differing from PAM250 (see [4]) should be tested.

1   S.F. Altschul, D.J. Lipman. *Trees, Stars and Multiple Biological Sequence Alignment*, SIAM J. Appl. Math. 49, 197-209 (1989)

2   H. Carillo, D. Lipman. *The Multiple Sequence Alignment Problem in Biology*, SIAM J. Appl. Math. 48, 1073-1082 (1988)

3   S.C. Chan, A.K.C. Wong, D.K.Y. Chiu. *A Survey of Multiple Sequence Comparison Methods*, Bull. Math. Biol. 54, 563-598 (1992)

4   A. Dress, S. Perrey. *Statistics for Fragment Comparison – A Biologically Motivated Approach to Sequence Alignment*, manuscript, submitted for ISMB 1995

5   Xiaoqiu Huang. *Alignment of Three Sequences in Quadratic Space*, Applied Computing Review, 1(2): 7-11 (1993)

6   J.D. Kececioglu. *The maximum weight trace problem in multiple sequence alignment*. In Proc. of the 4th Symp. on Comb. Pattern Matching, LNCS 684, pp. 106-119 (1993)

7   Th. Lengauer et al. "Toplign" Software Package (1994)

8   U. Lessel, D. Schomburg. *Similarities between 3D Structures*. Protein Engineering, Okt/Nov 1994

9   D.J. Lipman, S.F. Altschul, J.D. Kececioglu. *A tool for multiple sequence alignment*. Proc. Natl. Acad. Sci. USA, 86: 4412-4415 (1989)

10 S.B. Needleman, C.D. Wunsch. *A General Method Applicable to the Search for Similarities in the Amino-Acid Sequence of Two Proteins*, J. Molec. Biol. 48, 443-453 (1970)

Figure 1.

Divide and Conquer:

```
X: --4--3--4--5--2--4--5------3--4--5--1--4--5--3--4--5--2--4--5--3--4--5--0--4---4--3--4--5--2--4--5--3--4--5--2--4--5--3--4---1---4--3--4--5--2--4--5--3--4--5--
GDAAAGEKVSK-KCLACHT---F---DQGGANKVGPNLFGVFENTAAHKDNYAYSESYTEMKAKGLTWTEANLAAYVKNPKAFVLEKSGDPKAKSKMTFK-LTKDDEIENVIAYL-KTLK
GDAAKGEKEFN-KCKTCHSIIAPDGTEIVKGAKTGPNLYGVVGRTAGTYPEFKYKDSIVALGASGFAWTEEDIATYVKDPGAFLKEKLDDKKAKTGMAFK---LAKGGEDVAAYL-ASVV
GNPKAGEKIFKTKCAQCHT---V----DKGAGHKQGPNLFGRQSGTTPGYSYSTADKNMAV----IWEENTLYDYLLNPKKYIPGT-------KMVFPGLKKPQERADLISYLKEATS
*********** ********||    ****  ****************************** ***   *********** ****   ||||  *********  *
```

Structurally verified:

```
GDAAAGEKVSK-KCLACHTFDQGGAN------KVGPNLFGVFENTAAHKDNYAYSESYTEMKAKGLTWTEANLAAYVKNPKAFVLEKSGD-PKAKSKMTF-KLTKDDEIENVIAYLKTL-K
GDAAKGEKEFN-KCKTCHSIIAPDGTEIVKGAKTGPNLYGVVGRTAGTYPEFKYKDSIVALGASGFAWTEEDIATYVKDPGAFLKEKLDDKKA-KTGMAF-KLAK--GGEDVAAYLASV-V
GNPKAGEKIFKTKCAQCHTVDKGAGH------KQGPNLFGRQSGTTPGYSYSTADKNMAV----IWEENTLYDYLLNPKKYIP----------GTKMVFPGLKKPQERADLISYLKEATS
*********** ********              ****************************** ***   ***********    ****        ||   *********
```

MSA:

```
GDAAAGEKVSK-KCLACHTF------DQGGANKVGPNLFGVFENTAAHKDNYAYSESYTEMKAKGLTWTEANLAAYVKNPKAFVLEKSGDPKAKSKMTFK-LTKDDEIENVIAYLKTLK-
GDAAKGEKEFN-KCKTCHSIIAPDGTEIVKGAKTGPNLYGVVGRTAGTYPEFKYKDSIVALGASGFAWTEEDIATYVKDPGAFLKEKLDDKKAKTGMAFK-LAKGGE-DVAAYLASVV-
GNPKAGEKIFKTKCAQCHTV------DKGAGHKQGPNLFGRQSGTTPGYSYSTADKNM---AVIWEENTLYDYLLNPKKY------------PGTKMVFPGLKKPQERADLISYLKEATS
*********** ********        ****************************** ***   ***********          ****   || ***  *
```

Dynamic programming pairwise alignments:

```
GDAAAGEKVSKKCLACHTF------DQGGANKVGPNLFGVFENTAAHKDNYAYSESYTEMKAKGLTWTEANLAAYVKNPKAFVLEKSGDPKAKSKMTFKLTKDDEIENVIAYLKTLK
GDAAKGEKEFNRCKTCHSIIAPDGTEIVKGAKTGPNLYGVVGRTAGTYPEFKYKDSIVALGASGFAWTEEDIATYVKDPGAFLKEKLDDKKAKTGMAFKLAKGG--EDVAAYLASVV
*********** *********      ****************************** ***   ***********  ||  *********

GDAAAGEKVSK-KCLACHTFDQGGANKVGPNLFGVFENTAAHKDNYAYSESYTEMKAKGLTWTEANLAAYVKNPKAFVLEKSGDPKAKSKMTFK-LTKDDEIENVIAYL-TLK
GNPKAGEKIFKTKCAQCHTVDKGAGHKQGPNLFGRQSGTTPGYSYS---TADKNMAVIWEENTLYDYLLNPKKYI-------PGTKMVFPGLKKPQERADLISYLKEATS
*********** *********      ||||||||||  ************            |            *********   ||  *

GDAAKGEKEFN-KCKTCHSIIAPDGTEIVKGAKTGPNLYGVVGRTAGTYPEFKYKDSIVALGASGFAWTEEDIATYVKDPGAFLKEKLDDKKAKTGMAF---KLAKGGEDVAAYL-ASVV
GNPKAGEKIFKTKCAQCHTV-----DKGAGHKQGPNLFGRQSGTTPGYSYSTADKNMAV----IWEENTLYDYLLNPKKYI-------PGTKMVFPGLKKPQERADLISYLKEATS
*********** *********      ****************************** ***   ||||  *********
```

Figure 2.

Divide and Conquer:

```
X:  ---3-4--2---3--4--1-4--3--4--2-4--3--4-------------0-4--3--4--2---3--4--2---3--4--1-4--3--4---2-4--3--4---
MLNKGAEGAMVFEPAYIKANPGDTVTFIPVDKGHNVESIK------DMI--PEGAEKFKSKINENYVLTVTQPGAYLVKCT--PHYAMGMIALIAVG
-L-GGDDGSLAFVPNNITVGAGESIEFINNAGFPHNIVFDEDAVPAGVDADAISAEDYLNSKGQTVVRKLTTPGTYGVYCD--PHSGAGMKMTITVQ
-W-TFNTESW---PKGKRFRAGDILLFNYNPTMHNVVVNQGGFSTCNT--PAGAKVYTSGRDQ---IKLPKGQSYFI-CNFPGHCQSGMKIAVNAL
 !!                                      !! !!!            *** +++++++++**!  ** ********
```

Structurally verified:

```
MLN---KGAEGAMVFEPAYIK--AANPGDTVTFIPVDK-GHNVESIKDMIPEGAEK---P------KSK-INE-----NYVLTVTQPGAYLVKCTP--HYAMGMIALIAVG
LGGDDGS-----LAFVPNNIT--VGAGESIEFINNAGFPHNIVFDEDAVPAGVDADAIS------AEDYLNSKGQTVVRKLTTPGTYGVYCDP--HSGAGMKMTITVQ
WTF----NTESW-------PKGKRFRAGDILLFNYNPT-MHNVVVV--NQGGFSTC---NTPAGAKVYT-S-G-----RDQIKLP-KGQSYFICNFPGHCQSGMKIAVNAL
**                                      **            ** ****** +++++++++** ** ********
```

MSA:

```
MLNKGAEGAMVFEPAYIKANPGDTVTFI-PVDKGHNVESIKDMIPE------------GAEKFKSKINENYVLTVTQPGAYLVKCT--PHYAMGMIALIAVG
L--GGDDGSLAFVPNNITVGAGESIEFINNAGFPHNIVFDEDAVPAGVDADAISAEDYLNSKGQTVVRKLTTPGTYGVYCD--PHSGAGMKMTITVQ
W--TFNTESW---PKGKRFRAGDILLFNYNPTMHNVVVVNQGGFSTCNTPA--GAKVYTSGRDQ---IKLPKGQSYFI-CNFPGHCQSGMKIAVNAL
 !!                                      !! !!!             *** +++++++++**!  ** ********
```

Dynamic programming pairwise alignments:

```
MLNKGAEGAMVFEPAYIKANPGDTVTFIPVDK-GHNVESIKDMIPEG-------AEKFKSKINENYVLTVTQPGAYLVKCTPHYAMGMIALIAVG
L--GGDDGSLAFVPNNITVGAGESIEFINNAGFPHNIVFDEDAVPAGVDADAISAEDYLNSKGQTVVRKLTTPGTYGVYCDPHSGAGMKMTITVQ
*!                                          !!     ****** ********** ** ********
```

```
MLNKGAEGAMVFE-----PAYIKANPGDTVTFIPVDKGHNVESIKD------MIPEGAEKFKSKINENYVLTVTQPGAYLVKCT--PHYAMGMIALIAVG
-------WTFNTESWPKGKRFRAGDILLFNYNPTMHNVVVNQGGFSTCNTPAGAKVYTSGRDQ---IKLPKGQSYFI-CNFPGHCQSGMKIAVNAL
 !!                                      !! !!!     ***  +++++++++**  ** ********
```

```
LGGDDGSLAFVPNNITVGAGESIEFINNAGFPHNIVFDEDAVPAGVDADAISAEDYLNSKGQTVVRKLTTPGTYGVYCD--PHSGAGMKMTITVQ
WTFNTESW---PKGKRFRAGDILLFNYNPTMHNVVVNQGGFSTC---NTPAGAKVYTSGRDQI---KLPKGQSYFI-CNFPGHCQSGMKIAVNAL
**                                      !! !!!!    ** +++++++++**  ** ********
```

Figure 3.

Divide and Conquer:

```
X:  -6------5---4-6-5--3-6-5-6--4-6-5-6-4-6-5-6--2-6-5--6--4-6-5-6--3-6-5-6--4-6-5-6--5-6-5-6--1-6-5-6-4-6-5-6-4
GL----AKDAWEIPRESLRLEAKLGQGCPGEVWMGTWNDTTR-VAIKTLKPGTMSPEAFLQEAQVMKKLRHEKLVQLYAVVSEEP-IYIVIEYMSKGSL
TIYGVSPNYDKWEMERTDITMKHKLGGGQYGEVYEGVWKKYSLTVAVKTLKEDTMEVEEFLKEAAVMKEIKHPNLVQLLGVCTREPPFYIITEFMTYGNL
SS----YYWK--MEASEVMLSTRIGSGSPGTVYKGKWHGDVAVKILKVVDPTPEQLQAFRNEVAVLRKTRHVNILLFMGYMTKDN-LAIVTQWCEGSSL
!!!!!!!!!!!****************************************************!!!!!!**********!!!*****************

-6-5-6--2--5-6-4-6-5--3-6-5-6--4-6-5-6--0-6-5-6--4-6-5-6--3-6-5-6--4-6-5-6--2-6-5-6--4-6-5----6--3-6-5-
LDFLKGEMGKYLRLPQLVDMAAQIASGMAYVERMNYVHRDLRAANILVGENLVCKVADFGLARLIE-DNEYTARQGAKFPIKWTAPEAAL---YGRFTIKSDV
LDYLRECNRQFVSAVVLLYMATQISSAMEYLEKKNFIERDLAARNCLVGENHLVKVADFGLSRLMT-GDTYTAHAGAKFPIKWTAPESLA---YNKFSIKSDV
YKHLHVQETK-FQMFQLIDIARQTAQGMDYLHAKNIIHRDMKSNNIFLHEGLTVKIGDFGLATVKSRWSGSQQVEQPTGSVLWMAPEVIRMQDDNPFSFQSDV
******!!!!!****************************************************!!!!!!!!!!!***************!!!!!!!!!!

6-4-6-5-6--1--5-6-4-6-5--6--3-6-5-6-4-6-----5-6--2-6-5-6--4-6-5-6--3-6-5-6--4-6-5-6---
WSFGILLTELTTKGRVPYPGMVNR-EVLDQVERGYRMPCP----PECPESLHDLMCQCWRKDPEERPTFKYLQAQ-LLPACVLEVAE
WAFGVLLWEIATYGMSPYPGIDLS-QVYELLEKDYRMERP---EGCPEKVYELMRACWQWNPSDRPSFAEIHQA-FETMFQESSIS
YSYGIVLYELMAGEL-PYAHINNRDQIIFMVGRGYASPDLSRLYKNCPKAIKRLVADCVKVKEERPLFPQILSSIELLQHSLPKIN
**********************!!!!!**************************************!!!!!!!!!!!!!!!!!!!!!!
```

Maximum weight trace (originally 6 sequences):

```
---GLA-K-DAWEIPRESLRLEAKLGQGCPGEVWMGTWND-TTRVAIKTLK-PGTMSPEAFLQEAQVMKKLRHEKLVQLYAVVSE-EPIYIVIEYMSKGSL
TIYGVSPNYDKWEMERTDITMKHKLGGGQYGEVYEGVWKKYSLTVAVKTLKE-DTMEVEEFLKEAAVMKEIKHPNLVQLLGVCTREPPFYIITEFMTYGNL
-----S-S-YYWKMEASEVMLSTRIGSGSPGTVYKGKWHG-DVAVKILKVVDPTPEQLQAFRNEVAVLRKTRHVNILLFMGYMTK-DNLAIVTQWCEGSSL
**********************************************************************************************

LLDFLKGEMGKYLRLPQLVDMAAQIASGMAYVERMNYVHRDLRAANILVGENLVCKVADFGLARLIEDNEYTARQGAK-FPIKWTAPEA--ALY-GRFTIKSDV
LLDYLRECNRQFVSAVVLLYMATQISSAMEYLEKKNFIERDLAARNCLVGENHLVKVADFGLSRLMTGDTYTAHAGAK-FPIKWTAPES--LAY-NKFSIKSDV
LYKHLHV-QETKFQMFQLIDIARQTAQGMDYLHAKNIIHRDMKSNNIFLHEGLTVKIGDFGLATVKSRWSGSQQVEQPTGSVLWMAPEVIRMQDDNPFSFQSDV
**********************************************************************************************

WSFGILLTELTTKGRVPYPGMVNRE-VLDQVERGY---RMPCP-PECPESLHDLMCQCWRKDPEERPTFKYLQAQLLPACVLEVAE-----
WAFGVLLWEIATYGMSPYPGIDLSQ-VYELLEKDY---RMERP-EGCPEKVYELMRACWQWNPSDRPSFAEIH-----Q-AFETMFQESS-IS
YSYGIVLYELMA-GELPYAHINNRDQIIFMVGRGYASPDLSRLYKNCPKAIKRLVADCVKVKEERPLFPQIL----S-SIELLQHSLPKIN
******************************************
```

Figure 3 (continued)

MSA:

```
---GLAKDA--WEIPRESLRLEAKLGQGCPGEVWMGTWND-TTRVAI-KTLKPGTMSPEAFLQEAQVMKKLRHEKLVQLYAVVSEEP-IYIVIEYMSKGSL
TIYGVSPNYDKWEMERTDITMKHKLGGGQYGEVYEGVWKKYSLTVAV-KTLKEDTMEVEEFLKEAAVMKEIKHPNLVQLLGVCTREPPFYIITEFMTYGNL
-----SSYY--WKMEASEVMLSTRIGSGSFGTVYKGKWHG-DVAVKILKVVDPTPEQLQAFRNEVAVLRKTRHVNILLFMGYMTKDN-LAIVTQWCEGSSL
    !!!!!!*******************************************************************!!!!********

LDFLKGEMGKYLRLPQLVDMAAQIASGMAYVERMNYVHRDLRAANILVGENLVCKVADFGLARLIED-NEYTARQGAKFPIKWTAPEAALY---GRFTIKSDV
LDYLRECNRQFVSAVVLLYMATQISSAMEYLEKKNFIHRDLAARNCLVGENHLVKVADFGLSRLMTG-DTYTAHAGAKFPIKWTAPESLAY---NKFSIKSDV
YKHLHVQETKF-QMFQLIDIARQTAQGMDYLHAKNIIHRDMKSNNIFLHEGLTVKIGDFGLATVKSRWSGSQQVEQPTGSVLWMAPEVIRMQDDNPFSFQSDV
*****!!!!!!!*********************************************!!!!!!!!!!!*****!!!!!********

WSFGILLTELTTKGRVPYPGMVNR-EVLDQVERGYRMPCPP-----ECPESLHDLMCQCWRKDPEERPTFKYLQAQ---LLPACVLEVAE
WAPGVLLWELATYGMSPYPGIDLS-QVYELLEKDYRMERPE-----GCPEKVYELMRACWQWNPSDRPSFAEIHQAFETMFQESSIS---
YSYGIVLYELMA-GELPYAHINRRDQIIFMVGRGYASPDLSRLYKNCPKAIKRIVADCVKKVKEERPLFPQILSSIE-LLQHSLPKIN-
***********************!!!*********************************************!!!!!!!!!!!!
```