Chaining pairwise matches using the program chain2dim Manual

Stefan Kurtz*

August 4, 2014

1 Introduction

The following paper gives an in-depth introduction to the problem of chaining matches between two or more sequences. It also reports on different applications motivating the algorithms. The global and local chaining algorithms implemented in our program *chain2dim* are also described there.

> Chaining Algorithms and Applications to Comparative Genomics. *Enno Ohlebusch & Mohamed I. Abouelhoda*. Accepted for publication.

2 The program *chain2dim* and its options

chain2dim finds different kinds of chains in a given set of matches, namely:

- global chains without gap costs
- global chains with gap costs
- local chains

The program is called as follows:

chain2dim [*options*] *matchfile*

And here is a description of the options:

*Zentrum für Bioinformatik, Universität Hamburg, Bundesstrasse 43, 20146 Hamburg, Germany, E-mail: kurtz@zbh.uni-hamburg.de

`-global` [`gc`|`ov`]

Compute global chains. If the additional argument `gc` is used, then global chains with gap costs according to the L_1-model are computed. If the additional argument `ov` is used, then global chains with overlaps are computed.

`-local` [*lspec*]

Compute local chains with gaps costs according to the L_1-model. If there is no optional argument, then compute all local chains with a maximum score among all local chains. If there is an optional argument *lspec*, this is allowed to have three different forms:

- If *lspec* is a positive integer, then this is a minimum score. All local chains whose score is larger than or equal to the minimum score are reported.
- If *lspec* is a positive integer, say k, directly followed by the character `b`, then k specifies the number of best (i.e. largest) scores for which local chains are reported. Suppose that S is the set of all different scores of local chains. If $|S| \leq k$, then let $S_k = S$. If $|S| > k$, then let S_k be the subset of S consisting of the k largest scores in S. Then all local chains which have a score $s \in S_k$ are reported.
- If *lspec* is a positive integer smaller or equal to 100, say q, directly followed by the character `p`, then q specifies the percentage below the maximum score. Let m be the maximum score. Then all local chains with score larger than or equal to $m \cdot (1 - \lfloor \frac{q}{100} \rfloor)$ are reported.

`-wf` *weightfactor*

Specify a positive floating point value *weightfactor* by which the weight of each match is multiplied when computing the score of a chain. This option requires either option `-local` or option `-global gc`. The default *weightfactor* is 1.0. The weight factor is important to influence the weight of matches relative to the gap costs in local chains. The smaller the weight factor, the smaller the gaps between neighboring matches in a local chain. That is, if you only want to see chains where the gaps between the matches are small, then specify a small *weightfactor* smaller than 1.0. If you also want to see chains where the gaps between the matches are long, then specify a large *weightfactor*.

`-maxgap` *mg*

Specify the maximal width *mg* of a gap between two consecutive fragments of a chain. For example, if the first fragment ends at position q_1 and the second starts at position $q_2 > q_1$, then the gap size $q_2 - q_1 - 1$ must be at least *mg*. This constraint must hold in both dimensions.

`-outprefix` *prefix*

Specify that each chain is output into a seperate file whose name starts with *prefix*. In particular, the *i*th chain (counting from 0) is output into a file *filename*-*i*`.chain`

`-silent`

Only report the length and the score of chains, but not the chains themselves.

`-v`
: Be verbose, that is, report about the different steps of the computation as well as the resource requirements of the computation.

`-version`
: Show the version of the Vmatch version, the program is part of. Also report the compilation date and the compilation options.

`-help`
: Show a summary of all options and terminate.

Either option `-global` or `-local` must be used.

3 Input and output format

The matchfile specifies matches line by line. Two formats are allowed to specify matches, namely *Vmatch*-format and *simple* format. Both formats allow comment lines beginning with the character `#`. All lines which are not comment lines are match lines.

- The *Vmatch*-format is produced by the program `vmatch`. The first comment line is mandatory. Each match line reports a match by its length and the start position in the first and the second sequence. Among other values, the weight of the match is given. For a detailed description of the *Vmatch*-format, see the corresponding manual pages.
- Each match line in the simple format reports a match by four or five integers. The integers are separated by white spaces. The first two integers give the start position and the end position of the match in the first sequence. The third and the fourth integers give the start and the end position of the match in the second sequence. The optional fifth integer in the line specifies the weight. Let l_1 be the length of the match in the first sequence and l_2 be the length of the match in the second sequence. If the fifth integer in the line is missing, then the weight is $2 \cdot \min\{l_1, l_2\} - |l_1 - l_2|$. This is the largest score which can be achieved when aligning two sequences of length l_1 and l_2, where each pair of matching characters in the alignment is scoring 2, each mismatch is scoring -2, and each indel is scoring -1.

All match line in a matchfile must be in the same format. The matches a chain consists of, are reported in the same format as the format used in the matchfile.

4 Examples

Consider a matchfile `ecolicmp.vm` in *Vmatch*-format, reporting 14 matches between the *E.coli* K12 genome and the *Ecoli O157:H7* genome.

```
$ cat ecolicmp.vm
# args=-l 50 -q EcoliO157H7 EcoliK12
  165    0      64   D   165    0       64   0    6.53e-87    330   100.00
  165    0     393   D   165    0      410   0    6.53e-87    330   100.00
  116    0    1884   D   116    0     1901   0    2.07e-57    232   100.00
  167    0    2508   D   167    0     2525   0    4.08e-88    334   100.00
  117    0    2760   D   117    0     2777   0    5.17e-58    234   100.00
  128    0    1077   D   128    0     1094   0    1.23e-64    256   100.00
  183    0    1623   D   183    0     1640   0    9.50e-98    366   100.00
  117    0     393   D   117    0  5471259   0    5.17e-58    234   100.00
  117    0    3244   D   117    0     3261   0    5.17e-58    234   100.00
  128    0    3622   D   128    0     3639   0    1.23e-64    256   100.00
  165    0      64   D   165    0  5470913   0    6.53e-87    330   100.00
  173    0    3763   D   173    0     3780   0    9.96e-92    346   100.00
  164    0    3937   D   164    0     3954   0    2.61e-86    328   100.00
  134    0    4201   D   134    0     4218   0    3.01e-68    268   100.00
```

The first program call (following the prompt `$`) delivers the highest scoring global chain of length 12 with score 3514.

```
$ chain2dim -global ecolicmp.vm
# chain 0: length 12 score 3514
  165    0      64   D   165    0       64   0    6.53e-87    330   100.00
  165    0     393   D   165    0      410   0    6.53e-87    330   100.00
  128    0    1077   D   128    0     1094   0    1.23e-64    256   100.00
  183    0    1623   D   183    0     1640   0    9.50e-98    366   100.00
  116    0    1884   D   116    0     1901   0    2.07e-57    232   100.00
  167    0    2508   D   167    0     2525   0    4.08e-88    334   100.00
  117    0    2760   D   117    0     2777   0    5.17e-58    234   100.00
  117    0    3244   D   117    0     3261   0    5.17e-58    234   100.00
  128    0    3622   D   128    0     3639   0    1.23e-64    256   100.00
  173    0    3763   D   173    0     3780   0    9.96e-92    346   100.00
  164    0    3937   D   164    0     3954   0    2.61e-86    328   100.00
  134    0    4201   D   134    0     4218   0    3.01e-68    268   100.00
```

Note that only two matches (where the second instances of the match are at large positions) are missing in the global chain. The matches making up the chain are reported in order of their start positions. Because it is a chain, this is consistent in both sequences. Note that the original comment line from the matchfile is echoed in the output. This allows to use the output file for the program *Genalyzer* or other programs from the *Vmatch*-software suite.

Now suppose we have the original input in simple format file `ecolicmp.of`:

```
$ cat ecolicmp.of
# args=-l 50 -q EcoliO157H7 EcoliK12
     64     228      64     228     330
    393     557     410     574     330
   1884    1999    1901    2016     232
   2508    2674    2525    2691     334
   2760    2876    2777    2893     234
   1077    1204    1094    1221     256
   1623    1805    1640    1822     366
    393     509 5471259 5471375     234
   3244    3360    3261    3377     234
   3622    3749    3639    3766     256
     64     228 5470913 5471077     330
   3763    3935    3780    3952     346
   3937    4100    3954    4117     328
   4201    4334    4218    4351     268
```

The following program call delivers the same chain as above, but with a different score, due to the additional gap penalties (switched on by the argument `gc` for option `-global`. Because, we use option `-silent`, the chain is not reported:

```
$ chain2dim -silent -global gc ecolicmp.of
# chain 0: length 12 score -5468705
```

Now lets turn to local chains. The simplest call reports the optimal local chain:

```
$ chain2dim -local ecolicmp.of
# chain 0: length 4 score 964
3622 3749 3639 3766 256
3763 3935 3780 3952 346
3937 4100 3954 4117 328
4201 4334 4218 4351 268
```

Using a weight factor 1.8, the optimal local chain extends to the left by three matches:

```
$ chain2dim -wf 1.8 -local ecolicmp.of
# chain 0: length 7 score 1931
2508 2674 2525 2691 601
2760 2876 2777 2893 421
3244 3360 3261 3377 421
3622 3749 3639 3766 460
3763 3935 3780 3952 622
3937 4100 3954 4117 590
4201 4334 4218 4351 482
```

Using a weight factor 0.5, the last match is separated from the optimal local chain which was computed with the default weight factor 1.0:

```
$ chain2dim -wf 0.5 -local ecolicmp.of
# chain 0: length 3 score 433
3622 3749 3639 3766 128
3763 3935 3780 3952 173
3937 4100 3954 4117 164
```

Alternatively, we can specify a maximum gap value to cut chains. For example the following only outputs a chain with maximum gap 20. As a consequence the last fragment `4201 4334 4218 4351 268` which has distance 100 to the previous fragment.

```
$ chain2dim -local -maxgap 20 ecolicmp.of
# chain 0: length 3 score 898
3622 3749 3639 3766 256
3763 3935 3780 3952 346
3937 4100 3954 4117 328
```

To obtain different local chains with some minimum score we add an extra argument to the option `-local`. For example, to obtain the chains with the two largest scores, we use the argument `2b`. In addition we use option `-v`, which reports the different steps of the computation.

```
$ chain2dim -local 2b -v ecolicmp.of
# match file "ecolicmp.of" (open format) read
# compute chain scores
# retrieve optimal chains
# compute optimal local chains with score >= 440
# chain 0: length 2 score 440
1623 1805 1640 1822 366
1884 1999 1901 2016 232
# chain 1: length 4 score 964
3622 3749 3639 3766 256
3763 3935 3780 3952 346
3937 4100 3954 4117 328
4201 4334 4218 4351 268
# overall space peak: main=0.10 MB
```

The same chains are computed if we use the arguments `440` or `55p` to the option `-local`.

```
$ chain2dim -local 55p -silent ecolicmp.of
# chain 0: length 2 score 440
# chain 1: length 4 score 964
```