APPENDIX

A MAIN DATA STRUCTURE

Successor tables (ST). The successor tables $\{ST_1, ST_2, ..., ST_d\}$ of the sequence set $T = \{S_1, S_2, ..., S_d\}$ are built to support the compression of the data and quick search for the immediate successors of the points. For a sequence $S_l = x_1, x_2, ..., x_n$ from the sequence set T over a finite alphabet $\Sigma = (c_1, c_2, ..., c_k)$, its successor table ST_l is a two-dimensional array, where $ST_l[i, j]$ (the element of the ith row and the jth column) is defined as

$$ST_l[i, j] = min\{r | x_r = c_i, r \geqslant 1, r \geqslant j, 1 \leqslant i \leqslant |\Sigma|, 0 \leqslant j \leqslant n\} \quad (6)$$

From Eq. 6, we can see that $ST_l[i, j]$ denotes the minimal position r (the rth character position) of the sequence S_l with $x_r = c_i$ after position j. See the examples in Fig. 6.

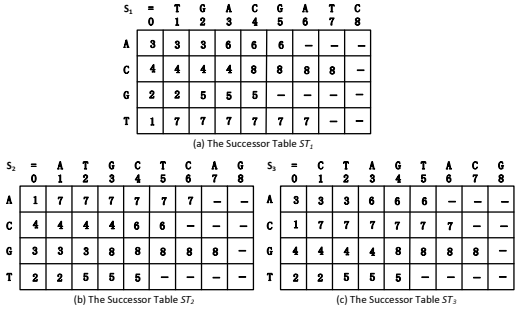

Figure 6: The constructed successor tables ST_1, ST_2 and ST_3 corresponding to the sequences S_1, S_2 and S_3 (in the paper), where "−" indicates $\emptyset$.

The set S_{suc} of immediate successors of a d-dimensional point $p = (p_1, p_2, ..., p_d)$ can be obtained efficiently in $O(d|\Sigma|)$ time. For a d-dimensional point p, the operation for producing its S_{suc} can be characterized by Eq. 7.

$$S_{suc} = \{(ST_1[i', p_1], ST_2[i', p_2], ..., ST_d[i', p_d])\}$$
$$s.t. 1 \leqslant i' \leqslant |\Sigma|, \forall ST_l[i', p_i] \neq \emptyset, 1 \leqslant l, i \leqslant d \quad (7)$$

For example, for the dominant $(2, 3, 4)$ of the sequences S_1, S_2 and S_3 (in the paper), we can couple the corresponding rows 1-4 of the second, third and forth columns from the successor tables ST_1, ST_2 and ST_3 to obtain all its immediate successors $(3, 7, 6)$, $(4, 4, 7)$, $(5, 8, 8)$ and $(7, 5, 5)$ corresponding to the characters A, C, G, and T, respectively. There is no immediate successor for the dominant $(6, 7, 3)$ due to the coupling results $(\_, \_, 6)$, $(8, \_, 7)$, $(\_, 8, 4)$ and $(7, \_, 5)$, which indicates none of the points is an immediate successor according to Eq. 7.

B EXPERIMENTAL RESULTS

We evaluate the performance of the approximate algorithms, whose performances vary in terms of not only efficiency but also precision. Here, the precision is measured by Eq. 3. The results for all the tested approximate algorithms, including the state-of-the-art *CRO*, *SA_MLCS* as well as our algorithm *HA-MLCS* are shown in Tables 3 and 4.

Table 3: Precisions (P) and running times (T) of *CRO* (A1), *SA-MLCS* (A2) and *HA-MLCS* (A3) for 5 sequences with various lengths ($|S_i|$).

	$\|\Sigma\|$=4							$\|\Sigma\|$=20					
	A1		A2		A3 (m = 100)			A1		A2		A3 (m = 100)	
$\|S_i\|$	P	T(s)	P	T(s)	P	T(s)	$\|S_i\|$	P	T(s)	P	T(s)	P	T(s)
1.0E+3	0.524	0.08	0.822	0.97	0.988	0.71	1.0E+3	0.411	0.09	0.893	7.43	0.992	0.63
2.0E+3	0.496	0.11	0.818	1.94	0.981	1.42	2.0E+3	0.366	0.12	0.877	17.31	0.981	1.26
5.0E+3	0.391	1.01	0.792	2.41	0.985	1.35	5.0E+3	0.357	5.79	0.857	47.99	0.978	2.90
1.0E+4	0.347	3.48	0.743	9.33	0.975	5.50	1.0E+4	0.334	2.27	0.822	113.70	0.967	4.35
2.0E+4	0.322	5.86	0.717	32.59	0.972	9.22	2.0E+4	0.327	8.67	0.802	239.00	0.971	8.12
5.0E+4	0.314	33.01	0.701	87.06	0.968	27.56	5.0E+4	0.305	53.94	0.791	541.80	0.965	18.79
1.0E+5	0.296	133.20	0.678	156.79	0.959	52.86	1.0E+5	0.279	218.00	0.751	5272.00	0.952	39.87
1.0E+7	+	+	0.269	1609.00	0.951	557.20	1.0E+7	+	+	0.741	44190.00	0.962	403.60
1.0E+8	+	+	+	+	0.948	3430.00	1.0E+8	+	+	+	+	0.963	2985.00

Symbol '+' indicates the memory overflow leading to calculating failure.

Table 4: Precisions (P) and running times (T) of *CRO* (A1), *SA-MLCS* (A2) and *HA-MLCS* (A3) for d sequences with lengths ($|S_i|$) 1000 and 2000, respectively.

	$\|\Sigma\|$=4, $\|S_i\|$=1000							$\|\Sigma\|$=20, $\|S_i\|$=2000					
	A1		A2		A3 (m = 100)			A1		A2		A3 (m = 100)	
d	P	T(s)	P	T(s)	P	T(s)	d	P	T(s)	P	T(s)	P	T(s)
10	0.689	0.03	0.792	0.62	0.988	0.41	10	0.698	0.03	0.801	0.12	0.987	0.08
50	0.667	0.06	0.756	0.82	0.977	0.51	50	0.671	0.06	0.811	0.11	0.971	0.20
100	0.645	0.08	0.758	0.97	0.975	0.63	100	0.655	0.07	0.801	0.10	0.978	0.13
400	0.632	0.17	0.742	1.02	0.973	0.74	400	0.589	0.17	0.812	0.23	0.975	0.19
800	0.617	0.19	0.738	1.33	0.973	0.88	1000	0.623	0.35	0.805	0.32	0.972	0.25
1000	0.602	0.26	0.717	2.26	0.969	1.24	5000	0.611	1.51	0.798	0.41	0.966	0.35
3000	0.587	0.45	0.703	3.11	0.967	1.64	10000	0.594	2.98	0.785	0.43	0.966	0.41
5000	0.577	0.70	0.695	3.57	0.965	1.76	14000	0.568	4.26	0.773	0.50	0.964	0.47
10000	0.534	1.11	0.686	3.83	0.959	2.18	20000	0.536	5.57	0.758	1.21	0.959	0.73

C THE EVOLUTION TREES OF COVID-19 VIRUSES FROM THE USA AND ENGLAND

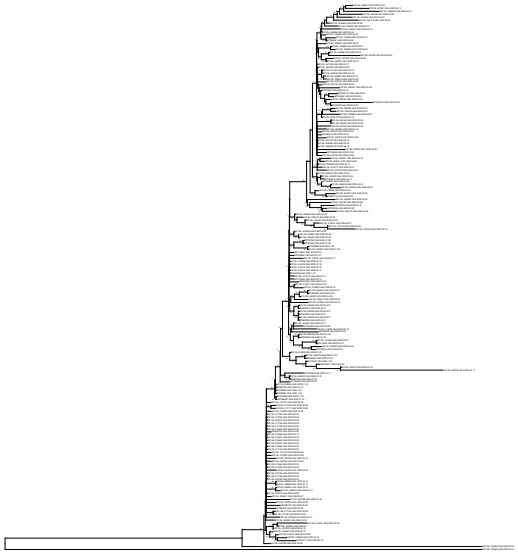

Figure 7: The evolution tree of COVID-19 viruses in human hosts from USA.

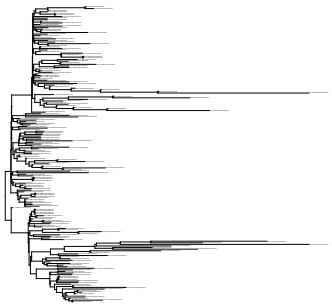

Figure 8: The evolution tree of COVID-19 viruses in human hosts from England.

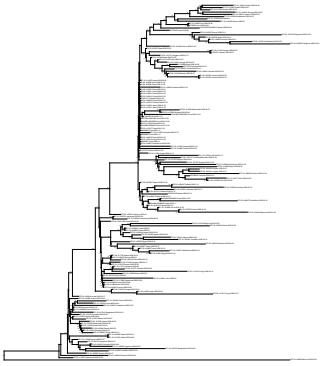

Figure 9: The evolution tree of COVID-19 viruses in human hosts from different 79 countries.

D ALGORITHM PSEUDOCODE

Algorithm 1 HA-MLCS(T, Σ)

$ST \leftarrow$ Construct Successor Tables of sequence set T with Σ;
$k \leftarrow 0$; *MLCS-ODAG* $\leftarrow \varnothing$; $D^k \leftarrow \{(0, 0, ..., 0)\}$;
$D^{k+1}_{init} \leftarrow Successor(D^k, ST)$; //calculate successor point set D^{k+1}_{init} of D^k
while $D^{k+1}_{init} \neq \varnothing$ **do** //Constructing *MLCS-ODAG* with layer by layer
  $(D^{k+1}_{init})_{1st} \leftarrow$ BestNondominatedSorting(D^{k+1}_{init});
  Calculate score(p) by Eq. 1, $p \in (D^{k+1}_{init})_{1st}$;
  $D^{k+1} \leftarrow$ Keep top m points with the minimum score in $(D^{k+1}_{init})_{1st}$;
  MLCS-ODAG $\leftarrow$ *MLCS-ODAG* $\cup\, D^{k+1}$; $k \leftarrow k + 1$;
  $D^{k+1}_{init} \leftarrow Successor(D^k, ST)$;
end while
$maxlevel \leftarrow k - 1$; $k \leftarrow 0$; $D^0 \leftarrow \{(\infty, \infty, ..., \infty)\}$;
while $D^k \neq \varnothing$ **do** //Algorithm BackwardTopSorting
  $D^{k+1} \leftarrow \varnothing$;
  for $q \in D^k$ **do**
    for $p \in precursor[q]$ **do**
      if $tlevel[p] + k \neq maxlevel$ **then**
        Delete p from *MLCS-ODAG*;
      else
        $D^{k+1} \leftarrow D^{k+1} \cup \{p\}$;
      end if
    end for
  end for
  $k \leftarrow k + 1$;
end while
return all of the *MLCSs* of sequence set T;

The proposed algorithm *HA-MLCS* is implemented in Java JDK1.8. Where m is a user-customized parameter ($1 \leqslant m \in \mathbb{Z}$), which represents how many number of key points to be retained in each layer when constructing *MLCS-ODAG*. The source code of algorithm *HA-MLCS* is available at: https://github.com/HA-MLCS/HA-MLCS

E THE SIMILARITY BETWEEN COVID-19 AND RELATED VIRUSES

Table 5: The similarity matrices.

LCS-Based

	HCoV229E	HCoVhku1	HCoVnl63	HCoVoc43	SARS	Lassa	MERS	Victoria	Yamagata	Ebola	Dengue	Rhinolophine	Pangolin	Environment
2019.12	0.652594	0.691539	0.658072	0.678947	0.826398	0.340651	0.694165	0.425849	0.425918	0.499802	0.340251	0.741884	0.875237	0.999240
2020.01	0.652732	0.691539	0.659161	0.680932	0.826633	0.340785	0.694067	0.425989	0.426070	0.499965	0.340377	0.741971	0.875355	0.998743
2020.02	0.652987	0.691313	0.659465	0.681191	0.826681	0.341012	0.694763	0.426234	0.426302	0.500241	0.340610	0.741924	0.875033	0.997878
2020.03	0.654413	0.689322	0.660977	0.681283	0.823739	0.342154	0.694857	0.427542	0.427530	0.501517	0.341783	0.740702	0.870977	0.992975
2020.04	0.650424	0.675613	0.651367	0.687941	0.827384	0.342254	0.694740	0.428357	0.428087	0.502606	0.342201	0.740289	0.875600	0.993509
2020.05	0.658049	0.682661	0.658083	0.688351	0.828885	0.344117	0.695210	0.430231	0.429576	0.503550	0.343054	0.740601	0.878438	0.970557
Average	**0.653533**	**0.686998**	**0.658004**	**0.683107**	**0.826620**	**0.341829**	**0.694634**	**0.427367**	**0.427247**	**0.501280**	**0.341379**	**0.741229**	**0.875107**	**0.992150**

LD-Based

	HCoV229E	HCoVhku1	HCoVnl63	HCoVoc43	SARS	Lassa	MERS	Victoria	Yamagata	Ebola	Dengue	Rhinolophine	Pangolin	Environment
2019.12	0.555877	0.561025	0.560896	0.561579	0.796821	0.339617	0.570439	0.418336	0.418386	0.471776	0.339098	0.663919	0.861980	0.999239
2020.01	0.555879	0.561095	0.560922	0.561698	0.796944	0.339750	0.570588	0.418456	0.418508	0.471892	0.339220	0.663868	0.861901	0.998727
2020.02	0.555845	0.561248	0.560880	0.561874	0.796624	0.339974	0.570760	0.418670	0.418719	0.472045	0.339446	0.663574	0.861409	0.997861
2020.03	0.555188	0.561446	0.560093	0.562361	0.796700	0.341071	0.571322	0.419872	0.419881	0.472709	0.340569	0.660890	0.856334	0.992766
2020.04	0.555563	0.561800	0.560262	0.562394	0.795090	0.341193	0.571710	0.422083	0.422847	0.472724	0.341657	0.660874	0.853377	0.993442
2020.05	0.553189	0.560186	0.558221	0.562513	0.796969	0.342833	0.571981	0.423131	0.423201	0.477284	0.342929	0.661864	0.854125	0.970490
Average	**0.555257**	**0.561133**	**0.560212**	**0.562070**	**0.796525**	**0.340740**	**0.571133**	**0.420091**	**0.420257**	**0.473072**	**0.340486**	**0.662498**	**0.858188**	**0.992087**

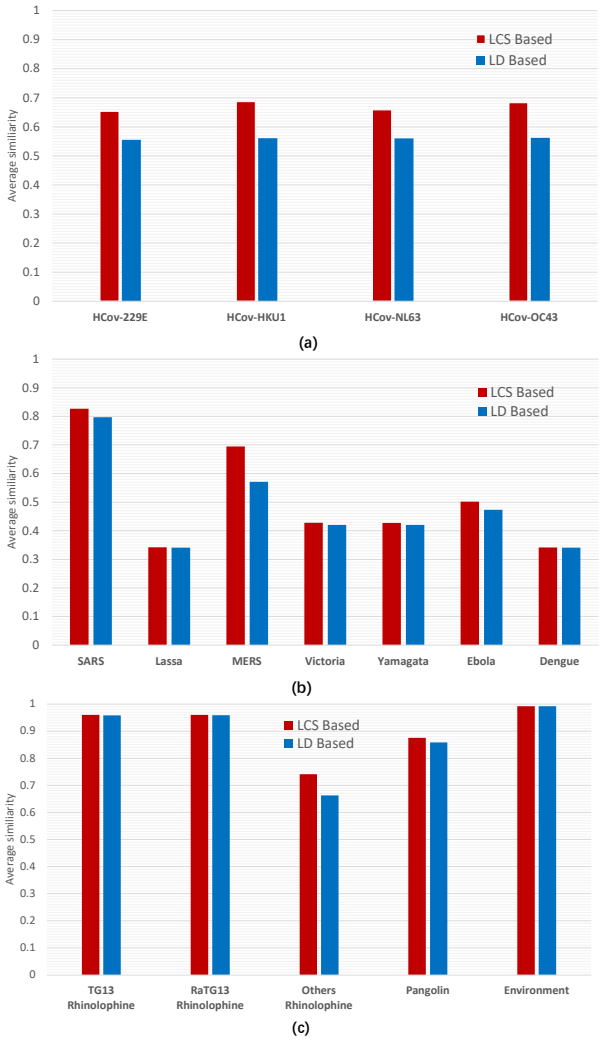

Figure 10: The schematic diagram between COVID-19 strains and other viruses.