SUPPLEMENTARY MATERIALS

Pseudocode of the Algorithm for Computing MCS

Below is the pseudocode of our algorithm for computing the MCS between a pair of graphs. In this recursive algorithm every call will cause the initial common subgraph m to increment by one vertex correspondence. If a *maximal* match is reached, then the identified maximal common subgraph m will be compared to the previously identified maximal common subgraphs between the two input structures. When the entire search space has been explored, the largest maximal common subgraph is returned as the MCS of the input structures.

```
MATCH$(G_1, G_2, m, t)$
  **if** $upperBound(G_1, G_2, m) < candidate$
    **then**
          **return**
  $t' \leftarrow t$
  **while** $true$
    **do**
        $v_1 \leftarrow order(G_1 - m)$
        $t' \leftarrow t \bigcup \{v_1\}$
        **if** $v1 = None$
          **then**
              $updateCandidate(m)$
              **return**
        **for** $v_2 \in G_2 - m$
          **do**
              **if** $compatible(v_1, v_2)$
                **then** $m' \leftarrow m + \{v_1 : v_2\}$
                      $Match(G_1, G_2, m', t')$
        **return**
```

The inputs in the above algorithm are the two graphs G_1 and G_2, a partial solution m represented as a set of correspondences, and a list of tested vertices t that need to be excluded in this search step. A vertex in t is either included in the partial solution m or it is chosen not to be used along this branch of the search tree. The variable t is introduced to avoid searching the same partial solution more than once. The $upperBound$ step is a subroutine for estimating the upper bound of the detected MCS size if its solution is based on the current partial match m. The $order$ component is a subroutine for ordering the unmatched vertices in the graph G_1. This subroutine returns one of the unmatched vertices of G_1 that will be used in the current matching round, and therefore controls in which order the solution space is searched. It also controls the termination of the search process along the current branch of the search tree. For example, when none of the unmatched vertices in G_1 match to any unmatched vertex in G_2 then the above $order$ subroutine will cause the algorithm to backtrack. The $updateCandidate$ subroutine is called when the search reaches a leaf of the search tree, and it uses the current partial solution to update the global candidate solution when the current one is better than the previous one. The $compatible$ step tests whether vertex v_1 from G_1 and vertex v_2 from G_2 match each other. It utilizes the labels of the vertices (atoms), the induced edges (bonds), and other important structural feature constraints.

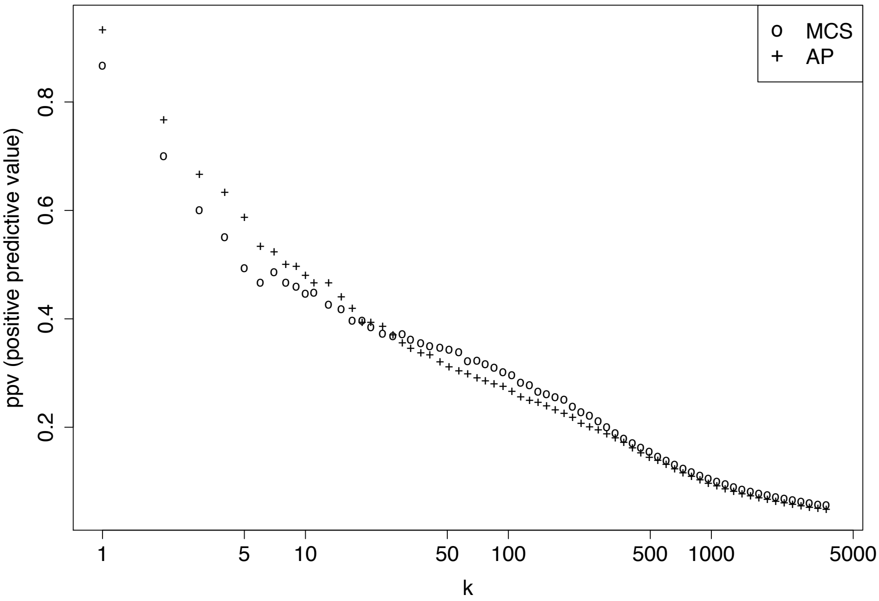
Models	*Training set size*			
	400	*2000*	*5000*	*10000*
MCS	58.5(3.0)	64.3(2.4)	67.2(1.3)	69.8(0.9)
MCS c1	58.8(3.1)	65.2(1.7)	68.2(1.4)	70.0(1.9)
MCS c2	59.7(3.2)	67.0(1.5)	69.2(1.0)	71.0(0.9)
MCS c3	59.2(2.7)	65.8(1.7)	68.5(1.1)	70.5(1.9)
hybrid	61.3(3.4)	67.0(1.9)	69.7(1.3)	71.5(1.2)
hybrid c1	60.1(3.3)	66.6(1.6)	69.4(1.3)	71.8(1.7)
hybrid c2	60.8(3.4)	67.6(1.7)	70.4(1.2)	72.3(0.9)
hybrid c3	60.2(3.2)	67.0(1.7)	69.9(1.2)	72.3(1.2)

(a)

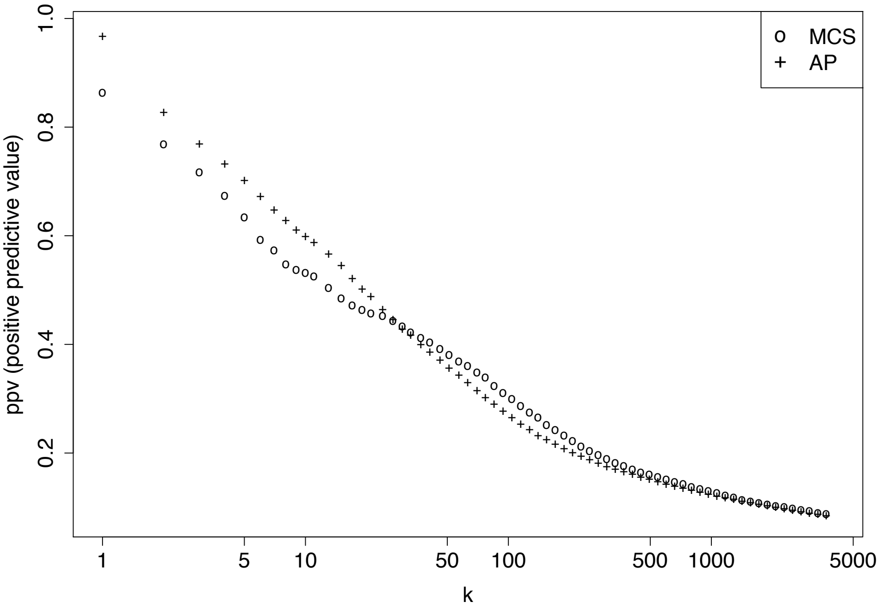
Models	*Training set size*			
	300	*1000*	*2000*	*5000*
MCS	60.0(2.8)	64.5(1.8)	66.8(1.7)	69.9(1.3)
MCS c1	59.5(3.2)	64.7(1.8)	67.7(1.5)	71.1(1.3)
MCS c2	59.4(3.1)	64.6(1.8)	67.5(1.5)	71.0(1.2)
MCS c3	58.2(3.0)	64.2(1.7)	67.4(1.4)	71.4(0.9)
hybrid	62.7(3.2)	67.6(1.8)	70.4(1.4)	73.8(1.2)
hybrid c1	61.5(3.2)	67.2(1.6)	70.2(1.5)	73.8(1.2)
hybrid c2	61.7(3.5)	67.4(1.7)	70.7(1.3)	74.4(1.1)
hybrid c3	60.4(3.4)	66.8(1.7)	70.2(1.5)	74.2(1.3)

(b)

Table A-1. Average AUC values using prediction models based on different MCS coefficients and different training set sizes. Standard deviations are given in parentheses. Table (a) lists the result for the NCI antiviral dataset, and Table (b) lists the result for the NCI anti-cancer dataset. The *MCS* model uses the absolute MCS sizes. The models *MCS c1*, *MCS c2* and *MCS c3* use the MCS coefficients listed in Equations 2, 3 and 4, respectively. The *hybrid* model uses the absolute MCS sizes and the AP information. The models *hybrid c1*, *hybrid c2* and *hybrid c3* use the MCS coefficients listed in Equations 2, 3 and 4, respectively, and the AP information.



(a)



(b)

Fig. A-1: Performance Comparisons of AP- and MCS-based search methods. The average PPVs from all simulated similarity searches are plotted against the k values. Part (a) provides the results for the NCI antiviral dataset and part (b) for the NCI anti-cancer dataset.

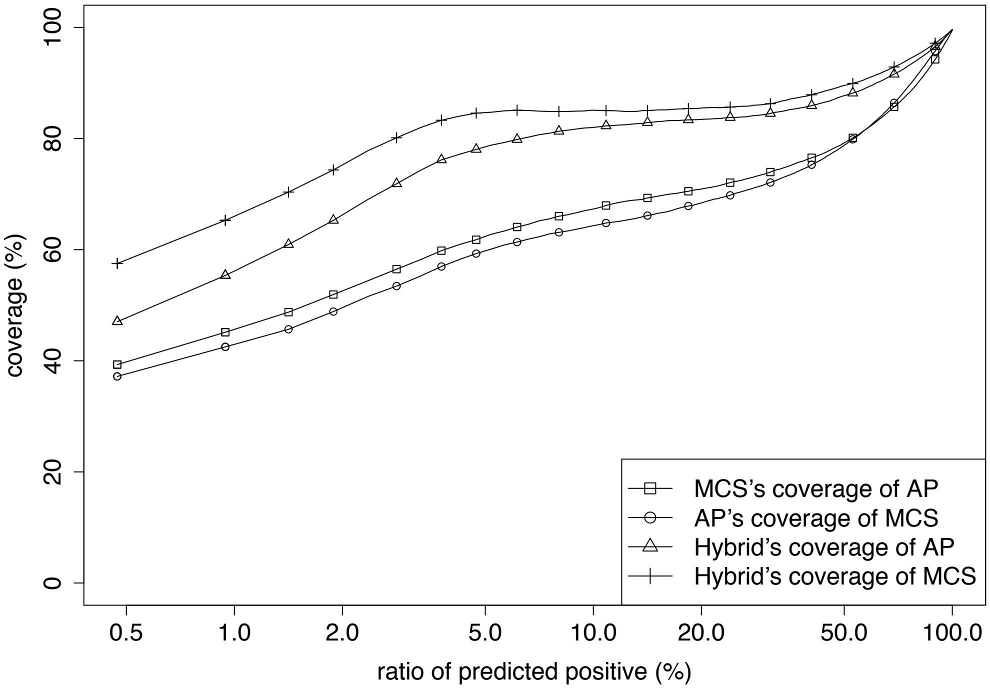


Fig. A-2: Cross coverages between pairs of prediction models. The x axis is the number of predicted positives over the total number of compounds in the dataset. The y axis is one model's coverage of another model (see Equation 1). For example, when 10% of the compounds are predicted to be active, 82% of the compounds identified by the AP model are also identified by the hybrid model. This means that under this condition the hybrid model's coverage of the AP model is 82%. This corresponds to point $(10, 82)$ in the curve.