SimSite3D Software User Guide version 3.0

Protein Structural Analysis & Design Lab
MSU
J. Van Voorst and L. Kuhn

This guide shows how to use the SimSite3D software[1] to generate sitemaps (templates) of protein binding sites, to compare sitemaps of protein binding site, and how to interpret the results. Text in <angular brackets> should be replaced by actual names provided by the user (without the angular brackets). Text in [square brackets] denotes optional parameters to be replaced by actual names provided by the user. For help with installation and usage, or to provide feedback on the software, please contact Jeffrey Van Voorst (vanvoor4@msu.edu) or Leslie Kuhn (KuhnL@msu.edu).

1 Introduction

The SimSite3D Software tools are designed to quickly search a database of three dimensional structures, in Protein Data Bank format, with protein-ligand binding sites to determine which binding sites in the database have steric and chemical similarities to the query binding site. To realize this goal, SimSite3D performs some offline computations for each protein-ligand binding site, and the actual searches make use of the precomputed representation of the database binding sites.

More particularly, SimSite3D makes use of protein-ligand binding site sitemaps. A sitemap is computed from a user specified volume and protein coordinate file (in PBD format). Sitemaps contain the computed positions of favorable interaction in the ligand binding volume. The computed positions are those from which good hydrogen bonds or hydrophobic interactions can be made with the protein. Because SimSite3D was designed and tested on protein-ligand binding sites, the sitemap volume should contain the volume of interest that could be occupied by potential binders.

After one or more sitemaps is built, a sitemap may be used to query a database of sitemaps for binding sites that are significantly similar to the query binding site. During a database search, the best 3D alignment is determined between each database sitemap and the query sitemap. The top scoring alignments are saved in a user specified directory. Besides searching a database, one has the option of performing a comparison between two sitemaps.

SimSite3D can be used without computing alignments, that is, SimSite3D can also score user-provided alignments. Thus, the database proteins, ligands, and sitemaps are all assumed to be aligned with respect to the query protein, ligand, and sitemap. Finally, the methods SimSite3D uses to compare binding sites are computationally comparable to protein-ligand docking methods and require similar computational resources.

[1] Note: The name for SimSite3D at MSU is SimSite3D.

2 Generating sitemaps

Sitemap representation: The program gen_points, with usage described below, can be used to generate an unbiased sitemap of a binding site based on a given volume (the supported volumes are ligand bounding boxes using the minimum and maximum coordinate values in the coordinate system of the mol2 file and spheres defined by a point and radius). An unbiased sitemap is very similar to the template representation described in M. I. Zavodszky, P. C. Sanschagrin, R. S. Korde, and L. A. Kuhn (2002) "Distilling the Essential Features of a Protein Surface for Improving Protein-Ligand Docking, Scoring, and Virtual Screening", *Journal of Computer-Aided Molecular Design*, **16**, 883-902. Each sitemap is split into four files by SimSite3D. These files are the sitemap labels file (`XXXXXX_s.csv`), the sitemap points file (`XXXXXX_s.pdb`), the sitemap atoms file (`XXXXXX_a.pd`b), and the sitemap rad file (`XXXXXX_rad.pdb`). The file naming convention used by SimSite3D can be found in more detail in section 5.

The sitemap labels file and points files are generated by SimSite3D from the residues of a given protein. The sitemap labels file is the main sitemap file in terms of the information contained in the sitemap representation and is formatted using XML-like labeled sections. The sitemap points file lists the sitemap interaction points identifying favorable positions (and in the case of H-bonds, angles) for placing ligand hydrophobic atoms (labeled H), hydrogen-bond or salt-bridge acceptor atoms (labeled A), hydrogen-bond or salt-bridge donor atoms (labeled D), and positions at which a ligand atom could either accept or donate an H-bond/salt bridge from the protein ("doneptor" sites, labeled N). The sitemap points file is a PDB-formatted file containing the sitemap points (which can be visualized using standard molecular graphics software that respect a '#' character in the first column of a line as a comment line and ignore REMARK lines). In a sitemap points file, hydrophobic points are assigned B-values (last numeric column of PDB file) of 100 and residue numbers ranging 1-1000, A points have B-values of 0 and residue numbers ranging 1001-2000, D points have B-values of 50 and residue numbers ranging 2001-3000, and N points have B-values of 25 and residue numbers ranging 3001-4000. Assigning B-values to sitemap points based on their interaction type makes it easy to differentially color or render the sitemap points in most molecular graphics software according to their chemistry. The sitemap represents favorable ligand atom chemistry and placement for interacting with the protein and can be used to guide ligand optimization as well as compare binding sites, as is performed by SimSite3D.

The sitemap atoms and rad files are a reduced set of the lines from the protein structure file (PDB file). The sitemap atoms file is a PDB file containing the ATOM and HETATM records of the subset of protein atoms (used to generate the sitemap interaction points) that have the potential to make favorable interactions with positions in the sitemap. The sitemap rad file contains all of the protein residues that have at least 1 atom within 4.5 Å of the ligand defining the binding site volume or within 4.5 Å of the sitemap volume if defined by a sphere. The rad file holds those residues that have the potential to form favorable interactions with ligand atoms in the sitemap volume. The purpose of the rad file is to provide protein-ligand scoring functions (such as DrugScore) with a reduced number of residues to consider.

To enable matching of favored H-bond directions between sitemaps, a unit vector is associated with each H-bond sitemap point; the vector represents the direction from the heavy atom (typically N or O) to the sitemap point. Goodness of match between a matched query and database polar point pair is computed as the dot product between their vectors. The vectors are written into the sitemap points file (XXXXXX_s.pdb) as a #HBVEC line. An example of a #HBVEC line is:

```
"#HBVEC 4020  O   HOH  4020       0.321  -0.825  -0.465  1.00200.00"
```

The important items are the residue number and the coordinates. The residue number `4020` denotes that this line contains the unit vector in the preferred direction for the 20th hbond template point. The coordinates is the unit vector representing the preferred direction.

Input file pre-processing:

- The ligand file (in mol2 format) is assumed to have the atomic coordinates that were determined as part of the protein structure provided in the protein file and to be in the same reference frame as the protein. The hydrogen atoms of the ligand are ignored when determining the volume of the binding site. Care should be taken when labeling the database ligand atoms (field 2 in the mol2 atom records) because the atom names in the database ligand files will be used throughout the search and will be present in the results.

- By default, all water molecules in the protein PDB file are ignored. The "--waters" flag may be used to specify a listing (to gen_points) of those water molecules you wish to consider as a part of the binding site (i.e. the water molecules to consider as part of the protein surface, both sterically and chemically). Because the hydrogen atoms of water molecules are used to infer the orientation of the water molecules, inclusion of bound waters in the binding site requires protonation of the water molecules. In particular, up to 5 sitemap points may be set down for each hydrogen atom and each lone pair of electrons of a water molecule.

- By default, all metal atoms present in a PDB structure are ignored. Using the "--include_metals" flag will cause gen_points to consider all metal atoms near the binding site as part of the protein (if you wish to include only a subset of the metals near the binding site, please remove the undesired metal atoms from the structure file). gen_points models the interactions of a metal atom by setting down a number of points on a sphere centered at the center of the metal atom with a radius dependent on the metal. Because metals are usually positively charged, the sitemap points on the sphere are labeled as hydrogen bond acceptor points to denote sites of favorable interaction with the metal. The supported metal elements are: Ca, Co, Cu, Fe, K, Mg, Mn, Na and Zn. The sphere radii were determined as the radius corresponding to the first maximum in the metal-ligand distance distributions in the

Metalloprotein Database and Browser (The Scripps Research Institute; http://metallo.scripps.edu): Ca, 2.4 Å; Co, 1.9 Å; Cu, 2.1 Å; Fe, 2.2 Å; K, 2.4 Å; Mg, 2.1 Å; Mn, 2.2 Å; Na, 2.4 Å; and Zn, 2.1 Å.

- HETATMs in the protein PDB file that are part of a chain (e.g., nonstandard residues and peptydyl inhibitors) are considered as part of the protein, sterically, but their chemistry does not contribute to the chemistry labels of sitemap points. For handling RCSB PDB files in which peptidyl inhibitors often cannot be distinguished as a ligand rather than as a separate chain of the protein, peptides with fewer than 10 residues do not contribute to the sitemap. We recommend not including the ligand itself, and other HETATMs/ligands that are not essentially part of the protein, in the PDB file. Pfizer crystal structures have already been prepared this way.

- Hydrogen atoms on protein residues in the input protein file (in PDB format) are ignored. The sitemap generation program implicitly determines the positions of hydrogen atoms with respect to protein donor and donor antecedent atoms.

- NOTE: after creation of any sitemap it is crucial that the corresponding protein, ligand and sitemap are not altered in any way. SimSite3D searches results depend on these files being unmodified, and they can be used to rank hits using an external protein-ligand scoring method such as DrugScore. Each sitemap labels file only contains the names of the protein file (and possibly ligand file) used to generate the sitemap, and the directory information is provided at search time either through the command line, SimSite3D environment variables, or from settings in an simsite3d.conf file.

Sitemap generation command and arguments for one sitemap:

The intended use of gen_points is to generate query sitemaps or temporary sitemaps. To generate a large number of sitemaps, we recommend using the python script, auto_gen_sitemaps.py. To simplify file management, the recommended location to create query and temporary sitemaps is the directory from which you intend to run searches.

The supported methods of defining the sitemap volumes of interest are by a ligand volume bounding box or by a sphere. Note, the bounding box can have a large volume, because it uses the minimum and maximum coordinates of the ligand in the mol2 file, which are generally do not align well with the major and minor axes of the ligand.

To generate a sitemap using a ligand bounding box:

```
gen_points -p <XXXXXX_p.pdb> -l <XXXXXX_l.mol2> [<XXXXXX_s.csv>]

Ex: gen_points -p 1eqm_ADP_p.pdb -l 1eqm_ADP_l.mol2 [1eqm_ADP_s.csv]
```

In the above example, the program will first look for 1eqm_ADP_p.pdb and 1eqm_ADP_l.mol2 in your current working directory. If a file is not found locally, gen_points will look for the protein in the **$SIMSITE3D_DBASE_PROTS** directory and the ligand in the **$SIMSITE3D_DBASE_LIGS** directory. If no output path is given, the generated sitemap points file, labels file, atoms file and rad file will be written to the **$SIMSITE3D_OUPUT_DIR** in the file named XXXXXX_s.pdb, XXXXXX_s.csv, XXXXXX_a.pdb, XXXXXX_rad.pdb respectively. Note: if given, the output file name must follow the XXXXXX_s.csv template.

In the current implementation, the sitemaps generated using a sphere to specify the pocket volume are supported only as query sitemaps. This assumption is based on the use of SimSite3D for pocket mining and the desire of not returning database hits without ligands in the query pocket. In addition, for database sitemaps, the need to determine the center and radius of a sphere for each binding site

To generate a sitemap using a spherical volume:

```
gen_points –p <XXXXXX_p.pdb> --sphere "<center x,y,z and radius>" \
      [<XXXXXX_s.csv>]

Ex: gen_points –p 1b38_atp_p.pdb --sphere "0.5 29 8 4" output_sitemap_s.csv
```

In this example, the generated sitemap volume is the sphere centered at (0.5, 29.0, 8.0) with radius 4.0Å, and the volume captures the interactions of the adenine binding pocket of cdk2 in the PDB structure 1b38. If no output sitemap file name is given, the computed sitemap files are written to the **$SIMSITE3D_OUTPUT_DIR** directory.

Example input and output files for gen_points to generate a sitemap using a ligand bounding box to define the sitemap volume may be found in **$SIMSITE3D_INSTALL_DIR**/examples/gen_sitemap/(in,out). Running the examples is recommended to test for a successful installation of SimSite3D.

Explicitly including bound water molecules in a query binding site:

By default, gen_points ignores all water molecules in the protein structure file. In the event that your binding site has one or more water molecules known to be conserved or essential for binding, the "--waters" flag may be used to specify a listing of the water molecules. The listed water molecules will be considered as part of the protein in terms of hydrogen bond acceptors and donors and in terms of protein volume.

```
gen_points –p <XXXXXX_p.pdb> -l <YYYYYY_l.mol2> --waters "<list of waters>" \
      <query_s.csv>

Ex: gen_points –p 1ere_est_p.pdb –l 1ere_est_l.mol2 --waters "55W 101 A64W" \
      1ere_est_s.csv
```

In this example, gen_points will consider the three water molecules (residue 55 with insertion code W and blank chain identifier; residue 101 with blank insertion code and blank chain identifier; and residue 64 from chain A and with insertion code W) as part of the protein. The effect is to remove the sitemap points corresponding to the protein's interactions with the listed water molecules (55W, 101 and A64W), add the sitemap points corresponding to the potential hydrogen bonding interactions of the listed water molecules and consider the listed water molecules as part of the protein surface. Typically, all of the specifed water residues would have the same format—the different formats are shown in the above example to display 3 likely ways that water residues are specified in PDB formatted files.

NOTE: gen_points infers the orientation of the water molecules from the location of the oxygen and hydrogen atoms. In other words, the included waters residues require positions for hydrogen atoms.

Explicitly including all metal atoms in a query binding site:

By default, gen_points ignores all metal atoms in the protein structure file. In the event that your pet binding site has one or more metal atoms known to be essential for binding, the --include_metals" flag may be used to have gen_points consider all metal atoms as part of the protein. If you desire to include only a subset of metal atoms near the binding site, please only include in the PDB file those metal atoms you wish to include.

```
gen_points –p <XXXXXX_p.pdb> -l <YYYYYY_l.mol2> --include_metals <query_s.csv>

Ex: gen_points –p 1ere_est_p.pdb –l 1ere_est_l.mol2 --include_metals \
      1ere_est_s.csv
```

In this example, gen_points will consider all metal atoms in the protein structure file and near the binding site as part of the protein.

Sitemap generation tool to create a database of searchable sitemaps:

```
auto_gen_sitemaps.py
```

The auto_gen_sitemaps.py script uses the **$SIMSITE3D_DBASE_PROTS** and **$SIMSITE3D_DBASE_LIGS** directories to generate one sitemap for each protein, ligand pair (including multiple ligands per protein structure file is not currently supported). The computed sitemaps represent the entire binding site of each ligand and are stored in the **$SIMSITE3D_DBASE_SITES** directory. To allow incremental builds of sitemap databases, auto_gen_sitemaps.py checks the **$SIMSITE3D_DBASE_SITES** directory and only builds sitemaps for those structures in **$SIMSITE3D_DBASE_PROTS** that do not have an existing sitemap in **$ASCBAS_DBASE_SITES**.

3 Comparing sitemaps

The SimSite3D tool search_sitemaps aligns and ranks database sitemaps (e.g., the sitemaps created for the Pfizer CSDB) according to their shape and chemistry match to a query sitemap. Search_sitemaps identifies an optimal (rigid) orientation for each database sitemap relative to the query sitemap, determines the fragment of the database ligand in the query pocket and outputs statistics about the dockings.

Docking: Sitemap matches are based upon testing each set of neighboring triplets of sitemap points in the database sitemap for the ability to match a triplet of sitemap points in the query, requiring that the two triplets (triangles) have similar side lengths (within a preset tolerance), matching (A, D, N, H) chemistry labels at all three points, and have similarly oriented polar groups (the dot product between the vectors representing the favored hydrogen-bonding direction for the two matched polar sitemap points needs to be positive). All such triangle matches (dockings) are tested between the two sitemaps.

Orientation/Alignment Scoring: For a pair of query and database sitemaps, each saved triangle match is scored according to shape and chemistry. Pseudo code of the scoring method for a query,database sitemap pair is:

```
• For each saved triangle match:
    1. Initialize the match print to be false (zero) for each query sitemap point
    2. Initialize the ligand fragment binary string to be true (1) for each ligand atom
    3. Transform the database sitemap into the reference frame of the query sitemap (the transformations are computed the docking step).
    4. For each point in the query sitemap:
        a. Find the closest database sitemap point to the given query sitemap point
        b. If the distance is greater than 1.5 angstroms, skip this query point (this query point does not have a corresponding point in the database sitemap)
        c. Else, if the two sitemaps points are labeled as hydrophobic, add 1 to the hydrophobic count, set the match print bit corresponding to the query hydrophobic point to true (1)
        d. Else, if one of the sitemap points is labeled hydrophobic and the other is a polar point, add 1 to the unmatched polar count
        e. Else, the matched points are both polar, and calculate the dot product of the two vectors representing the optimal direction of the covalent bond between a ligand donor atom and ligand hydrogen atom or the direction from a ligand acceptor atom and a lone pair of its electrons.
            o If the dot product is negative, skip this query point (this query point does not have a corresponding point in the database sitemap)
            o Else, if one of the points is D and the other is A, add the dot product of their vectors to the polar mismatch sum
            o Else, if both points are A or D, add the dot product to the A:A & D:D polar match sum (giving a weight between 0 and 1, according to the similarity of the vector alignment). Set the match print bit corresponding to the query's polar sitemap point to true (1).
            o Else, add the dot product to the doneptor polar match sum (giving a weight between 0 and 1, according to the similarity of the vector alignment). Set the match print bit corresponding to the query's polar sitemap point to true (1).
    5. The alignment score of the (query, database) pair for the current triangle match is determined by multiplying the sums by their respective weights, summing the weighted sums and adding a constant term. The sum weights and constant term were found by a least squares error fit to a training set.
    6. If the score is worse than the score cutoff and or is not in the top N scores (where N can be set by the user) for the current (query, database) pair, skip this alignment.
    7. Else, determine the part of the database ligand that is in the query pocket. If fewer than 5 ligand atoms are in the query pocket, skip this alignment
• Optional step: If the external protein-ligand scoring function is not "NONE", score each saved ligand fragment with its corresponding query protein using the user selected external scoring function.
```

Match prints:

Given a query sitemap with *n* interaction points, the query sitemap labels file will have *n* labels (the *i*th label corresponds to the *i*th point in the sitemap points file). In a similar manner, each hit (alignment or orientation) to the query sitemap that is listed in the results file has an associated match print or bit string of length *n* where the *i*th bit corresponds to the *i*th query sitemap point. If a match print bit is zero, the hit sitemap (for the given alignment) did not have a point in common with that query point. On the other hand, if a bit is one, the hit does have a corresponding and complementary point for that query point.

Ligand fragment binary string:

Given a hit between the query sitemap and a database sitemap, suppose the corresponding database ligand has *n* atoms. Then it is likely that the ligand fragment (corresponding to the hit) has fewer than *n* atoms. A binary string of length *n* is used to denote the presence or absence of the database ligand atoms in the ligand fragment. That is, if the *i*th ligand atom is present or absent from the ligand fragment, the *i*th position in the binary string will be 1 or 0 respectively.

Orientation score normalization:

To allow normalization of the score between matched sitemaps to indicate the score's significance, we have built a database of 140 structurally diverse proteins that have biologically relevant ligands bound. Sitemaps are generated from the ligand binding sites in these crystal structures chosen as the intersection between Nh3D high-resolution structures that represent a unique structural fold at the CATH topological domain level and also have a valid ligand in the sense of BindingMOAD; invalid ligands include peptides longer than 10 residues, crystallization additives (for that particular structure), and ligands that are covalently bound. This set of sitemaps allows SimSite3D to evaluate the distribution of similarity scores for a particular query when matched against a set "random" sitemaps. "Promiscuous" sitemaps with average shape and average polar and nonpolar atom distributions could match many sites in the diverse proteins' ligand binding sites, and this would be indicated by a significant spread (large standard deviation) relative to the mean score. Thus, before a query sitemap is used, its highest-scoring alignment is calculated for each of the 140 diverse sitemaps, and the mean μ_q and standard deviation σ_q of the scores are computed. Each similarity score *s* (see Scoring section, above) between the query and a database sitemap is then normalized to create the normalized score $S = (s - \mu_q)/\sigma_q$, indicating how many standard deviations this database entry scores relative to the mean in the normalization database. The default value of w_t is -1.5, where more negative indicates a better score. For example, -1 means a score that is 1 standard deviation better than the mean score. If no database sitemap matches meeting the threshold are identified (i.e., no matches appear in the .out file), the σ threshold could also be automatically adjusted to keep the top few matches. User feedback on this would be appreciated, since there are potential advantages and disadvantages.

Search results directory:

You may specify an output directory on the command line to search_sitemaps (the directory is created if the last entry in the path does not exist). If an output directory is specified on the command line, the results will be placed in that directory. Otherwise an output directory will be created in the **$SIMSITE3D_PROJ_OUTPUT** directory with name given by the query structure's name with "_results" appended to the name (e.g. for a query named `1ua2_A_atp_s.csv`, the output directory would be **$SIMSITE3D_PROJ_OUTPUT**`/1ua2_A_atp_results`). The results directory will hold a results file containing the results of the search (a .out file), a subdirectory containing the database ligand fragments and a subdirectory holding the full ligand corresponding to each hit in the results file. (Note: if the search was too stringent it is possible that the results file will not contain any hits and the ligand subdirectories will be empty).

The fragmenting of the ligands can be used as an indication of which portion of the ligand is likely to be complementary to the query pocket. This allows one to focus on the similarities between two binding sites without seeing the portions of the ligand which are a poor match. Thus, a ligand fragment is generated for each hit in the results file. The corresponding database ligand for each hit has the portions of the ligand which fall outside of the query pocket removed. The resulting fragments are written to the ligands subdirectory of the output directory with the tag "`_NNNNN_f`" inserted between the structure name and the `.mol2` extension. For example, if the database sitemap has the name "`1b38_atp_s.pdb`", the ligand fragment file corresponding to the second best SimSite3D hit (for `1b38_atp_s.pdb`) would have the name "`1b38_atp_00002_f.mol2`".

You may also view the entire ligand with respect to the query pocket. Thus, for each hit, the `ligands` subdirectory has the corresponding ligand written in `.mol2` format with the tag "`_NNNNN_l`". In the above example, in addition to the fragment file, the transformed full ligand file "`1b38_atp_00002_l.mol2`" could be found in the `moved_ligands` subdirectory (of the main results directory).

Search results data file (.out):

The first 20 or so lines of the results file represent a small comment header to help reproduce and/or compare results from one run to the next. Each line in the .out file that is not whitespace nor has the # or % character in the first column is a score record and contains the following information:

1) name of the ligand fragment file
2) normalized score of this orientation of the database sitemap with respect to the query (see Scoring section, above)
3) 3x3 rotation matrix written row-wise for operations on column vectors
4) 3x1 translation vector which is to be applied after the rotation
5) A binary string representing the match print of the hit sitemap to the query sitemap
6) A binary string denoting the database ligand atoms present in the ligand fragment
7) external score of database ligand, for this orientation, with respect to the query protein (if an external scoring method is not set or explicitly requested to be

omitted, this column will also be omitted). Please refer to section 4 for an explanation of external scoring functions.

If no hits were found, the results file will not have any score records. If up to **N** scores for each database sitemap are requested, the number of records for each database sitemap will depend on how many hits for that particular sitemap were better than the score threshold and will be at most **N**.

Examples showing search_sitemaps command line options

The C++ program that forms the basis for the comparison of a query sitemap versus one or more database sitemaps is called search_sitemaps. The useful command line parameters are explained and several examples command lines are described in this section. However, search_sitemaps is in development, and there are a number of command line parameters not described here but implemented in the code for development purposes. For best results, please ignore these unsupported parameters.

This software can be run in one of two modes. The first mode is to compare the query sitemap versus a second sitemap, where the query sitemap is specified first on the command line. The second mode is to compare the query sitemap against a database of sitemaps.

Comparing two sitemaps:

To compare a query sitemap versus an individual sitemap you can use the command:

```
search_sitemaps <query_s.csv> <second_sitemap_s.csv>
Ex: search_sitemaps 1osh_fex_s.csv 1ie9_vdx_s.csv
```

The results of the search, in the above example, may be found in the **$SIMSITE3D_PROJ_OUTPUT**/`1ie9_vdx_results` directory. Explicitly specifying a directory is covered in a following example.

Note: because the primary focus of SimSite3D is to identify ligands or ligand fragments that are predicted to be complementary to a query site, search_sitemaps requires that the second sitemap have an associated ligand. In addition, this allows protein-ligand scoring functions to be used as a secondary method to scoring SimSite3D hits.

Comparing a query sitemap against the standard database:

To search the "standard" database (the directory specified by the **$SIMSITE3D_DBASE_SITES** variable) for significant matches to a query sitemap:

```
search_sitemaps <query_s.csv>
Ex: search_sitemaps 1ere_est_s.csv
```

In this example, the sitemap 1ere_est_s.csv is queried against the sitemaps in **$SIMSITE3D_DBASE_SITES** to find sites similar to the estradiol binding site in the human estrogen receptor (PDB 1ere).

Comparing a query sitemap against a user specified database:

To search a user specified database for hits to the given query sitemap

```
search_sitemaps --dbase_sites /path/to/sitemaps --dbase_ligs /path/to/ligands \
      <query_s.csv>
```

```
Ex: search_sitemaps --dbase_sites ~/data/test/sites --dbase_ligs \
      ~/data/test/ligands 1ere_est_s.csv
```

In this example, the sitemap 1ere_est_s.csv is used to search for similar sitemaps in the directory ~/data/test/sites and the corresponding ligands (for the database sitemaps) are in ~/data/test/ligands.

Specifying the search results directory:

To explicitly specify a directory to hold the search results, use the --proj_output flag:

```
search_sitemaps --proj_output path/to/my/results <query_s.csv> \
      [second_sitemap_s.csv]
```

```
Ex: search_sitemaps --proj_output ~/my_results/1osh_fex_01012007 \
      1osh_fex_s.csv 1ie9_vdx_s.csv
```

For the given example, if the output directory does not exist, the directory `1osh_fex_01012007` will be created in `~/my_results/`. If the directory `~/my_results` does not exist, the search program will fail. The results file (`1osh_fex_to_1ie9_vdx.out`) will be placed in the `~/my_results/1osh_fex_01012007` directory and the `ligands` directories will also be in `~/my_results/1osh_fex_01012007`.

Specifying a name for the results file:

The explicit naming of the results file itself can be specified using the -o flag:

```
search_sitemaps -o path/to/my/results.out <query_s.csv> [second_sitemap_s.csv]
```

```
Ex: search_sitemaps –o ~/result_files/1osh_fex_01012007.out 1osh_fex_s.csv \
     1ie9_vdx_s.csv
```

The results are written to `~/result_files/1osh_fex_01012007.out`, and the ligands are written to the directory **$SIMSITE3D_PROJ_OUTPUT**`/1osh_fex_results`.

Specifying a name for the results file AND specifying a name for the results directory:

The two flags may be used together:

```
Ex: search_sitemaps –o ~/result_files/1osh_fex.out --proj_output \
      ~/result_ligs/1osh_fex_ligs 1osh_fex_s.csv 1ie9_vdx_s.csv
```

The effect is to write the results to `~/result_files/1osh_fex.out` and the ligands and ligand fragments subdirectories to `~/result_ligs/1osh_fex_ligs`.

Keeping multiple hits for each database sitemap (if they exist):

By default, only the best scoring (SimSite3D score) hit for each database sitemap is kept if its score is better than the score threshold. To save the top N hits for each database sitemap (each must have a score better than the score threshold)

```
search_sitemaps --keep_n_scores N <query_s.csv> [second_sitemap_s.csv]
```

```
Ex: search_sitemaps --keep_n_scores 5 1osh_fex_s.csv
```

The results file will contain up to 5 hits for each database sitemap that is similar to `1osh_fex_s.csv`.

Changing the score threshold:

The default score threshold is a normalized score of –1.5 or better. A normalized score of –1.5 for a query-database sitemap pair orientation implies that the given orientation has a raw score which is 1.5 standard deviations better than the mean of the query sitemaps score versus the 140 diverse sitemaps (used for normalization) , and is more stringent Conversely, if the small number of hits returned indicates the threshold of –1.5 is too strict for this query and database, the score threshold can be decreased to a number of magnitude less than -1.5. See the following example:

```
search_sitemaps --score_threshold +/-N.N <query_s.csv> [second_sitemap_s.csv]
Ex: search_sitemaps --score_threshold –0.75 1osh_fex_s.csv
```

The score threshold in this example is –0.75 instead of the default value of –1.5.

Changing the external protein-ligand scoring function:

The method of specifying external protein-ligand scoring functions is explained in more detail in the external scoring function section. The default value is the scoring function specified by the first scoring function line in the `ext_prot_lig_score_fcns.conf` file in **$SIMSITE3D_INSTALL_DIR**`/SimSite3DSoftParams`. In order to use a different (other than the default) external protein-ligand scoring function, the name of the scoring function (the first column for the desired entry) must be specified on the search_sitemaps command line

```
search_sitemaps --prot_lig_score SF_name <query_s.csv> [second_sitemap_s.csv]
Ex: search_sitemaps --prot_lig_score DrugScore_PAIRSURF 1osh_fex_s.csv
```

If there is a line in the ext_prot_lig_score_fcns.conf file with the name "DrugScore_PAIRSURF" in the first column, the first line with that name is used to define the external scoring function to use. If no such line is found in ext_prot_lig_score_fcns.conf, the search will fail. If you wish to search without using any protein-ligand scoring functions, you can use the name "NONE" as the parameter to the ExtScore flag. At the present, the default protein-ligand scoring function is NONE.

SFCscore supports scoring protein-ligand interactions using multiple scoring functions. SimSite3D supports this method of running SFCscore by appending additional fields to each column of the output file. As an example, if SFCscore is run with 3 scoring functions each score line in the SimSite3D output file will have 9 fields.

A search_sitemaps example:

Please see the corresponding input and output example files in **$SIMSITE3D_INSTALL_DIR**`/examples/search_sitemaps`. The **$SIMSITE3D_INSTALL_DIR**`/examples/search_sitemaps/out` directory contains an additional file "PAIR_10_1ua2_ATP_SimSite3D_frag_l.scr" which is a DrugScore 1.2 score file containing the score between the, query, ATP binding site of cdk2 (PDB code 1b38) and the database ligand ATP from cdk7 (PDB structure 1ua2) transformed to the coordinate system of the query binding site (ATP binding site of cdk2). This DrugScore example file is given to show the DrugScore format expected by SimSite3D.

Transforming the structures corresponding to database matches for a query sitemap into the query reference frame:

In addition to reporting the score for the best match(s) meeting the σ threshold between two sitemaps, the results file includes the transformation matrix and vector to move the database objects (sitemap points and protein and ligand atom coordinates) into the reference frame of the query. The transformation is defined by a rotation matrix $\boldsymbol{A}$ (written row-wise) and a translation (column) vector $\boldsymbol{t}$. To transform the coordinates $\boldsymbol{x}$ (e.g., the x y z coordinates of an atom) for a database object into the space of the query sitemap/protein/ligand coordinates, calculate $\boldsymbol{y} = \boldsymbol{xA} + \boldsymbol{b},$ where $\boldsymbol{b}$ is the transpose of $\boldsymbol{t}.$

4 External Protein-Ligand Scoring Functions

To aid in the evaluation of hits, an external protein-ligand scoring function can be used to give an initial approximation on the compatibility of the the database ligand fragment for the query binding site, considering the atomic interactions rather than sitemap (pharmacophore-like) correspondences.

Specifying Available Protein-Ligand Scoring Functions:

The interface to the protein-ligand scoring functions is via a comma separated text file (ext_prot_lig_score_fcns.conf) in the **$SIMSITE3D_INSTALL_DIR**/SimSite3DSoftParams directory. Each non comment line in the protein-ligand scoring functions file should have exactly 3 fields or it will be ignored. The fields are:

1. The first field is to provide a unique name to the scoring function as it should be specified on the command line to search_sitemaps. If two lines have the same value (name) in the first field, only the first line with that name will be used.
2. This field is used to specify (internally to SimSite3D) which type of scoring function will be called. SimSite3D 3.0 supports DrugScore and has an SFCscore interface that needs a bit more testing.
3. The last field holds the explicit command line to run the protein-ligand scoring function where the protein and ligand file names are replaced with the variables **$PROTEIN** and **$LIGAND** respectively. As an example, to run DrugScore in our lab we might use

   ```
   "/soft/linux/drugscore/bin/drugscore.lnx PAIR 1b38_atp_p.pdb
        1b38_atp_l.mol2"
   ```

   to score ATP with respect to cdk2 in the PDB structure 1b38. In this case, the command to place in this field for our DrugScore line would be

   ```
   "/soft/linux/drugscore/bin/drugscore.lnx PAIR $PROTEIN $LIGAND"
   ```

The first scoring function specified in the ext_prot_lig_score_fcns.conf file is taken to be the default external scoring function. Besides the supported scoring functions, there is a special scoring function with the name "NONE" in the first field; the "NONE" scoring function implies that no protein-ligand scoring function is to be used.

Supported Protein-Ligand Scoring Functions:

The following protein-ligand scoring functions are supported

1. SFCscore default protein-ligand atom contacts scoring.
2. DrugScore 1.2 PAIR.
3. NONE: This scoring function is a place holder used to indicate the desire to not use an external protein-ligand scoring function.

Adding New Protein-Ligand Scoring Functions:

Adding additional protein-ligand scoring functions requires parsing of the scoring function output files. To do so, add a line to the ext_prot_lig_score_fcns.conf file and write a C++ derived class of the C++ class ExternalScoringFunction that can determine the name of the score file (from the new protein-ligand scoring function) and parse the score file to obtain the score(s).

Notes about SFCscore:

Based on discussions between the Protein Structural Analysis and Design Lab at Michigan State and the Pocket Mining Group in CADD at Pfizer Ann Arbor, the “default” SFCscore .spf file included with SFCscore will result in SFCscore scoring protein-ligand interactions in approximately the same as manner as DrugScore 1.2 PAIR scores protein-ligand interations. In particular, both the default SFCscore and DrugScore 1.2 PAIR scoring functions consider protein-ligand contacts alone. One is not limited to using a specific .spf file with SimSite3D, and one method of supporting multiple SFCscore scoring methods would be to create a .spf file for each method. Note: at the present SimSite3D calls the scoring functions for each ligand fragment (rather than concatenating ligands in the same coordinate file).

Simultaneous SimSite3D searches

A number of external protein-ligand scoring function programs do not allow a user to provide a name for the output file(s). If multiple searches (search_sitemaps runs) are conducted simultaneously and such a protein-ligand scoring function is used, the search_sitemaps runs must be carried out with different **$SIMSITE3D_PROJ_OUTPUT** directories or the results might be undefined. As an example, DrugScore 1.2 uses the ligand filename as part of the DrugScore output filenames; if two or more searches are run at the same time and in the same directory the searches might overwrite or delete their neighbors’ output files.

5 SimSite3D file naming convention

The SimSite3D protein, ligand and sitemap file naming scheme is:

- XXXXXX_p.pdb for proteins (e.g., `HIV-pr_K41R_V64I_0200758_LIG_1_p.pdb`)
- XXXXXX_l.mol2 for ligands (e.g., `HIV-pr_K41R_V64I_0200758_LIG_1_l.mol2`)
- Sitemaps files generated by SimSite3D corresponding to the ligand binding site (`e.g., LIG_1`) in the given protein, where XXXXXX is the Pfizer crystal structure name

    - XXXXXX_s.csv is the sitemap labels file (e.g., `HIV-pr_K41R_V64I_0200758_LIG_1_s.csv)`
    - XXXXXX_s.pdb contains the sitemap interaction points
    - XXXXXX_a.pdb holds the atoms from XXXXXX_p.pdb which have a corresponding sitemap interaction point (in XXXXXX_s.pdb)
    - XXXXXX_rad.pdb contains those residues from the protein (XXXXXX_p.pdb) that have at least one heavy atom within 4.5 Å of any atom in the ligand (XXXXXX_l.mol2.)
- Ligand files generated by SimSite3D during a search have the following naming
    - XXXXXX_NNNNN_l.mol2 is the ligand from the database structure XXXXXX in the reference frame of the query pocket given by the Nth hit from XXXXXX's pocket to the query pocket; where NNNNN is N written as 5 digits (zero padded) (e.g, `HIV-pr_K41R_V64I_0200758_LIG_1_00012_f.mol2)`
    - XXXXXX_NNNNN_f.mol2 is the fragment(s) of the database ligand (from XXXXXX); that is the portion of XXXXXX_NNNNN_l.mol2 that "fit" in the query's pocket.

Appendix: Changes Between Versions

Version 3.0:

- Additional changes exist in the code to pursue a variety of ideas alternative methods. We are evaluating whether the computational cost incurred is reasonable with respect to the increase in accuracy, as well as tuning these methods. They include: MSMS surface complementarity assessment of binding sites; pose clustering to group transformations (dockings) for a given query/database sitemap pair that are so similar as to be reasonably represented by a single orientation (requires a bit of debugging and much tuning); energy scoring of matched hydrogen bonds; and projection of Hbond points to spherical caps, similar to the work of M. Rarey, B. Kramer, F. Lengauer, and G. Klebe (1996), "A Fast Flexible Docking Method Using an Incremental Construction Algorithm", *JMB*, 261:470-489.

- Surface complementarity prototype compares MSMS surfaces by 4 values. The first step is to align the database surface to the query surface using a transformation saved in the triangle matching step. Next, all database vertices that are farther than 1.0 Å from the query surface (triangle patch) are removed, and all faces (triangles) missing 1 or more vertices are removed from the database triangular mesh. The sum of the areas of the remaining database triangles is used to estimate the amount of complementary surface area between the two surfaces. The RMSE of the remaining database vertices to the query surface is used to estimate "goodness" of fit. The number of remaining database points gives a bit more coarse value than the estimate surface area of how "large" the matched surface patches are. The number of remaining faces in the database mesh as opposed to the number of points in the database mesh give an indication of how fragmented the matched surface patches are.

- The matching of points to spherical caps is similar to that in the 1996 Flexx paper, but we are trying to line up query protein donors and acceptors to database protein donors and acceptors (in orientations similar to the query protein). The authors in Flexx used the spherical caps to determine if protein atoms could make hbonds with ligand atoms.

- Updated the default scoring method to use the weights attained in the latest round of training and testing, corresponding to the addition of hydrophobic spheres, which changed the relative sampling and weighting of hydrophobic versus polar matches.
- Added the ability to generate random perturbations (alignments) from the original orientation of the target sitemap (uses a poisson-like distribution rather than using a strictly uniform distribution). Using a poisson-like distribution helps to generate samples that are some distance from each other to help sample the entire space without using as large a number of samples as required if sampling from a uniform distribution.
- Added the ability to score sitemaps as given (i.e. as aligned by an outside method).
- Additional scoring methods -- energy-based scoring of matched H-bonding groups -- needs better placement of probe points before being pursued further
- Hydrophobic shells around hydrophobic atoms added as a way to more thoroughly sample hydrophobic surface
- Fixed handling of HETATMs to be a bit more standard
- Updated the python class utils.py to reflect the current environment variable processing
- The method of BKP Horn is used to determine the transformation (quaternion + translation vector) for the 3 pt alignments of sitemaps. This eliminates the need to support a routine to find eigen values/vectors in the case of 3 pt alignments.
- The Timer class now uses sigaction() in the place of signal(). The main reason for the change is sigaction is the preferred method of signal handling and is more portable from the Unix perspective.

Version 2.7.2:

- Cleaned up misc. files that were no longer used.
- Added support for metals in sitemaps
- Repaired issue with fragmenting mol2 files containing multiple substructures.
- Atoms which were labeled as HETATMs in the input PDB files should now be correctly labeled as HETATMs in any output PDB files.
- Having the names of runtime variables in the config files not prefixed with SIMSITE3D_ was deemed a possible place of confusion. The example config files and parsing was updated accordingly.
- Blank chainIDs and insertion codes are omitted (not represented by a character) from the label in the sitemap labels file. If an insertion code is not blank, it is appended to the residue number; if a chainID is blank, "__" appears instead of "_C_", where C is given as an example of a nonblank chainID.

- Added support for water molecules to be considered as part of the protein when generating a sitemap.
- Adjusted the scoring functions templates {do you mean parsing here, or how the scoring function lines are specified by the user?}
- Template points now have a unique label in the corresponding csv file.
- Added a field to the output for a hit. Field 6 now contains the ligand fragment binary string indicating which atoms in the ligand are/are not part of the fragment

Version 2.7.1:

- Fixed a copy-n-paste bug in method used to load environment variables
- Programs no longer support loading of a conf file from command line
- conf files are no longer required
- system conf file is now /etc/simsite3d/simsite3d.conf
- local conf file is now ${HOME}/.simsite3d/simsite3d.conf
- Write Ligand id as mol2 Molecule Name: mol2File::write() now puts the "struct_id" + _NNNNN as the molecule name (instead of "*****"). mol2File::write() now places the atom-type + orbit in a 6 char column.
- Allow mol2 Files to be in dos Format: We cannot assume that the input files (structures, ligands, etc) will be created only on *nix platforms. As such, the mol2File class had to be modified to ignore the '\r' Windows includes to denote the end of a line.
- General Handling of Runtime Environment: The environment is handled in a somewhat general fashion so that it is relatively easy to add or change environment and configuration variables. Of course, this has nothing to do with the actual implementation of the ideas the variables represent. Added conf file, environment variables and conf file on cmdline
- Updated Cmdline Flags: Renamed some of the SimSite3D Software cmdline parameters (flags) to better fit the updated cmdline options
- Ligand Fragmenting Update: Ligand fragments with few than N heavy atoms are discarded. If no ligand fragments remain, the alignment is discarded. Looking at a radius of 1.5Å around template points shows that (especially for pockets) a number of small holes exist in the "interior" of the template volume. Bumping up to 2.0Å seems to fill in the holes and should not increase the volume much. However, this method is relatively costly and is not ideal. If we could use an ellipse or set of bounding boxes to represent the template volume, it is likely we would be better off.
- Sitemap Atoms File: fixed bug that was putting additional atoms in the sitemap atoms file.
- Normalization Dataset: added environment variables (and the associated conf entries) for the diverse dataset and ligands.

- Update Filenames in Sitemap Labels Files: sitemap .csv files only have the name of the files and not the path to the ligand and protein files. In addition, we assume that all of the sitemap files are in the same directory. Given that assumption we do not need to write the _s.pdb and _a.pdb and _rad.pdb in the XXXXXX_s.csv file.
- Repaired Installation Script: repaired the install.py script so that it does not install the auto_gen_sitemaps.pl script, does not install the diverse sitemaps and does not blow away the python and examples directories if it is installing to the source dir.
- External Prot-Lig SF Changes: changed the way that the external scoring functions are called. The explicit command line is listed in external_scoring_functions.txt with **$PROTEIN** and **$LIGAND** denoting where SimSite3D will stuff in the chosen protein and ligand files.

Version 2.7.0:

- Sitemap Volume: Ligand hydrogen atoms are ignored when using a ligand to define the volume of a sitemap.
- Match Prints: Given the sitemap points of a query binding pocket, each hit has an associated binary string (in the score file). A 1 in a position indicates the hit has a similar interaction point which corresponding to the given query point. The match prints can be used as another ranking method besides the SimSite3D score for the hit. In particular, hierarchical clustering techniques can use the match prints to generate a dendogram of the hits based on the satisfied sitemap points of the query pocket.
- Sitemap Files: The sitemaps now include the protein atoms making the interactions associated with the sitemap points and labels for the sitemap points. This additional information is placed into 2 additional files (for a total of 3 files per sitemap).
    - The "original" sitemap file is relatively unchanged from previous versions of SimSite3D. This file is denoted as the "sitemap points" file and is in PDB format.
    - The "sitemap labels" file contains additional information about the sitemap and is used to tie the points file to the atoms file. The main use for this file outside of SimSite3D is to provide the labels given to the sitemap points. More information in stored in this file and depending on development needs, this file is the one which is mostly likely to change in the future.
    - The "sitemap atoms" file contains those protein atoms that gave rise to the sitemap points and is a subset of the protein PDB file.
- Ligand Files: The ligand fragment file associated with a hit is now generated by keeping only those ligand atoms which are within interaction distance of a query pocket protein atom giving rise a query sitemap point and do not have significant overlap with any protein atom.
- Working/Output Directory: At the present, the Linux distributions used to test SimSite3D require execute permissions on a particular in order to "cd" into that directory. In addition, SimSite3D makes use of several temporary files that need read/write access. An environment variable and a command line parameter were added

to direct SimSite3D as to which directory to build the results directory and to create/delete temporary files.

- The examples directory is now recursively copied (by install.py) to **$SIMSITE3D_INSTALL_DIR** so that the installation files of SimSite3D need not be accessible by users.

Version 2.6.1:

- External Scoring Functions: The options column in external_scoring_fuctions.txt was split into two columns. Column 3 is for those options which must or may precede the protein and ligand on the command line for the external scoring function. Column 6 is for the options which must or may follow the protein and ligand on the command line (e.g. SFCscore requires that the user given .sfc file be specified as the last option to SFCscore. In the case of SFCscore, the path to the .sfc file should be the last item in column 6 of the SFCscore line in external_scoring_functions.txt).

Version 2.6:

- Ligand Fragmentation: The method to handle the fragmentation of ligands has changed from considering flexible bonds as cleavable to a brute-force ring search. The current handling is to keep all rings for which at least one bond of the ring should be kept. The ligand fragmentation method is a preliminary one which is based on relatively loose constraints used to determine if a ligand atom is in the query pocket.
- External Scoring Functions: To be able to handle a variety of external protein-ligand scoring functions or none at all, the method has been generalized a bit. For an explanation please see the section on external scoring functions.
- Sitemap Headers: Sitemaps need not be regenerated, but to allow for comments which are compatible with those expected by tools such as MovIt, the comments have been changed from using the '%' and '#' characters to "REMARK 10".
- Normalization of Query Sitemaps: In the interest of keeping the number of steps for a sitemap search to a minimum, the query sitemaps are normalized at creation time and the statistics are saved in the generated sitemap file. Because of the current computational load to normalize a sitemap, the perl script (auto_gen_sitemaps.pl) to generate the database sitemaps does not normalize the database sitemaps.
- C++ Program Defaults: At the request of some, the C++ programs now have default values for most of the command line parameters. Adding the defaults, allows the use of C++ programs interchangeably with the Perl wrapper scripts and allows for more flexibility in running a search. Ideally, the more obscure parameters and those which modify the sitemap parameters should be left to their default values. Those parameters which are deemed the most useful to change are explained in detail in the sections describing the C++ program search_sitemaps.

Version 2.5:

- Ligand fragmentation: for each hit to a given query, SimSite3D will write out 2 .mol2 files. One file contains the ligand, corresponding to the given hit, as it exists in the database, but with a rigid transformation applied to move the ligand to the coordinate space of the query pocket (sitemap). The second file contains the subset of atoms and bonds representing the portion(s) of the ligand hit that fall inside the query pocket[2]. This fragmentation is an initial implementation and is not expected to be ideal in every case. In particular, the ligand can only be cut at flexible points. Thus, rigid fragments of ligands are included or excluded on an all or nothing basis (e.g. for a phenyl ring the entire ring is included if any one of the atoms falls inside the query pocket).
- DrugScore: after an SimSite3D search, each hit is given a DrugScore based on the database ligand fragment (for that hit) and the query protein structure. The DrugScore field gives another method of ranking the order of the hits (besides the SimSite3D orientation score).
- Search results: to help organize the results of a search, SimSite3D will create a subdirectory in the directory where the search was executed. This subdirectory will hold the results (.out file) of the search and a directory holding the ligand fragments.
- Results file: A header was added to help reproduce runs when developing and to help compare results across different version of SimSite3D. Each result (line) has an additional field for the DrugScore (or empty if DrugScore is not used) for the database ligand fragment in the query pocket. At present, the profile time field has been dropped.
- Ligand fragments directory: For each line in the results file there are 2 .mol2 files written to the ligands subdirectory of the results directory (for more detail please refer to the ligand fragmentation section above).
- Sitemaps: the sitemap files have been added a header listing the parameters which were used to generate them. This header allows the developers and testers to compare sitemaps between the different versions of SimSite3D.

[2] The volume of a query pocket is defined as a ball. The center of the ball is given by the average position or centroid of the query sitemap. The radius of the ball is taken to be the distance from the centroid to the point of the sitemap which is farthest away from the centroid plus a buffer of 0.5 Å.