

GraSPI

Graph-based Structure-Property Interrogator

Last updated on June 2020

Motivation: molecular vocabulary large pool of descriptors/features exists

Dragon molecular descriptor list: 51 pages with 4885 descriptors

List of molecular descriptors calculated by Dragon

ID	Name	Description	Block
1	MW	molecular weight	Constitutional indices
2	AMW	average molecular weight	Constitutional indices
3	Sv	sum of atomic van der Waals volumes (scaled on Carbon atom)	Constitutional indices
4	Se	sum of atomic Sanderson electronegativities (scaled on Carbon atom)	Constitutional indices
44	nCIC	number of rings (cyclomatic number)	Ring descriptors
45	nCIR	number of circuits	Ring descriptors
46	TRS	total ring size	Ring descriptors
47	Rperim	ring perimeter	Ring descriptors
94	DBI	Dragon branching index	Topological indices
95	SNar	Narumi simple topological index (log function)	Topological indices
96	HNar	Narumi harmonic topological index	Topological indices
97	GNar	Narumi geometric topological index	Topological indices
98	Xt	total structure connectivity index	Topological indices
99	Dz	Pogliani index	Topological indices
100	Ram	ramification index	Topological indices
101	BLI	Kier benzene-likeness index	Topological indices
173	MPC04	molecular path count of order 4	Walk and path counts
174	MPC05	molecular path count of order 5	Walk and path counts
175	MPC06	molecular path count of order 6	Walk and path counts
176	MPC07	molecular path count of order 7	Walk and path counts
197	X0	connectivity index of order 0	Connectivity indices
198	X1	connectivity index of order 1 (Randic connectivity index)	Connectivity indices
199	X2	connectivity index of order 2	Connectivity indices
200	X3	connectivity index of order 3	Connectivity indices
832	ATS1m	Broto-Moreau autocorrelation of lag 1 (log function) weighted by mass	2D autocorrelations
833	ATS2m	Broto-Moreau autocorrelation of lag 2 (log function) weighted by mass	2D autocorrelations

Do we need so many?

- Data automation lowers energy barriers

Can we find the smallest set of features

- Feature engineering and data analytics enables the reduction

Descriptors, features, salient/hidden variables

In machine learning, a **feature** is an individual measurable property or characteristic of a phenomenon being observed. Features are usually numeric (but also strings and graphs).

Data win over model sophistication:

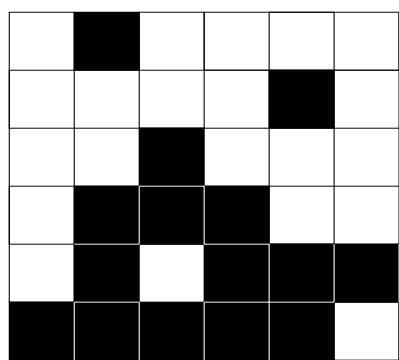
- The More Features, The Merrier
- Feature Man-Months Aren't Mythical

Watson (DeepQA, Jeopardy): 20 engineers and researchers +
3 years of work.

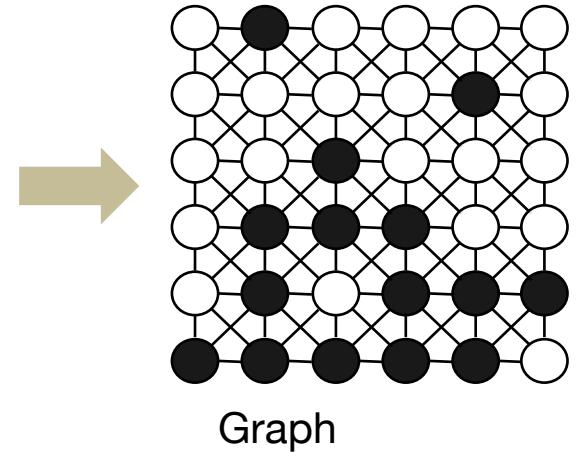
Features may seem quite ordinary ingredients of the pipeline, but writing features can be extremely painful.

We propose a two-level representation

Level 1: voxel-based representation

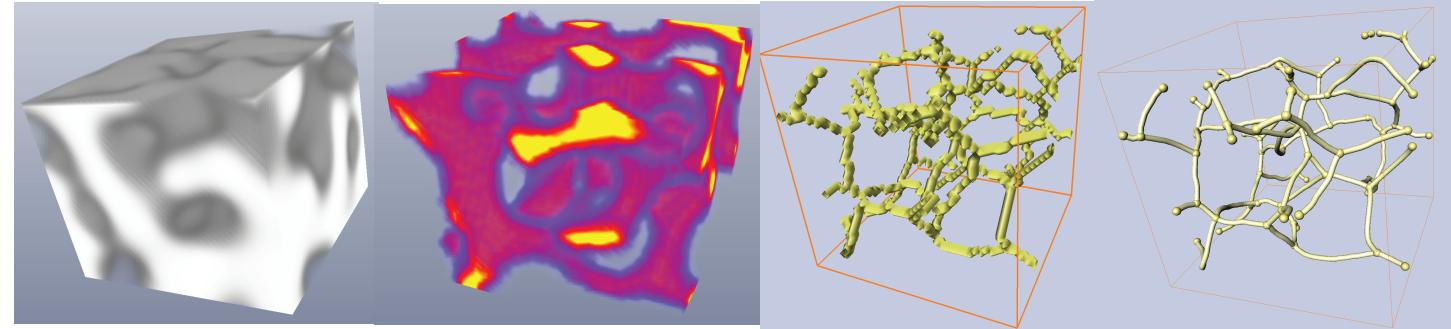


Microstructure



Graph

Level 2: topology preserving representation



Microstructure

Distance map

Medial axis

Graph

Steps involved:

- For each voxel in the microstructure image create vertex in graph
- Connect vertices in the graph following neighborhood of the image (first and second order neighbors)

Steps involved:

- calculate distance field to the interface
- Remove voxel based on the distance field to get medial axis
- Convert the medial axis into graph representation

GraSPI: Graph-based Structure descriptor Interrogator

What is GraSPI?

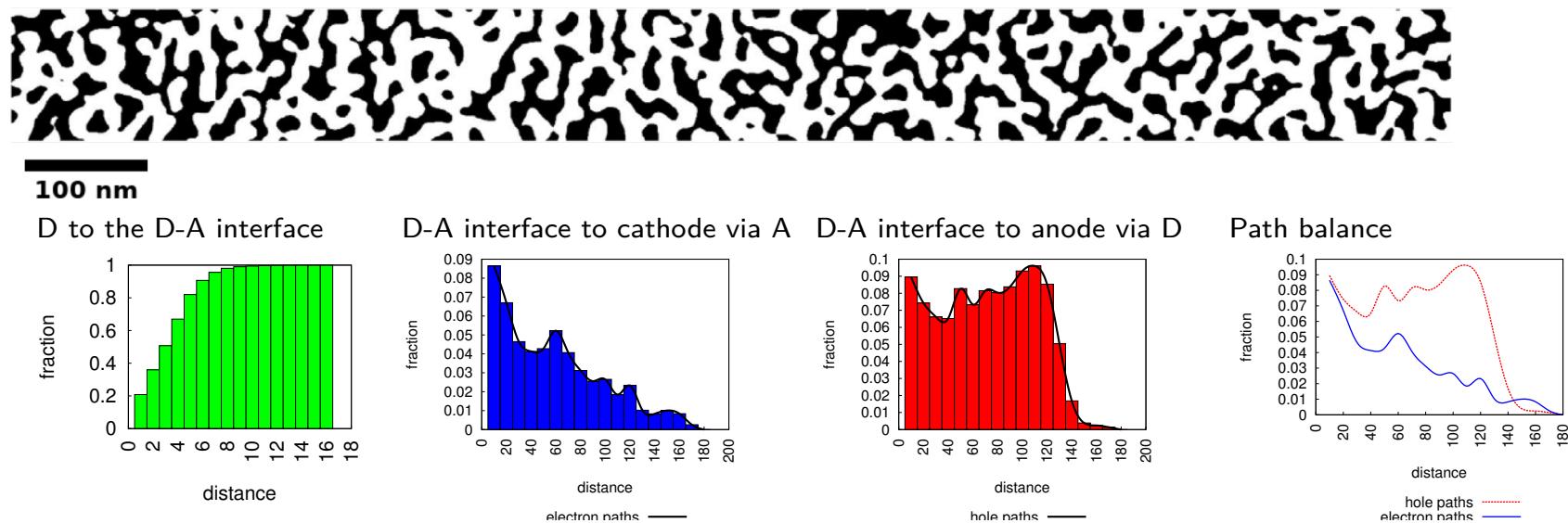
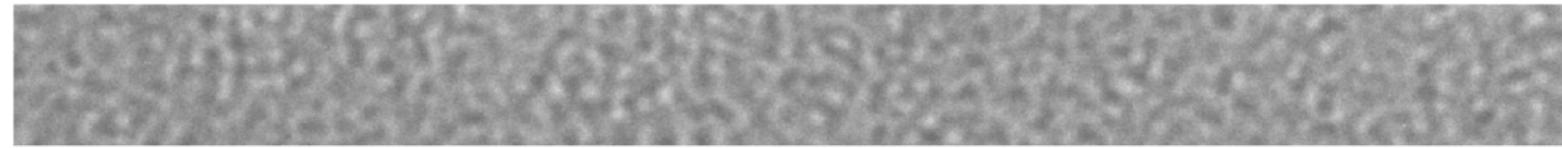
- GraSPI annotates microstructure with the set of descriptors/features.
- Given a segmented micrograph (black and white image), the set of descriptors is calculated.
- The set of descriptors captures size, shape and topological characteristics of the morphology.
- The set of descriptors has been motivated by the application for organic solar cells but can be generalized for other applications
- To enable the generalization, we plan to group the descriptors into mathematical descriptors and physics-encoded descriptors.

What problem GraSPI solves?

- As of now, given the segmented image, compute set of descriptors (scalars, histograms) and physics-weighted descriptors (scalars and histograms)
- In the future, expand the set of descriptors to include more descriptors, expand the representation to include the skeletons (topological features), expand the physics-based weighting functions, add representation-specific distance measures (beyond the Euclidean distance) and couple it with dimensionality reduction techniques.

From micrograph to set of statistics and descriptors

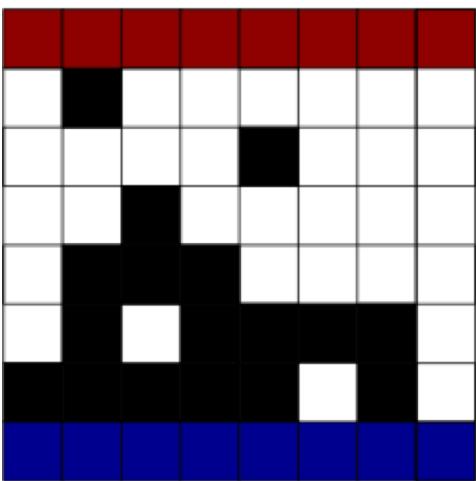
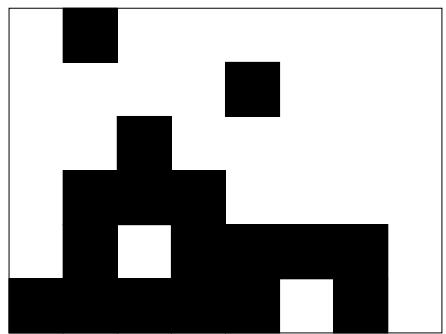
Micrograph



Set of descriptors
/features

- ① Fraction of light absorbing material: **0.583**.
- ② Fraction of photoactive material whose distance to the interface is within exciton diffusion length ($d < 10\text{nm}$): **0.994**.
- ③ Fraction of donor domains connected to anode: **0.97**.
- ④ Fraction of acceptor domains connected to cathode: **0.54**.
- ⑤ Distance from interface to anode via donor: $\mu = 67.82 \text{ nm} \sigma = 38.45 \text{ nm}$.
- ⑥ Distance from interface to cathode via acceptor: $\mu = 53.61 \text{ nm} \sigma = 39.38 \text{ nm}$.

How it works?



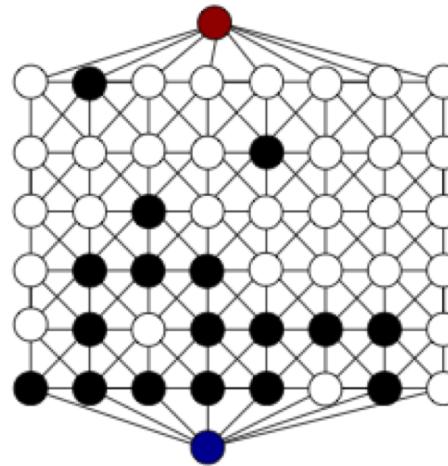
□ electron-accepting material

■ electron-donor material

■ anode

■ cathode

Represent image as graph to quickly calculate
the descriptors



○ electron-accepting material

● electron-donor material

● anode

● cathode

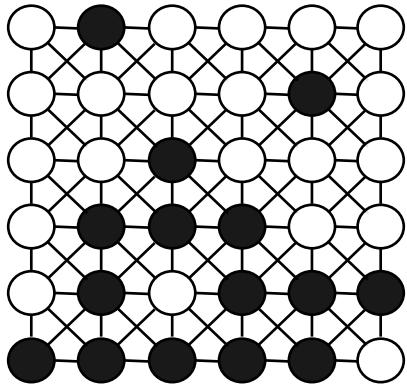
Descriptors:

- Fraction of Black and white vertices
- Fraction of Black vertices connected to Red (top electrode)
- Fraction of White vertices connected to Blue (bottom electrode)
- Fraction of the interface (Green) that has connection to both Red and Blue vertex
- ...
- Histogram of the shortest path from any Black vertex towards Red vertex
-
-
-

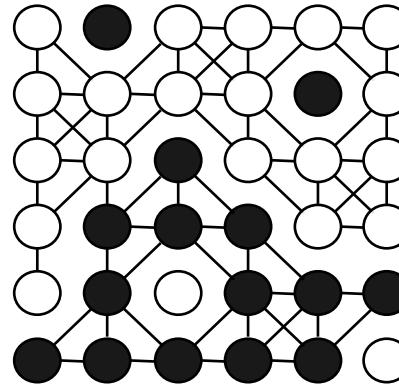
How it works

- Represent morphology as graph
- Add meta vertices to facilitate the computations of the descriptor (top/bottom surface, interface)
- Learn how to filter the graph to facilitate graph queries for various features
- Execute basic algorithms: e.g., compute connected components, compute the shortest paths

Example of graph queries using filtering



Filter graph



- ① Construct undirected graph $G = (V, E)$, where V is a set of vertices and E is a set of edges.
- ② Define function $c : V \rightarrow \{0, \dots, 3\}$, which assigns a color to each vertex in V , where:
 - 0 represents electron-donor material (black),
 - 1 represents electron-accepting material (white),
 - 2 represents cathode (blue),
 - 3 represents anode (red),
 - 4 represents interface (green).
- ③ Construct $G' = (V, E')$, where $E' = \{e = (u, v) \in E \mid c(u) = c(v)\}$, i.e. E' is a set of edges connecting vertices of the same color.
- ④ Identify connected components in G' .

```
1 class edge_same_color_predicate {
2 public:
3     edge_same_color_predicate() : G_(0), color_(0) { }
4     edge_same_color_predicate(const gt::graph_t& G, const std::vector<COLOR>& color)
5 : G_(&G), color_(&color) { }
6     bool operator()(const gt::edge_t& e) const {
7         return ((*color_)[boost::source(e, *G_)] == (*color_)[boost::target(e, *G_)]);
8     }
9 private:
10    const gt::graph_t* G_;
11    const std::vector<COLOR>* color_;
12};
```



```
1 void DetermineConnectedComponents( gt::graph_t* G, const std::vector<COLOR>& color,
2 std::vector<int>& components){
3     edge_same_color_predicate p( *G, color);
4     boost::filtered_graph<gt::graph_t, edge_same_color_predicate> FG(*G, p);
5     boost::connected_components(FG, &components[0]);
6 }
```

Another example of graph queries using filtering

Find distances from all BLACK vertices to the RED vertex (n), with the constrain that paths consist of BLACK nodes only:

- ① Construct $G'' = (V, E'')$, where
 $E'' = \{e = (u, v) \in E \mid ((c(u) = BLACK) \vee (c(u) = RED)) \wedge ((c(v) = BLACK) \vee (c(v) = RED))\}$, i.e. E'' is a set of edges connecting BLACK or RED vertices only.
- ② For vertex n (e.g. electrode) in G'' find all shortest paths with source in $n \rightarrow$ Dijkstra's algorithm.

```
1  class edge_color_BLACK_or_RED_predicate {
2  public:
3  edge_color_BLACK_or_RED_predicate() : G_(0), color_(0) { }
4  edge_color_BLACK_or_RED_predicate(const gt::graph_t& G, const std::vector<COLOR>& color)
5  : G_(&G), color_(&color) { }
6  bool operator()(const gt::edge_t& e) const {
7  if ( (*color_)[boost::source(e, *G_)] == RED ) && ( (*color_)[boost::target(e, *G_)] == BLACK )
8  return true;
9  if ( (*color_)[boost::target(e, *G_)] == RED ) && ( (*color_)[boost::source(e, *G_)] == BLACK )
10 return true;
11 return ((*color_)[boost::source(e, *G_)] == (*color_)[boost::target(e, *G_)]);
12 }
13 private:
14 const gt::graph_t* G_;
15 const std::vector<COLOR>* color_;
16 };

1 void FindDistancesToVertexN( gt::graph_t* G, const std::vector<COLOR>& color, gt::vertex_t n ){
2
3 edge_color_BLACK_or_RED_predicate pred( *G, color );
4 boost::filtered_graph<gt::graph_t, edge_color_BLACK_or_RED_predicate> FG(*G, pred);
5
6 std::vector<double> d(boost::num_vertices(*G));
7 std::vector<gt::vertex_t> p(boost::num_vertices(*G));
8 std::fill(d.begin(), d.end(), 0.0);
9 for (unsigned int i = 0; i < n; ++i) p[i] = i;
10
11 boost::dijkstra_shortest_paths(FG, top_electrode, boost::predecessor_map(&p[0]).distance_map(&d[0]));
12
13 }
```

Computational complexity

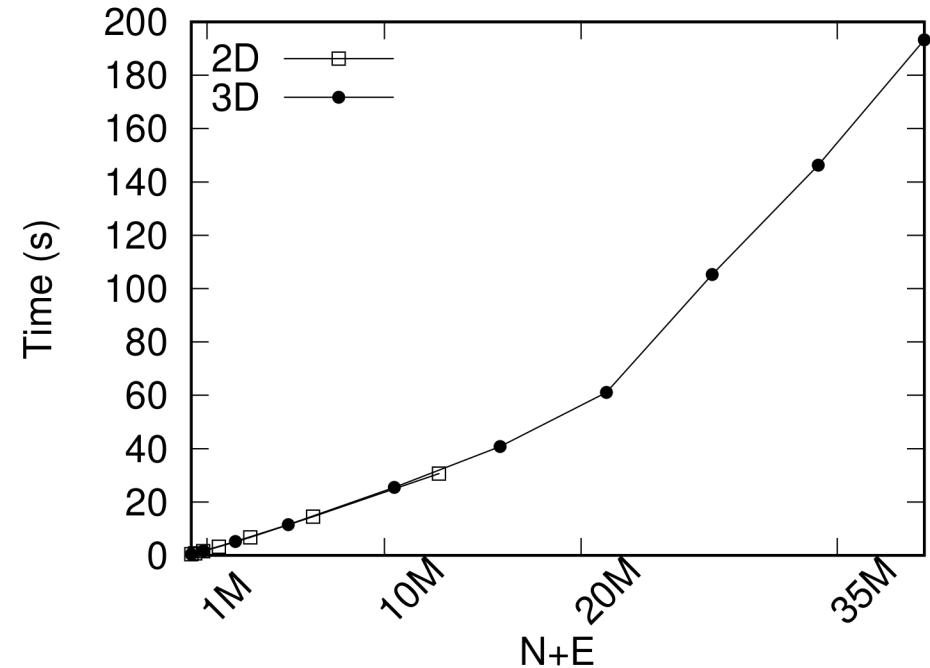
GraSPI computes the set of descriptors at the low cost:

2D examples – order of seconds

3D examples – order of minutes

Note linear complexity of the execution

The largest problem solved on the desktop machine (35M) under 3 minutes

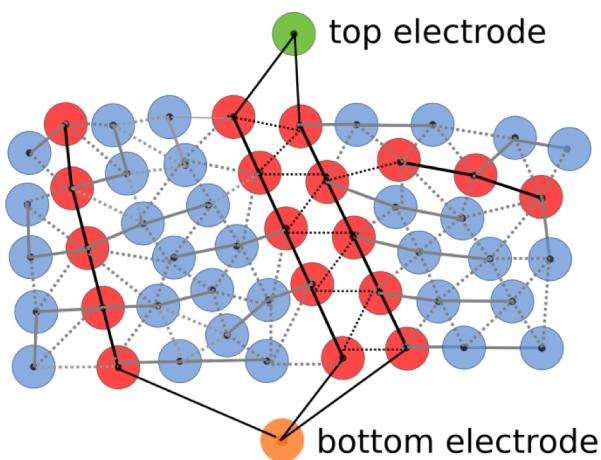
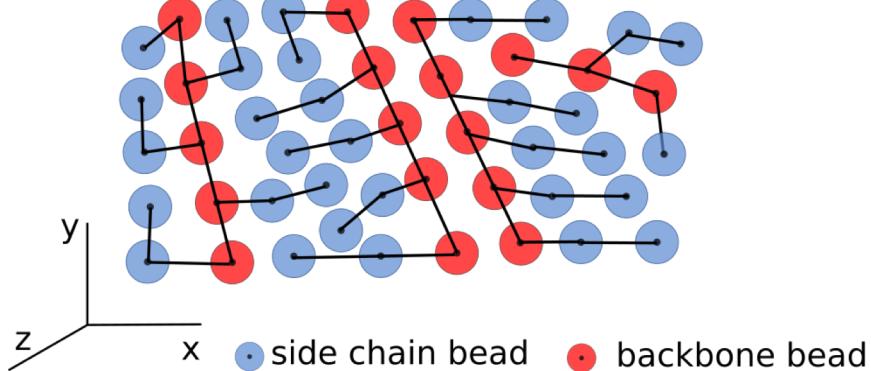


N- number of vertices
E- number of edges

Technical details: Coded in C/C++ using boost graph library

Important features

- Works for structured (micrograph, tomography data) and unstructured data (molecular dynamics, atom probe data)
- Graphs can be encoded to include information from other levels (e.g., DFT calculations)



— backbone edge
····· intrachain edge
— side chain edge
····· sidechain intrachain edge