■ OBJECT AND MATERIAL EIGENFUNCTIONS — Cont'd.

• Laplacian eigenfunctions and neural networks: ... Concerning this classification-adapted summarized version of a simulated annealing approach, crucial values influencing the algorithms behavior and output value(s) are the values of ΔW and of T. "Good values" for these parameters must also be determined experimentally. The second figure shown on the previous page (bottom-left) illustrates a viable method for executing simulated annealing with multiple initial values of the classification system's parameter. The allowable region in multi-dimensional domain space of the system's parameters could be populated with a set of initial values (= points in the multi-dimensional domain) that are uniformly distributed, "covering the allowable region." Thus, one can execute the simulated annealing algorithm for each of these initial W-values, which can be done in a parallel fashion. The result will be a corresponding output value(s) for each initial W-value. Using such a more exhaustive and computationally more expensive approach, one would define the W-value with the largest associated performance value $p(W)$ as the **(currently!)** best-possible parameter value setting for the system. **The optimization of W is supremely important, and it should be a continually ongoing process.** ...

■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks:...

EVOLUTIONARY ALGORITHMS.

(This class of algorithms is based on the concept of evolution as studied in biology. An evolutionary ALGORITHM is another cost function optimization method that attempts to determine a location(s) in the function's multi-dimensional domain where the function is (nearly) globally optimal.) Summarized at a high and general level, evolutionary algorithms operate on a "population" over time. Members of the "population," i.e., genotypical member characteristics, are "combined" and produce new members. Members have a (generally) finite life-span and are eliminated from the "population" when they have exhausted their life-span. Members are ranked / prioritized based on a "fitness" function (the cost function to be optimized), and members with better "fitness" values are granted higher probabilities for being selected for "combination" and production of new members.
In the context of optimizing classification system performance $p(\mathbb{W})$, the member "fitness function" is defined by and as $p(\mathbb{W})$. For example, a member could be understood as a $\mathbb{W}$-tuple, i.e., the tuple representing the system's parameter value setting.

...

OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

- Laplacian eigenfunctions and neural networks:... Often, an "encoding" is used to represent the (genotypical) characteristcs of a member, e.g., via a bit-string. In our application — near-optimal parameter value determination for a classification system — such an "encoding" does not seem to be necessary or desirable. Concerning our parameter value optimization objective, we can apply the typical operations — crossover, (re-)combination, mutation — directly to the W-tuples associated with a group of members (= a group of system parameter value settings). For classification performance function optimization, an evolutionary algorithm can be summarized at a high level as follows:

- define an initial population (>2 members);
- assign an age to each member;
- compute fitness (= p(W)) for each member;
- order the population members based on p(W);
- WHILE (p(W) "not good enough" AND
  no. members in population > 2 AND
  noIterations < maxIterations)
  - select member subsets, using fitness as probabilistic selection criterion;
  - combine the members in the subsets:

...

OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

- Laplacian eigenfunctions and neural networks: ...

  - combine the members in the subsets:
    - perform crossover combinations of characteristics of members in the selected subsets;
    - apply mutations to the characteristics of the new members obtained by crossover combinations;
  - compute fitness ($= p(W)$) for new members;
  - insert new members into ordered member list based on $p(W)$;
  - increment age of members;
  - FOR each member with age > maxAge
    - delete member from population;
- return member with "optimal" fitness, i.e., return this member's W and $p(W)$ values;

/* Note. One must consider also those */
/* members that are no longer in */
/* the last population, due to the */
/* enforcement of the maxAge limit. */

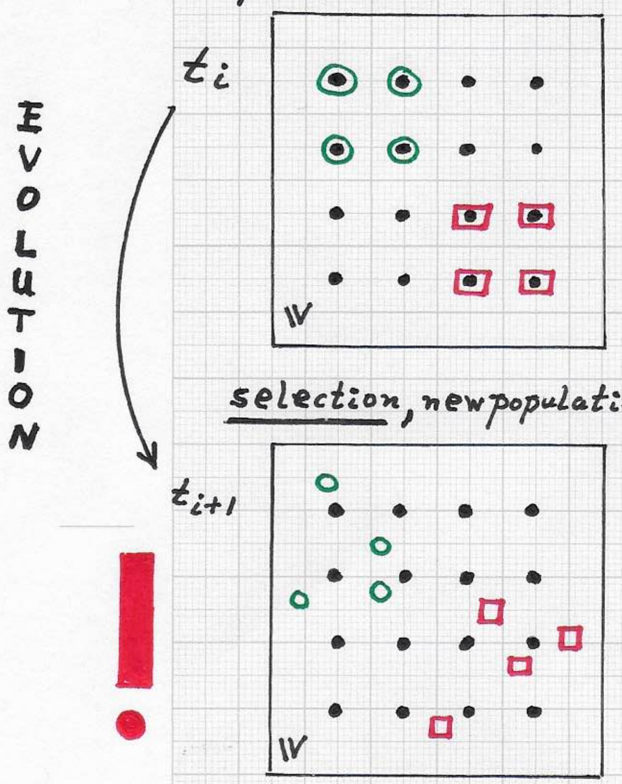
This high-level description captures the adaptation of the generic evolutionary algorithm to the material classification setting. In the following, we discuss important steps in more detail.

...

▪ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ...

parameter domain:



$\mathbb{V}_1 = (v_1^1, v_2^1, v_3^1, v_4^1)$

$\mathbb{V}_2 = (v_1^2, v_2^2, v_3^2, v_4^2)$

crossover:

$\bar{\mathbb{V}}_1 = (v_1^1, v_2^1, v_3^2, v_4^2)$

$\bar{\mathbb{V}}_2 = (v_1^2, v_2^2, v_3^1, v_4^1)$

mutation:

$\tilde{\mathbb{V}}_1 = (v_1^1 +/- \varepsilon, v_2^1, v_3^2, v_4^2)$

$\tilde{\mathbb{V}}_2 = (v_1^2, v_2^2 +/- \varepsilon, v_3^1, v_4^1)$

+ potential eliminations

The left figure illustrates a hypothetical transition of a "population" from time step t_i to t_{i+1}. The set $\{\bullet\}$ is the set of members at time t_i. Two subsets are selected from the eight fittest members, one four-member subset $\{\odot\}$ and one four-member subset $\{\boxdot\}$. These two subsets generate two new four-member sets, i.e., $\{\circ\}$ and $\{\square\}$, shown for time t_{i+1}. **It is possible to keep or eliminate the selected "old" fittest members used to generate the two new four-member sets.**

One can understand the figure as an **abstract** representation of the finite, multi-dimensional parameter space of the classification system ($\mathbb{V}$-space). The shown "populations" for t_i and t_{i+1} consist of **16** and **24** members ($\mathbb{V}$-tuples). The simple crossover-and-mutation example (left) shows how $\mathbb{V}_1$ and $\mathbb{V}_2$ could be used to generate two new members $\tilde{\mathbb{V}}_1$ and $\tilde{\mathbb{V}}_2$. ...