■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:...

The initial state of an evolutionary algorithm (its input) is an **initial population**. Regarding our classification application, we must optimize the classification system's (hyper-)parameter values. Thus, a randomly generated set of parameter w-tuples are used as initial population members. We generate **at least two members** and randomly assign an **age** value to each member (**between 0 and maxAge**). The initial members should "nicely cover" the bounded multi-dimensional w-space domain and ideally should be distributed nearly uniformly. The overarching optimization goal is the execution of the evolutionary algorithm such that the continually changing population of w-tuples moves to a location(s) where the system's performance function $p(w)$ is (nearly) optimal, i.e., maximal. The computationally most expensive calculation is the computation of the performance (=fitness) **$p(w)$ for each member** in the initial population. Despite the overall cost of an evolutionary algorithm, it is a global function optimization method we can use, since system performance optimization is a pre-processing step. **Fitness (= $p(w)$)** value computation for each initial member can be done in a parallelized fashion. ...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ... We **order the population members** based on their assigned p(IV) values, placing the fittest member at the top of the resulting ordered list. Next, we perform the actual evolution of the population via a **WHILE loop**, for example, that definitely terminates after at most **maxIterations** repetitions. This loop can also terminate when the fitness (p(IV) value) of at least one member is larger than a specified minimally "required" p(IV) "standard expectation", or when the size of the population has possibly gone down to less than two members. These last two requirements define the loop's Boolean expression **p(IV) "not good enough" AND no. members in population > 2**. In order to **select member subsets** for the generation of new members, one uses fitness as selection criterion. Current members are ordered in a list; thus, one can, for example, use the list order to assign linearly decreasing probability values (for being selected) to the members – with the member at the top of the list receiving the highest probability value. Alternatively, one can map a member's fitness (p(IV) value) directly to a p(IV)-value-specific selection probability.

...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:...

It is important to note that a subset of selected "very fit" members can have two, three or more members in it. Further, subsets can have different numbers of members in them. Once a certain percentage of the fittest members has been identified and subsets have been established, we can **combine the members** in a specific subset to generate new members (potentially having to satisfy certain minimal fitness requirements). "Combination of members" in our context refers to the random selection of IV-tuple component values – of IV-tuples associated with the subset members – and combining these values to generate a new IV-tuple, i.e., a new member. For example, if a subset had three members with associated IV-tuples $(v_1', v_2', v_3', v_4', v_5', v_6')$, $(v_1^2, v_2^2, v_3^2, v_4^2, v_5^2, v_6^2)$ and $(v_1^3, v_2^3, v_3^3, v_4^3, v_5^3, v_6^3)$, then they could produce the member $(v_1^2, v_2^2, v_3', v_4', v_5^3, v_6')$. (Such a combination is a generalization of the standard crossover-based combination where two members would be used to produce a new member.) When a **crossover combination is done**, we apply (any number of) **mutations** to the IV-tuple resulting from a combination. For example, we could apply three random mutations to $(v_1^2, v_2^2, v_3', v_4', v_5^3, v_6')$.

...

OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ... The result after mutation could be $(v_1^2 +/- \varepsilon, v_2^2, v_3', v_4' +/- \varepsilon, v_5^3, v_6' +/- \varepsilon)$. Of course, one must ensure that all w-tuple component values remain in their allowed value ranges when performing mutations. For each new member we compute its fitness (= p(w)) value. We can then insert new members into the ordered list of all current members. If we decide to use age as a criterion for upating the population, then we will have to increment the age of each population member as part of the WHILE loop that controls the population's evolution. Further, if age is considered and a member must be eliminated when its age attribute satisfies the condition age > maxAge, then we will also delete members from the list when they have reached their age limit, done as part of the WHILE loop.

• **Note.** In the context of our application and overarching goal – producing populations that over time continually improve their performance p, i.e., improve their ability to detect specific materials – it might not be desirable or advantageous to enforce an age limit. The w-tuples with the very best p(w)-values should NOT be eliminated from the population.

...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ... If one disscarded age as a member attribute, one could consider a variety of other options to control the evolution of a population. We describe some possibilities at a high level.

1) As long as a population of w-tuples is "relatively small" and the percentage of new members is also "relatively small," one can simply expand the initial population one-by-one, by always inserting a newly produced member into an unlimited list.

2) Enforce a fixed population size, using the size of the initial population as the needed size constant. When new members are produced, insert them into the list of all members. After each iteration of the WHILE loop controlling the evolution process, delete all members "at the bottom" of the list, i.e., those members that are in positions beyond the allowed list size.

3) Assuming that a WHILE loop iteration produces k new members, delete exactly k members from the member list – randomly or via some criteria.

** AT ALL TIMES, STORE THE BEST w-TUPLE **
** OBTAINED SO FAR, I.E., THE ONE WITH THE **
** BEST p(w) - VALUE. THIS w-TUPLE IS THE OUTPUT. **
...