Okay, based on the patterns in the past papers and the tutorial, here's a list of *new* MCQ-style facts and concepts. I've tried to deduce the kind of distinctions and specific details they might quiz you on, similar to what we've seen.

New MCQ-Style Facts & Concepts (Predicted based on Past Paper Patterns):

General Metaheuristics & Evolutionary Algorithms:

1. **Population Initialization in EAs:** While typically random, "sensible initialization" (e.g., using a heuristic to generate the first population) can be a valid strategy to potentially speed up convergence, but it's not strictly part of the core random EA definition. (Derived from 2023 Paper, Q8 options for EA population generation).
2. **Metaheuristic Characteristics:** They are strategies that guide the search, can escape local optima, are often stochastic, and generally non-problem-specific (though they can incorporate domain knowledge). (COS314 Metaheuristics_Notes.pdf, Section 1.2)
3. **Single-Point vs. Multi-Point Search:**
    - **Single-Point:** Iteratively improves a single solution (e.g., Iterated Local Search, Simulated Annealing).
    - **Multi-Point (Population-Based):** Works with a collection of solutions simultaneously (e.g., GA, GP, PSO, ACO).
4. **"No Free Lunch" Theorem (Implied Concept):** No single optimization algorithm is universally best for all problems. This is why different metaheuristics exist and why hybridization can be beneficial.

1. Genetic Algorithm (GA):

5. Chromosome Length: In a standard GA, chromosomes have a fixed length, unlike in Grammatical Evolution.

6. Elitism: A strategy where the best individual(s) from the current generation are guaranteed to be passed to the next generation, preventing the loss of good solutions.

7. Steady-State vs. Generational Replacement:

* Generational: The entire old population is replaced by the new offspring.

* Steady-State: Only a few individuals (often the worst) are replaced by new offspring in each generation, leading to more overlap between generations.

8. Handling Constraints: GAs may require penalty functions in the fitness evaluation or specialized operators to handle constrained optimization problems effectively.

2. Grammatical Evolution (GE):

9. Role of BNF Grammar: The BNF grammar in GE defines the structure and syntax of the programs that can be evolved, essentially constraining the search to syntactically correct programs.

10. Codon Interpretation: The interpretation of a codon (integer value) is context-dependent based on the current non-terminal being expanded and the number of production rules available for that non-terminal.

11. Wrapping's Purpose: Wrapping allows shorter chromosomes to potentially generate complex programs by reusing codons. However, excessive wrapping can indicate an inefficient mapping or a poor genotype.

3. Genetic Programming (GP):

12. Function Arity: The arity of a function in the GP function set specifies the number of

arguments (child nodes) that function takes. This is crucial for constructing valid syntax trees.
13. Protected Operations: Operations like "protected division" (e.g., returning 1 or a predefined constant on division by zero) are used to ensure the closure property and prevent runtime errors during fitness evaluation.
14. Mutation Depth (in GP): When performing mutations that grow a subtree (like point mutation replacing a node with a new subtree), a "mutation depth" parameter can limit the size of the newly generated subtree to control bloat.
15. Sufficiency vs. Closure Property:
* Sufficiency: The chosen terminal and function sets must be capable of representing a solution to the problem.
* Closure: All functions must be able to accept the output of any other function or terminal as input (often requires type consistency or protected operations).
4. Particle Swarm Optimization (PSO):
16. Inertia Weight (w): A high inertia weight encourages global exploration (particles tend to continue in their current direction), while a low inertia weight encourages local exploitation (particles are drawn more strongly to their pbest and gbest).
17. Neighborhood Topology (Besides Global Best):
* Local Best (lbest): Particles are influenced by the best particle in their defined local neighborhood, not just the single global best. This can help maintain diversity and prevent premature convergence to a single point.
18. Velocity Clamping: A mechanism to limit the maximum velocity of particles, preventing them from flying out of the search space or oscillating too wildly.
5. Ant Colony Optimisation (ACO):
19. Pheromone Evaporation (ρ):
* A high evaporation rate causes pheromone trails to diminish quickly, encouraging more exploration of new paths.
* A low evaporation rate allows trails to persist longer, leading to stronger exploitation of known good paths (but can risk premature convergence).
20. Heuristic Information (ηij): Provides problem-specific greedy information to guide ants, often an estimate of the "desirability" of a path component (e.g., inverse of distance/cost). It complements the dynamic pheromone information.
21. ACO vs. Greedy Search: ACO is not purely greedy because path selection is probabilistic, influenced by both pheromones (learned experience) and heuristic information (greedy choice). A purely greedy ant would always pick the path with the best heuristic value.
6. K-Nearest Neighbor (KNN):
22. Choice of K:
* A small K can make the model sensitive to noise (high variance).
* A large K can oversmooth the decision boundaries and misclassify (high bias).
* K is often chosen via cross-validation.
23. Curse of Dimensionality: In high-dimensional spaces, the concept of "nearness" becomes less meaningful as all points tend to be far apart and almost equidistant from each other, which can degrade KNN's performance.
24. Computational Cost: KNN's prediction phase can be slow for large datasets because it

needs to compute distances to all training points for each new prediction.
7. Hopfield Neural Network:
25. Storage Capacity: The Hopfield network has a limited storage capacity. If too many patterns are stored (typically around 0.14N, where N is the number of neurons), the network may fail to retrieve them correctly, leading to spurious states.
26. Energy Function: Hopfield networks have an associated energy function. During the recall process, the network dynamics tend to move towards states that are local minima of this energy function, which correspond to the stored patterns.
27. Synchronous vs. Asynchronous Update:
* Asynchronous (standard): Updates one unit at a time; guaranteed to converge to a stable state (local minimum of energy function).
* Synchronous: Updates all units simultaneously; may lead to oscillations between states instead of converging to a single stable state.
28. Role of Zero Diagonals (wii=0): Setting the diagonal weights to zero in the weight matrix (Hebb's rule) helps prevent neurons from self-exciting excessively and can improve the stability and recall of stored patterns.
These "new facts" are derived from the principles shown in your notes and the way questions in the past papers probe specific details and distinctions between algorithms or within components of a single algorithm. They focus on the *implications* of certain design choices or parameter settings.