Okay, I've reviewed the past papers you provided. Here's an updated set of exam notes, incorporating insights and areas of emphasis from these documents, with a particular focus on common question types and key distinctions highlighted in the papers.

Exam Notes: Metaheuristics and Machine Learning (Enhanced with Past Paper Insights)

1. Genetic Algorithm (GA)

- **Theoretical Knowledge (Points of Emphasis from Past Papers):**
  - **Search Space:** GAs search in the **solution space**, not the program space. This is a key distinction from GP and GE.
  - **Nature:** A GA can be described as a stochastic search method (though a question in the 2024 paper "A genetic algorithm is a stochastic hill-climbing search in which a large population of states is maintained" was marked, its correctness depends on the exact interpretation of "hill-climbing" in this context; GAs are more global search).
  - **Exploration vs. Exploitation:** Genetic operators are crucial for balancing this. **Crossover** is typically associated with **exploitation** (searching within the neighborhood of existing good solutions/genetic material) and **Mutation** with **exploration** (introducing new genetic material to search new areas).
- **Application Knowledge (from Past Papers):**
  - **Problem Example (Vehicle Routing Problem - 2022 Memo):**
    - **Chromosome Representation:** A sequence of supermarkets representing the route, starting and ending at the depot.
    - **Fitness Function:** A distance metric (e.g., total time or distance of the route) to be minimized.
    - **Selection:** Tournament selection is a common choice.
    - **Mutation:** Any method that consistently changes the sequence of supermarkets in an ordered manner (e.g., swapping pairs, multiple changes).
    - **Crossover:** Single-point crossover can be used, but care must be taken to avoid duplicate genes (supermarkets) in offspring, often requiring repair mechanisms or specialized crossover operators for permutation-based problems.
  - The 2024 paper asks to identify an AI approach that "mimics evolutionary processes to generate increasingly better solutions of a problem in a solution space," with "Genetic algorithm" as the correct answer.

2. Grammatical Evolution (GE)

- **Theoretical Knowledge (Points of Emphasis from Past Papers):**
  - **Representation:** Uses variable-length individuals (chromosomes) made up of

codons. It maps a genotype (binary string) to a phenotype (derivation tree/program) using a grammar.
    - **Search Space:** Searches in the **program space**.
    - **Genotype-to-Phenotype Mapping:**
        - The genotype (binary string) is interpreted as a series of codons (typically 8-bit integers).
        - Each codon's decimal value is used with a modulus operator against the number of available production rules for the current non-terminal in the BNF grammar to select a rule.
        - The derivation process expands non-terminals using selected rules until a complete program (phenotype) consisting only of terminals is formed.
        - "Wrapping" occurs if the end of the codon string is reached before the program is complete; the process reuses codons from the beginning of the string.
    - **Selection:** Based on fitness score. A 2024 question suggests "A combination of fitness and randomness" or "Fitness score only" were options, implying fitness is the primary driver.
- **Application Knowledge (from Past Papers):**
    - **Symbolic Regression (2023 & 2024 Papers):** GE can be used for symbolic regression, similar to GP, to find a mathematical expression.
    - **Derivation Process Example (2023 Paper):** A detailed question involved showing the step-by-step derivation of a phenotype z + x / z * x / z from a given genotype (list of codon values) and a BNF grammar. This highlights the importance of understanding the mapping mechanics.
    - A question in the 2023 paper (Question 21) asks to identify what is *not* a genetic programming process, with "production rule mapping" being the answer, implying it's specific to GE.

3. Genetic Programming (GP)

- **Theoretical Knowledge (Points of Emphasis from Past Papers):**
    - **Representation:** Traditionally uses syntax trees.
    - **Search Space:** Searches in the **program space**.
    - **Functionality:** Can evolve executable expressions and may perform feature selection automatically.
    - **Bloat:** A phenomenon associated with GP where programs grow excessively large without a corresponding improvement in fitness (2023 Paper).
    - **Tree Generation Methods:** Full, Grow, Ramped half-and-half are standard. "Graph tree growth" was listed as *not* a GP tree generation method in the 2023 paper.
    - When GP is configured to evolve decision trees, it acts as a **classification algorithm** (2023 Paper).
- **Application Knowledge (from Past Papers):**
    - **Symbolic Regression (2024 & 2023 Papers):**

        - **Function Set:** Contains operators like +, -, *, / (protected division). Powers like x2 or x3 can be formed by combinations (e.g., x∗x).
        - **Terminal Set:** Contains variables (e.g., x) and constants (e.g., a, b, c, d if they are part of the expression to be discovered). y (the dependent variable) should not be in the terminal set as GP evolves f(x).
        - **Fitness Function:** For regression, use error measures like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or Mean Absolute Error, calculated over all fitness cases (data points). The goal is to minimize this error.
    - **Distinction from ANNs for Classification (2022 Memo):** GP *is* a classification algorithm that evolves a classifier, while an ANN *can be configured as* a classifier. GP itself doesn't classify in the way an already-trained ANN does; it *produces* the classifier.

4. Particle Swarm Optimization (PSO)

- **Theoretical Knowledge (Points of Emphasis from Past Papers):**
    - **Suitability:** Well-suited for optimization problems with **continuous functions** (2024 Paper, Q2 - "Continuous and differentiable functions" was the selected answer, though "differentiable" is not strictly always required for PSO, continuous is key).
    - **Solution Representation:** Each solution is typically represented as a **particle**.
    - **Velocity Update:** Particles update their velocity based on their own **pbest (personal best)** and the **gbest (global best position)** found by the swarm.
- **Application Knowledge:**
    - Focus on understanding the core mechanism: particles moving in a search space, influenced by their own best-found positions and the overall best-found position in the swarm.

5. Ant Colony Optimization (ACO)

- **Theoretical Knowledge (Points of Emphasis from Past Papers):**
    - **Core Idea:** A probabilistic technique for solving problems reducible to finding good paths through graphs.
    - **Path Choice Influence:** The main factor influencing an ant's path choice is the **presence and intensity of pheromones on the path**.
    - **Probability Calculation:** Understanding how to calculate the probability of an ant choosing a specific path is crucial. This involves:
        - Pheromone value (τ) on the path.
        - Heuristic value (η) of the path (often 1/cost or 1/distance).
        - Parameters α and β controlling the influence of pheromone and heuristic information, respectively.
        - The probability of choosing path j from node i is: $Pij=\Sigma k(\tau ik)\alpha(\eta ik)\beta(\tau ij)\alpha(\eta ij)\beta$ for all feasible paths k.
        - The 2024 paper (Question 16) required calculating these probabilities for

different paths given costs, pheromone values, and $\alpha=\beta=1$.

- **Application Knowledge (from Past Papers):**
  - **Travelling Salesman Problem (TSP):** A classic problem that can be solved "seamlessly" using ACO (2024 Paper). ACO is well-suited for finding shortest paths in graphs.

6. K-Nearest Neighbor (KNN)

- **Theoretical Knowledge (Points of Emphasis from Past Papers):**
  - The 2024 paper (Question 7) involved a practical KNN-like clustering scenario (though it's essentially K-Means steps being tested). It requires:
    - Calculating distances (implicitly, to assign entities to the closest centroid).
    - Understanding how entities are assigned to clusters based on proximity to centroids.
    - Recalculating cluster mean values (centroids) after initial assignment.
    - Tracking which entities belong to which cluster after an iteration.
  - While the question tests K-Means, the underlying concept of assigning based on "nearness" is related to KNN.
- **Application Knowledge:**
  - Be prepared for questions that involve distance calculations and assignments based on those distances, especially in a clustering context which shares principles with KNN's neighbor identification.

7. Hopfield Neural Network

- **Theoretical Knowledge (Points of Emphasis from Past Papers):**
  - **Connectivity:** It is a **fully connected** network (not sparsely connected).
  - **Nature:** It is a **recurrent ANN** and performs **associative memory**.
  - **State Updates:** States in a Hopfield Network are updated **asynchronously** (one unit at a time, often randomly selected).
  - **Autoassociation Example (2022 Memo):** Given a pattern to store (e.g., [1010]), calculate the weight matrix. Then, given a new (potentially corrupted) input (e.g., [1110]), show the step-by-step update process for each unit (randomly selected) until the network converges to a stable state (hopefully the original stored pattern or the closest one).
    - Remember to convert binary patterns (0,1) to bipolar (-1,1) for weight matrix calculation if that's the convention used (e.g., $W=\Sigma(2p-1)(2p-1)T$ for binary, or $W=\Sigma ppT$ for bipolar, with $wii=0$). The 2022 memo implies converting to bipolar first.
- **Application Knowledge:**
  - The ability to manually calculate the weight matrix for a given pattern (or patterns).
  - The ability to step through the recall/convergence process for a given input and weight matrix, updating units one by one.

General Machine Learning & Neural Network Concepts (from Past Papers)

- **Overfitting (Large gap between training and test accuracy):**
    - Common methods to reduce this gap include **Dropout** (2024 Paper, Question 5). Other options often include regularization, getting more data, or early stopping. "Sigmoid activation" or "RMSprop optimizer" are not direct solutions for overfitting.
- **Linear Perceptron Limitation:**
    - A linear perceptron **cannot learn the OR function if "linear" implies no non-linear activation function** (like a step function) that can create a decision boundary. However, a simple perceptron *with* a step activation function *can* learn OR. The question (2024 Paper, Q4) "Which of the following cannot learn the OR function?" with options "A network of sigmoid neurons with one hidden layer," "A linear perceptron," "A single sigmoid neuron," and "None of the stated." If "linear perceptron" implies a purely linear output without a threshold/step activation, it would struggle. A single sigmoid neuron can learn OR. "None of the stated" was the selected answer, implying all listed options *can* learn OR. A single McCulloch-Pitts neuron or a single perceptron with a step/threshold activation can learn OR.
- **Perceptron Learning Rule (2022 & 2024 Papers):**
    - Be familiar with the basic perceptron learning rule for updating weights and bias: $w_i(new) = w_i(old) + \alpha(t - o)p_i$ and $b(new) = b(old) + \alpha(t - o)$, where t is target, o is output, p_i is input, and α is learning rate.
    - Questions involved finding the least values of $w1, w2, b$ for convergence for simple logic functions like AND or custom functions, given an activation rule (e.g., $f(n)=1$ if $n>\theta$, $f(n)=0$ if $n \leq \theta$).
- **Backpropagation (2023 Paper):**
    - A detailed question required calculating an updated weight ($w11$) in a 2-layer network using the generalized delta rule. This involves:
        - Feedforward pass: Calculate net inputs and activations for hidden and output layers.
        - Calculating error at the output layer ($\delta k = (target - actual) \cdot f'(net)$).
        - Calculating weight updates for the output layer weights ($\Delta w_{ik} = \alpha \delta_k \cdot \text{hidden\_activation}_i$).
        - (If updating hidden layer weights: Propagating error back to hidden layer $\delta i = (\Sigma k \delta k w i k) \cdot f'(neth)$).
        - The activation function was sigmoid $1/(1+e-n)$, so its derivative $f'(n) = f(n)(1 - f(n))$ is needed.
- **Classification vs. Regression:**
    - Predicting an exact temperature is **regression learning** (2024 Paper). Classifying emails as spam/not spam is classification.

- **Desirable ANN Characteristic: Generalization** is a key desirable characteristic of a trained ANN classifier (2022 Memo).
- **Multi-objective Optimization:**
    - **Pareto Front:** A set of optimal solutions where no objective can be improved without degrading at least one other objective (2023 Paper).

This enhanced set of notes should provide a more targeted preparation based on the types of questions and concepts emphasized in the past papers. Good luck with your exam!