The University of Nottingham Ningbo China

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, Spring SEMESTER 2018-2019

ARTIFICIAL INTELLIGENCE METHODS

Time allowed **TWO Hours**

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer ALL questions

Each question carries 25 marks

Only silent, self-contained calculators with a Single-Line Display are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn your examination paper over until instructed to do so

ADDITIONAL MATERIAL:
None.

INFORMATION FOR INVIGILATORS:
Please collect the exam papers at the end of the exam.

Question 1. [25 marks] Basics of Artificial Intelligence (AI) Methods.

(A) Which one of the following statements is **incorrect**?

(i) 1D knapsack problem is NP-Hard.

(ii) 1D bin packing problem is NP-Hard.

(iii) Maximum/largest saturation degree first heuristic is used to solve maximum flow problem.

(iv) Maximum/largest saturation degree first is a heuristic method that can be used to find a good quality solution for the graph coloring problem.

(5 marks)

(B) Which one of the following statements regarding linear programming is **incorrect**?

(i) All constraints in a linear programming model are linear.

(ii) The objective function is a linear combination of decision variables.

(iii) All the constraints must be linear, and the objective function must be either linear or quadratic.

(iv) The problem can be solved by the Simplex method efficiently.

(5 marks)

(C) Which one of the following statements about Simulated Annealing (SA) is **correct**?

(i) SA balances the exploitation and exploration by using a short-term memory.

(ii) When the temperature is reduced to zero, SA is degenerated into a simple first descent local search method.

(iii) The starting temperature of SA does not have impact on the performance of the solution.

(iv) SA is a deterministic approach and always returns the same results from multiple runs.

(5 marks)

(D) What are the main differences between strong Artificial Intelligence and weak Artificial Intelligence?

(4 marks)

(E) Name **three** major categories of artificial intelligence research and **two** topics in each category.

(6 marks)

Question 2. [25 marks] Local search (Hill climbing).

(A) List 5 design components of a local search.
(5 marks)

A simple and effective way to encode a solution for the bin packing problem is to use an **array-based assignment representation**.
Each item is assigned to a bin using an integer index.
For example, if we have s = [0, 1, 0, 2], it means:item 0 and 2 are placed in bin 0,item 1 is in bin 1,item 3 is in bin 2.
Why this encoding is suitable: Easy to implement and update during local search (e.g., move or swap items).Allows quick computation of bin load and total number of bins.Helps generate neighborhood solutions efficiently.Maintains a direct mapping from items to bins.

(B) Bin packing is a classic combinatorial optimisation problem that has practical applications in many fields. The one-dimensional bin packing problem aims to pack a set of ***n*** items of different sizes into minimum number of homogeneous fixed capacity bins. Answer the following questions regarding the implementation details of a simple local search algorithm for bin packing.

(i). Describe how a solution to a bin packing problem instance can be best encoded for the ease of implementation of a local search method and why.
(5 marks)

(ii). Suggest two neighbourhood operators for the local search method tackling this problem.
(4 marks)

(iii). Explain the drawbacks of simple local search methods to combinatorial optimization problems in general. How can those drawbacks be addressed, at least partially? (You may use bin packing problem as an example).
(6 marks)

(iv). Explain why best-fit heuristic cannot guarantee to solve the problem optimally.
(5 marks)

1. **Geometric Cooling Schedule**

$T_{k+1} = \alpha \cdot T_k \quad \text{where } \alpha \in (0, 1)$

This is the most widely used cooling schedule. Temperature decreases exponentially, offering a smooth and controlled decay.

2. **Non-linear Cooling Schedule**

$T_k = \frac{T_0}{(1 + \beta \cdot k^p)} \quad \text{for constants } \beta > 0, p \geq 1$

This method allows more flexible cooling rates. It is useful when fine control over convergence speed is needed and can avoid too fast cooling.

Question 3. [25 marks] Metaheuristics.

(A) Provide two most widely used cooling functions in simulated annealing approach.

(5 marks)

(B) A simulated annealing approach is adopted to solve a combinatorial optimisation problem. The experimental results show that, after a few iterations, virtually all new solutions are rejected and hence the search stagnates at the current solution in hand thereafter. Please give **two** possible reasons that could lead to this phenomenon. For each of them, how can the issue be addressed?

(6 marks)

(C) Explain the use of the tabu tenure parameter in a tabu search approach. What are factors that should be considered when setting this parameter?

(6 marks)

(D) Both simulated annealing (SA) and tabu search (TS) are single based point neighbourhood search methods for optimisation problems. Describe the mechanisms by which SA and TS escape from local optima. What are the main differences between SA and TS?

(8 marks)

Question 4. [25 marks] Metaheuristics and Hyper-heuristics.

p-median problem is a classic combinatorial optimisation problem with many real-world applications, for example, determining the optimal locations of public or industrial facilities (such as petrol stations). The problem can be formally defined as follows: consider a set I={1,…,*n*} of potential locations for *p* facilities of identical functions, and a set J={1,…*m*} of customers. All customers have a same priority and a same unit demand. Denote g_{ij} as the transportation cost of using a facility at node i ($i \in I$) to service customer j ($j \in J$). The problem is to determine the location of the *p* facilities in order to minimise the total transportation costs for satisfying the demands of all customers. Each customer is serviced by the closest open facility.

(A) Assume that a genetic algorithm is chosen to tackle this problem, what solution representation (or solution encoding scheme) should be used for this problem? Why?

(5 marks)

(B) Describe two different methods to generate the initial population.

(4 marks)

(C) Considering the solution representation that you adopted in (A), how would you design the crossover and mutation operators for the genetic algorithm? List and explain one crossover operator and one mutation operator.

(5 marks)

(D) Explain how the constraints (i.e. exactly *p* facilities are open) should be handled for this problem.

(5 marks)

(E) Considering the solution representation that you adopted in (A), what is the size of the search space?

(2 marks)

(F) Suppose you decide to use an iterative hyper-heuristic approach to solve this problem, give two possible low-level heuristics.

(4 marks)

> **Issue 1: Cooling is too fast (temperature drops too quickly)**
> When the temperature decreases too rapidly, the acceptance probability for worse solutions becomes extremely low. As a result, the algorithm quickly loses its ability to escape local optima and stagnates.

Solution:
Use a **slower cooling schedule** such as a higher geometric factor (e.g., α = 0.99 instead of 0.90) or switch to a **non-linear cooling schedule** to maintain acceptance flexibility for longer.

> **Issue 2: Poor move generation / weak neighborhood structure**
> If the algorithm generates new solutions that are significantly worse or infeasible, they are likely to be rejected regardless of the temperature.

Solution:
Design a **better neighborhood operator** that produces more meaningful or slight modifications. For example, use small perturbations or domain-specific heuristics to guide neighbor generation.

> **Tabu tenure** is a key parameter in Tabu Search that defines the number of iterations a move or solution attribute remains "tabu" (forbidden). It helps prevent the search from cycling back to recently visited solutions, thus encouraging exploration of new regions in the solution space.

When setting the tabu tenure, the following factors should be considered:

1. **Problem size**:
Larger problems may require a longer tenure to prevent quick revisits.
2. **Neighborhood structure**:
If the neighborhood is small, a short tenure is sufficient; if it's large, a longer tenure helps explore more broadly.
3. **Search intensification vs diversification**:
   - Short tenure → more intensification (exploiting local regions)
   - Long tenure → more diversification (exploring new areas)
4. **Risk of cycling**:
If frequent cycling is observed, the tenure should be increased.
5. **Dynamic adjustment**:
In practice, tenure can be adjusted dynamically to balance intensification and diversification.

> **Both SA and TS are single-point, neighborhood-based metaheuristics** designed to escape local optima in combinatorial optimization problems.

🔷 **How they escape local optima:**

- **Simulated Annealing (SA):**
SA probabilistically accepts worse solutions based on a temperature-controlled probability function. The acceptance probability decreases over time, allowing SA to occasionally move to worse solutions and thus escape local optima.

$$P(\text{accept}) = \exp\left(-\frac{\Delta}{T}\right)$$

- **Tabu Search (TS):**
TS escapes local optima by **prohibiting recently visited solutions or moves** using a tabu list. Even if all neighbors are worse, TS can explore non-improving solutions, as long as they are not tabu (or if they satisfy an aspiration criterion).

🔷 **Main differences between SA and TS:**

Feature	Simulated Annealing (SA)	Tabu Search (TS)
Acceptance of worse solutions	Probabilistic (based on temperature)	Deterministic (accept best non-tabu)
Memory usage	No memory	Uses memory (Tabu List)
Escape strategy	Randomness (stochastic)	Prohibited revisits (memory-based)
Parameters	Temperature, cooling rate	Tabu tenure, aspiration criterion
Nature	Stochastic	Deterministic (with exceptions)