AYDIN ADNAN MENDERES UNIVERSITY CSE 419 ARTIFICIAL INTELLIGENCE ASSIGNMENT#1

Problem Definition

The Cutting Stock Problem (CSP) is a well-known optimization problem encountered in various industries, particularly in manufacturing. It involves cutting large stock materials—such as rolls of paper, metal, or wood—into smaller pieces of specific lengths required to meet customer demand. The main objective is to minimize the amount of leftover material (waste) while fulfilling all quantity requirements efficiently.

For instance, given a 100-meter stock roll and a set of piece lengths such as 10, 15, and 20 meters, the goal is to determine how to cut the rolls in such a way that all demands are satisfied using the fewest number of rolls and producing the least waste.

This project aims to address CSP using a multi-agent system enhanced by local search optimization algorithms. Efficient solutions to CSP help reduce material costs and promote better resource utilization.

Algorithms Used

The solution architecture utilizes multiple agents, each operating with a different heuristic optimization strategy. The project incorporates two key strategies:

1. **Collaborative Agent-Based Optimization:** Each agent independently solves the CSP using its assigned heuristic and then shares its best solution with the others. Through this collaboration, agents iteratively refine and improve the overall cutting strategy by exchanging information.
2. **Hyper Meta-Heuristic Approach:** Each agent employs a different heuristic algorithm—Simulated Annealing (SA), Hill Climbing (HC), or Genetic Algorithm (GA). The most efficient solution among these is selected, allowing the system to leverage the strengths of diverse heuristics for optimal waste minimization.

This dual approach enables both dynamic collaboration and effective comparison of different optimization techniques within the multi-agent framework.

Experimental Setup

The experiments were conducted using a Python implementation that allows users to specify the stock roll length, piece sizes, and quantities. The system supports flexible parameter settings, enabling adaptation to various CSP scenarios. For each test

case, the selected algorithms (SA, HC, GA) were executed both independently and within a collaborative agent-based framework.

Example Code Blocks

The following code snippets are directly taken from the actual Python implementation of the algorithms.

```python
# === SA & HC: Waste evaluation used in both ===
def evaluate_solution(solution):
    rolls = []
    current = []
    for piece in solution:
        if sum(current) + piece <= roll_length:
            current.append(piece)
        else:
            rolls.append(current)
            current = [piece]
    if current:
        rolls.append(current)
    waste = sum([roll_length - sum(r) for r in rolls])
    return waste, rolls

# === GA: decode and fitness functions ===
def decode(individual):
    sequence = [total_pieces[i] for i in individual]
    solution = []
    current_roll = []
    for piece in sequence:
        if sum(current_roll) + piece <= roll_length:
            current_roll.append(piece)
        else:
            solution.append(current_roll)
            current_roll = [piece]
    if current_roll:
        solution.append(current_roll)
    return solution

# Fitness calculation for GA

def fitness(individual):
    sequence = [total_pieces[i] for i in individual]
    piece_counts = {l: 0 for l in demand}
    for l in sequence:
        if l in piece_counts:
            piece_counts[l] += 1

    penalty = 0
    for l in demand:
        diff = piece_counts[l] - demand[l]
        if diff < 0:
            penalty += abs(diff) * 1000
        elif diff > 0:
            penalty += diff * 500

    solution = decode(individual)
    waste = sum([roll_length - sum(roll) for roll in solution])
    return waste + penalty
```

Experimental Results & Analysis

In this section, we present two distinct multi-agent configurations based on the strategies described: Collaborative Agent-Based Optimization and Hyper Meta-Heuristic Approach.

Instance 1: Collaborative Agent-Based Optimization

Each agent (SA, HC, GA) runs independently to solve the CSP. Afterward, they share their best solution, and all agents refine using the best shared encoding.

EXAMPLE 1:

Roll Length		Piece lengths and quantities	Algorithm(s)			
100		10-5,15-3,20-2	SA, HC, GA			
Algorithm	Rolls Used	Rolls	Used	Waste	Total Waste	Runtime (sec)
SA	2	Roll 1: [10, 15, 10, 20, 20, 10, 15]	100	0	65	0.005
		Roll 2: [10, 10, 15]	35	65		
HC	2	Roll 1: [10, 15, 10, 10, 10, 15, 15, 10]	95	5	65	0.004
		Roll 2: [20, 20]	40	60		
GA	2	Roll 1: [10, 15, 20, 15, 10, 10, 10, 10]	100	0	65	0.022
		Roll 2: [15, 20]	35	65		
Colloborative						
SA (refined)	2	Roll 1: [10, 15, 10, 20, 20, 10, 15]	100	0	65	0.005
		Roll 2: [10, 10, 15]	35	65		
HC (refined)	2	Roll 1: [10, 15, 10, 20, 20, 10, 15]	100	0	65	0.004
		Roll 2: [10, 10, 15]	35	65		
GA (refined)	2	Roll 1: [15, 15, 20, 10, 10, 20]	90	10	65	0.022
		Roll 2: [15, 10, 10, 10]	45	55		

Observation: All three algorithms used the same number of rolls and produced identical total waste. The refined versions showed slight differences in the distribution of pieces across rolls, but no significant reduction in waste was observed. Runtimes remained similarly low across all algorithms, with Hill Climbing being marginally the fastest, followed by Simulated Annealing, while the Genetic Algorithm took slightly longer.

EXAMPLE2:

Roll Length		Piece lengths and quantities	Algorithm(s)			
135		22-2, 37-3, 31-2, 15-2,7-4,16-5	SA, HC, GA			
Algorithm	Rolls Used	Rolls	Used	Waste	Total Waste	Runtime (sec)
SA	3	Roll 1: [22, 22, 16, 16, 15, 37]	128	7	50	0.007
		Roll 2: [16, 37, 31, 7, 31, 7]	129	6		
		Roll 3: [15, 16, 37, 7, 16, 7]	98	37		
HC	3	Roll 1: [7, 15, 37, 16, 7, 15, 16]	113	22	50	0.005
		Roll 2: [31, 7, 31, 22, 7, 22]	120	15		
		Roll 3: [16, 37, 16, 37, 16]	122	13		
GA	3	Roll 1: [7, 37, 16, 16, 16, 7, 31]	130	5	50	0.034
		Roll 2: [7, 22, 15, 16, 37, 31, 7]	135	0		
		Roll 3: [16, 15, 37, 22]	90	45		
Colloborative						
SA (refined)	3	Roll 1: [22, 22, 16, 16, 15, 37]	128	7	50	0.007
		Roll 2: [16, 37, 31, 7, 31, 7]	129	6		
		Roll 3: [15, 16, 37, 7, 16, 7]	98	37		
HC (refined)	3	Roll 1: [22, 22, 16, 16, 15, 37]	128	7	50	0.007
		Roll 2: [16, 37, 31, 7, 31, 7]	129	6		
		Roll 3: [15, 16, 37, 7, 16, 7]	98	37		
GA (refined)	3	Roll 1: [37, 22, 15, 7, 15, 7, 16, 16]	135	0	50	0.034
		Roll 2: [7, 16, 22, 37, 7, 31]	120	15		
		Roll 3: [37, 31, 16, 16]	100	35		

Observation: All three algorithms utilized the same number of rolls and produced identical total waste values. The refined versions of the algorithms showed slight differences in their cutting patterns, but no significant waste reduction was achieved. The runtimes remained low across all algorithms, with the Genetic Algorithm showing a slightly longer runtime compared to the Simulated Annealing and Hill Climbing algorithms.

Instance 2: Hyper Meta-Heuristic Approach

Each agent runs a different heuristic (SA, HC, GA). The best solution is selected among them without refinement or collaboration.

EXAMPLE 1:

Roll length		Piece lengths and quantities	Algorithm(s)			
120		10-6, 20-4, 30-2	SA, HC, GA			
Algorithm	Rolls Used	Rolls	Used	Waste	Total Waste	Runtime (sec)
SA	2	Roll 1: [10, 10, 30, 20, 20, 20, 10]	120	0	40	0.005
		Roll 2: [30, 10, 10, 20, 10]	80	40		
HC	2	Roll 1: [10, 20, 30, 10, 10, 10, 10, 20]	120	0	40	0.004
		Roll 2: [20, 10, 20, 30]	80	40		
GA	2	Roll 1: [30, 20, 10, 10, 10, 30]	110	10	40	0.024
		Roll 2: [20, 10, 20, 10, 20, 10]	90	30		

Observation: All three algorithms used the same number of rolls. While the Simulated Annealing (SA) and Hill Climbing (HC) algorithms produced identical total waste values of 40, the Genetic Algorithm (GA) produced a total waste of 40 as well, but with a different distribution of waste between the rolls (10 on the first roll and 30 on the second). The cutting patterns varied slightly between algorithms, but no reduction in waste was observed. The runtimes remained low across all algorithms, with Hill Climbing being the fastest, followed by Simulated Annealing, while the Genetic Algorithm took slightly longer.

EXAMPLE 2:

Roll length		Piece lengths and quantities	Algorithm(s)			
142		38-2,27-2,19-3,14-2	SA, HC, GA			
Algorithm	Rolls Used	Rolls	Used	Waste	Total Waste	Runtime (sec)
SA	2	Roll 1: [27, 19, 19, 19, 38, 14]	136	6	69	0.005
		Roll 2: [38, 14, 27]	79	63		
HC	2	Roll 1: [19, 27, 38, 27, 19]	130	12	69	0.003
		Roll 2: [14, 38, 14, 19]	85	57		
GA	2	Roll 1: [19, 27, 19, 27, 38]	130	12	69	0.021
		Roll 2: [38, 14, 19, 14]	85	57		

Observation: All three algorithms used the same number of rolls. Simulated Annealing (SA) produced a total waste of 69, with a distribution of 6 waste units in the first roll and 63 in the second. Both Hill Climbing (HC) and Genetic Algorithm (GA) also generated a total waste of 69, but with different waste distributions—12 in the first roll and 57 in the second. The cutting patterns varied slightly across algorithms, but no reduction in waste was observed. The runtimes were very low across all algorithms, with Hill Climbing being the fastest, followed closely by Simulated Annealing, while the Genetic Algorithm took slightly longer.

Conclusion

In this study, the efficiency of different heuristic algorithms and multi-agent systems in solving the Cutting Stock Problem was examined. Trials using Simulated Annealing, Hill Climbing, and Genetic Algorithms showed that each algorithm provided effective results in specific scenarios. The collaborative agent-based approach allowed agents to share solutions and achieve better results, while the hyper meta-heuristic approach, combining different algorithms, helped find the most efficient cutting strategy. The results indicate that these strategies were successful in minimizing material waste. In the future, further testing with additional parameters could yield more optimized solutions.