Report
Fundamentals of Optimization

Harvest Planning

PREPARED & SUBMITTED BY

Do Nghiem Duc 20214892 duc.dn214892@sis.hust.edu.vn
Trinh Hoang Giang 20214893 giang.th214893@sis.hust.edu.vn
Bui Khanh Linh 20214910 linh.bk214910@sis.hust.edu.vn

SUBJECT

Fundamentals of Optimization

PROJECT NAME

Harvest Planning

INSTRUCTOR

Bui Quoc Trung

Content

1. Problem

a. Problem description

- There are N fields of 1, 2,..., N of the same agricultural product planted.
- Each field i has a yield of d(i) and must be harvested between days s(i) and e(i).
- The agricultural product processing plant has a maximum processing capacity of M. (total production of agricultural products that can be processed in 1 day). Furthermore, the factory starts the machine only when the total output to be processed in a day must be greater than or equal to m.
- Planning the harvest of agricultural products (what day each field is harvested) so that maximum and minimum power constraints are complied and the difference in the number of agricultural products to be harvested between the days are the smallest.

b. Input

Line 1: N, m, M
Line 1+i (i = 1,…,N): d(i), s(i) và e(i)

2. Algorithms

a. Exhaustive Search - backtracking

- It goes through each solution of the problem, and for each solution, it calculates its value of the objective function, then compares values of objective functions for all solutions to find the optimal solution whose objective function is minimal.
- **Variable:**
  +) X[i] contains fields harvested in day i
  +) Avai_field[i] contains all possible fields in day i
- **Constraints:**
  +) X[i] ⊂ Avai_field[i]
  +) m <= sum(d[X[i]]) <= M or X[i] = empty
  +) X[i] intercept X[j] == empty for i != j
- **Objective function:**
  minimize f = max(d[X[i]]) - min(d[X[j]])

b. Local Search

i. Simulated Annealing

1. Initialize: Randomly assign harvest days for each field between its harvesting window (s[i] and e[i]).

2. Define the loss function: Determine the total threshold violation with respect to the minimum processing threshold m and the maximum processing capacity M.
3. Define the neighbor: Move a randomly selected field to a different random day between its harvesting window (s[i] and e[i]).
4. Local search for feasible solutions: Repeat step 3 until a number of feasible solutions are found, where the loss function is equal to 0.
5. Define the optimized function: Calculate the deviation between the maximum and minimum yields.
6. Local search for optimal solutions with some acceptance of higher deviation: Repeat step 3, but this time the neighborhood must maintain a loss function of 0. If the deviation of the new solution is higher than the current deviation, it may still be accepted with some probability, defined as exp(-abs(new_dev - curr_dev)), which is similar to the principle of simulated annealing.
7. Select the best solution: From these optimal deviations of those feasible solutions found, choose the one with the smallest deviation.

ii. Genetic Algorithm

1. Initialize the population: Randomly generate a defined number of harvest plans (population size) and store them in a population.
2. Calculate the loss function: Determine the total threshold violation with respect to the minimum processing threshold m and the maximum processing capacity M. Sort the population in ascending order of loss.
3. Select the mating pool: Choose individuals for mating by randomly selecting individuals from the elite (top performers in the population) and one normal individual (to introduce diversity through mutation).
4. Breed the offspring: Generate offspring by combining information from its parents. For each day, randomly choose between that day's harvested fields of the first parent or the second parent (avoiding repeated fields and ensuring all fields are harvested in the end).
5. Replace the worst individuals: Replace the worst individual in the population with the newly bred offspring, sort the population again, and repeat the process.
6. Feasible solutions: After a number of generations, choose the best individuals from the population (with the lowest loss). Repeat the whole process for other generated populations until we have several feasible solutions (loss = 0).
7. Select the best solution: From feasible solutions, select the one with the lowest deviation as the final solution.

c. Integer Linear Programming (ILP) & Constraint Programming (CP)

- We use **OR-tools** to implement these two programs.
- They have general form:
    1. Import library: ortools, numpy, …
    2. Create solver object
    3. Define decision variables
    4. Set constraints
    5. Set objective function
    6. Solve the problem
- But they have a few differences:
  Note that: Equality constraint can be transferred to inequality constraint with same lower bound and upper bound ($A = B \Leftrightarrow B \leq A \leq B$).

	ILP	**CP**
	The constraints must be linear form	The constraints can be statements
Variable	$x(i,j) = 1$ if j^{th} field is harvested on i^{th} day, = 0 otherwise $t(i)$ = 1 if the machine is used on i^{th} day, = 0 otherwise ma: maximum capacity mi: minimum capacity	$x(i,j) = 1$ if j^{th} field is harvested on i^{th} day, = 0 otherwise ma: maximum capacity mi: minimum capacity
Constraints	1, $mt(i) \leq \sum_{j=1}^{N} x(i,j)d\,(j) \leq Mt(i) \forall i$ 2, $m \leq mi \leq M$ 3, $m \leq ma \leq M$ 4, $ma \geq \sum_{j=1}^{N} x(i,j) \cdot d(j) \forall i$ 5, $M(t(i)-1) + mi \leq \sum_{j=1}^{N} x(i,j) \cdot d(j)$ 6, x(i, j) = 0 if i not in [s(j), e(j)] 7, $\sum_{i=s(j)}^{e(j)} x(i,j) = 1 \forall j$	1, $m \leq mi \leq M$ 2, $m \leq ma \leq M$ 3, $(m \leq \sum_{j=1}^{N} x(i,j)d\,(j) \leq M) \vee (\sum_{j=1}^{N} x(i,j) = 0)$ 4, $ma \geq \sum_{j=1}^{N} x(i,j) \cdot d(j) \forall i$ 5, $\sum_{j=1}^{N} x(i,j) \cdot d(j) < m \Rightarrow mi \leq \sum_{j=1}^{N} x(i,j)\,d(j)$ 6, x(i, j) = 0 if i not in [s(j), e(j)] 7, $\sum_{i=s(j)}^{e(j)} x(i,j) = 1 \forall j$
Objective function	$f(x) = ma - mi$ -> min	$f(x) = ma - mi$ -> min

The fifth constraint of the MIP model may be difficult to understand. We will explain it in more detail:

- If t(i)=1 means that the factory starts the machine on i^{th} day, then the inequality equivalent to mi is less than or equal to the total product on the i^{th} day.
- Otherwise, if the factory doesn't start the machine (t(i)=0 and the total product must be 0), the we have an inequality $mi - M \leq 0$ which is always true.

3. Comparison

- We test these algorithms for 6 different instances.
- Local Search algorithms can take a long time. Therefore, we will limit the running time of these algorithms and we will take the first 5 feasible solutions that the algorithms return.

Bộ dữ liệu	Kích thước	Thuật toán giải đúng				Thuật toán heuristic									
		IP		CP		Simulated Annealing					Genetic Algorithm				
		f	t(s)	f	t(s)	f_min	f_max	f_avg	std_dev	t_avg(s)	f_min	f_max	f_avg	std_dev	t_avg(s)
Ins1	5	2	0.02	2	0.02	2	2	2	0	0.0052	2	2	2	0	0.022
Ins2	10	0	0.06	0	0.06	1	7	4.6	2.41	0.0054	2	5	3.8	1.17	0.024
Ins3	15	0	0.09	0	0.07	3	5	3.8	0.75	0.068	5	8	6.4	1.02	0.038
Ins4	20	1	15.66	1	0.73	4	8	5.8	1.54	0.44	9	12	10.4	1.02	0.096
Ins5	25	0	0.54	0	0.65	5	9	7.8	1.47	0.222	7	15	11.2	2.16	0.114
Ins6	30	0	2.66	0	0.43	8	10	9.2	0.75	0.334	13	18	14.8	1.94	0.206

- IP & CP & Backtracking give us the best solution but consume too much time, especially backtracking.
- Local Search doesn't return the best solution, but an acceptable one with much less time-consuming.
-

4. Conclusion

- If we consider the optimal solution, IP & CP & Backtracking is the best. But if we consider the time-consuming with an acceptable solution, Local search is the best.
- After studying this project, we have learned more about algorithms to solve

optimization problems than just learning theory and also have our experience in analyzing problems. If we had more time, we'd like to do a more thorough analysis of the issue.

5. List of tasks

Tasks		Participant
Analytic tasks	Designing and illustrating slides	Linh
	Generating and testing programs	Giang
	Writing report	Linh
Programming tasks	Implement exhaustive search	Duc
	Implement linear programing	Giang
	Implement constraint programing	Giang
	Implement local search	Duc
	Creating input data generator	Linh