Algorithms PA2 Report

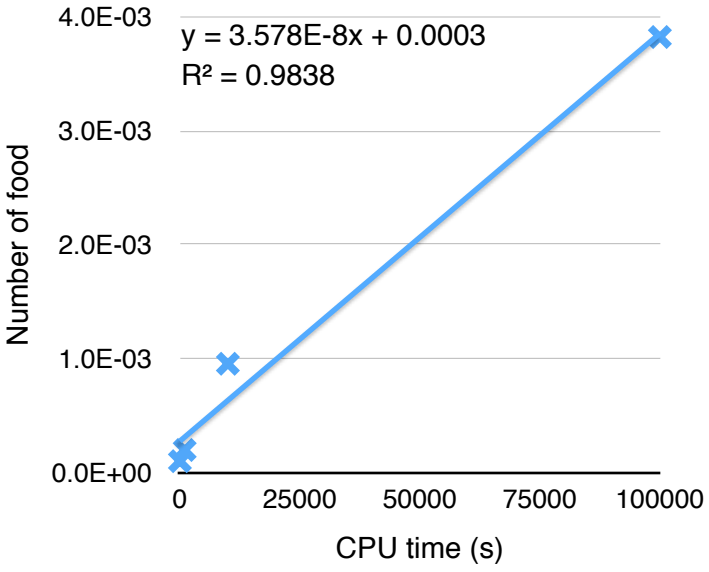
B01501061 NTUEE 盧柏儒

1. Data Analysis

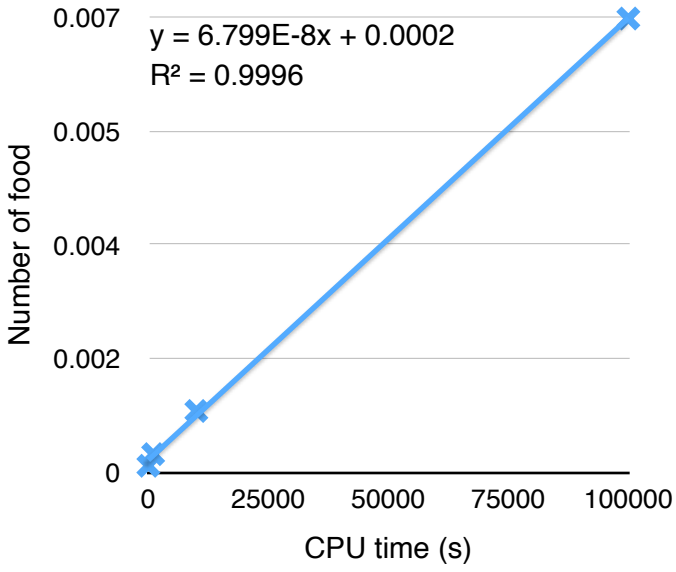
Table 1

case	*number* of food	*GD total distance*	GD CPU time (s)	GD Memory (KB)	*DP Total distance*	DP CPU time (s)	DP Memory (KB)
case1	4	20	9.8E-05	12580	14	0.000101	12580
case2	1000	154742	0.000196	12580	150426	0.00028	12580
case3	10000	1550932	0.000953	12740	1512940	0.000944	12880
case4	100000	30931698	0.003821	14916	30092520	0.006981	15700

In the table 1, there are Greedy (GD) Algorithm and Dynamic programming (DP) Algorithm to implement the WorkerAnt program respectively. Oberviously, in these case, the DP's total distance is shorter than the GD's. However, DP takes more time and more memory to achieve the best solution (it will be proved later). The time complexity chart is as following.



time complexity for GD chart 1



time complexity for DP chart 2

2. Jobs

(1) Prove the Optimal Substructure

Assume the mode $d[i]$ is the best sub-solution,

$$d[i] = \min_{0 \le j < i} \{d[j] + dist2origin(\_\_\_\_\_) + dist(\_\_\_\_\_, i) + dist2origin(i)\}$$

Let the $d[j]$ is not the best sub-solution, and then the $d[i]$ will have better sub-solution. It is contradicted to our assumption. There the $d[i]$ is the best sub-solution.

(2) Dynamic Programming

In my DP program, there is two primary part. First, since the best sub-solution is that the ant can take all of food at one time, I use the maximum capacity to find the last food (key food) which the ant can take food at one time and record the point. In this part, it's time complexity is O(n), because the program only execute the function "is_over_cap" for n times to find the food in the worst case (the sum capacity of all food is smaller than the ant maximum capacity). The reason why I add the first step will be explain in the experiment part of this report. In another part, I assume each sub-solution after the key food is the sum of the previous one best sub-solution and the distance from original point to the food position. In the second part, the time complexit is the O(cn) (c is the maximum capacity of the ant). Therefore, the total complexity is O(cn). When c is a constant, the time complexity is O(cn) = O(n). But when c is related to n, the time complexity is $O(n^2)$.

(3) Greedy Choice Property

The Greedy algorithm the best return point is not always previous point. But in the Greedy algorithm, the ant return if it can't take more food, it result in the previous point is a return point. Therefore, the Greedy algorithm is solution for ant to take all of food in order but it does not always produce the optimal solution.

(4) Greedy Implementation

The time complexity is O(n). The ant does not worry the previous return point and only care about its current capacity. If over the maximum capacity, it return. If not, it keep going to take food. Therefore, It only take a loop to finish the question.

3. Experiment

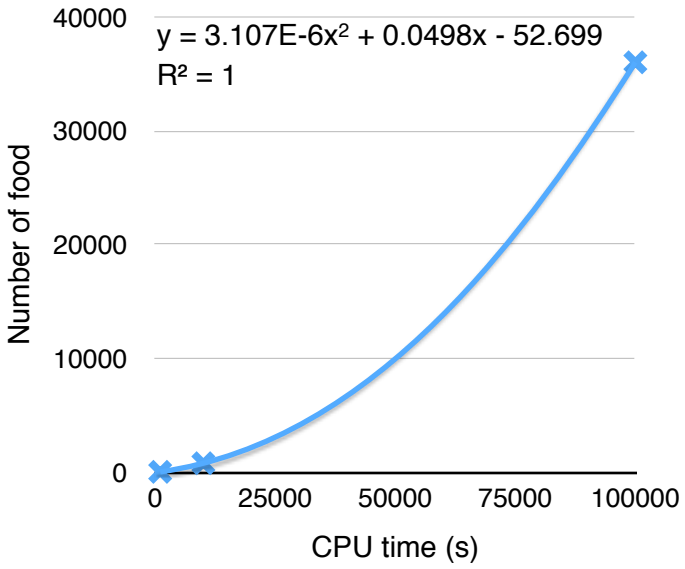
In this experiment, I edit the maximum capacity of the ant, and then execute the DP program. The experiment data table an chart is as following.

When the maximum capacity is bigger but not too bigger, it means the c is related to n. Therefore, the time complexity is $O(n^2)$.

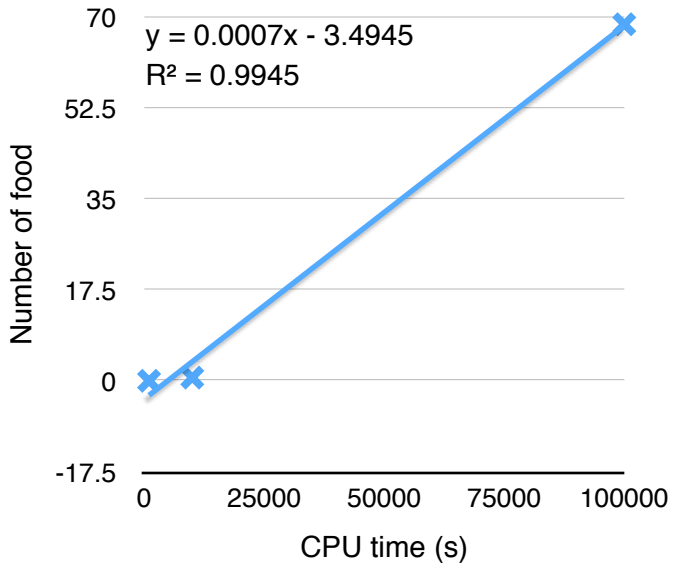
But if the maximum capacity is much bigger, in my program first part, the function "is_over_cap" will be execute n times nearly. Thus, the complexity is about O(n).

Table 2

case	number of food	big but not too big	maximum capacity	very big	maximum capacity
case2	1000	0.216682	100000	0.006027	10000000
case3	10000	756.106	1000000	0.622765	100000000
case4	100000	36000	10000000	68.5653	2140000000



big but not too big chart 3



very big chart4