

A. INTRODUCTION

Consider the Pancho's Burritos restaurant. Figure 1 shows the operational sequence to complete one burrito order.

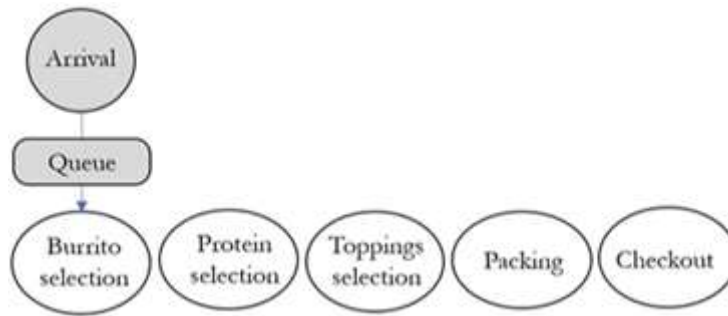


Figure 1 – Operational sequence to complete one burrito order.

As per figure 1, after the customer arrives and completes queue, there are 5 stations to be completed. Currently, the restaurant has the capacity to serve 92 customers per hour. Figure 2 represents the current weighted distributed tasks among the 5 stations; for example, Station 1 includes Base, Rice, and Beans while station 5 corresponds to checkout. Figure 2 also presents the completion times for each task and the total completion time for each station.

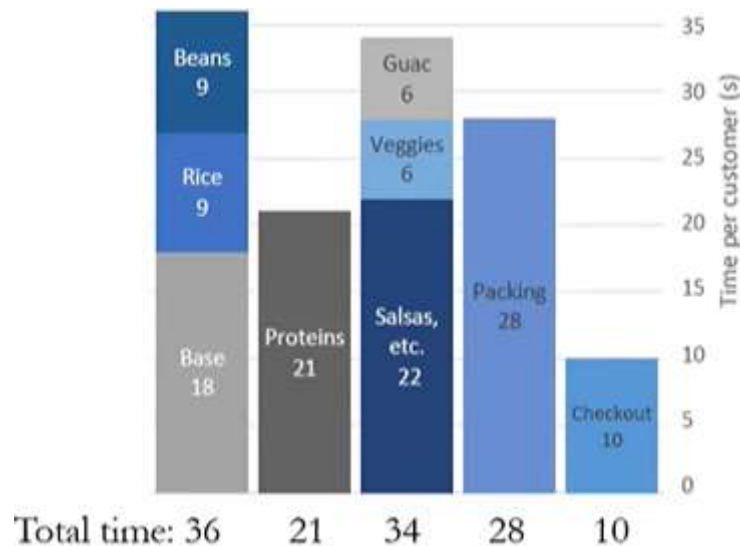


Figure 2 – Processing times, total completion time per station, and task distribution per station.

The problem described can be also compared to assembly line balancing applications, in which, the order of tasks takes precedence. For example, the tasks necessary to complete a customer order, depend on one another. In the case described above, the addition of Salsa, Veggie, or Guac, would not be feasible if done after packing, therefore to complete packing and subsequently checkout, the selection of food toppings must be done first.

Furthermore, an important measure of assembly lines applications is the cycle time or the maximum completion time at a specific station. As shown in figure 2, station 1 presents the current maximum completion time. This indicates that every 36 seconds the restaurant can serve one customer. The production capacity per hour is then calculated by dividing 3600 seconds over the cycle time. The current model for Pancho's Burritos produces 100 burritos per hour.

In this assignment 2 questions will be analyzed:

- What is the maximum capacity with 5 workers?
- Is it possible to meet the current demand (92 customers) with four workers?
- What is the required number of workers to meet a demand of 125 customers per hour?

In assembly problems, assignment of tasks among stations is important. For each of the above questions, a new task distribution chart is presented to show the best task assignment to meet each question's objective.

B. OBJECTIVES:

- i. To understand the process of sequencing and scheduling.
- ii. To understand the concept and preparation of Mathematical Model
- iii. To find the cycle time to calculate the maximum capacity of and meeting the demand of the customer.

C. QUESTIONS:

Question 1

What is the maximum capacity with 5 Workers?

Solution Setup:

- a. We used IBM CPLEX write codes and find the solution to the given problem.
- b. Firstly, we prepared the mathematical model as following:

Objective: Minimize Cycle time

$X_{i,j}$: Binary decision variable taking the value "1" when task "I" is done by worker "j";
0 otherwise

Since, one tasks can be done only by a given worker, we have:

$$\sum_{j=0}^m X_{i,j} = 1$$

For every: $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$

- c. We exclude worker "5" from this optimization since we know that packaging(28sec) and checkout(10sec) must be done last, which both sum up to 38sec, which is not optimal so we consider only the first 4 workers.

We have the following processing time which could be re-assigned to any worker within the base and packaging time as:

$$P = [9, 9, 21, 6, 6, 22]$$

In this case we will have the workers with the base and packaging time as:

$$\text{Add} = [18, 0, 0, 28]$$

$$\text{Cycle time} \geq X_{i,j} + \text{Add}$$

CPLEX Codes:

```

int workers=4;
int tasks=6;
int min2hr=60*60;
range n=1..tasks;
range m=1..workers;

int p[n]=[9,9,21,6,6,22];
int add[m]=[18,0,0,28]; /*compulsory added task that must be performed in worker 1 and 4*/

dvar boolean x[n][m];
dvar int+ cycle_time;
dvar int+ Wcyc_time[n][m];
minimize cycle_time;

subject to {
    c1: forall(i in n) sum(j in m)x[i][j]==1;
    c2: forall(i in n,j in m) Wcyc_time[i][j]==x[i][j]*p[i];
    c3: forall(j in m) cycle_time>=sum(i in n)Wcyc_time[i][j]+add[j];
}

float cap=round(min2hr/cycle_time); /*Since each Burritos must be a complete product*/
execute
{
    writeln("Number of Workers = ",workers+1);
    writeln("Minimum Cycle time = ",cycle_time);
    write("Maximum Capacity = ", cap, " Burritos/hr");
}

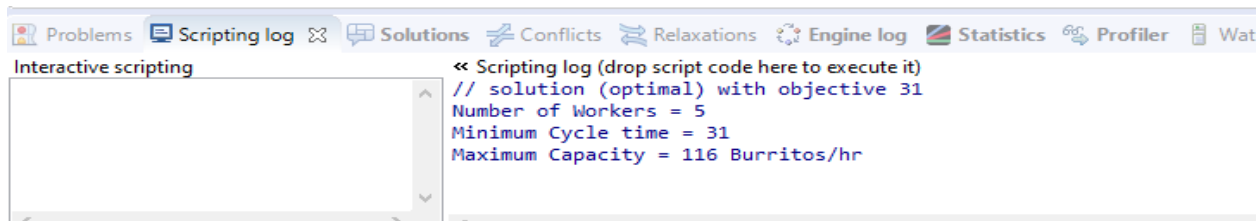
```

Output on the Problem Browser:

Solution with objective 31		
	Name	Value
Data (7)		
	add	[18 0 0 28]
	m	1.4
	min2hr	3600
	n	1.6
	p	[9 9 21 6 6 22]
	tasks	6
	workers	4
Decision variables (3)		
	cycle_time	31
	Wcyc_time	[[0 0 9 0] [0 9 0 0] [0 21 0 0] [6 0 0 0] [6 0 0 0] [0 0 22 0]]
	x	[[0 0 1 0] [0 1 0 0] [0 1 0 0] [1 0 0 0] [1 0 0 0] [0 0 1 0]]
Constraints (3)		
	c1	forall(i in 1..6) sum(j in 1..4) x[i][j] == 1
	c2	forall(i in 1..6, j in 1..4) Wcyc_time[i][j] == x[i][j]*p[i]
	c3	forall(j in 1..4) cycle_time >= sum(i in 1..6) Wcyc_time[i][j]+add[j]
Result data (1)		
	cap	116

Figure 3 – Output for Question 1

Output on Script Log:



```

<< Scripting log (drop script code here to execute it)
// solution (optimal) with objective 31
Number of Workers = 5
Minimum Cycle time = 31
Maximum Capacity = 116 Burritos/hr

```

Figure 4 – Script Log for Question 1

Conclusion to Question 1:

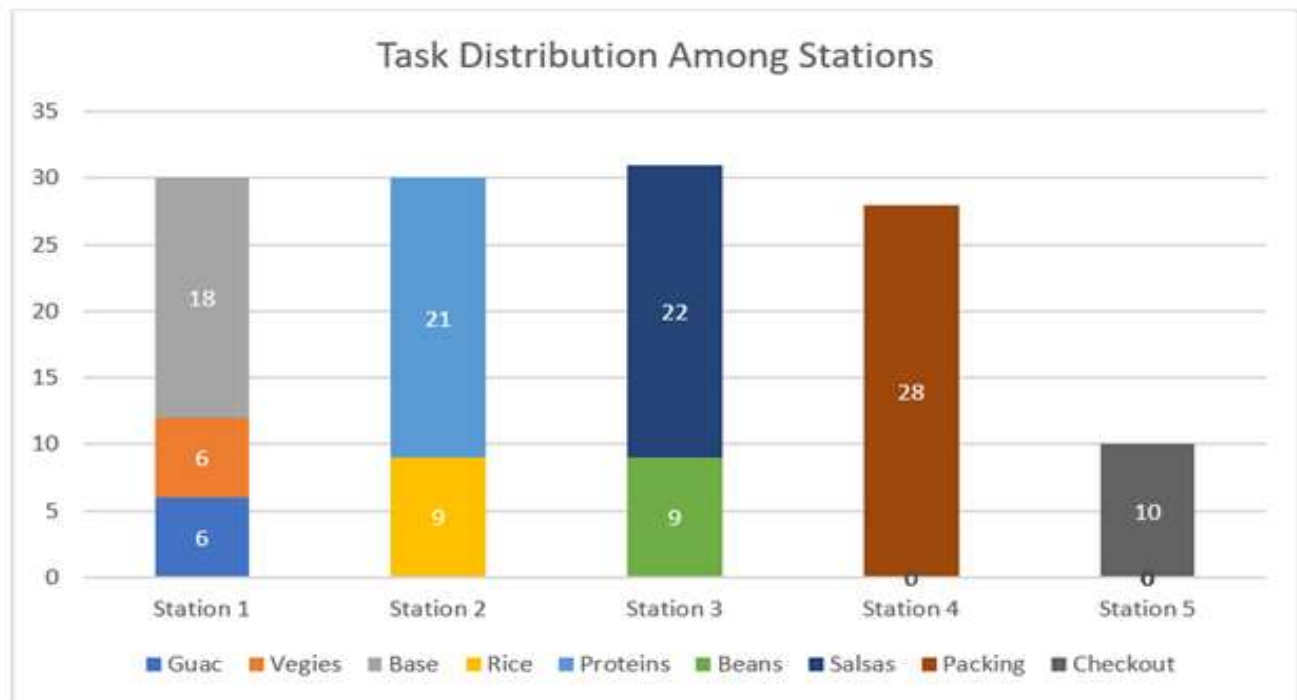


Figure 5 – Task Distribution among Stations, Question 1

Table 1 – Total Time per station, Question 1

Station	Total Time
1	30
2	30
3	31
4	28
5	10

- We got the cycle time = 31 seconds
Capacity = $3600 / 31$
= 116.129032258 Burritos
- The maximum capacity with 5 workers = 116.129032258 = approx. 116 Burritos.

Question 2

Is it possible to meet the current demand (92 customers) with four workers?

Solution Setup:

- We used IBM CPLEX write codes and find the solution to the given problem.
- For this optimization we reduced the number of workers to 3, since the last (i.e., 4th worker) will have a cycle time of 38sec for packaging(28sec) and checkout(10sec)*

CPLEX Codes:

```
int workers=3;
int tasks=6;
int min2hr=60*60;
int worker4=38;

range n=1..tasks;
range m=1..workers;
range cnt=1..2;
int p[n]=[9,9,21,6,6,22];
int add[m]=[18,0,0]; /*compulsory added task that must be performed in worker 1*/

dvar boolean x[n][m];
dvar int+ cycle_time;
dvar int+ Wcyc_time[n][m];
minimize cycle_time;

subject to
{
  c1: forall(i in n) sum(j in m)x[i][j]==1;
  c2: forall(i in n,j in m) Wcyc_time[i][j]==x[i][j]*p[i];
  c3: forall(j in m) cycle_time>=sum(i in n)Wcyc_time[i][j]+add[j];
}

int cyc_time[cnt]=[cycle_time,worker4];
/*Below line, compares the optimized cycle time for the 1st three stations and station 4*/
int actual_cyc_time=max(i in cnt)cyc_time[i];
float cap=round(min2hr/actual_cyc_time); /*Since each Burritos must be a complete
product*/
execute
{
  writeln("Minimum Cycle time = ",actual_cyc_time);
  write("Maximum Capacity = ", cap, " Burritos/hr");
}
```

Output on the Problem Browser:

	Name	Value
▼	Data (9)	
📄	add	[18 0 0]
↔	cnt	1.2
↔	m	1.3
11	min2hr	3600
↔	n	1.6
📄	p	[9 9 21 6 6 22]
11	tasks	6
11	worker4	38
11	workers	3
▼	Decision variables (3)	
11	cycle_time	31
📄	Wcyc_time	[[0 0 9] [0 9 0] [0 21 0] [6 0 0] [6 0 0] [0 0 22]]
📄	x	[[0 0 1] [0 1 0] [0 1 0] [1 0 0] [1 0 0] [0 0 1]]
▼	Constraints (3)	
>	c1	forall(i in 1..6) sum(j in 1..3) x[i][j] == 1
>	c2	forall(i in 1..6, j in 1..3) Wcyc_time[i][j] == x[i][j]*p[i]
>	c3	forall(j in 1..3) cycle_time >= sum(i in 1..6) Wcyc_time[i][j]+add[j]
▼	Result data (3)	
11	actual_cyc_time	38
11	cap	95
📄	cyc_time	[31 38]

Figure 6 – Output for Question 2

Output on Script Log:

	Scripting log (drop script code here to execute it)
Interactive scripting	<pre>// solution (optimal) with objective 31 Minimum Cycle time = 38 Maximum Capacity = 95 Burritos/hr</pre>

Figure 7 – Script Log for Question 2

Conclusion:

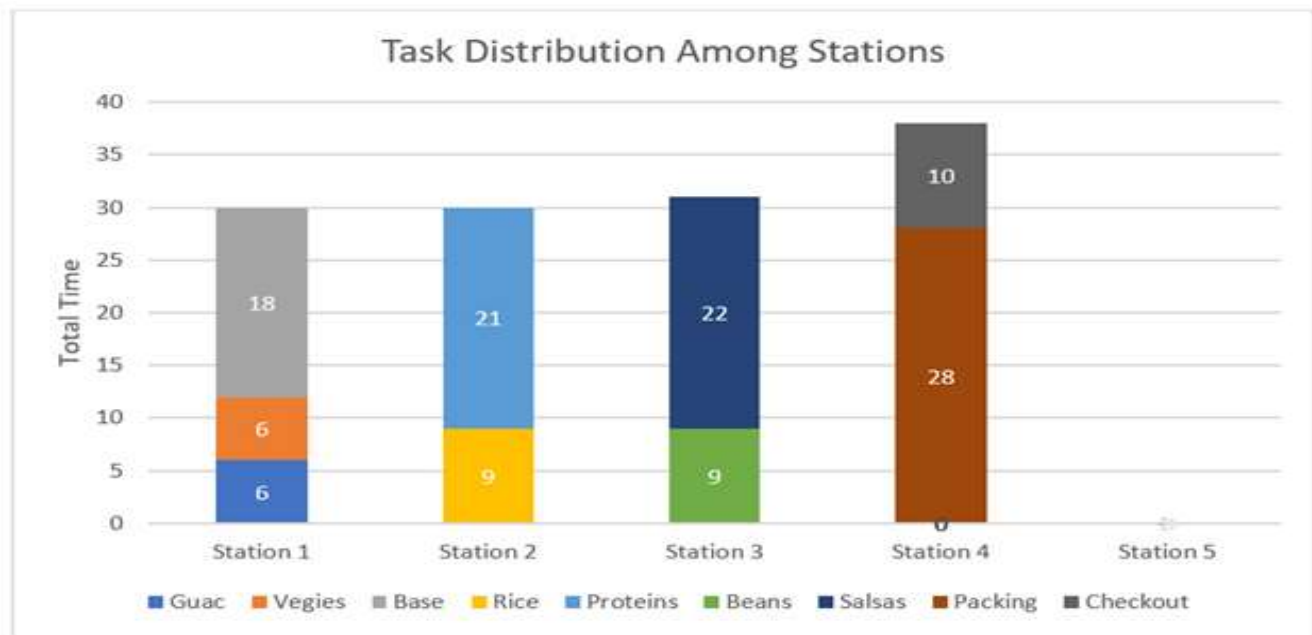


Figure 8 – Task Distribution among Stations, Question 2

Table 2 – Total Time per station, Question 2

Station	Total Time
1	30
2	30
3	31
4	38
5	0

- We got the cycle time = 38 seconds
Capacity = $3600 / 38$
= 94.736 = Approx. 95 Burritos
- Yes, we can meet the current demand of 92 customers with 4 workers.

Question 3

What is required number of workers to meet a demand of 125 customers per hour?

CPLEX Code:

```
int workers=5;
int tasks=6;
int min2hr=60*60;
range n=1..tasks;
range m=1..workers;

int p[n]=[9,9,21,6,6,22];
int add[m]=[18,0,0,0,28]; /*compulsory added task that must be performed in worker 1 and
5*/

dvar boolean x[n][m];
dvar int+ cycle_time;
dvar int+ Wcyc_time[n][m];
minimize cycle_time;

subject to {
    c1: forall(i in n) sum(j in m)x[i][j]==1;
    c2: forall(i in n,j in m) Wcyc_time[i][j]==x[i][j]*p[i];
    c3: forall(j in m) cycle_time>=sum(i in n)Wcyc_time[i][j]+add[j];
}

float cap=round(min2hr/cycle_time); /*Since each Burritos must be a complete product*/
execute
{
    writeln("Number of Workers = ",workers+1);
    writeln("Minimum Cycle time = ",cycle_time);
    write("Maximum Capacity = ", cap, " Burritos/hr");
}
```

Output on the Problem Browser:

Problem browser [x] Variables [y] Breakpoints		
Solution with objective 28		
	Name	Value
Data (7)	add	[18 0 0 0 28]
	m	1..5
	min2hr	3600
	n	1..6
	p	[9 9 21 6 6 22]
	tasks	6
	workers	5
Decision variables (3)	cycle_time	28
	Wcyc_time	[[0 9 0 0 0] [0 9 0 0 0] [0 0 21 0 0] [6 0 0 0 0] [0 6 0 0 0] [0 0 0 22 0]]
	x	[[0 1 0 0 0] [0 1 0 0 0] [0 0 1 0 0] [1 0 0 0 0] [0 1 0 0 0] [0 0 0 1 0]]
Constraints (3)	c1	forall(i in 1..6) sum(j in 1..5) x[i][j] == 1
	c2	forall(i in 1..6, j in 1..5) Wcyc_time[i][j] == x[i][j]*p[i]
	c3	forall(j in 1..5) cycle_time >= sum(i in 1..6) Wcyc_time[i][j]+add[j]
Result data (1)		
	cap	129

Figure 9 – Output for Question 3

Output on Script Log:

```

<< Scripting log (drop script code here to execute it)
// solution (optimal) with objective 28
Number of Workers = 6
Minimum Cycle time = 28
Maximum Capacity = 129 Burritos/hr
  
```

Figure 10 – Script Log for Question 3

Conclusion:

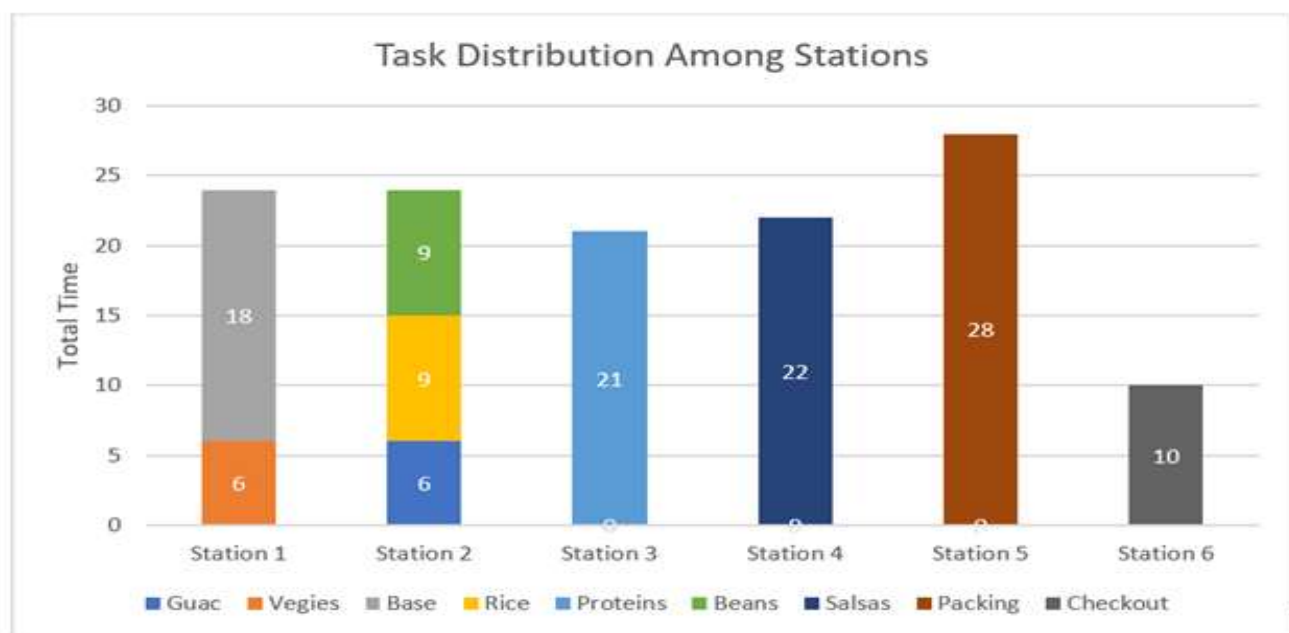


Figure 11 – Task Distribution Among Stations, Question 3

Table 3 – Total Time per station, Question 3

Station	Total Time
1	24
2	24
3	21
4	22
5	28
6	10

- We got the cycle time = 28 seconds
Capacity = $3600 / 28$
= 128.5714 = Approx. 128 Burritos
- We need 6 workers to meet a demand of 125 customers per hour.