NU Industries: A Case Study in the Benefits of Linear Programming for Supply Chain Optimization

By: Billy Burke, Julia Ma, Manoj Pillai, Kay Quiballo, Lauren Vaught

Abstract:

Unprecedented supply chain challenges have created huge hurdles in recent years due to high demand and expectations from customers among all the manufacturers. It is vital to have an optimal supply chain solution to make it efficient. Linear programming is one of the most efficient optimization techniques and has been a powerful tool to address a wide range of optimization problems in various industries. In this study, an optimization problem is formulated to maximize NU Industries' profit considering all of the constraints related to the entire supply chain process using a linear programming model. Using sensitivity analysis, the linear programming model will allow decision-makers to gain key insights into the impact of slight changes to constraints when solving complex issues with large numbers of constraints. We have attempted to solve the problem using integer programming as well to compare and contrast the outcomes of the model to verify any business recommendations need to be revisited. Ultimately, linear programming proved to be an effective method for optimizing the NU Industries supply chain and marketing strategies, for both integer and non-integer solutions.

Keywords:

Optimization, Linear programming, integer programming, sensitivity analysis, supply chain

Introduction

Supply chain management is paramount for businesses to maximize profits and avoid supply shortages. With successful management of the supply chain, companies can gain efficiency from raw materials to all the way to customer deliverables. This is especially true given the effects the COVID-19 pandemic has had on global supply chains. Reuters estimates that supply chain disruptions may have caused nearly $4 trillion in lost revenues.[1] Given this fact, optimization has been a key priority for all manufacturing industries to improve business performance while maximizing business value. Manufacturing Optimization is a business-focused holistic discipline that delivers both cost and business growth simultaneously in the entire supply chain process.

NU Industries operates two manufacturing plants responsible for producing widgets, gadgets, and flugels. Once manufactured, products are shipped to a centralized distribution center. Due to

contractual agreements, the sales department has placed production requirements during five periods (Appendix: Table 1). Additionally, the marketing department believes that they can gain additional demand with effective advertising.

The total allocated budget for advertising is heavily dependent on the highest yield for the period in part or in full. Each product can be manufactured more than the demand and the excess manufactured product can be stored in inventory. The capacity of Plant A is 70 units, whereas Plant B can store up to 50 units. An additional constraint of inventory is it can store only finished products and raw materials should be purchased based on the demand.

NU Industries' pricing for the different items is Widgets - $2490, Gadgets - $1990, and Flugels - $2970. The goal of this study is to maximize the profit based on operation efficiency, costs, and advertising budgets. We use Linear programming to model the problem of maximizing the profit for NU Industries. The questions we tried to resolve are:

- Determine optimal product mix to maximize profitability
- What is the profit potential of plants for producing each product?
- Identify the impact of transportation cost

The assumption being made during the analysis is that the products are distributed unevenly between the two plants based on their raw material and labor requirements, and their associated costs. The modeling is performed using Python programming. A sensitivity analysis is also conducted to test how slight changes in coefficients of our objective function impact the optimal solution.

Literature Review

Supply chain optimization amongst complex lines of business is key to managing any large-scale operation. We find that in order for businesses to be successful and effectively manage their cost, the key operators or business stakeholders must have a solid grasp of many complicated workflows. NU industries is no stranger to this decision problem. We looked to compare the supply chain study of our company in question with other literature to see what works, what doesn't, and where certain approaches were most effective in solving these types of problems.

It isn't difficult to find literature surrounding supply chain optimization, but the key is to find comparable studies to effectively measure and compare to the problem you are working on. A clear front runner in this category is the paper "Supply Chain Optimization Studies: A Literature Review and Classification" by Kocaoglu Y., Gumus, and Kocaoglu B[2]. In this paper, they focus on the strategic, tactical and operational decisions of researching supply chain topics. We are able to identify optimization methods of mention, primarily with regard to solving by linear programming and sensitivity analysis. This paper gave us concrete and tactical solutions to

compare companies working with distributors and retailers. Decision levels and optimization methods are key to solving the NU industries problem and this paper was a great tool as we worked to solve it.

We found additional similarities in literature provided by Eren Ozceylan and Turan Paksoy[3]. “A Mixed Integer Programming Model for a Closed-loop Supply Chain Network” showed us examples of a model that looks for the optimal solution of transportation amounts in a closed-loop supply chain network. While NU industries isn’t a closed-loop business as defined by GEP (one that reuses production waste to create new products), we are able to find examples and draw similarities to the mixed integer solutions. Having to balance the locations of plants and retailers amongst sorting the transportation values optimally is most similar to our problem with NU industries and our need to balance delivery of raw materials, our labor, and inventory.

While working to solve our problem relating to inventory and materials, we were also able to find literature on optimizing mixed-integer linear programming models as it related to LNG (liquefied natural gas) supply chain planning. In a paper by Sangiah, Tirkolaee, and Goli, we understand the role of management and its need to focus on linear programming to model LNG sales planning over time horizons to minimize cost[4]. This piece of literature is the most similar to our problem being solved. Linear programming is the most effective method we lean on to solve the current NU industries problem. We also found that our sensitivity analysis was similar and can be considered a worthwhile expenditure of a team's effort to help drive the decision-making process.

A fourth article, and worth mentioning only because it helps us identify weakness in the resiliency while showing the vulnerability of the supply chain, is a paper by Elleuch, Dafoui, and co[5]. We did not go into as much detail as this paper provides, certainly not in the terms of “state of the art research”, but I would argue that our method of solving the problem does not rely upon or need the complex methods mentioned. This was a good example for us of a drawback that does not allow us to express the solution as elegantly or effectively as we attempted.

Methodology

Given the linear nature of the constraints that NU Industries faces, the team decided to approach this problem as a linear programming problem. Linear programming is one of the fastest ways to optimize a problem with linear constraints, and there are many ways to solve a linear program. The team chose to leverage Python for this application as it is a widely used free and open-source software (FOSS) that makes code sharing easy. Python has many libraries available for solving linear programs, but arguably the most popular library is PuLP. With PuLP, you can add LpVariables and LpConstraints to a problem and use a variety of solvers to figure out what the optimal solution would be. PuLP can also be leveraged to perform a sensitivity analysis on

the constraints, which was helpful for the team's purposes of providing business recommendations.

Of course, as with any approach to modeling a real-life scenario, the linear program does have its limitations. First, and most glaringly is that the world is rarely able to be captured perfectly in linear equations. Even though the NU Industries constraints are able to be written as linear expressions, it's highly unlikely that the assumptions made in drawing the constraints up will hold true exactly. The primary example of this is the marketing constraint. Though we can predict the budget for marketing, it's very difficult to accurately predict the increase in demand as a result of any marketing that takes place. The projections may be generous, leading to an excess of products and nowhere to store them. On the other hand, the projections may be conservative, leading to demand not being met, missed profits, and potentially larger crises as well. We don't have any insight into what gadgets, widgets, and flugels are used for, but if it's a critical component to something very important (perhaps gadgets are the key components used in ventilators used to treat COVID-19 patients!) then a shortage could be life-threatening. For the purposes of this paper, any approach used to solve the optimization problem would face this limitation, since every approach would be subject to these assumptions. Therefore, the team selected linear programming regardless, knowing that it has many other benefits to offer.

The other main limitation of linear programming is that it often will result in non-integer solutions. Some variables in a problem can be represented by a floating number. Money, for example, can be spent in increments of .01. Depending on the labor tracking system, hours of labor can also be recorded in small increments, like .1 hours. Other variables, like the number of gadgets produced, make much more sense as an integer. Unfortunately, linear programs don't always have integer solutions. Despite this limitation, the team decided to proceed given the many benefits of linear programs.

Methods – Constraints

Supply and Demand

NU Industries has varying quotas for each type of product and for each period (Appendix: Table 1).

The Marketing & Forecasting Department can cultivate additional demand through effective advertising. The total amount of demand will equal the baseline quota plus any additional demand cultivated through advertising.

- *total demand (by product per period) = quota demand + advertising demand*

Supply will be equal to the number of units produced for an individual period plus any units that are carried over by inventory of the previous period.

- *total supply (by product per period) = units produced + units inventoried*

The total demand (by product per period) cannot exceed the corresponding total supply.

- *total supply (by product per period) >= total demand (by product per period)*

Advertising
Additional advertising can increase the demand of a product for the following period. The current total advertising budget for all 5 periods is $70,000

- *advertising budget <= 70,000*

Resources
Each type of product requires a different amount of resources to produce (which varies between Plant A and Plant B). These resources are Raw Material 1, Raw Material 2, and hours in Labor. The amount of resources consumed may not exceed the total amount of resources available.

- *raw material 1 consumed (by product per period) <= raw material 1 available*
- *raw material 2 consumed (by product per period) <= raw material 2 available*
- *labor consumed (by product per period) <= labor available*

Labor
Labor is a resource used to create products. The cost of labor varies between Plant A and Plant B. Regular time labor is limited to 2500 hours for Plant A and 3800 hours for Plant B per period. Any amount of time greater than that will count as overtime labor which has a higher cost than regular time labor.

- *regular time labor at Plant A (per period) <= 2500*
- *regular time labor at Plant B (per period) <= 3800*

Inventory
Some periods have varying costs so it may be advantageous to make products in lower-cost periods and inventory them to sell during a higher cost period. The maximum inventory varies between Plant A and Plant B.

- *inventory at Plant A (per period) <= 70*
- *inventory at Plant B (per period) <= 50*

Delivery - Raw Materials
Raw material must be delivered to each plant with varying delivery costs. The total amount of each raw material available is below:

- *raw material 1 (per plant per period) <= 70 tons*
- *raw material 2 (per plant per period) <= 2.5 tons*

Methods – Objective Function

Profit
The objective is to optimize the profit function. There is 1 source of positive income, and 5 different types of costs (below). Profit will be money made from revenue minus the 5 costs.

- *Total profit = total revenue - (sum of 5 costs)*

Revenue

Revenue is the amount of money made for each of the 3 products.

- *total revenue = revenue of widgets, gadgets, and flugels*

Advertising (cost)

- *advertising costs (by product per period)*

Labor (cost)

The overtime labor costs more, and the labor costs are projected to increase after the second period of operation, so these costs are factored into the problem in this stage.

- *regular time labor + overtime labor (per plant per period)*

Inventory (cost)

Though it costs money to store products, it is sometimes more cost-effective to have some stored rather than have to produce the full amount of products to meet the demand in the same period. Furthermore, as demonstrated by current events and unforeseen supply chain stoppages, it can sometimes be advantageous to have products in inventory.

- *inventory cost (per plant per period)*

Delivery - Raw Materials (cost)

- *delivery cost raw material 1 (per plant per period)*
- *delivery cost raw material 2 (per plant per period)*

Transportation - Finished Products (cost)

- *transportation cost (by product per plant)*

Methods - Implementation

Using the PuLP library, the team was able to add each variable and constraint to the Linear Program, resulting in a total of 146 variables and 118 constraints, plus the final objective function. Most of the Python implementation is straightforward according to the documentation provided by PuLP, although there were several equality constraints that were written as two separate constraints, one greater than or equal to and one less than or equal to. The two constraints effectively create an equality constraint for the solver.

The team also explored solving this problem as an integer problem. Because many of the variables work better in a real-life scenario as integers, the team explored this route as well. In PuLP, you can simply denote the variable type to be an integer when declaring them and the program will output integer values. In general, integer problems are much more difficult for solvers to figure out because the solution set is not continuous. This will often result in the solver

spending hours and hours attempting to converge on a solution. PuLP allows the user to set a tolerance level for how far the solution can be from the optimal solution. The team used .01 tolerance for this purpose.
The PuLP library also enables a sensitivity report to be generated while solving the linear program. The team was able to generate this report and use it to provide business recommendations to the NU Indsutries team, which are discussed later in this work.

Computational Experiment and Results

The linear program was able to be solved by Python in under 1 second and using less than .3 Mb of memory. Despite having a large number of variables and constraints, the problem was able to be solved quickly and efficiently.

The maximum revenue given the constraints and variables is $7,097,279.52

Raw material 1 will be split between both plants. Plant A will account for 78256, 55898.1, 73897.1, 68755.1, and 71455.2 raw material 1in periods 1-5 respectively. Plant B will account for 32448, 84101.9, 66102.9, 71244.9, and 68544.8 raw material 1in periods 1-5 respectively.

Raw material 2 will be split between both plants. Plant A will account for 206.4, 535.96, 924.73, 720.59 raw material 2 in periods 1, 3, 4, and 5 respectively. Plant B will account for 1935.2, 5000, 3862.9, 4056.25, and 4279.41 raw material 2 in periods 1-5 respectively. Please see Results Table 1 in the Appendix.

Regular labor hours will be relied on heavily to create the flugels, gadgets, and widgets. Plant B will be used for all flugel production during period 1 (1484), period 2 (3778.7), period 3 (2683.62), period 4 (2241.42), and period 5 (3800). Plant A will be used for all the gadget production during period 1 (2272), period 2 (1725.55), period 3 (1907.95), period 4 (1478.5), and period 5 (1704). The widget labor will be split between both plants. Plant A will be used for period 1 (228), period 3 (592.05), period 4 (1021.5), and period 5 (796), while Plant B will use labor for period 1 (418.6), period 2 (21.3), period 3 (1116.38), and period 4 (1558.58).

Plant B will be utilized for all the overtime labor hours. Overtime will prove beneficial for flugels in periods 4 (193.19) and 5 (152.65). Additionally, overtime labor hours will be used to make widgets in periods 2 (1116.2) and 5 (249.44). Please see Results Table 2 in the Appendix.

The units produced for flugels will all come from plant B at 140, 356.48, 253.18, 229.68, and 372.89 during periods 1-5 respectively. The units produced for gadgets will all come from plant A at, 320, 243.04, 268.73, 208.24, and 240 and during periods 1-5 respectively. The widgets produced will be split between both plants. Plant A will be used for period 1 (24), period 3

(62.32), period 4 (107.53), and period 5 (83.79), while Plant B will be used for period 1 (46), period 2 (125), period 3 (122.68), period 4 (171.27), and period 5 (27.41). Please see Results Table 3 in the Appendix.

Flugel inventory will be stored in Plant A during period 3 (36.52) and period 4 (31.2). Gadget inventory will be stored in both plants in period 1-3. Plant A will hold 70 gadgets in period 1, 63.04 in 2 and 33.48 in 3. Plant B will hold 50 gadgets in period 1 and 3.28 gadgets in 3. Widget inventory will be stored in both plants during period 4. 38.8 widgets will be stored in plant A and 50 in plant B. Please see Results Table 4 in the Appendix.

Advertising will only be profitable for flugels and will happen in periods 1 and 2 at cost of $32,666.70 and $2097.18 respectively.

The integer solution with a tolerance of .01 came very close to the optimal solution, with a profit of $7,071,758.90. The program was able to be solved by Python in 10.3 seconds and using 11.2 Mb of memory. Even with the more difficult task of converging on an integer solution, linear programming with PuLP was still able to come within .01% of the optimal solution in a relatively short amount of time.

Discussion and Conclusions

Ultimately, the linear program was able to converge on an optimal solution for NU Industries' supply chain and marketing strategies in under 1 second. The ability to modify constraints and rerun the program without worrying about limited time or computational resources would be very valuable to any corporation. Any mathematical model is going to be imperfect and based on assumptions that may not always hold true, but the value of the linear program is that it's not only easy to understand at face value, and it's easy (relatively speaking) for a computer to solve. Given the rapidly changing supply chain landscape that the pandemic has created, it is especially valuable for companies to be able to optimize their enterprise operations and also stay flexible to changing conditions. Because you can rerun the program so quickly, it's very easy for data scientists to try different conditions and constraints to reflect the changing situation of their company.

From the sensitivity analysis, we can see that there is room for additional utilization of the advertising budget to potentially generate more revenue as only flugels are being advertised in the first two periods. The activity range shows that there is additional advertising budget available for other periods for flugels, gadgets and widgets. The limiting variable of increasing the advertising is the inventory and the labor. If we could use a portion of the excess advertising budget to increase the labor needed or produce additional inventory, the budget could not only support the advertising demand but also have the ability to produce it. With regards to the raw

materials needed, we see that the labor is the limiting variable as to why the maximum of the activity range is unable to be met. We would propose increasing the labor to overcome this.

Future directions of this project include reporting back our findings to the NU team responsible for making decisions. We could discuss the feasibility of implementing these changes and get their feedback on whether they think our model makes practical sense. Our team could help spearhead an experimental trial to test this idea. Lastly, we could also re-run different scenarios and update the model to reflect real world changes over time.

References

1. Mitchell, Sophie. "Supply chain disruptions may have caused up to $4 trillion in lost revenues." Reuters Events. May 19, 2021. Accessed May 31, 2019. https://www.reutersevents.com/supplychain/supply-chain/supply-chain-disruptions-may-have-caused-4-trillion-lost-revenues

2. Kocaoğlu, Y. , Gümüş, A. T. & Kocaoğlu, B. "Supply Chain Optimization Studies: A Literature Review and Classification." Doğuş Üniversitesi Dergisi. 19 (1). 79-98. 2018. https://dergipark.org.tr/en/pub/doujournal/issue/66678/1043168

3. Özceylan, Eran. Paksoy, Turan. "A Mixed Integer Programming Model for a Closed-loop Supply-chain Network." International Journal of Production Research. 51:3. 718-734. DOI: 10.1080/00207543.2012.661090. 2013. https://www.tandfonline.com/doi/abs/10.1080/00207543.2012.661090

4. Sangaiah, A.K., Tirkolaee, E.B., Goli, A. *et al.* "Robust optimization and mixed-integer linear programming model for LNG supply chain planning problem." *Soft Comput* 24, 7885–7905. https://doi.org/10.1007/s00500-019-04010-6. 2020. https://link.springer.com/article/10.1007/s00500-019-04010-6

5. H. Elleuch, E. Dafaoui, A. Elmhamedi, H. Chabchoub. "Resilience and Vulnerability in Supply Chain: Literature review". IFAC-PapersOnLine, Volume 49, Issue 12. Pages 1448-1453. ISSN 2405-8963. https://doi.org/10.1016/j.ifacol.2016.07.775. 2016. https://www.sciencedirect.com/science/article/pii/S2405896316310564

Appendix

Table 1: Production requirements

Product	Period 1	Period 2	Period 3	Period 4	Period 5
Widgets	70	125	185	190	200
Gadgets	200	300	295	245	240
Flugels	140	175	205	235	230

Table 2: Advertising investment (limited to $70,000)

Product	Period 1-4
Widgets	160
Gadgets	120
Flugels	180

Results Table 1: Raw materials (pounds)

	Raw Material 1		Raw Material 2	
Period	Plant A	Plant B	Plant A	Plant B
1	78,256	32,448	206.4	935.2
2	55,898.1	84,101.9		5000
3	73,897.1	66,102.9	535.96	3862.9
4	68,755.1	71,244.9	924.73	4,056.25
5	71,455.2	68,544.8	720.59	4,279.41

Results Table 2: Labor (hours)

	Regular Labor		Overtime Labor	
Period	Plant A	Plant B	Plant A	Plant B
1	2,272 G 228 W	1,484 F 418.6 W		
2	1,725.6 G	3,778.7 F 21.3 W		1,116.2 W
3	1,908 G 592.1 W	2,683.6 F 1,116.4 W		
4	1,478.5 G 1,021.5 W	2,241.4 F 1558.6 W		193.2 F
5	1,704 G 796 W	3,800 F		152.7 F 249.4 W

Results Table 3: Units Produced

	Units Produced	
Period	Plant A	Plant B
1	320 G 24 W	140 F 46 W
2	243 G	356.5 F 125 W
3	268.7 G 63.3 W	253.2 F 122.7 W
4	208.2 G 107.5 W	229.7 F 171.3 W
5	240 G 83.8 W	372.9 F 27.4 W

Results Table 4: Inventory

	Inventory	
Period	Plant A	Plant B
1	70 G	50 G
2	63 G	
3	33.5 G 36.5 F	3.3 G
4	31.2 F 38.8 W	50 W
5		

Source Code:

```
from pulp import LpVariable, LpProblem,LpMaximize,LpStatus,value, GLPK
import pandas as pd
# Units produced at plant a
u_w1_a = LpVariable("u_w1_a",0,None)
u_w2_a = LpVariable("u_w2_a",0,None)
u_w3_a = LpVariable("u_w3_a",0,None)
u_w4_a = LpVariable("u_w4_a",0,None)
u_w5_a = LpVariable("u_w5_a",0,None)
u_g1_a = LpVariable("u_g1_a",0,None)
u_g2_a = LpVariable("u_g2_a",0,None)
u_g3_a = LpVariable("u_g3_a",0,None)
u_g4_a = LpVariable("u_g4_a",0,None)
u_g5_a = LpVariable("u_g5_a",0,None)
u_f1_a = LpVariable("u_f1_a",0,None)
u_f2_a = LpVariable("u_f2_a",0,None)
```

```
u_f3_a = LpVariable("u_f3_a",0,None)
u_f4_a = LpVariable("u_f4_a",0,None)
u_f5_a = LpVariable("u_f5_a",0,None)
# Units produced at plant b
u_w1_b = LpVariable("u_w1_b",0,None)
u_w2_b = LpVariable("u_w2_b",0,None)
u_w3_b = LpVariable("u_w3_b",0,None)
u_w4_b = LpVariable("u_w4_b",0,None)
u_w5_b = LpVariable("u_w5_b",0,None)
u_g1_b = LpVariable("u_g1_b",0,None)
u_g2_b = LpVariable("u_g2_b",0,None)
u_g3_b = LpVariable("u_g3_b",0,None)
u_g4_b = LpVariable("u_g4_b",0,None)
u_g5_b = LpVariable("u_g5_b",0,None)
u_f1_b = LpVariable("u_f1_b",0,None)
u_f2_b = LpVariable("u_f2_b",0,None)
u_f3_b = LpVariable("u_f3_b",0,None)
u_f4_b = LpVariable("u_f4_b",0,None)
u_f5_b = LpVariable("u_f5_b",0,None)
# advertising
a_w1 = LpVariable("a_w1",0,None)
a_w2 = LpVariable("a_w2",0,None)
a_w3 = LpVariable("a_w3",0,None)
a_w4 = LpVariable("a_w4",0,None)
a_g1 = LpVariable("a_g1",0,None)
a_g2 = LpVariable("a_g2",0,None)
a_g3 = LpVariable("a_g3",0,None)
a_g4 = LpVariable("a_g4",0,None)
a_f1 = LpVariable("a_f1",0,None)
a_f2 = LpVariable("a_f2",0,None)
a_f3 = LpVariable("a_f3",0,None)
a_f4 = LpVariable("a_f4",0,None)
# labor regular a
rl_w1_a = LpVariable("rl_w1_a",0,None)
rl_w2_a = LpVariable("rl_w2_a",0,None)
rl_w3_a = LpVariable("rl_w3_a",0,None)
rl_w4_a = LpVariable("rl_w4_a",0,None)
rl_w5_a = LpVariable("rl_w5_a",0,None)
rl_g1_a = LpVariable("rl_g1_a",0,None)
rl_g2_a = LpVariable("rl_g2_a",0,None)
```

```
rl_g3_a = LpVariable("rl_g3_a",0,None)
rl_g4_a = LpVariable("rl_g4_a",0,None)
rl_g5_a = LpVariable("rl_g5_a",0,None)
rl_f1_a = LpVariable("rl_f1_a",0,None)
rl_f2_a = LpVariable("rl_f2_a",0,None)
rl_f3_a = LpVariable("rl_f3_a",0,None)
rl_f4_a = LpVariable("rl_f4_a",0,None)
rl_f5_a = LpVariable("rl_f5_a",0,None)
# labor overtime a
ol_w1_a = LpVariable("ol_w1_a",0,None)
ol_w2_a = LpVariable("ol_w2_a",0,None)
ol_w3_a = LpVariable("ol_w3_a",0,None)
ol_w4_a = LpVariable("ol_w4_a",0,None)
ol_w5_a = LpVariable("ol_w5_a",0,None)
ol_g1_a = LpVariable("ol_g1_a",0,None)
ol_g2_a = LpVariable("ol_g2_a",0,None)
ol_g3_a = LpVariable("ol_g3_a",0,None)
ol_g4_a = LpVariable("ol_g4_a",0,None)
ol_g5_a = LpVariable("ol_g5_a",0,None)
ol_f1_a = LpVariable("ol_f1_a",0,None)
ol_f2_a = LpVariable("ol_f2_a",0,None)
ol_f3_a = LpVariable("ol_f3_a",0,None)
ol_f4_a = LpVariable("ol_f4_a",0,None)
ol_f5_a = LpVariable("ol_f5_a",0,None)
# inventory a
i_w1_a = LpVariable("i_w1_a",0,None)
i_w2_a = LpVariable("i_w2_a",0,None)
i_w3_a = LpVariable("i_w3_a",0,None)
i_w4_a = LpVariable("i_w4_a",0,None)
i_g1_a = LpVariable("i_g1_a",0,None)
i_g2_a = LpVariable("i_g2_a",0,None)
i_g3_a = LpVariable("i_g3_a",0,None)
i_g4_a = LpVariable("i_g4_a",0,None)
i_f1_a = LpVariable("i_f1_a",0,None)
i_f2_a = LpVariable("i_f2_a",0,None)
i_f3_a = LpVariable("i_f3_a",0,None)
i_f4_a = LpVariable("i_f4_a",0,None)
# labor regular b
rl_w1_b = LpVariable("rl_w1_b",0,None)
rl_w2_b = LpVariable("rl_w2_b",0,None)
```

```
rl_w3_b = LpVariable("rl_w3_b",0,None)
rl_w4_b = LpVariable("rl_w4_b",0,None)
rl_w5_b = LpVariable("rl_w5_b",0,None)
rl_g1_b = LpVariable("rl_g1_b",0,None)
rl_g2_b = LpVariable("rl_g2_b",0,None)
rl_g3_b = LpVariable("rl_g3_b",0,None)
rl_g4_b = LpVariable("rl_g4_b",0,None)
rl_g5_b = LpVariable("rl_g5_b",0,None)
rl_f1_b = LpVariable("rl_f1_b",0,None)
rl_f2_b = LpVariable("rl_f2_b",0,None)
rl_f3_b = LpVariable("rl_f3_b",0,None)
rl_f4_b = LpVariable("rl_f4_b",0,None)
rl_f5_b = LpVariable("rl_f5_b",0,None)
# labor overtime b
ol_w1_b = LpVariable("ol_w1_b",0,None)
ol_w2_b = LpVariable("ol_w2_b",0,None)
ol_w3_b = LpVariable("ol_w3_b",0,None)
ol_w4_b = LpVariable("ol_w4_b",0,None)
ol_w5_b = LpVariable("ol_w5_b",0,None)
ol_g1_b = LpVariable("ol_g1_b",0,None)
ol_g2_b = LpVariable("ol_g2_b",0,None)
ol_g3_b = LpVariable("ol_g3_b",0,None)
ol_g4_b = LpVariable("ol_g4_b",0,None)
ol_g5_b = LpVariable("ol_g5_b",0,None)
ol_f1_b = LpVariable("ol_f1_b",0,None)
ol_f2_b = LpVariable("ol_f2_b",0,None)
ol_f3_b = LpVariable("ol_f3_b",0,None)
ol_f4_b = LpVariable("ol_f4_b",0,None)
ol_f5_b = LpVariable("ol_f5_b",0,None)
# inventory b
i_w1_b = LpVariable("i_w1_b",0,None)
i_w2_b = LpVariable("i_w2_b",0,None)
i_w3_b = LpVariable("i_w3_b",0,None)
i_w4_b = LpVariable("i_w4_b",0,None)
i_g1_b = LpVariable("i_g1_b",0,None)
i_g2_b = LpVariable("i_g2_b",0,None)
i_g3_b = LpVariable("i_g3_b",0,None)
i_g4_b = LpVariable("i_g4_b",0,None)
i_f1_b = LpVariable("i_f1_b",0,None)
i_f2_b = LpVariable("i_f2_b",0,None)
```

```
i_f3_b = LpVariable("i_f3_b",0,None)
i_f4_b = LpVariable("i_f4_b",0,None)
# raw materials a
rm1_1_a = LpVariable("rm1_1_a",0,None)
rm1_2_a = LpVariable("rm1_2_a",0,None)
rm1_3_a = LpVariable("rm1_3_a",0,None)
rm1_4_a = LpVariable("rm1_4_a",0,None)
rm1_5_a = LpVariable("rm1_5_a",0,None)
rm2_1_a = LpVariable("rm2_1_a",0,None)
rm2_2_a = LpVariable("rm2_2_a",0,None)
rm2_3_a = LpVariable("rm2_3_a",0,None)
rm2_4_a = LpVariable("rm2_4_a",0,None)
rm2_5_a = LpVariable("rm2_5_a",0,None)
# rmb
rm1_1_b = LpVariable("rm1_1_b",0,None)
rm1_2_b = LpVariable("rm1_2_b",0,None)
rm1_3_b = LpVariable("rm1_3_b",0,None)
rm1_4_b = LpVariable("rm1_4_b",0,None)
rm1_5_b = LpVariable("rm1_5_b",0,None)
rm2_1_b = LpVariable("rm2_1_b",0,None)
rm2_2_b = LpVariable("rm2_2_b",0,None)
rm2_3_b = LpVariable("rm2_3_b",0,None)
rm2_4_b = LpVariable("rm2_4_b",0,None)
rm2_5_b = LpVariable("rm2_5_b",0,None)

prob = LpProblem("problem",LpMaximize)
# demand and advertising constraints
# widgets
prob += u_w1_a+u_w1_b >= 70
prob += i_w1_a + i_w1_b <= (u_w1_a+u_w1_b) - (70)
prob += i_w1_a + i_w1_b >= (u_w1_a+u_w1_b) - (70)
prob += u_w2_a+u_w2_b + i_w1_a + i_w1_b >=125 + a_w1*(1/160)
prob += i_w2_a + i_w2_b >= (u_w2_a+u_w2_b  + i_w1_a + i_w1_b) - (125 + a_w1*(1/160))
prob += i_w2_a + i_w2_b <= (u_w2_a+u_w2_b  + i_w1_a + i_w1_b) - (125 + a_w1*(1/160))
prob += u_w3_a+u_w3_b + i_w2_a + i_w2_b  >= 185 + a_w2*(1/160)
prob += i_w3_a + i_w3_b >= (u_w3_a+u_w3_b + i_w2_a + i_w2_b) - (185 + a_w2*(1/160))
prob += i_w3_a + i_w3_b <= (u_w3_a+u_w3_b + i_w2_a + i_w2_b) - (185 + a_w2*(1/160))
prob += u_w4_a+u_w4_b + i_w3_a + i_w3_b  >= 190 + a_w3*(1/160)
prob += i_w4_a + i_w4_b >= (u_w4_a+u_w4_b + i_w3_a + i_w3_b) - (190 + a_w3*(1/160))
prob += i_w4_a + i_w4_b <= (u_w4_a+u_w4_b + i_w3_a + i_w3_b) - (190 + a_w3*(1/160))
```

```
prob += u_w5_a+u_w5_b + i_w4_a + i_w4_b  >= 200 + a_w4*(1/160)
# gadgets
prob += u_g1_a+u_g1_b >= 200
prob += i_g1_a + i_g1_b >= (u_g1_a+u_g1_b) - (200)
prob += i_g1_a + i_g1_b <= (u_g1_a+u_g1_b) - (200)
prob += u_g2_a+u_g2_b + i_g1_a + i_g1_b >= 300 + a_g1*(1/120)
prob += i_g2_a + i_g2_b >= (u_g2_a+u_g2_b + i_g1_a + i_g1_b) - (300 + a_g1*(1/120))
prob += i_g2_a + i_g2_b <= (u_g2_a+u_g2_b + i_g1_a + i_g1_b) - (300 + a_g1*(1/120))
prob += u_g3_a+u_g3_b + i_g2_a + i_g2_b >= 295 + a_g2*(1/120)
prob += i_g3_a + i_g3_b >= (u_g3_a+u_g3_b + i_g2_a + i_g2_b) - (295 + a_g2*(1/120))
prob += i_g3_a + i_g3_b <= (u_g3_a+u_g3_b + i_g2_a + i_g2_b) - (295 + a_g2*(1/120))
prob += u_g4_a+u_g4_b + i_g3_a + i_g3_b >= 245 + a_g3*(1/120)
prob += i_g4_a + i_g4_b >= (u_g4_a+u_g4_b + i_g3_a + i_g3_b) - (245 + a_g3*(1/120))
prob += i_g4_a + i_g4_b <= (u_g4_a+u_g4_b + i_g3_a + i_g3_b) - (245 + a_g3*(1/120))
prob += u_g5_a+u_g5_b + i_g4_a + i_g4_b >= 240 + a_g4*(1/120)
# flugels
prob += u_f1_a+u_f1_b >= 140
prob += i_f1_a + i_f1_b >= (u_f1_a+u_f1_b) - (140)
prob += i_f1_a + i_f1_b <= (u_f1_a+u_f1_b) - (140)
prob += u_f2_a+u_f2_b + i_f1_a + i_f1_b >= 175 + a_f1*(1/180)
prob += i_f2_a + i_f2_b >= (u_f2_a+u_f2_b + i_f1_a + i_f1_b) - (175 + a_f1*(1/180))
prob += i_f2_a + i_f2_b <= (u_f2_a+u_f2_b + i_f1_a + i_f1_b) - (175 + a_f1*(1/180))
prob += u_f3_a+u_f3_b + i_f2_a + i_f2_b >= 205 + a_f2*(1/180)
prob += i_f3_a + i_f3_b >= (u_f3_a+u_f3_b + i_f2_a + i_f2_b) - (205 + a_f2*(1/180))
prob += i_f3_a + i_f3_b <= (u_f3_a+u_f3_b + i_f2_a + i_f2_b) - (205 + a_f2*(1/180))
prob += u_f4_a+u_f4_b + i_f3_a + i_f3_b >= 235 + a_f3*(1/180)
prob += i_f4_a + i_f4_b >= (u_f4_a+u_f4_b + i_f3_a + i_f3_b) - (235 + a_f3*(1/180))
prob += i_f4_a + i_f4_b <= (u_f4_a+u_f4_b + i_f3_a + i_f3_b) - (235 + a_f3*(1/180))
prob += u_f5_a+u_f5_b + i_f4_a + i_f4_b >= 230 + a_f4*(1/180)
prob += (a_w1+a_w2+a_w3+a_w4) + (a_g1+a_g2+a_g3+a_g4) + (a_f1+a_f2+a_f3+a_f4) <=
70000
# labor a
prob += rl_w1_a + rl_g1_a + rl_f1_a <= 2500
prob += rl_w2_a + rl_g2_a + rl_f2_a <= 2500
prob += rl_w3_a + rl_g3_a + rl_f3_a <= 2500
prob += rl_w4_a + rl_g4_a + rl_f4_a <= 2500
prob += rl_w5_a + rl_g5_a + rl_f5_a <= 2500
# inventory a
prob += i_w1_a+i_g1_a+i_f1_a <= 70
prob += i_w2_a+i_g2_a+i_f2_a <= 70
```

```
prob += i_w3_a+i_g3_a+i_f3_a <= 70
prob += i_w4_a+i_g4_a+i_f4_a <= 70
# labor b
prob += rl_w1_b + rl_g1_b + rl_f1_b <= 3800
prob += rl_w2_b + rl_g2_b + rl_f2_b <= 3800
prob += rl_w3_b + rl_g3_b + rl_f3_b <= 3800
prob += rl_w4_b + rl_g4_b + rl_f4_b <= 3800
prob += rl_w5_b + rl_g5_b + rl_f5_b <= 3800
# inventory b
prob += i_w1_b+i_g1_b+i_f1_b <= 50
prob += i_w2_b+i_g2_b+i_f2_b <= 50
prob += i_w3_b+i_g3_b+i_f3_b <= 50
prob += i_w4_b+i_g4_b+i_f4_b <= 50
# raw materials
prob += rm1_1_a + rm1_1_b <= 70*2000
prob += rm1_2_a + rm1_2_b <= 70*2000
prob += rm1_3_a + rm1_3_b <= 70*2000
prob += rm1_4_a + rm1_4_b <= 70*2000
prob += rm1_5_a + rm1_5_b <= 70*2000
prob += rm2_1_a + rm2_1_b <= 2.5*2000
prob += rm2_2_a + rm2_2_b <= 2.5*2000
prob += rm2_3_a + rm2_3_b <= 2.5*2000
prob += rm2_4_a + rm2_4_b <= 2.5*2000
prob += rm2_5_a + rm2_5_b <= 2.5*2000
# production at plant a
# raw materials for rm1 and rm2
prob += 194*u_w1_a + 230*u_g1_a + 178*u_f1_a <= rm1_1_a
prob += 194*u_w2_a + 230*u_g2_a + 178*u_f2_a <= rm1_2_a
prob += 194*u_w3_a + 230*u_g3_a + 178*u_f3_a <= rm1_3_a
prob += 194*u_w4_a + 230*u_g4_a + 178*u_f4_a <= rm1_4_a
prob += 194*u_w5_a + 230*u_g5_a + 178*u_f5_a <= rm1_5_a
prob += 8.6*u_w1_a + 0*u_g1_a + 11.6*u_f1_a <= rm2_1_a
prob += 8.6*u_w2_a + 0*u_g2_a + 11.6*u_f2_a <= rm2_2_a
prob += 8.6*u_w3_a + 0*u_g3_a + 11.6*u_f3_a <= rm2_3_a
prob += 8.6*u_w4_a + 0*u_g4_a + 11.6*u_f4_a <= rm2_4_a
prob += 8.6*u_w5_a + 0*u_g5_a + 11.6*u_f5_a <= rm2_5_a
# labor for w/g/f
prob += 9.5*u_w1_a <= rl_w1_a + ol_w1_a
prob += 9.5*u_w2_a <= rl_w2_a + ol_w2_a
prob += 9.5*u_w3_a <= rl_w3_a + ol_w3_a
```

```
prob += 9.5*u_w4_a <= rl_w4_a + ol_w4_a
prob += 9.5*u_w5_a <= rl_w5_a + ol_w5_a
prob += 7.1*u_g1_a <= rl_g1_a + ol_g1_a
prob += 7.1*u_g2_a <= rl_g2_a + ol_g2_a
prob += 7.1*u_g3_a <= rl_g3_a + ol_g3_a
prob += 7.1*u_g4_a <= rl_g4_a + ol_g4_a
prob += 7.1*u_g5_a <= rl_g5_a + ol_g5_a
prob += 11.1*u_f1_a <= rl_f1_a + ol_f1_a
prob += 11.1*u_f2_a <= rl_f2_a + ol_f2_a
prob += 11.1*u_f3_a <= rl_f3_a + ol_f3_a
prob += 11.1*u_f4_a <= rl_f4_a + ol_f4_a
prob += 11.1*u_f5_a <= rl_f5_a + ol_f5_a
# production at plant b
# raw materials for rm1 and rm2
prob += 188*u_w1_b + 225*u_g1_b + 170*u_f1_b <= rm1_1_b
prob += 188*u_w2_b + 225*u_g2_b + 170*u_f2_b <= rm1_2_b
prob += 188*u_w3_b + 225*u_g3_b + 170*u_f3_b <= rm1_3_b
prob += 188*u_w4_b + 225*u_g4_b + 170*u_f4_b <= rm1_4_b
prob += 188*u_w5_b + 225*u_g5_b + 170*u_f5_b <= rm1_5_b
prob += 9.2*u_w1_b + 0*u_g1_b + 10.8*u_f1_b <= rm2_1_b
prob += 9.2*u_w2_b + 0*u_g2_b + 10.8*u_f2_b <= rm2_2_b
prob += 9.2*u_w3_b + 0*u_g3_b + 10.8*u_f3_b <= rm2_3_b
prob += 9.2*u_w4_b + 0*u_g4_b + 10.8*u_f4_b <= rm2_4_b
prob += 9.2*u_w5_b + 0*u_g5_b + 10.8*u_f5_b <= rm2_5_b
# labor for w/g/f
prob += 9.1*u_w1_b <= rl_w1_b + ol_w1_b
prob += 9.1*u_w2_b <= rl_w2_b + ol_w2_b
prob += 9.1*u_w3_b <= rl_w3_b + ol_w3_b
prob += 9.1*u_w4_b <= rl_w4_b + ol_w4_b
prob += 9.1*u_w5_b <= rl_w5_b + ol_w5_b
prob += 7.8*u_g1_b <= rl_g1_b + ol_g1_b
prob += 7.8*u_g2_b <= rl_g2_b + ol_g2_b
prob += 7.8*u_g3_b <= rl_g3_b + ol_g3_b
prob += 7.8*u_g4_b <= rl_g4_b + ol_g4_b
prob += 7.8*u_g5_b <= rl_g5_b + ol_g5_b
prob += 10.6*u_f1_b <= rl_f1_b + ol_f1_b
prob += 10.6*u_f2_b <= rl_f2_b + ol_f2_b
prob += 10.6*u_f3_b <= rl_f3_b + ol_f3_b
prob += 10.6*u_f4_b <= rl_f4_b + ol_f4_b
prob += 10.6*u_f5_b <= rl_f5_b + ol_f5_b
```

```
revenue = 2490*(u_w1_a+u_w2_a+u_w3_a+u_w4_a+u_w5_a +
u_w1_b+u_w2_b+u_w3_b+u_w4_b+u_w5_b) +
1990*(u_g1_a+u_g2_a+u_g3_a+u_g4_a+u_g5_a + u_g1_b+u_g2_b+u_g3_b+u_g4_b+u_g5_b)
+ 2970*(u_f1_a+u_f2_a+u_f3_a+u_f4_a+u_f5_a + u_f1_b+u_f2_b+u_f3_b+u_f4_b+u_f5_b)
ad_cost = (a_w1+a_w2+a_w3+a_w4) + (a_g1+a_g2+a_g3+a_g4) + (a_f1+a_f2+a_f3+a_f4)
labor_a_cost = 11*((rl_w1_a + rl_g1_a + rl_f1_a) + (rl_w2_a + rl_g2_a + rl_f2_a))+
16.5*((ol_w1_a + ol_g1_a + ol_f1_a) + (ol_w2_a + ol_g2_a + ol_f2_a)) + 11*1.05*((rl_w3_a +
rl_g3_a + rl_f3_a) + (rl_w4_a + rl_g4_a + rl_f4_a) + (rl_w5_a + rl_g5_a + rl_f5_a)) +
16.5*1.05*((ol_w3_a + ol_g3_a + ol_f3_a) + (ol_w4_a + ol_g4_a + ol_f4_a) + (ol_w5_a +
ol_g5_a + ol_f5_a))
inv_a_cost = 7.5*(i_w1_a+i_w2_a+i_w3_a+i_w4_a) + 5.5*(i_g1_a+i_g2_a+i_g3_a+i_g4_a) +
6.5*(i_f1_a+i_f2_a+i_f3_a+i_f4_a)
labor_b_cost = 11*((rl_w1_b + rl_g1_b + rl_f1_b) + (rl_w2_b + rl_g2_b + rl_f2_b)) +
16.5*((ol_w1_b + ol_g1_b + ol_f1_b) + (ol_w2_b + ol_g2_b + ol_f2_b)) + 11*1.10*((rl_w3_b +
rl_g3_b + rl_f3_b) + (rl_w4_b + rl_g4_b + rl_f4_b) + (rl_w5_b + rl_g5_b + rl_f5_b)) +
16.5*1.10*((ol_w3_b + ol_g3_b + ol_f3_b) + (ol_w4_b + ol_g4_b + ol_f4_b) + (ol_w5_b +
ol_g5_b + ol_f5_b))
inv_b_cost = 7.8*(i_w1_b+i_w2_b+i_w3_b+i_w4_b) + 5.7*(i_g1_b+i_g2_b+i_g3_b+i_g4_b) +
7.0*(i_f1_b+i_f2_b+i_f3_b+i_f4_b)
material_cost = 1.25*(rm1_1_a + rm1_2_a + rm1_3_a + rm1_4_a + rm1_5_a) + 2.65*(rm2_1_a
+ rm2_2_a + rm2_3_a + rm2_4_a + rm2_5_a) + 1.45*(rm1_1_b + rm1_2_b + rm1_3_b +
rm1_4_b + rm1_5_b) + 2.9*(rm2_1_b + rm2_2_b + rm2_3_b + rm2_4_b + rm2_5_b)
delivery_cost = 6.30*(u_w1_a+u_w2_a+u_w3_a+u_w4_a+u_w5_a) +
6.50*(u_w1_b+u_w2_b+u_w3_b+u_w4_b+u_w5_b) +
4.60*(u_g1_a+u_g2_a+u_g3_a+u_g4_a+u_g5_a) +
5.00*(u_g1_b+u_g2_b+u_g3_b+u_g4_b+u_g5_b) +
5.50*(u_f1_a+u_f2_a+u_f3_a+u_f4_a+u_f5_a) +
5.70*(u_f1_b+u_f2_b+u_f3_b+u_f4_b+u_f5_b)

prob += revenue - ad_cost - labor_a_cost - inv_a_cost - labor_b_cost - inv_b_cost -
material_cost - delivery_cost

prob.writeLP("NUIndustriesLP.txt")
prob.solve(GLPK(options=['--ranges', 'NUIndustriesSen.txt']))
print("Status: ", LpStatus[prob.status])

print("Profit: $", value(prob.objective))
```

Source Code for Integer Solution:

```
import pulp
u_w1_a = LpVariable("u_w1_a",0,None,cat="Integer")
u_w2_a = LpVariable("u_w2_a",0,None,cat="Integer")
u_w3_a = LpVariable("u_w3_a",0,None,cat="Integer")
u_w4_a = LpVariable("u_w4_a",0,None,cat="Integer")
u_w5_a = LpVariable("u_w5_a",0,None,cat="Integer")
u_g1_a = LpVariable("u_g1_a",0,None,cat="Integer")
u_g2_a = LpVariable("u_g2_a",0,None,cat="Integer")
u_g3_a = LpVariable("u_g3_a",0,None,cat="Integer")
u_g4_a = LpVariable("u_g4_a",0,None,cat="Integer")
u_g5_a = LpVariable("u_g5_a",0,None,cat="Integer")
u_f1_a = LpVariable("u_f1_a",0,None,cat="Integer")
u_f2_a = LpVariable("u_f2_a",0,None,cat="Integer")
u_f3_a = LpVariable("u_f3_a",0,None,cat="Integer")
u_f4_a = LpVariable("u_f4_a",0,None,cat="Integer")
u_f5_a = LpVariable("u_f5_a",0,None,cat="Integer")
# Units produced at plant b
u_w1_b = LpVariable("u_w1_b",0,None,cat="Integer")
u_w2_b = LpVariable("u_w2_b",0,None,cat="Integer")
u_w3_b = LpVariable("u_w3_b",0,None,cat="Integer")
u_w4_b = LpVariable("u_w4_b",0,None,cat="Integer")
u_w5_b = LpVariable("u_w5_b",0,None,cat="Integer")
u_g1_b = LpVariable("u_g1_b",0,None,cat="Integer")
u_g2_b = LpVariable("u_g2_b",0,None,cat="Integer")
u_g3_b = LpVariable("u_g3_b",0,None,cat="Integer")
u_g4_b = LpVariable("u_g4_b",0,None,cat="Integer")
u_g5_b = LpVariable("u_g5_b",0,None,cat="Integer")
u_f1_b = LpVariable("u_f1_b",0,None,cat="Integer")
u_f2_b = LpVariable("u_f2_b",0,None,cat="Integer")
u_f3_b = LpVariable("u_f3_b",0,None,cat="Integer")
u_f4_b = LpVariable("u_f4_b",0,None,cat="Integer")
u_f5_b = LpVariable("u_f5_b",0,None,cat="Integer")
# advertising
a_w1 = LpVariable("a_w1",0,None,cat="Integer")
a_w2 = LpVariable("a_w2",0,None,cat="Integer")
a_w3 = LpVariable("a_w3",0,None,cat="Integer")
a_w4 = LpVariable("a_w4",0,None,cat="Integer")
a_g1 = LpVariable("a_g1",0,None,cat="Integer")
```

```
a_g2 = LpVariable("a_g2",0,None,cat="Integer")
a_g3 = LpVariable("a_g3",0,None,cat="Integer")
a_g4 = LpVariable("a_g4",0,None,cat="Integer")
a_f1 = LpVariable("a_f1",0,None,cat="Integer")
a_f2 = LpVariable("a_f2",0,None,cat="Integer")
a_f3 = LpVariable("a_f3",0,None,cat="Integer")
a_f4 = LpVariable("a_f4",0,None,cat="Integer")
# labor regular a
rl_w1_a = LpVariable("rl_w1_a",0,None,cat="Integer")
rl_w2_a = LpVariable("rl_w2_a",0,None,cat="Integer")
rl_w3_a = LpVariable("rl_w3_a",0,None,cat="Integer")
rl_w4_a = LpVariable("rl_w4_a",0,None,cat="Integer")
rl_w5_a = LpVariable("rl_w5_a",0,None,cat="Integer")
rl_g1_a = LpVariable("rl_g1_a",0,None,cat="Integer")
rl_g2_a = LpVariable("rl_g2_a",0,None,cat="Integer")
rl_g3_a = LpVariable("rl_g3_a",0,None,cat="Integer")
rl_g4_a = LpVariable("rl_g4_a",0,None,cat="Integer")
rl_g5_a = LpVariable("rl_g5_a",0,None,cat="Integer")
rl_f1_a = LpVariable("rl_f1_a",0,None,cat="Integer")
rl_f2_a = LpVariable("rl_f2_a",0,None,cat="Integer")
rl_f3_a = LpVariable("rl_f3_a",0,None,cat="Integer")
rl_f4_a = LpVariable("rl_f4_a",0,None,cat="Integer")
rl_f5_a = LpVariable("rl_f5_a",0,None,cat="Integer")
# labor overtime a
ol_w1_a = LpVariable("ol_w1_a",0,None,cat="Integer")
ol_w2_a = LpVariable("ol_w2_a",0,None,cat="Integer")
ol_w3_a = LpVariable("ol_w3_a",0,None,cat="Integer")
ol_w4_a = LpVariable("ol_w4_a",0,None,cat="Integer")
ol_w5_a = LpVariable("ol_w5_a",0,None,cat="Integer")
ol_g1_a = LpVariable("ol_g1_a",0,None,cat="Integer")
ol_g2_a = LpVariable("ol_g2_a",0,None,cat="Integer")
ol_g3_a = LpVariable("ol_g3_a",0,None,cat="Integer")
ol_g4_a = LpVariable("ol_g4_a",0,None,cat="Integer")
ol_g5_a = LpVariable("ol_g5_a",0,None,cat="Integer")
ol_f1_a = LpVariable("ol_f1_a",0,None,cat="Integer")
ol_f2_a = LpVariable("ol_f2_a",0,None,cat="Integer")
ol_f3_a = LpVariable("ol_f3_a",0,None,cat="Integer")
ol_f4_a = LpVariable("ol_f4_a",0,None,cat="Integer")
ol_f5_a = LpVariable("ol_f5_a",0,None,cat="Integer")
# inventory a
```

```
i_w1_a = LpVariable("i_w1_a",0,None,cat="Integer")
i_w2_a = LpVariable("i_w2_a",0,None,cat="Integer")
i_w3_a = LpVariable("i_w3_a",0,None,cat="Integer")
i_w4_a = LpVariable("i_w4_a",0,None,cat="Integer")
i_g1_a = LpVariable("i_g1_a",0,None,cat="Integer")
i_g2_a = LpVariable("i_g2_a",0,None,cat="Integer")
i_g3_a = LpVariable("i_g3_a",0,None,cat="Integer")
i_g4_a = LpVariable("i_g4_a",0,None,cat="Integer")
i_f1_a = LpVariable("i_f1_a",0,None,cat="Integer")
i_f2_a = LpVariable("i_f2_a",0,None,cat="Integer")
i_f3_a = LpVariable("i_f3_a",0,None,cat="Integer")
i_f4_a = LpVariable("i_f4_a",0,None,cat="Integer")
# labor regular b
rl_w1_b = LpVariable("rl_w1_b",0,None,cat="Integer")
rl_w2_b = LpVariable("rl_w2_b",0,None,cat="Integer")
rl_w3_b = LpVariable("rl_w3_b",0,None,cat="Integer")
rl_w4_b = LpVariable("rl_w4_b",0,None,cat="Integer")
rl_w5_b = LpVariable("rl_w5_b",0,None,cat="Integer")
rl_g1_b = LpVariable("rl_g1_b",0,None,cat="Integer")
rl_g2_b = LpVariable("rl_g2_b",0,None,cat="Integer")
rl_g3_b = LpVariable("rl_g3_b",0,None,cat="Integer")
rl_g4_b = LpVariable("rl_g4_b",0,None,cat="Integer")
rl_g5_b = LpVariable("rl_g5_b",0,None,cat="Integer")
rl_f1_b = LpVariable("rl_f1_b",0,None,cat="Integer")
rl_f2_b = LpVariable("rl_f2_b",0,None,cat="Integer")
rl_f3_b = LpVariable("rl_f3_b",0,None,cat="Integer")
rl_f4_b = LpVariable("rl_f4_b",0,None,cat="Integer")
rl_f5_b = LpVariable("rl_f5_b",0,None,cat="Integer")
# labor overtime b
ol_w1_b = LpVariable("ol_w1_b",0,None,cat="Integer")
ol_w2_b = LpVariable("ol_w2_b",0,None,cat="Integer")
ol_w3_b = LpVariable("ol_w3_b",0,None,cat="Integer")
ol_w4_b = LpVariable("ol_w4_b",0,None,cat="Integer")
ol_w5_b = LpVariable("ol_w5_b",0,None,cat="Integer")
ol_g1_b = LpVariable("ol_g1_b",0,None,cat="Integer")
ol_g2_b = LpVariable("ol_g2_b",0,None,cat="Integer")
ol_g3_b = LpVariable("ol_g3_b",0,None,cat="Integer")
ol_g4_b = LpVariable("ol_g4_b",0,None,cat="Integer")
ol_g5_b = LpVariable("ol_g5_b",0,None,cat="Integer")
ol_f1_b = LpVariable("ol_f1_b",0,None,cat="Integer")
```

```
ol_f2_b = LpVariable("ol_f2_b",0,None,cat="Integer")
ol_f3_b = LpVariable("ol_f3_b",0,None,cat="Integer")
ol_f4_b = LpVariable("ol_f4_b",0,None,cat="Integer")
ol_f5_b = LpVariable("ol_f5_b",0,None,cat="Integer")
# inventory b
i_w1_b = LpVariable("i_w1_b",0,None,cat="Integer")
i_w2_b = LpVariable("i_w2_b",0,None,cat="Integer")
i_w3_b = LpVariable("i_w3_b",0,None,cat="Integer")
i_w4_b = LpVariable("i_w4_b",0,None,cat="Integer")
i_g1_b = LpVariable("i_g1_b",0,None,cat="Integer")
i_g2_b = LpVariable("i_g2_b",0,None,cat="Integer")
i_g3_b = LpVariable("i_g3_b",0,None,cat="Integer")
i_g4_b = LpVariable("i_g4_b",0,None,cat="Integer")
i_f1_b = LpVariable("i_f1_b",0,None,cat="Integer")
i_f2_b = LpVariable("i_f2_b",0,None,cat="Integer")
i_f3_b = LpVariable("i_f3_b",0,None,cat="Integer")
i_f4_b = LpVariable("i_f4_b",0,None,cat="Integer")
# raw materials a
rm1_1_a = LpVariable("rm1_1_a",0,None,cat="Integer")
rm1_2_a = LpVariable("rm1_2_a",0,None,cat="Integer")
rm1_3_a = LpVariable("rm1_3_a",0,None,cat="Integer")
rm1_4_a = LpVariable("rm1_4_a",0,None,cat="Integer")
rm1_5_a = LpVariable("rm1_5_a",0,None,cat="Integer")
rm2_1_a = LpVariable("rm2_1_a",0,None,cat="Integer")
rm2_2_a = LpVariable("rm2_2_a",0,None,cat="Integer")
rm2_3_a = LpVariable("rm2_3_a",0,None,cat="Integer")
rm2_4_a = LpVariable("rm2_4_a",0,None,cat="Integer")
rm2_5_a = LpVariable("rm2_5_a",0,None,cat="Integer")
# rmb
rm1_1_b = LpVariable("rm1_1_b",0,None,cat="Integer")
rm1_2_b = LpVariable("rm1_2_b",0,None,cat="Integer")
rm1_3_b = LpVariable("rm1_3_b",0,None,cat="Integer")
rm1_4_b = LpVariable("rm1_4_b",0,None,cat="Integer")
rm1_5_b = LpVariable("rm1_5_b",0,None,cat="Integer")
rm2_1_b = LpVariable("rm2_1_b",0,None,cat="Integer")
rm2_2_b = LpVariable("rm2_2_b",0,None,cat="Integer")
rm2_3_b = LpVariable("rm2_3_b",0,None,cat="Integer")
rm2_4_b = LpVariable("rm2_4_b",0,None,cat="Integer")
rm2_5_b = LpVariable("rm2_5_b",0,None,cat="Integer")
```

```
prob1 = LpProblem("problem1",LpMaximize)
# demand and advertising constraints
# widgets
prob1 += u_w1_a+u_w1_b >= 70
prob1 += i_w1_a + i_w1_b <= (u_w1_a+u_w1_b) - (70)
prob1 += i_w1_a + i_w1_b >= (u_w1_a+u_w1_b) - (70)
prob1 += u_w2_a+u_w2_b + i_w1_a + i_w1_b >=125 + a_w1*(1/160)
prob1 += i_w2_a + i_w2_b >= (u_w2_a+u_w2_b  + i_w1_a + i_w1_b) - (125 + a_w1*(1/160))
prob1 += i_w2_a + i_w2_b <= (u_w2_a+u_w2_b  + i_w1_a + i_w1_b) - (125 + a_w1*(1/160))
prob1 += u_w3_a+u_w3_b + i_w2_a + i_w2_b  >= 185 + a_w2*(1/160)
prob1 += i_w3_a + i_w3_b >= (u_w3_a+u_w3_b + i_w2_a + i_w2_b) - (185 + a_w2*(1/160))
prob1 += i_w3_a + i_w3_b <= (u_w3_a+u_w3_b + i_w2_a + i_w2_b) - (185 + a_w2*(1/160))
prob1 += u_w4_a+u_w4_b + i_w3_a + i_w3_b  >= 190 + a_w3*(1/160)
prob1 += i_w4_a + i_w4_b >= (u_w4_a+u_w4_b + i_w3_a + i_w3_b) - (190 + a_w3*(1/160))
prob1 += i_w4_a + i_w4_b <= (u_w4_a+u_w4_b + i_w3_a + i_w3_b) - (190 + a_w3*(1/160))
prob1 += u_w5_a+u_w5_b + i_w4_a + i_w4_b  >= 200 + a_w4*(1/160)
# gadgets
prob1 += u_g1_a+u_g1_b >= 200
prob1 += i_g1_a + i_g1_b >= (u_g1_a+u_g1_b) - (200)
prob1 += i_g1_a + i_g1_b <= (u_g1_a+u_g1_b) - (200)
prob1 += u_g2_a+u_g2_b + i_g1_a + i_g1_b >= 300 + a_g1*(1/120)
prob1 += i_g2_a + i_g2_b >= (u_g2_a+u_g2_b + i_g1_a + i_g1_b) - (300 + a_g1*(1/120))
prob1 += i_g2_a + i_g2_b <= (u_g2_a+u_g2_b + i_g1_a + i_g1_b) - (300 + a_g1*(1/120))
prob1 += u_g3_a+u_g3_b + i_g2_a + i_g2_b >= 295 + a_g2*(1/120)
prob1 += i_g3_a + i_g3_b >= (u_g3_a+u_g3_b + i_g2_a + i_g2_b) - (295 + a_g2*(1/120))
prob1 += i_g3_a + i_g3_b <= (u_g3_a+u_g3_b + i_g2_a + i_g2_b) - (295 + a_g2*(1/120))
prob1 += u_g4_a+u_g4_b + i_g3_a + i_g3_b >= 245 + a_g3*(1/120)
prob1 += i_g4_a + i_g4_b >= (u_g4_a+u_g4_b + i_g3_a + i_g3_b) - (245 + a_g3*(1/120))
prob1 += i_g4_a + i_g4_b <= (u_g4_a+u_g4_b + i_g3_a + i_g3_b) - (245 + a_g3*(1/120))
prob1 += u_g5_a+u_g5_b + i_g4_a + i_g4_b >= 240 + a_g4*(1/120)
# flugels
prob1 += u_f1_a+u_f1_b >= 140
prob1 += i_f1_a + i_f1_b >= (u_f1_a+u_f1_b) - (140)
prob1 += i_f1_a + i_f1_b <= (u_f1_a+u_f1_b) - (140)
prob1 += u_f2_a+u_f2_b + i_f1_a + i_f1_b >= 175 + a_f1*(1/180)
prob1 += i_f2_a + i_f2_b >= (u_f2_a+u_f2_b + i_f1_a + i_f1_b) - (175 + a_f1*(1/180))
prob1 += i_f2_a + i_f2_b <= (u_f2_a+u_f2_b + i_f1_a + i_f1_b) - (175 + a_f1*(1/180))
prob1 += u_f3_a+u_f3_b + i_f2_a + i_f2_b >= 205 + a_f2*(1/180)
prob1 += i_f3_a + i_f3_b >= (u_f3_a+u_f3_b + i_f2_a + i_f2_b) - (205 + a_f2*(1/180))
prob1 += i_f3_a + i_f3_b <= (u_f3_a+u_f3_b + i_f2_a + i_f2_b) - (205 + a_f2*(1/180))
```

```
prob1 += u_f4_a+u_f4_b + i_f3_a + i_f3_b >= 235 + a_f3*(1/180)
prob1 += i_f4_a + i_f4_b >= (u_f4_a+u_f4_b + i_f3_a + i_f3_b) - (235 + a_f3*(1/180))
prob1 += i_f4_a + i_f4_b <= (u_f4_a+u_f4_b + i_f3_a + i_f3_b) - (235 + a_f3*(1/180))
prob1 += u_f5_a+u_f5_b + i_f4_a + i_f4_b >= 230 + a_f4*(1/180)
prob1 += (a_w1+a_w2+a_w3+a_w4) + (a_g1+a_g2+a_g3+a_g4) + (a_f1+a_f2+a_f3+a_f4) <=
70000
# labor a
prob1 += rl_w1_a + rl_g1_a + rl_f1_a <= 2500
prob1 += rl_w2_a + rl_g2_a + rl_f2_a <= 2500
prob1 += rl_w3_a + rl_g3_a + rl_f3_a <= 2500
prob1 += rl_w4_a + rl_g4_a + rl_f4_a <= 2500
prob1 += rl_w5_a + rl_g5_a + rl_f5_a <= 2500
# inventory a
prob1 += i_w1_a+i_g1_a+i_f1_a <= 70
prob1 += i_w2_a+i_g2_a+i_f2_a <= 70
prob1 += i_w3_a+i_g3_a+i_f3_a <= 70
prob1 += i_w4_a+i_g4_a+i_f4_a <= 70
# labor b
prob1 += rl_w1_b + rl_g1_b + rl_f1_b <= 3800
prob1 += rl_w2_b + rl_g2_b + rl_f2_b <= 3800
prob1 += rl_w3_b + rl_g3_b + rl_f3_b <= 3800
prob1 += rl_w4_b + rl_g4_b + rl_f4_b <= 3800
prob1 += rl_w5_b + rl_g5_b + rl_f5_b <= 3800
# inventory b
prob1 += i_w1_b+i_g1_b+i_f1_b <= 50
prob1 += i_w2_b+i_g2_b+i_f2_b <= 50
prob1 += i_w3_b+i_g3_b+i_f3_b <= 50
prob1 += i_w4_b+i_g4_b+i_f4_b <= 50
# raw materials
prob1 += rm1_1_a + rm1_1_b <= 70*2000
prob1 += rm1_2_a + rm1_2_b <= 70*2000
prob1 += rm1_3_a + rm1_3_b <= 70*2000
prob1 += rm1_4_a + rm1_4_b <= 70*2000
prob1 += rm1_5_a + rm1_5_b <= 70*2000
prob1 += rm2_1_a + rm2_1_b <= 2.5*2000
prob1 += rm2_2_a + rm2_2_b <= 2.5*2000
prob1 += rm2_3_a + rm2_3_b <= 2.5*2000
prob1 += rm2_4_a + rm2_4_b <= 2.5*2000
prob1 += rm2_5_a + rm2_5_b <= 2.5*2000
# production at plant a
```

```
# raw materials for rm1 and rm2
prob1 += 194*u_w1_a + 230*u_g1_a + 178*u_f1_a <= rm1_1_a
prob1 += 194*u_w2_a + 230*u_g2_a + 178*u_f2_a <= rm1_2_a
prob1 += 194*u_w3_a + 230*u_g3_a + 178*u_f3_a <= rm1_3_a
prob1 += 194*u_w4_a + 230*u_g4_a + 178*u_f4_a <= rm1_4_a
prob1 += 194*u_w5_a + 230*u_g5_a + 178*u_f5_a <= rm1_5_a
prob1 += 8.6*u_w1_a + 0*u_g1_a + 11.6*u_f1_a <= rm2_1_a
prob1 += 8.6*u_w2_a + 0*u_g2_a + 11.6*u_f2_a <= rm2_2_a
prob1 += 8.6*u_w3_a + 0*u_g3_a + 11.6*u_f3_a <= rm2_3_a
prob1 += 8.6*u_w4_a + 0*u_g4_a + 11.6*u_f4_a <= rm2_4_a
prob1 += 8.6*u_w5_a + 0*u_g5_a + 11.6*u_f5_a <= rm2_5_a
# labor for w/g/f
prob1 += 9.5*u_w1_a <= rl_w1_a + ol_w1_a
prob1 += 9.5*u_w2_a <= rl_w2_a + ol_w2_a
prob1 += 9.5*u_w3_a <= rl_w3_a + ol_w3_a
prob1 += 9.5*u_w4_a <= rl_w4_a + ol_w4_a
prob1 += 9.5*u_w5_a <= rl_w5_a + ol_w5_a
prob1 += 7.1*u_g1_a <= rl_g1_a + ol_g1_a
prob1 += 7.1*u_g2_a <= rl_g2_a + ol_g2_a
prob1 += 7.1*u_g3_a <= rl_g3_a + ol_g3_a
prob1 += 7.1*u_g4_a <= rl_g4_a + ol_g4_a
prob1 += 7.1*u_g5_a <= rl_g5_a + ol_g5_a
prob1 += 11.1*u_f1_a <= rl_f1_a + ol_f1_a
prob1 += 11.1*u_f2_a <= rl_f2_a + ol_f2_a
prob1 += 11.1*u_f3_a <= rl_f3_a + ol_f3_a
prob1 += 11.1*u_f4_a <= rl_f4_a + ol_f4_a
prob1 += 11.1*u_f5_a <= rl_f5_a + ol_f5_a
# production at plant b
# raw materials for rm1 and rm2
prob1 += 188*u_w1_b + 225*u_g1_b + 170*u_f1_b <= rm1_1_b
prob1 += 188*u_w2_b + 225*u_g2_b + 170*u_f2_b <= rm1_2_b
prob1 += 188*u_w3_b + 225*u_g3_b + 170*u_f3_b <= rm1_3_b
prob1 += 188*u_w4_b + 225*u_g4_b + 170*u_f4_b <= rm1_4_b
prob1 += 188*u_w5_b + 225*u_g5_b + 170*u_f5_b <= rm1_5_b
prob1 += 9.2*u_w1_b + 0*u_g1_b + 10.8*u_f1_b <= rm2_1_b
prob1 += 9.2*u_w2_b + 0*u_g2_b + 10.8*u_f2_b <= rm2_2_b
prob1 += 9.2*u_w3_b + 0*u_g3_b + 10.8*u_f3_b <= rm2_3_b
prob1 += 9.2*u_w4_b + 0*u_g4_b + 10.8*u_f4_b <= rm2_4_b
prob1 += 9.2*u_w5_b + 0*u_g5_b + 10.8*u_f5_b <= rm2_5_b
# labor for w/g/f
```

```
prob1 += 9.1*u_w1_b <= rl_w1_b + ol_w1_b
prob1 += 9.1*u_w2_b <= rl_w2_b + ol_w2_b
prob1 += 9.1*u_w3_b <= rl_w3_b + ol_w3_b
prob1 += 9.1*u_w4_b <= rl_w4_b + ol_w4_b
prob1 += 9.1*u_w5_b <= rl_w5_b + ol_w5_b
prob1 += 7.8*u_g1_b <= rl_g1_b + ol_g1_b
prob1 += 7.8*u_g2_b <= rl_g2_b + ol_g2_b
prob1 += 7.8*u_g3_b <= rl_g3_b + ol_g3_b
prob1 += 7.8*u_g4_b <= rl_g4_b + ol_g4_b
prob1 += 7.8*u_g5_b <= rl_g5_b + ol_g5_b
prob1 += 10.6*u_f1_b <= rl_f1_b + ol_f1_b
prob1 += 10.6*u_f2_b <= rl_f2_b + ol_f2_b
prob1 += 10.6*u_f3_b <= rl_f3_b + ol_f3_b
prob1 += 10.6*u_f4_b <= rl_f4_b + ol_f4_b
prob1 += 10.6*u_f5_b <= rl_f5_b + ol_f5_b

revenue = 2490*(u_w1_a+u_w2_a+u_w3_a+u_w4_a+u_w5_a +
u_w1_b+u_w2_b+u_w3_b+u_w4_b+u_w5_b) +
1990*(u_g1_a+u_g2_a+u_g3_a+u_g4_a+u_g5_a + u_g1_b+u_g2_b+u_g3_b+u_g4_b+u_g5_b)
+ 2970*(u_f1_a+u_f2_a+u_f3_a+u_f4_a+u_f5_a + u_f1_b+u_f2_b+u_f3_b+u_f4_b+u_f5_b)
ad_cost = (a_w1+a_w2+a_w3+a_w4) + (a_g1+a_g2+a_g3+a_g4) + (a_f1+a_f2+a_f3+a_f4)
labor_a_cost = 11*((rl_w1_a + rl_g1_a + rl_f1_a) + (rl_w2_a + rl_g2_a + rl_f2_a))+
16.5*((ol_w1_a + ol_g1_a + ol_f1_a) + (ol_w2_a + ol_g2_a + ol_f2_a)) + 11*1.05*((rl_w3_a +
rl_g3_a + rl_f3_a) + (rl_w4_a + rl_g4_a + rl_f4_a) + (rl_w5_a + rl_g5_a + rl_f5_a)) +
16.5*1.05*((ol_w3_a + ol_g3_a + ol_f3_a) + (ol_w4_a + ol_g4_a + ol_f4_a) + (ol_w5_a +
ol_g5_a + ol_f5_a))
inv_a_cost = 7.5*(i_w1_a+i_w2_a+i_w3_a+i_w4_a) + 5.5*(i_g1_a+i_g2_a+i_g3_a+i_g4_a) +
6.5*(i_f1_a+i_f2_a+i_f3_a+i_f4_a)
labor_b_cost = 11*((rl_w1_b + rl_g1_b + rl_f1_b) + (rl_w2_b + rl_g2_b + rl_f2_b)) +
16.5*((ol_w1_b + ol_g1_b + ol_f1_b) + (ol_w2_b + ol_g2_b + ol_f2_b)) + 11*1.10*((rl_w3_b +
rl_g3_b + rl_f3_b) + (rl_w4_b + rl_g4_b + rl_f4_b) + (rl_w5_b + rl_g5_b + rl_f5_b)) +
16.5*1.10*((ol_w3_b + ol_g3_b + ol_f3_b) + (ol_w4_b + ol_g4_b + ol_f4_b) + (ol_w5_b +
ol_g5_b + ol_f5_b))
inv_b_cost = 7.8*(i_w1_b+i_w2_b+i_w3_b+i_w4_b) + 5.7*(i_g1_b+i_g2_b+i_g3_b+i_g4_b) +
7.0*(i_f1_b+i_f2_b+i_f3_b+i_f4_b)
material_cost = 1.25*(rm1_1_a + rm1_2_a + rm1_3_a + rm1_4_a + rm1_5_a) + 2.65*(rm2_1_a
+ rm2_2_a + rm2_3_a + rm2_4_a + rm2_5_a) + 1.45*(rm1_1_b + rm1_2_b + rm1_3_b +
rm1_4_b + rm1_5_b) + 2.9*(rm2_1_b + rm2_2_b + rm2_3_b + rm2_4_b + rm2_5_b)
delivery_cost = 6.30*(u_w1_a+u_w2_a+u_w3_a+u_w4_a+u_w5_a) +
6.50*(u_w1_b+u_w2_b+u_w3_b+u_w4_b+u_w5_b) +
```

```
4.60*(u_g1_a+u_g2_a+u_g3_a+u_g4_a+u_g5_a) +
5.00*(u_g1_b+u_g2_b+u_g3_b+u_g4_b+u_g5_b) +
5.50*(u_f1_a+u_f2_a+u_f3_a+u_f4_a+u_f5_a) +
5.70*(u_f1_b+u_f2_b+u_f3_b+u_f4_b+u_f5_b)

prob1 += revenue - ad_cost - labor_a_cost - inv_a_cost - labor_b_cost - inv_b_cost -
material_cost - delivery_cost

solver = pulp.GLPK(options=['--mipgap', '0.01'])
prob1.solve(solver)
print("Status: ", LpStatus[prob1.status])

print("Profit: $", value(prob1.objective))
```