MIE262_Warp_Shoe_Project

Authors' information: Tianhao_Huang_huan2546_1007659856, Jialuo_Chen_chenj926_1008897419

Abstract:

Warp Shoe Company is one of Canada's oldest shoe companies, and they believe that it is occasionally profitable to make the University of Toronto's Industrial Engineering student to compute production plans. Hence this report will go through the methodology of constructing an optimal production plan of 2006's February. Through the Gurobi solver, an optimal profit of [x] was obtained, along with the production quantity for each shoe type. Furthermore, this report will answer the questions proposed in Section Result. For example, the change in optimal solution under the addition of certain constraint upper bounds.

Introduction:

The shoe company WARP seeks to capitalize on a competitor's bankruptcy by doubling its February demand. To answer the consulting requirement of WARP, the team considered a wide range of data, including the production demand, various machines, raw material, labor cost, and sales price, and combined them to make a AMPL model. KMore specifically, the relevant data are read from Microsoft Access tables. Furthermore, the team implemented linear programming and integer programming knowledge on the model to make the recommendation on the WARP's production plan for February of 2006.

Methodology:

The team formulated a model that consists of an objective function and sets of constraints, based on assumptions clarified. Specifically, the .mod file includes all the variables, objective function, and constraints. And the .dat file reads the data provided by the Access tables, and stores them in the model. Furthermore, the .run file solves the model with Gurobi.

Assumptions:

The team assumed that the WARP company would not produce shoes more than demand since it is possible for those shoes to become stock rather than cash flow. And considering that the transportation cost and operation cost of the warehouse is ignored, means it is possible to transport shoes to other warehouses without any cost, thus the total warehouse capacity is the only factor we are going to consider for the warehouse. The team also assumed that the revenue comes from the sale of shoes; the cost all comes from shoe production which only includes raw material cost, workers' salary, machine operation cost and the cost of failing to meet the demand. For machine operation time, the team used the average operation time provided in the table and assumed that there would be no failure when making shoes. We also assume that the WARP company would not purchase extra raw materials in any form.

Objective Function:

The model's objective function is made considering the summation of revenue through sales and the cost for shoes' production, which includes raw material cost, human resource cost, machine operation cost and the cost of failing to meet the demand. Since we assumed no failure while production, thus the cost for failure on production is excluded.

Constraint:

Five constraints are included based on the assumption and concern, which are raw material budget, raw material capacity, the machine and workers' production time limit, the demand and the warehouse capacity.

Results:

We created our objective function and constraints for our model after analyzing the information provided in the instructions and generated the .dat file to read table the data provided. After that, we runned our AMPL using the .run file and the two .mod and .dat files we wrote before, and implemented these data into our output file to answer the questions. The result we got as the array of x[s] has some 0 element. This is due to the reason that some of the shoes are relatively unprofitable compared with other types of shoes, and as a result of Linear Programming they are treated as degenerated solutions. The objective function value profit is 13466400 after rounding, this is a approximation we have based on the assumptions of the overall circumstances.

The detailed result outputs are shown in our outputfile.

Questions:

Question 1. How should you estimate the demand for the month of February?

We collected all of the product demand data for each shoe type from February during the year 1997 - 2003. Then we calculate the average of each year, and double this data to get our estimated demand for the month of February.

Question 2. How many variables and constraints do you have?

We have one variable x{S} and five constraints.

Question 3. If you had to relax your integer program to an LP, how many constraints were violated

After rounding the LP solution to the closest integer solution?

The amount of constraints before rounding (LP):

- Amount of budget constraint violated: 0
- Amount of raw material capacity constraint violated: 0
- Amount of machine production time limit constraint violated: 1
- Amount of approximate demand constraint violated: 0
- Amount of warehouse capacity constraint violated: 0

The amount of constraints after rounding (IP):

- Amount of budget constraint violated: 0
- Amount of raw material capacity constraint violated: 0
- Amount of machine production time limit constraint violated: 1
- Amount of approximate demand constraint violated: 0
- Amount of warehouse capacity constraint violated: 0

Question 4.Which constraints are binding, and what is the real-world interpretation of those binding constraints?

The Raw material capacity and the Demand constraints were binding. In total 83 raw materials and demand for 56 shoes were binded, details are shown below. This means that the following material is consumed quicker than any other materials and the following 56 types of shoes are the most profitable.

Budget:

The Budget constraint is not binding!

RawMaterialCapacity:

The Raw material 1's capacity constraint is binding!
The Raw material 5's capacity constraint is binding!
The Raw material 7's capacity constraint is binding!
The Raw material 10's capacity constraint is binding!
The Raw material 11's capacity constraint is binding!
The Raw material 12's capacity constraint is binding!
The Raw material 15's capacity constraint is binding!
The Raw material 18's capacity constraint is binding!
The Raw material 19's capacity constraint is binding!
The Raw material 21's capacity constraint is binding!
The Raw material 26's capacity constraint is binding!
The Raw material 30's capacity constraint is binding!
The Raw material 31's capacity constraint is binding!
The Raw material 35's capacity constraint is binding!
The Raw material 38's capacity constraint is binding!
The Raw material 40's capacity constraint is binding!
The Raw material 41's capacity constraint is binding!
The Raw material 43's capacity constraint is binding!
The Raw material 47's capacity constraint is binding!
The Raw material 50's capacity constraint is binding!
The Raw material 53's capacity constraint is binding!
The Raw material 56's capacity constraint is binding!
The Raw material 57's capacity constraint is binding!
The Raw material 60's capacity constraint is binding!
The Raw material 61's capacity constraint is binding!
The Raw material 63's capacity constraint is binding!
The Raw material 64's capacity constraint is binding!
The Raw material 65's capacity constraint is binding!
The Raw material 71's capacity constraint is binding!
The Raw material 73's capacity constraint is binding!
The Raw material 76's capacity constraint is binding!
The Raw material 77's capacity constraint is binding!
The Raw material 79's capacity constraint is binding!

The Raw material 80's capacity constraint is binding!
The Raw material 81's capacity constraint is binding!
The Raw material 82's capacity constraint is binding!
The Raw material 83's capacity constraint is binding!
The Raw material 84's capacity constraint is binding!
The Raw material 85's capacity constraint is binding!
The Raw material 87's capacity constraint is binding!
The Raw material 91's capacity constraint is binding!
The Raw material 94's capacity constraint is binding!
The Raw material 97's capacity constraint is binding!
The Raw material 98's capacity constraint is binding!
The Raw material 101's capacity constraint is binding!
The Raw material 103's capacity constraint is binding!
The Raw material 104's capacity constraint is binding!
The Raw material 105's capacity constraint is binding!
The Raw material 106's capacity constraint is binding!
The Raw material 108's capacity constraint is binding!
The Raw material 113's capacity constraint is binding!
The Raw material 115's capacity constraint is binding!
The Raw material 116's capacity constraint is binding!
The Raw material 117's capacity constraint is binding!
The Raw material 118's capacity constraint is binding!
The Raw material 119's capacity constraint is binding!
The Raw material 120's capacity constraint is binding!
The Raw material 123's capacity constraint is binding!
The Raw material 125's capacity constraint is binding!
The Raw material 126's capacity constraint is binding!
The Raw material 127's capacity constraint is binding!
The Raw material 129's capacity constraint is binding!
The Raw material 132's capacity constraint is binding!
The Raw material 133's capacity constraint is binding!
The Raw material 134's capacity constraint is binding!
The Raw material 135's capacity constraint is binding!
The Raw material 136's capacity constraint is binding!
The Raw material 137's capacity constraint is binding!
The Raw material 138's capacity constraint is binding!
The Raw material 140's capacity constraint is binding!
The Raw material 141's capacity constraint is binding!
The Raw material 143's capacity constraint is binding!
The Raw material 144's capacity constraint is binding!
The Raw material 145's capacity constraint is binding!
The Raw material 147's capacity constraint is binding!
The Raw material 151's capacity constraint is binding!
The Raw material 153's capacity constraint is binding!

The Raw material 154's capacity constraint is binding!
The Raw material 155's capacity constraint is binding!
The Raw material 158's capacity constraint is binding!
The Raw material 159's capacity constraint is binding!
The Raw material 161's capacity constraint is binding!
The Raw material 163's capacity constraint is binding!

Total:

There are 83 Raw material capacity constraints are binding!

MachineOperationTime:

None of the Machine Operation Time constraints are binding!

Demand:

Demand of shoe type SH002's constraint is binding!
Demand of shoe type SH027's constraint is binding!
Demand of shoe type SH032's constraint is binding!
Demand of shoe type SH034's constraint is binding!
Demand of shoe type SH044's constraint is binding!
Demand of shoe type SH050's constraint is binding!
Demand of shoe type SH058's constraint is binding!
Demand of shoe type SH062's constraint is binding!
Demand of shoe type SH063's constraint is binding!
Demand of shoe type SH087's constraint is binding!
Demand of shoe type SH095's constraint is binding!
Demand of shoe type SH096's constraint is binding!
Demand of shoe type SH105's constraint is binding!
Demand of shoe type SH121's constraint is binding!
Demand of shoe type SH142's constraint is binding!
Demand of shoe type SH144's constraint is binding!
Demand of shoe type SH146's constraint is binding!
Demand of shoe type SH165's constraint is binding!
Demand of shoe type SH197's constraint is binding!
Demand of shoe type SH201's constraint is binding!
Demand of shoe type SH221's constraint is binding!
Demand of shoe type SH223's constraint is binding!
Demand of shoe type SH224's constraint is binding!
Demand of shoe type SH226's constraint is binding!
Demand of shoe type SH234's constraint is binding!
Demand of shoe type SH235's constraint is binding!
Demand of shoe type SH247's constraint is binding!
Demand of shoe type SH253's constraint is binding!
Demand of shoe type SH261's constraint is binding!
Demand of shoe type SH262's constraint is binding!
Demand of shoe type SH282's constraint is binding!
Demand of shoe type SH300's constraint is binding!

Demand of shoe type SH314's constraint is binding!
Demand of shoe type SH320's constraint is binding!
Demand of shoe type SH325's constraint is binding!
Demand of shoe type SH329's constraint is binding!
Demand of shoe type SH348's constraint is binding!
Demand of shoe type SH361's constraint is binding!
Demand of shoe type SH367's constraint is binding!
Demand of shoe type SH369's constraint is binding!
Demand of shoe type SH378's constraint is binding!
Demand of shoe type SH407's constraint is binding!
Demand of shoe type SH415's constraint is binding!
Demand of shoe type SH417's constraint is binding!
Demand of shoe type SH427's constraint is binding!
Demand of shoe type SH430's constraint is binding!
Demand of shoe type SH447's constraint is binding!
Demand of shoe type SH453's constraint is binding!
Demand of shoe type SH456's constraint is binding!
Demand of shoe type SH466's constraint is binding!
Demand of shoe type SH477's constraint is binding!
Demand of shoe type SH479's constraint is binding!
Demand of shoe type SH490's constraint is binding!
Demand of shoe type SH511's constraint is binding!
Demand of shoe type SH516's constraint is binding!
Demand of shoe type SH534's constraint is binding!

Total:

56 shoe types' constraints are binding!

WareHouseCapacity:

None of the warehouse constraints are binding!

Question 5:

WareHouseCapacity.dual = 0

Means the product will not fill the total capacity of the warehouse, thus we would not need extra warehouse space.

The optimized objective function value is:

Profit = 13466400

Question 6:Imagine that machines were available for only 8 hours per day. How would your solution change? Which constraints are binding now? Does the new solution seem realistic to you?

The objective function value and x[s] value stays the same, and the raw material capacity, machine production time, and approximate demand are binding now. This seems realistic to me since the constraint of machine operation time shown in Q4 was not binding, indicating that it has a slackness. This slackness allows the obj function to be optimal as we changed available machine hours to 8.

Here is Q6 in our output file:
New Constraints are now:

The amount of constraints before rounding (LP):
Amount of budget constraint violated: 0
Amount of raw material capacity constraint violated: 56
Amount of machine production time limit constraint violated: 72
Amount of approximate demand constraint violated: 20
Amount of warehouse capacity constraint violated: 0

The amount of constraints after rounding (IP):
Amount of budget constraint violated: 0
Amount of raw material capacity constraint violated: 55
Amount of machine production time limit constraint violated: 72
Amount of approximate demand constraint violated: 39
Amount of warehouse capacity constraint violated: 0

x [*] :=
SH001 0 SH141 0 SH281 307.907 SH421 416.571
SH002 448.857 SH142 398.857 SH282 415.143 SH422 0
SH003 0 SH143 0 SH283 0 SH423 419.017
SH004 0 SH144 448.857 SH284 460.889 SH424 0
SH005 33.1429 SH145 0 SH285 0 SH425 458.571
SH006 0 SH146 489.143 SH286 436.548 SH426 0
SH007 0 SH147 68.4137 SH287 410.286 SH427 477.429
SH008 0 SH148 0 SH288 162.086 SH428 0
SH009 27 SH149 88.383 SH289 429.829 SH429 27.3735
SH010 0 SH150 425.429 SH290 129.57 SH430 430
SH011 433.429 SH151 0 SH291 0 SH431 0
SH012 289.045 SH152 0 SH292 136.365 SH432 0
SH013 425.429 SH153 129.82 SH293 0 SH433 0
SH014 0 SH154 437.143 SH294 0 SH434 0
SH015 0 SH155 0 SH295 403.429 SH435 245.975
SH016 0 SH156 0 SH296 0 SH436 0
SH017 0 SH157 0 SH297 0 SH437 0
SH018 247.219 SH158 0 SH298 110.829 SH438 96.8
SH019 200.088 SH159 0 SH299 408.014 SH439 0
SH020 0 SH160 21 SH300 427.429 SH440 0
SH021 0 SH161 0 SH301 0 SH441 0
SH022 0 SH162 0 SH302 0 SH442 0

SH023 0 SH163 0 SH303 0 SH443 0
SH024 399.379 SH164 0 SH304 0 SH444 175.467
SH025 404.286 SH165 460 SH305 0 SH445 462.857
SH026 0 SH166 0 SH306 0 SH446 0
SH027 434 SH167 0 SH307 81.2943 SH447 458.857
SH028 0 SH168 379.926 SH308 266.427 SH448 0
SH029 410.216 SH169 0 SH309 0 SH449 209.982
SH030 0 SH170 0 SH310 0 SH450 0
SH031 305.622 SH171 0 SH311 0 SH451 455.429
SH032 490.857 SH172 82.3521 SH312 476.857 SH452 0
SH033 0 SH173 308.491 SH313 483.429 SH453 470.857
SH034 372.571 SH174 0 SH314 391.143 SH454 0
SH035 0 SH175 0 SH315 0 SH455 0.515306
SH036 0 SH176 450.286 SH316 168.419 SH456 430
SH037 0 SH177 0 SH317 316.143 SH457 252.964
SH038 478.571 SH178 139.775 SH318 0 SH458 218.811
SH039 389.429 SH179 107.645 SH319 0 SH459 265.488
SH040 432.857 SH180 0 SH320 462.286 SH460 0
SH041 338.738 SH181 0 SH321 0 SH461 0
SH042 215.137 SH182 0 SH322 290.699 SH462 181.706
SH043 0 SH183 10.7182 SH323 0 SH463 0
SH044 442.286 SH184 105.163 SH324 0 SH464 403.218
SH045 374.306 SH185 370.617 SH325 475.143 SH465 270.282
SH046 0 SH186 68.0572 SH326 0 SH466 440.857
SH047 410.571 SH187 114.793 SH327 0 SH467 126.223
SH048 292.214 SH188 246.232 SH328 194.248 SH468 141.204
SH049 0 SH189 0 SH329 400 SH469 0
SH050 437.429 SH190 0 SH330 0 SH470 379.233
SH051 0 SH191 80.6177 SH331 0 SH471 394.489
SH052 0 SH192 0 SH332 0 SH472 292.349
SH053 0 SH193 0 SH333 0 SH473 194.343
SH054 0 SH194 303.543 SH334 386.571 SH474 281.762
SH055 0 SH195 243.066 SH335 0 SH475 0
SH056 0 SH196 0 SH336 0 SH476 416.857
SH057 204.81 SH197 437.429 SH337 0 SH477 422.571
SH058 432 SH198 0 SH338 109.157 SH478 354.294
SH059 7.67893 SH199 0 SH339 0 SH479 488.286
SH060 191.181 SH200 0 SH340 439.435 SH480 0
SH061 167.912 SH201 476.286 SH341 429.429 SH481 0
SH062 458.286 SH202 0 SH342 299.33 SH482 112.234
SH063 446 SH203 500.571 SH343 30.1097 SH483 434.571
SH064 18.2534 SH204 0 SH344 0 SH484 0
SH065 369.039 SH205 144.365 SH345 317.786 SH485 0
SH066 0 SH206 0 SH346 0 SH486 0

SH067 0 SH207 0 SH347 176.703 SH487 83.1042
SH068 0 SH208 0 SH348 488.857 SH488 0
SH069 0 SH209 436.423 SH349 40.6795 SH489 441.429
SH070 0 SH210 0 SH350 0 SH490 537.143
SH071 338.729 SH211 0 SH351 0 SH491 183.511
SH072 6.87916 SH212 0 SH352 396.571 SH492 39.3634
SH073 345.834 SH213 436.857 SH353 0 SH493 0
SH074 0 SH214 0 SH354 0 SH494 64.9048
SH075 0 SH215 0 SH355 0 SH495 136.333
SH076 64.9008 SH216 0 SH356 186.22 SH496 0
SH077 176.86 SH217 205.84 SH357 0 SH497 450.286
SH078 407.143 SH218 0 SH358 0 SH498 239.48
SH079 195.042 SH219 0 SH359 20.7196 SH499 104.378
SH080 409.429 SH220 0 SH360 0 SH500 425.429
SH081 163.884 SH221 468.286 SH361 417.429 SH501 420.571
SH082 288.762 SH222 2.78749 SH362 419.429 SH502 0
SH083 0 SH223 409.143 SH363 0 SH503 0
SH084 0 SH224 442.571 SH364 0 SH504 0
SH085 0 SH225 403.874 SH365 259.804 SH505 0
SH086 0 SH226 418.857 SH366 197.702 SH506 0
SH087 430 SH227 256.222 SH367 452.286 SH507 98.867
SH088 423.429 SH228 0 SH368 0 SH508 415.95
SH089 0 SH229 0 SH369 378 SH509 0
SH090 474.571 SH230 69.6662 SH370 0 SH510 0
SH091 0 SH231 0 SH371 0 SH511 398
SH092 0 SH232 418.972 SH372 273.751 SH512 30.4036
SH093 284.527 SH233 448.571 SH373 416.571 SH513 327.342
SH094 307.216 SH234 405.143 SH374 0 SH514 396.857
SH095 422.286 SH235 470.857 SH375 0 SH515 0
SH096 452.571 SH236 496.857 SH376 24.0941 SH516 425.143
SH097 0 SH237 0 SH377 148.671 SH517 0
SH098 0 SH238 396.465 SH378 422.571 SH518 0
SH099 0 SH239 109.914 SH379 254.092 SH519 0
SH100 0 SH240 107.109 SH380 0 SH520 123.142
SH101 0 SH241 418.571 SH381 17.8297 SH521 231.57
SH102 0 SH242 0 SH382 0 SH522 54.1516
SH103 0 SH243 0 SH383 0 SH523 134.922
SH104 0 SH244 241.934 SH384 504.571 SH524 96.5308
SH105 418 SH245 0 SH385 0 SH525 152.446
SH106 201.096 SH246 0 SH386 0 SH526 0
SH107 1.4354 SH247 422 SH387 0 SH527 258.751
SH108 0 SH248 78.5367 SH388 0.215179 SH528 0
SH109 0 SH249 405.429 SH389 0 SH529 0
SH110 417.143 SH250 0 SH390 0 SH530 0

SH111 0 SH251 0 SH391 127.685 SH531 0
SH112 0 SH252 0 SH392 0 SH532 475.429
SH113 454.286 SH253 381.143 SH393 0 SH533 0
SH114 0 SH254 479.429 SH394 0 SH534 406.286
SH115 0 SH255 202.693 SH395 448.376 SH535 440.571
SH116 199.755 SH256 434.286 SH396 0 SH536 31.9008
SH117 20.5311 SH257 0 SH397 0 SH537 280.044
SH118 293.557 SH258 0 SH398 0 SH538 0
SH119 0 SH259 0 SH399 442.857 SH539 0
SH120 0 SH260 1.93737 SH400 0 SH540 170.324
SH121 508.286 SH261 409.143 SH401 0 SH541 3.88484
SH122 0 SH262 411.143 SH402 43.8644 SH542 0
SH123 0 SH263 0 SH403 0 SH543 332.093
SH124 0 SH264 0 SH404 0 SH544 0
SH125 0 SH265 413.143 SH405 0 SH545 0
SH126 75.602 SH266 0 SH406 0 SH546 275.725
SH127 381.295 SH267 0 SH407 437.429 SH547 0
SH128 0 SH268 0 SH408 0 SH548 0
SH129 0 SH269 429.429 SH409 0 SH549 61.0804
SH130 93.6652 SH270 76.1212 SH410 262.168 SH550 0
SH131 54.726 SH271 0 SH411 399.904 SH551 0
SH132 0 SH272 0 SH412 87.6778 SH552 181.33
SH133 134.866 SH273 267.014 SH413 0 SH553 0
SH134 490.286 SH274 119.008 SH414 0 SH554 0
SH135 0 SH275 0 SH415 438.286 SH555 0
SH136 0 SH276 279.884 SH416 0 SH556 423.143
SH137 0 SH277 422.857 SH417 482 SH557 470.286
SH138 0 SH278 0 SH418 343.171
SH139 0 SH279 18.8891 SH419 289.92
SH140 0 SH280 0 SH420 89.7077
;
The optimized objective function valueis:

Profit = 13466400

Question 7:If in addition there was a $7,000,000 budget available to buy raw materials, what would you do? Change your formulation and solve again.

The Profit might remain the same as there is no increase in the demand, thus there is no extra profit resulting from more raw material budget. Or in other words, the constraint for the raw material is unbinding, thus it is no use to increase its RHS value.

Here is Q7 in our output file
New Constraints are now:

The amount of constraints before rounding (LP):

Amount of budget constraint violated: 0

Amount of raw material capacity constraint violated: 37

Amount of machine production time limit constraint violated: 72

Amount of approximate demand constraint violated: 20

Amount of warehouse capacity constraint violated: 0

The amount of constraints after rounding (IP):

Amount of budget constraint violated: 0

Amount of raw material capacity constraint violated: 55

Amount of machine production time limit constraint violated: 72

Amount of approximate demand constraint violated: 39

Amount of warehouse capacity constraint violated: 0

x [*] :=

SH001 0 SH141 0 SH281 307.907 SH421 416.571
SH002 448.857 SH142 398.857 SH282 415.143 SH422 0
SH003 0 SH143 0 SH283 0 SH423 419.017
SH004 0 SH144 448.857 SH284 460.889 SH424 0
SH005 33.1429 SH145 0 SH285 0 SH425 458.571
SH006 0 SH146 489.143 SH286 436.548 SH426 0
SH007 0 SH147 68.4137 SH287 410.286 SH427 477.429
SH008 0 SH148 0 SH288 162.086 SH428 0
SH009 27 SH149 88.383 SH289 429.829 SH429 27.3735
SH010 0 SH150 425.429 SH290 129.57 SH430 430
SH011 433.429 SH151 0 SH291 0 SH431 0
SH012 289.045 SH152 0 SH292 136.365 SH432 0
SH013 425.429 SH153 129.82 SH293 0 SH433 0
SH014 0 SH154 437.143 SH294 0 SH434 0
SH015 0 SH155 0 SH295 403.429 SH435 245.975
SH016 0 SH156 0 SH296 0 SH436 0
SH017 0 SH157 0 SH297 0 SH437 0
SH018 247.219 SH158 0 SH298 110.829 SH438 96.8
SH019 200.088 SH159 0 SH299 408.014 SH439 0
SH020 0 SH160 21 SH300 427.429 SH440 0
SH021 0 SH161 0 SH301 0 SH441 0
SH022 0 SH162 0 SH302 0 SH442 0
SH023 0 SH163 0 SH303 0 SH443 0
SH024 399.379 SH164 0 SH304 0 SH444 175.467
SH025 404.286 SH165 460 SH305 0 SH445 462.857
SH026 0 SH166 0 SH306 0 SH446 0
SH027 434 SH167 0 SH307 81.2943 SH447 458.857
SH028 0 SH168 379.926 SH308 266.427 SH448 0
SH029 410.216 SH169 0 SH309 0 SH449 209.982

SH030 0 SH170 0 SH310 0 SH450 0
SH031 305.622 SH171 0 SH311 0 SH451 455.429
SH032 490.857 SH172 82.3521 SH312 476.857 SH452 0
SH033 0 SH173 308.491 SH313 483.429 SH453 470.857
SH034 372.571 SH174 0 SH314 391.143 SH454 0
SH035 0 SH175 0 SH315 0 SH455 0.515306
SH036 0 SH176 450.286 SH316 168.419 SH456 430
SH037 0 SH177 0 SH317 316.143 SH457 252.964
SH038 478.571 SH178 139.775 SH318 0 SH458 218.811
SH039 389.429 SH179 107.645 SH319 0 SH459 265.488
SH040 432.857 SH180 0 SH320 462.286 SH460 0
SH041 338.738 SH181 0 SH321 0 SH461 0
SH042 215.137 SH182 0 SH322 290.699 SH462 181.706
SH043 0 SH183 10.7182 SH323 0 SH463 0
SH044 442.286 SH184 105.163 SH324 0 SH464 403.218
SH045 374.306 SH185 370.617 SH325 475.143 SH465 270.282
SH046 0 SH186 68.0572 SH326 0 SH466 440.857
SH047 410.571 SH187 114.793 SH327 0 SH467 126.223
SH048 292.214 SH188 246.232 SH328 194.248 SH468 141.204
SH049 0 SH189 0 SH329 400 SH469 0
SH050 437.429 SH190 0 SH330 0 SH470 379.233
SH051 0 SH191 80.6177 SH331 0 SH471 394.489
SH052 0 SH192 0 SH332 0 SH472 292.349
SH053 0 SH193 0 SH333 0 SH473 194.343
SH054 0 SH194 303.543 SH334 386.571 SH474 281.762
SH055 0 SH195 243.066 SH335 0 SH475 0
SH056 0 SH196 0 SH336 0 SH476 416.857
SH057 204.81 SH197 437.429 SH337 0 SH477 422.571
SH058 432 SH198 0 SH338 109.157 SH478 354.294
SH059 7.67893 SH199 0 SH339 0 SH479 488.286
SH060 191.181 SH200 0 SH340 439.435 SH480 0
SH061 167.912 SH201 476.286 SH341 429.429 SH481 0
SH062 458.286 SH202 0 SH342 299.33 SH482 112.234
SH063 446 SH203 500.571 SH343 30.1097 SH483 434.571
SH064 18.2534 SH204 0 SH344 0 SH484 0
SH065 369.039 SH205 144.365 SH345 317.786 SH485 0
SH066 0 SH206 0 SH346 0 SH486 0
SH067 0 SH207 0 SH347 176.703 SH487 83.1042
SH068 0 SH208 0 SH348 488.857 SH488 0
SH069 0 SH209 436.423 SH349 40.6795 SH489 441.429
SH070 0 SH210 0 SH350 0 SH490 537.143
SH071 338.729 SH211 0 SH351 0 SH491 183.511
SH072 6.87916 SH212 0 SH352 396.571 SH492 39.3634
SH073 345.834 SH213 436.857 SH353 0 SH493 0

SH074 0 SH214 0 SH354 0 SH494 64.9048
SH075 0 SH215 0 SH355 0 SH495 136.333
SH076 64.9008 SH216 0 SH356 186.22 SH496 0
SH077 176.86 SH217 205.84 SH357 0 SH497 450.286
SH078 407.143 SH218 0 SH358 0 SH498 239.48
SH079 195.042 SH219 0 SH359 20.7196 SH499 104.378
SH080 409.429 SH220 0 SH360 0 SH500 425.429
SH081 163.884 SH221 468.286 SH361 417.429 SH501 420.571
SH082 288.762 SH222 2.78749 SH362 419.429 SH502 0
SH083 0 SH223 409.143 SH363 0 SH503 0
SH084 0 SH224 442.571 SH364 0 SH504 0
SH085 0 SH225 403.874 SH365 259.804 SH505 0
SH086 0 SH226 418.857 SH366 197.702 SH506 0
SH087 430 SH227 256.222 SH367 452.286 SH507 98.867
SH088 423.429 SH228 0 SH368 0 SH508 415.95
SH089 0 SH229 0 SH369 378 SH509 0
SH090 474.571 SH230 69.6662 SH370 0 SH510 0
SH091 0 SH231 0 SH371 0 SH511 398
SH092 0 SH232 418.972 SH372 273.751 SH512 30.4036
SH093 284.527 SH233 448.571 SH373 416.571 SH513 327.342
SH094 307.216 SH234 405.143 SH374 0 SH514 396.857
SH095 422.286 SH235 470.857 SH375 0 SH515 0
SH096 452.571 SH236 496.857 SH376 24.0941 SH516 425.143
SH097 0 SH237 0 SH377 148.671 SH517 0
SH098 0 SH238 396.465 SH378 422.571 SH518 0
SH099 0 SH239 109.914 SH379 254.092 SH519 0
SH100 0 SH240 107.109 SH380 0 SH520 123.142
SH101 0 SH241 418.571 SH381 17.8297 SH521 231.57
SH102 0 SH242 0 SH382 0 SH522 54.1516
SH103 0 SH243 0 SH383 0 SH523 134.922
SH104 0 SH244 241.934 SH384 504.571 SH524 96.5308
SH105 418 SH245 0 SH385 0 SH525 152.446
SH106 201.096 SH246 0 SH386 0 SH526 0
SH107 1.4354 SH247 422 SH387 0 SH527 258.751
SH108 0 SH248 78.5367 SH388 0.215179 SH528 0
SH109 0 SH249 405.429 SH389 0 SH529 0
SH110 417.143 SH250 0 SH390 0 SH530 0
SH111 0 SH251 0 SH391 127.685 SH531 0
SH112 0 SH252 0 SH392 0 SH532 475.429
SH113 454.286 SH253 381.143 SH393 0 SH533 0
SH114 0 SH254 479.429 SH394 0 SH534 406.286
SH115 0 SH255 202.693 SH395 448.376 SH535 440.571
SH116 199.755 SH256 434.286 SH396 0 SH536 31.9008
SH117 20.5311 SH257 0 SH397 0 SH537 280.044

SH118 293.557 SH258 0 SH398 0 SH538 0
SH119 0 SH259 0 SH399 442.857 SH539 0
SH120 0 SH260 1.93737 SH400 0 SH540 170.324
SH121 508.286 SH261 409.143 SH401 0 SH541 3.88484
SH122 0 SH262 411.143 SH402 43.8644 SH542 0
SH123 0 SH263 0 SH403 0 SH543 332.093
SH124 0 SH264 0 SH404 0 SH544 0
SH125 0 SH265 413.143 SH405 0 SH545 0
SH126 75.602 SH266 0 SH406 0 SH546 275.725
SH127 381.295 SH267 0 SH407 437.429 SH547 0
SH128 0 SH268 0 SH408 0 SH548 0
SH129 0 SH269 429.429 SH409 0 SH549 61.0804
SH130 93.6652 SH270 76.1212 SH410 262.168 SH550 0
SH131 54.726 SH271 0 SH411 399.904 SH551 0
SH132 0 SH272 0 SH412 87.6778 SH552 181.33
SH133 134.866 SH273 267.014 SH413 0 SH553 0
SH134 490.286 SH274 119.008 SH414 0 SH554 0
SH135 0 SH275 0 SH415 438.286 SH555 0
SH136 0 SH276 279.884 SH416 0 SH556 423.143
SH137 0 SH277 422.857 SH417 482 SH557 470.286
SH138 0 SH278 0 SH418 343.171
SH139 0 SH279 18.8891 SH419 289.92
SH140 0 SH280 0 SH420 89.7077
;
The optimized objective function valueis:

Profit = 13466400

• **Conclusion**

In conclusion, based on the circumstance the Company has provided, their recommended production plan is suggested as the value of x[s]. The model uses linear approximation for x[s], thus the answer should be just an approximation based on our assumptions with a profit of 13466400 for February.

• **Appendices 6**

-Code for Model file:

```
#####################################################
# Tianhao_Huang_huan2546_1007659856 Jialuo_Chen_chenj926_1008897419_



#####################################################
## Define your sets here
set S; # shoe types
set M; # index of machine
```

```
set R; # raw material index
set W; # warehouse index


#####################################################
## Define your parameters here
param SP{S};                    # sale price
param DM{S};                    # demand of each type of shoes
param RQ{S,R} default 0;        # the amount of row material used
param RMCP{R};                          # the capacity of each type of raw material
param RMC{R};                           # cost per type of material
param OT{S,M} default 0;        # the operation time needed to produce one pair of the type of shoes
param OCP{M};                           # cost needed for operation machine
param WC{W};                            # warehosue capacity

#####################################################
## Define your decision variables here
var x{S} >= 0; #num of shoes produced per type

#####################################################
## Define your objective function here
maximize Profit: sum{s in S} min(x[s],DM[s])*SP[s]                                      #
revenue from sale
                        + sum{s in S} max(0, x[s] - DM[s])*(-10)
# cost if not meet demand
                        - sum{s in S , r in R} x[s]*RQ[s,r]*RMC[r]
# raw material cost
                        - sum{s in S,m in M} x[s]*(25/60 + OCP[m])*(OT[s,m]/60);       # production
cost

#####################################################
## Define your constraints here

s.t. Budget: sum{s in S, r in R} x[s]*RQ[s,r]*RMC[r] <= 10000000;
# the money spend on raw material need to less thant 10000000
s.t. RawMaterialCapacity{r in R}: sum{s in S} x[s]*RQ[s,r] <= RMCP[r];
# raw material capacity
s.t. MachineOperationTime{m in M}: sum{s in S} x[s]*OT[s,m] <= 28*12*3600;
# machine production time limit
s.t. WareHouseCapacity: sum{s in S} x[s] <= sum{w in W} WC[w];
        # the total production should less than warehouse capacity
s.t. Demand{s in S}:  x[s] <= DM[s];
# assume the production num will lower or equal to the approximate demand
```

-Code for Data file:

```
#####################################################
# Tianhao_Huang_huan2546_1007659856 Jialuo_Chen_chenj926_1008897419_



#####################################################
## Define your parameters and sets here
set S := 1..557;
set M := 1..72;
set R := 1..165;
set W := 1..8;


#####################################################
############ Read Data From Access ################
#### NOTE: Include the address of the .mdb file in your code.
####### You can use the .mdb file in S drive ("S:\ECFPC"). Check below for an example.
############ Read data from Product_Master table here:
table SalesPrice IN "ODBC" "W:/262/WARP2011W.mdb" "Product_Master":
        S <- [Product_Num], SP ~ Sales_Price;
read table SalesPrice;


############ Read data from Warehouse_Master
table wareC IN "ODBC" "W:/262/WARP2011W.mdb" "Warehouse_Master":
        W <- [Warehouse_Num], WC ~ Capacity;
read table wareC;

############ Read data from Machine_Master table here:
table MachineOprtCost IN "ODBC" "W:/262/WARP2011W.mdb" "Machine_Master":
        M <- [Machine_Num], OCP ~ OpCost_per_min;
read table MachineOprtCost;

############ Read data from RM_Master table here:
table RM_Cost IN "ODBC" "W:/262/WARP2011W.mdb" "RM_Master":
        R <- [RM_Num], RMC ~ Cost, RMCP ~ S_Quantity;
read table RM_Cost;

############ Read Machine_Assign data from table
## Before reading table, we should use let command to assign zero for machines that are not being used.
Here is the code:
/*for {i in Shoes, j in Machines}{
```

```
	let tMachine[i,j]:=0;
}*/


## Now read the data from Machine_Assign table
table opTime IN "ODBC" "W:/262/WARP2011W.mdb" "Machine_Assign":
	[Product_Num, Machine_Num], OT ~ Avg_Duration;
read table opTime;


############# Read data from BOM table. *Hint: You should use a similar for loop, the one we used above, here.
table RM_needed IN "ODBC" "W:/262/WARP2011W.mdb" "BOM":
	[Product_Num, RM_Num], RQ ~ Quantity;
read table RM_needed;



############# Read data from Product_Demand table here. only read month 2 for each year, and then each number should be divided by 6
table Product_Demand IN "ODBC" "W:/262/WARP2011W.mdb" "SQL=SELECT Product_Num, AVG(Demand) AS AvgDemand FROM Product_Demand WHERE Year BETWEEN 1997 AND 2003 AND Month = 2 GROUP BY Product_Num":
	S <- [Product_Num], DM ~ AvgDemand;
read table Product_Demand;



#########################################################
################ Predict Demand Here ################
# Hint: Use for loop over Shoes and use let command.
let {s in S} DM[s]:= DM[s]*20; # the DM we have is 6 years' month 2 thus we need to divided by 6, since demand will be doubled, so we just divided by 3.
```

-Code for Run file:

```
#########################################################
# Tianhao_Huang_huan2546_1007659856 Jialuo_Chen_chenj926_1008897419_
reset;


#########################################################
#### Read model and data files. Include the address of your .mod and .dat file. Here is an example:
#model '\\SRVB\Homes$\shourabi\Desktop\MIE262_Project\warp.mod';
```

```
model AMPLModelfile.mod;
data AMPLDatafile.dat;


####################################################
############## Solve your model here ##############
# NOTE: DO NOT TOUCH THIS PART.
option solver gurobi;
solve;

####################################################
#### Print the information of the solution here

#########################################################################
# Q3: The amount of constraint violated
#########################################################################
printf "Question 3\n" > warp.out;

# Calculating the amount of const violated before rounding (the LP)
printf "\nThe amount of constraints before rounding (LP):" > warp.out;
var ConB := 0;
if sum {s in S, r in R} ((x[s]*RQ[s,r]*RMC[r])) > 10000000 then
        let ConB := ConB + 1;

var ConRMCP := 0;
for {r in R} {
if sum{s in S} (x[s]*RQ[s,r]) > RMCP[r] then
        let ConRMCP := ConRMCP + 1;
}

var ConOT := 0;
for {m in M} {
if sum{s in S} (x[s]*OT[s,m]) > 28*12*60 then
        let ConOT := ConOT + 1;
}

var ConDM := 0;
for {s in S} {
if (x[s]) > DM[s] then
        let ConDM := ConDM + 1;
}

var ConWC := 0;
if sum{s in S} x[s] > sum{w in W} WC[w] then
```

```
	let ConWC := ConWC + 1;

printf "\n	Amount of budget constraint violated: %d", ConB > warp.out;
printf "\n	Amount of raw material capacity constraint violated: %d", ConRMCP > warp.out;
printf "\n	Amount of machine production time limit constraint violated: %d", ConOT > warp.out;
printf "\n	Amount of approximate demand constraint violated: %d", ConDM > warp.out;
printf "\n	Amount of warehouse capacity constraint violated: %d", ConWC > warp.out;

# Calculating the amount of const violated after rounding (the IP)
printf "\n\nThe amount of constraints after rounding (IP):" > warp.out;
let ConB := 0;
if sum {s in S, r in R} ((round((x[s]))*RQ[s,r]*RMC[r])) > 10000000 then
	let ConB := ConB + 1;

let ConRMCP := 0;
for {r in R} {
if sum{s in S} (round(x[s])*RQ[s,r]) > RMCP[r] then
	let ConRMCP := ConRMCP + 1;
}
let ConOT := 0;
for {m in M} {
if sum{s in S} (round(x[s])*OT[s,m]) > 28*12*60 then
	let ConOT := ConOT + 1;
}

let ConDM := 0;
for {s in S} {
if round(x[s]) > DM[s] then
	let ConDM := ConDM + 1;
}

let ConWC := 0;
if sum{s in S} round(x[s]) > sum{w in W} WC[w] then
	let ConWC := ConWC + 1;

printf "\n	Amount of budget constraint violated: %d", ConB > warp.out;
printf "\n	Amount of raw material capacity constraint violated: %d", ConRMCP > warp.out;
printf "\n	Amount of machine production time limit constraint violated: %d", ConOT > warp.out;
printf "\n	Amount of approximate demand constraint violated: %d", ConDM > warp.out;
printf "\n	Amount of warehouse capacity constraint violated: %d", ConWC > warp.out;

#########################################################################
# Q4: Indentify which constraint is violated
#########################################################################
```

```
printf "\n\nQuestion 4\n" > warp.out;

################################
printf "\nBudget:\n" > warp.out;
# Check whether the budget constrain is binding
if Budget.slack == 0 then
	printf "  The Budget constraint is binding!\n" > warp.out;
else
	printf "  The Budget constraint is not binding!\n" > warp.out;

############################################
printf "\RawMaterialCapacity:\n" > warp.out;
# Check whether the Raw matrial capacity const is binding
var ConRQ_NonBind := 0;
var ConRQ_Bind := 0;
for {r in R} {
	if RawMaterialCapacity[r].slack == 0 then
		printf "  The Raw material %d's capacity constraint is binding!\n", r > warp.out;

	else
		let ConRQ_NonBind := ConRQ_NonBind + 1;
}

# if all of the Raw matrial capacity constraint are not binding
if ConRQ_NonBind == 165 then
	printf "  None of the Raw material capacity constraints are binding!\n" > warp.out;
let ConRQ_Bind := 165 - ConRQ_NonBind;
printf "Total:\n  There are %d Raw material capacity constraints are binding!\n\n", ConRQ_Bind >
warp.out;

############################################
printf "\MachineOperationTime:\n" > warp.out;
# Check whether the Machine Operation Time is violated
var ConOT_NonBind := 0;
for {m in M} {
	if MachineOperationTime[m].slack == 0 then
		printf "  Machine %d Operation Time constraint is binding!\n", m > warp.out;
	else
		let ConOT_NonBind := ConOT_NonBind + 1;
}
# if all of the Machine Operation Time constraint are not binding
if ConOT_NonBind == 72 then
	printf "  None of the Machine Operation Time constraints are binding!\n" > warp.out;
############################################
```

```
printf "\Demand:\n" > warp.out;
# Check whether the Demand for each shoe type is violated
var ConD_NonBind := 0;
var ConD_Bind := 0;
for {s in S} {
	if Demand[s].slack == 0 then
		printf "  Demand of shoe type %s's constraint is binding!\n", s > warp.out;
	else
		let ConD_NonBind := ConD_NonBind + 1;
}
# if all of the Machine Operation Time constraint are not binding
if ConD_NonBind == 557 then
	printf "  None of the demand of shoe type's constraint are binding!\n" > warp.out;
let ConD_Bind := 557 - ConD_NonBind;
printf "Total:\n  %d shoe types' constraints are binding!\n", ConD_Bind > warp.out;

################################################
printf "\WareHouseCapacity:\n" > warp.out;
# Check whether the WareHouse Capacity is violated
var ConWC_NonBind := 0;
for {w in W} {
	if WareHouseCapacity.slack == 0 then
		printf "  Warehouse constraint %d's constraint is binding!\n", w > warp.out;
	else
		let ConWC_NonBind := ConWC_NonBind + 1;
}
# if all of the WareHouse Capacity constraint are not binding
if ConWC_NonBind == 8 then
	printf "  None of the warehouse constraint are binding!\n" > warp.out;

#########################################################################################
##########
# Q5 how many additional space to buy or is it even economical to buy it
#########################################################################################
##########
printf "\n\nQuestion 5\n" > warp.out;
printf "\nThe additional space to buy:\n\n" > warp.out;
display WareHouseCapacity.dual > warp.out;

printf "\n" > warp.out;
display x > warp.out;

#########################################################################################
#########
```

```
# Q6 and Q7
#############################################################################################
#########
# machine avaliable for 8 hrs only

# The change in soln, and new bindings
# We include the run code for Q6 and 7 here
# But we created a temp run file to just run the program under the change of acaliable hrs

#############################################################################################
#########
# Q6
#############################################################################################
#########
# The change in soln, and new bindings
printf "Question 6 and 7\n" > warp.out;
printf "New Constraints are now:\n" > warp.out;

# Calculating the amount of const violated before rounding (the LP)
printf "\nThe amount of constraints before rounding (LP):" > warp.out;
var ConB := 0;
if sum {s in S, r in R} ((x[s]*RQ[s,r]*RMC[r])) > 10000000 then
        let ConB := ConB + 1;

var ConRMCP := 0;
for {r in R} {
if sum{s in S} (x[s]*RQ[s,r]) > RMCP[r] then
        let ConRMCP := ConRMCP + 1;
}


var ConOT := 0;
for {m in M} {
if sum{s in S} (x[s]*OT[s,m]) > 28*8*60 then
        let ConOT := ConOT + 1;
}

var ConDM := 0;
for {s in S} {
if (x[s]) > DM[s] then
        let ConDM := ConDM + 1;
}

var ConWC := 0;
```

```
if sum{s in S} x[s] > sum{w in W} WC[w] then
	let ConWC := ConWC + 1;

printf "\n	Amount of budget constraint violated: %d", ConB > warp.out;
printf "\n	Amount of raw material capacity constraint violated: %d", ConRMCP > warp.out;
printf "\n	Amount of machine production time limit constraint violated: %d", ConOT > warp.out;
printf "\n	Amount of approximate demand constraint violated: %d", ConDM > warp.out;
printf "\n	Amount of warehouse capacity constraint violated: %d", ConWC > warp.out;

# Calculating the amount of const violated after rounding (the IP)
printf "\n\nThe amount of constraints after rounding (IP):" > warp.out;
let ConB := 0;
if sum {s in S, r in R} ((round((x[s]))*RQ[s,r]*RMC[r])) > 10000000 then
	let ConB := ConB + 1;

let ConRMCP := 0;
for {r in R} {
if sum{s in S} (round(x[s])*RQ[s,r]) > RMCP[r] then
	let ConRMCP := ConRMCP + 1;
}

let ConOT := 0;
for {m in M} {
if sum{s in S} (round(x[s])*OT[s,m]) > 28*8*60 then
	let ConOT := ConOT + 1;
}

let ConDM := 0;
for {s in S} {
if round(x[s]) > DM[s] then
	let ConDM := ConDM + 1;
}

let ConWC := 0;
if sum{s in S} round(x[s]) > sum{w in W} WC[w] then
	let ConWC := ConWC + 1;

printf "\n	Amount of budget constraint violated: %d", ConB > warp.out;
printf "\n	Amount of raw material capacity constraint violated: %d", ConRMCP > warp.out;
printf "\n	Amount of machine production time limit constraint violated: %d", ConOT > warp.out;
printf "\n	Amount of approximate demand constraint violated: %d", ConDM > warp.out;
printf "\n	Amount of warehouse capacity constraint violated: %d", ConWC > warp.out;

printf "\n" > warp.out;
```

```
display x  > warp.out;

#########################################################################################
#########
# Q7
#########################################################################################
#########
# The change in soln, and new bindings
printf "Question 6 and 7\n" > warp.out;
printf "New Constraints are now:\n" > warp.out;

# Calculating the amount of const violated before rounding (the LP)
printf "\nThe amount of constraints before rounding (LP):" > warp.out;
var ConB := 0;
if sum {s in S, r in R} ((x[s]*RQ[s,r]*RMC[r])) > 17000000 then
        let ConB := ConB + 1;

var ConRMCP := 0;
for {r in R} {
if sum{s in S} (x[s]*RQ[s,r]) > RMCP[r] then
        let ConRMCP := ConRMCP + 1;
}


var ConOT := 0;
for {m in M} {
if sum{s in S} (x[s]*OT[s,m]) > 28*12*60 then
        let ConOT := ConOT + 1;
}

var ConDM := 0;
for {s in S} {
if (x[s]) > DM[s] then
        let ConDM := ConDM + 1;
}

var ConWC := 0;
if sum{s in S} x[s] > sum{w in W} WC[w] then
        let ConWC := ConWC + 1;

printf "\n       Amount of budget constraint violated: %d", ConB > warp.out;
printf "\n       Amount of raw material capacity constraint violated: %d", ConRMCP > warp.out;
printf "\n       Amount of machine production time limit constraint violated: %d", ConOT > warp.out;
printf "\n       Amount of approximate demand constraint violated: %d", ConDM > warp.out;
```

```
printf "\n          Amount of warehouse capacity constraint violated: %d", ConWC > warp.out;

# Calculating the amount of const violated after rounding (the IP)
printf "\n\nThe amount of constraints after rounding (IP):" > warp.out;
let ConB := 0;
if sum {s in S, r in R} ((round((x[s]))*RQ[s,r]*RMC[r])) > 17000000 then
        let ConB := ConB + 1;

let ConRMCP := 0;
for {r in R} {
if sum{s in S} (round(x[s])*RQ[s,r]) > RMCP[r] then
        let ConRMCP := ConRMCP + 1;
}

let ConOT := 0;
for {m in M} {
if sum{s in S} (round(x[s])*OT[s,m]) > 28*12*60 then
        let ConOT := ConOT + 1;
}

let ConDM := 0;
for {s in S} {
if round(x[s]) > DM[s] then
        let ConDM := ConDM + 1;
}

let ConWC := 0;
if sum{s in S} round(x[s]) > sum{w in W} WC[w] then
        let ConWC := ConWC + 1;

printf "\n          Amount of budget constraint violated: %d", ConB > warp.out;
printf "\n          Amount of raw material capacity constraint violated: %d", ConRMCP > warp.out;
printf "\n          Amount of machine production time limit constraint violated: %d", ConOT > warp.out;
printf "\n          Amount of approximate demand constraint violated: %d", ConDM > warp.out;
printf "\n          Amount of warehouse capacity constraint violated: %d", ConWC > warp.out;

printf "\n" > warp.out;
display x  > warp.out;


# optiaml solution
printf "\n\n\nThe optimized objective function valueis:\n\n" > warp.out;
display Profit > warp.out;
```