Journal of Industrial and Management Optimization

JIMO, 22(4): 2014–2043.
DOI: 10.3934/jimo.2026074
Received: 05 September 2025
Revised: 27 January 2026
Accepted: 25 February 2026
Published: 23 March 2026

https://www.aimspress.com/journal/jimo

Research article

A Regularized Mixed Integer Linear Programming framework with penalties for integrated workforce, subcontracting and production optimization

P. K. Sudhakar[1,*], R. Muthucumaraswamy[1], P. Selvaraju[2] and S. Rukmani Devi[3]

[1] Department of Mathematics, Sri Venkateswara College of Engineering, (Autonomous), (Affiliated to Anna University), Sriperumbudur, Chennai – 602117, Tamil Nadu, India
[2] Department of Artificial Intelligence and Data Science, Jerusalem College of Engineering (Autonomous), Chennai -600100, India
[3] Department of Computer Science, Saveetha College of Arts and Sciences, SIMATS Deemed to be University, Chennai, India

* **Correspondence:** Email: sudhakarpk_susila@yahoo.com.

Appendix 1

```
timestudy3 <- read_excel("~/timestudy3.xlsx")
data.frame(timestudy3)
timestudy3 <- read_excel("~/timestudy3.xlsx")
<-data.frame(timestudy3)

set.seed(222)
ind<-sample(2,nrow(timestudy3),replace = T,prob = c(0.7,0.3))
train<-timestudy3[ind==1,]
test<-timestudy3[ind==2,]
```

```
custom<-trainControl(method = "repeatedcv",number = 5,repeats = 5,verboseIter = T)
lm<-train(PPH~MP+T1,train,method = 'lm',trControl=custom)

## Aggregating results
## Fitting final model on full training set

+ Fold1.Rep1: intercept=TRUE
## - Fold1.Rep1: intercept=TRUE
## + Fold2.Rep1: intercept=TRUE
## - Fold2.Rep1: intercept=TRUE
## + Fold3.Rep1: intercept=TRUE
## - Fold3.Rep1: intercept=TRUE
## + Fold4.Rep1: intercept=TRUE
## - Fold4.Rep1: intercept=TRUE
## + Fold5.Rep1: intercept=TRUE
## - Fold5.Rep1: intercept=TRUE
## + Fold1.Rep2: intercept=TRUE
## - Fold1.Rep2: intercept=TRUE
## + Fold2.Rep2: intercept=TRUE
## - Fold2.Rep2: intercept=TRUE
## + Fold3.Rep2: intercept=TRUE
## - Fold3.Rep2: intercept=TRUE
## + Fold4.Rep2: intercept=TRUE
## - Fold4.Rep2: intercept=TRUE
## + Fold5.Rep2: intercept=TRUE
## - Fold5.Rep2: intercept=TRUE
## + Fold1.Rep3: intercept=TRUE
## - Fold1.Rep3: intercept=TRUE
## + Fold2.Rep3: intercept=TRUE
## - Fold2.Rep3: intercept=TRUE
## + Fold3.Rep3: intercept=TRUE
## - Fold3.Rep3: intercept=TRUE
## + Fold4.Rep3: intercept=TRUE
## - Fold4.Rep3: intercept=TRUE
## + Fold5.Rep3: intercept=TRUE
## - Fold5.Rep3: intercept=TRUE
## + Fold1.Rep4: intercept=TRUE
## - Fold1.Rep4: intercept=TRUE
## + Fold2.Rep4: intercept=TRUE
## - Fold2.Rep4: intercept=TRUE
## + Fold3.Rep4: intercept=TRUE
## - Fold3.Rep4: intercept=TRUE
```

```
## + Fold4.Rep4: intercept=TRUE
## - Fold4.Rep4: intercept=TRUE
## + Fold5.Rep4: intercept=TRUE
## - Fold5.Rep4: intercept=TRUE
## + Fold1.Rep5: intercept=TRUE
## - Fold1.Rep5: intercept=TRUE
## + Fold2.Rep5: intercept=TRUE
## - Fold2.Rep5: intercept=TRUE
## + Fold3.Rep5: intercept=TRUE
## - Fold3.Rep5: intercept=TRUE
## + Fold4.Rep5: intercept=TRUE
## - Fold4.Rep5: intercept=TRUE
## + Fold5.Rep5: intercept=TRUE
## - Fold5.Rep5: intercept=TRUE
## Aggregating results
## Fitting final model on full training set
```

```
summary(lm)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.555 -2.952 -1.823  3.435 12.693
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 54.01250    3.02395  17.862 1.40e-14 ***
## MP          31.15940    4.77185   6.530 1.44e-06 ***
## T1          -0.47421    0.06049  -7.839 8.26e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.743 on 22 degrees of freedom
## Multiple R-squared:  0.7373, Adjusted R-squared:  0.7134
## F-statistic: 30.87 on 2 and 22 DF,  p-value: 4.11e-07
```

Appendix 2 (Ridge)

```
## + Fold1.Rep1: alpha=0, lambda=1
## - Fold1.Rep1: alpha=0, lambda=1
## + Fold2.Rep1: alpha=0, lambda=1
## - Fold2.Rep1: alpha=0, lambda=1
## + Fold3.Rep1: alpha=0, lambda=1
## - Fold3.Rep1: alpha=0, lambda=1
## + Fold4.Rep1: alpha=0, lambda=1
## - Fold4.Rep1: alpha=0, lambda=1
## + Fold5.Rep1: alpha=0, lambda=1
## - Fold5.Rep1: alpha=0, lambda=1
## + Fold1.Rep2: alpha=0, lambda=1
## - Fold1.Rep2: alpha=0, lambda=1
## + Fold2.Rep2: alpha=0, lambda=1
## - Fold2.Rep2: alpha=0, lambda=1
## + Fold3.Rep2: alpha=0, lambda=1
## - Fold3.Rep2: alpha=0, lambda=1
## + Fold4.Rep2: alpha=0, lambda=1
## - Fold4.Rep2: alpha=0, lambda=1
## + Fold5.Rep2: alpha=0, lambda=1
## - Fold5.Rep2: alpha=0, lambda=1
## + Fold1.Rep3: alpha=0, lambda=1
## - Fold1.Rep3: alpha=0, lambda=1
## + Fold2.Rep3: alpha=0, lambda=1
## - Fold2.Rep3: alpha=0, lambda=1
## + Fold3.Rep3: alpha=0, lambda=1
## - Fold3.Rep3: alpha=0, lambda=1
## + Fold4.Rep3: alpha=0, lambda=1
## - Fold4.Rep3: alpha=0, lambda=1
## + Fold5.Rep3: alpha=0, lambda=1
## - Fold5.Rep3: alpha=0, lambda=1
## + Fold1.Rep4: alpha=0, lambda=1
## - Fold1.Rep4: alpha=0, lambda=1
## + Fold2.Rep4: alpha=0, lambda=1
## - Fold2.Rep4: alpha=0, lambda=1
## + Fold3.Rep4: alpha=0, lambda=1
## - Fold3.Rep4: alpha=0, lambda=1
## + Fold4.Rep4: alpha=0, lambda=1
## - Fold4.Rep4: alpha=0, lambda=1
## + Fold5.Rep4: alpha=0, lambda=1
## - Fold5.Rep4: alpha=0, lambda=1
## + Fold1.Rep5: alpha=0, lambda=1
## - Fold1.Rep5: alpha=0, lambda=1
```

```
## + Fold2.Rep5: alpha=0, lambda=1
## - Fold2.Rep5: alpha=0, lambda=1
## + Fold3.Rep5: alpha=0, lambda=1
## - Fold3.Rep5: alpha=0, lambda=1
## + Fold4.Rep5: alpha=0, lambda=1
## - Fold4.Rep5: alpha=0, lambda=1
## + Fold5.Rep5: alpha=0, lambda=1
## - Fold5.Rep5: alpha=0, lambda=1
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0, lambda = 0.75 on full training set
```

glmnet

25 samples
2 predictor
No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 19, 20, 18, 22, 21, 20, ...
Resampling results across tuning parameters:

```
lambda    RMSE      Rsquared   MAE
##   0.000100  6.538540  0.9439220  4.883760
##   0.250075  6.282229  0.9439569  4.797878
##   0.500050  5.720257  0.9458089  4.615742
##   0.750025  5.631299  0.9463767  4.633441
##   1.000000  5.702405  0.9245410  4.753398
##
## Tuning parameter 'alpha' was held constant at a value of 0
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0 and lambda =
0.750025.
```

Appendix 3 (Lasso)

```
## + Fold1.Rep1: alpha=1, lambda=1
## - Fold1.Rep1: alpha=1, lambda=1
## + Fold2.Rep1: alpha=1, lambda=1
## - Fold2.Rep1: alpha=1, lambda=1
```

```
## + Fold3.Rep1: alpha=1, lambda=1
## - Fold3.Rep1: alpha=1, lambda=1
## + Fold4.Rep1: alpha=1, lambda=1
## - Fold4.Rep1: alpha=1, lambda=1
## + Fold5.Rep1: alpha=1, lambda=1
## - Fold5.Rep1: alpha=1, lambda=1
## + Fold1.Rep2: alpha=1, lambda=1
## - Fold1.Rep2: alpha=1, lambda=1
## + Fold2.Rep2: alpha=1, lambda=1
## - Fold2.Rep2: alpha=1, lambda=1
## + Fold3.Rep2: alpha=1, lambda=1
## - Fold3.Rep2: alpha=1, lambda=1
## + Fold4.Rep2: alpha=1, lambda=1
## - Fold4.Rep2: alpha=1, lambda=1
## + Fold5.Rep2: alpha=1, lambda=1
## - Fold5.Rep2: alpha=1, lambda=1
## + Fold1.Rep3: alpha=1, lambda=1
## - Fold1.Rep3: alpha=1, lambda=1
## + Fold2.Rep3: alpha=1, lambda=1
## - Fold2.Rep3: alpha=1, lambda=1
## + Fold3.Rep3: alpha=1, lambda=1
## - Fold3.Rep3: alpha=1, lambda=1
## + Fold4.Rep3: alpha=1, lambda=1
## - Fold4.Rep3: alpha=1, lambda=1
## + Fold5.Rep3: alpha=1, lambda=1
## - Fold5.Rep3: alpha=1, lambda=1
## + Fold1.Rep4: alpha=1, lambda=1
## - Fold1.Rep4: alpha=1, lambda=1
## + Fold2.Rep4: alpha=1, lambda=1
## - Fold2.Rep4: alpha=1, lambda=1
## + Fold3.Rep4: alpha=1, lambda=1
## - Fold3.Rep4: alpha=1, lambda=1
## + Fold4.Rep4: alpha=1, lambda=1
## - Fold4.Rep4: alpha=1, lambda=1
## + Fold5.Rep4: alpha=1, lambda=1
## - Fold5.Rep4: alpha=1, lambda=1
## + Fold1.Rep5: alpha=1, lambda=1
## - Fold1.Rep5: alpha=1, lambda=1
## + Fold2.Rep5: alpha=1, lambda=1
## - Fold2.Rep5: alpha=1, lambda=1
## + Fold3.Rep5: alpha=1, lambda=1
## - Fold3.Rep5: alpha=1, lambda=1
```

```
## + Fold4.Rep5: alpha=1, lambda=1
## - Fold4.Rep5: alpha=1, lambda=1
## + Fold5.Rep5: alpha=1, lambda=1
## - Fold5.Rep5: alpha=1, lambda=1
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 1, lambda = 0.75 on full training set
```

glmnet
25 samples
2 predictor
No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 20, 21, 21, 19, 19, 22, ...
Resampling results across tuning parameters

```
lambda    RMSE      Rsquared   MAE
##   0.000100  7.285337  0.9107285  4.984667
##   0.250075  6.579550  0.9326310  4.748512
##   0.500050  6.114768  0.9529303  4.668763
##   0.750025  5.969826  0.9199346  4.832852
##   1.000000  6.202412  0.7612166  5.301045
##
## Tuning parameter 'alpha' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda =
0.750025
```

Appendix 4 (Elastic – net)

```
## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1
```

```
## - Fold1.Rep5: alpha=9.1, lambda=1
## + Fold2.Rep5: alpha=0.1, lambda=1
## - Fold2.Rep5: alpha=0.1, lambda=1
## + Fold2.Rep5: alpha=1.1, lambda=1
```

```
## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold2.Rep5: alpha=1.1, lambda=1
## + Fold2.Rep5: alpha=2.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold2.Rep5: alpha=2.1, lambda=1
## + Fold2.Rep5: alpha=3.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold2.Rep5: alpha=3.1, lambda=1
## + Fold2.Rep5: alpha=4.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold2.Rep5: alpha=4.1, lambda=1
## + Fold2.Rep5: alpha=5.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :555555555555555555555555555555555555555555555555555555555
555555555555555555555555555555555555555555555555555555555555555555555
55555555555555555€ng in (function (x, y, family = c("gaussian",
"binomial", "poisson", :
## alpha >1; set to 1

## - Fold2.Rep5: alpha=6.1, lambda=1
## + Fold2.Rep5: alpha=7.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold2.Rep5: alpha=7.1, lambda=1
## + Fold2.Rep5: alpha=8.1, lambda=1
```

```
## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold2.Rep5: alpha=8.1, lambda=1
## + Fold2.Rep5: alpha=9.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold2.Rep5: alpha=9.1, lambda=1
## + Fold3.Rep5: alpha=0.1, lambda=1
## - Fold3.Rep5: alpha=0.1, lambda=1
## + Fold3.Rep5: alpha=1.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold3.Rep5: alpha=1.1, lambda=1
## + Fold3.Rep5: alpha=2.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold3.Rep5: alpha=2.1, lambda=1
## + Fold3.Rep5: alpha=3.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold3.Rep5: alpha=3.1, lambda=1
## + Fold3.Rep5: alpha=4.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold3.Rep5: alpha=4.1, lambda=1
## + Fold3.Rep5: alpha=5.1, lambda=1
```

```
## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold3.Rep5: alpha=5.1, lambda=1
## + Fold3.Rep5: alpha=6.1, lambda=1

## Warning in (function (x, y, family = c("gaussian", "binomial",
"poisson", :
## alpha >1; set to 1

## - Fold3.Rep5: alpha=6.1, lambda=1
## + Fold3.Rep5: alpha=7.1, lambda=1

## Warning in (function (x, y, family = c(
## Selecting tuning parameters
## Fitting alpha = 0.1, lambda = 0.5 on full training set

##
glmnet .30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.
30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+
*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.3
0+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*
.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30
+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.
30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+
*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.3
0+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*
.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30
+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.
30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+
*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.3
0+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*
.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30+*.30
+*.30+*.30+*.30+*.30+*##

///’
## 25 samples
##  2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 21, 20, 19, 19, 21, 20, ...
## Resampling results across tuning parameters:
```

```
##
##   alpha  lambda    RMSE      Rsquared   MAE
##   0.1    0.000100  6.938719  0.9025724  4.938104
##   0.1    0.250075  5.915933  0.9308462  4.654569
##   0.1    0.500050  5.653699  0.9506032  4.609234
##   0.1    0.750025  5.694777  0.9541081  4.659761
##   0.1    1.000000  5.842480  0.9362192  4.813601
##   1.1    0.000100  6.939160  0.9026434  4.938139
##   1.1    0.250075  6.335735  0.9266664  4.791385
##   1.1    0.500050  5.957466  0.9504582  4.746826
##   1.1    0.750025  5.940707  0.9383797  4.809648
##   1.1    1.000000  6.393658  0.8024887  5.395651
##   2.1    0.000100  6.939160  0.9026434  4.938139
##   2.1    0.250075  6.335735  0.9266664  4.791385
##   2.1    0.500050  5.957466  0.9504582  4.746826
##   2.1    0.750025  5.940707  0.9383797  4.809648
##   2.1    1.000000  6.393658  0.8024887  5.395651
##   3.1    0.000100  6.939160  0.9026434  4.938139
##   3.1    0.250075  6.335735  0.9266664  4.791385
##   3.1    0.500050  5.957466  0.9504582  4.746826
##   3.1    0.750025  5.940707  0.9383797  4.809648
##   3.1    1.000000  6.393658  0.8024887  5.395651
##   4.1    0.000100  6.939160  0.9026434  4.938139
##   4.1    0.250075  6.335735  0.9266664  4.791385
##   4.1    0.500050  5.957466  0.9504582  4.746826
##   4.1    0.750025  5.940707  0.9383797  4.809648
##   4.1    1.000000  6.393658  0.8024887  5.395651
##   5.1    0.000100  6.939160  0.9026434  4.938139
##   5.1    0.250075  6.335735  0.9266664  4.791385
##   5.1    0.500050  5.957466  0.9504582  4.746826
##   5.1    0.750025  5.940707  0.9383797  4.809648
##   5.1    1.000000  6.393658  0.8024887  5.395651
##   6.1    0.000100  6.939160  0.9026434  4.938139
##   6.1    0.250075  6.335735  0.9266664  4.791385
##   6.1    0.500050  5.957466  0.9504582  4.746826
##   6.1    0.750025  5.940707  0.9383797  4.809648
##   6.1    1.000000  6.393658  0.8024887  5.395651
##   7.1    0.000100  6.939160  0.9026434  4.938139
##   7.1    0.250075  6.335735  0.9266664  4.791385
##   7.1    0.500050  5.957466  0.9504582  4.746826
##   7.1    0.750025  5.940707  0.9383797  4.809648
##   7.1    1.000000  6.393658  0.8024887  5.395651
```

```
##   8.1    0.000100  6.939160  0.9026434  4.938139
##   8.1    0.250075  6.335735  0.9266664  4.791385
##   8.1    0.500050  5.957466  0.9504582  4.746826
##   8.1    0.750025  5.940707  0.9383797  4.809648
##   8.1    1.000000  6.393658  0.8024887  5.395651
##   9.1    0.000100  6.939160  0.9026434  4.938139
##   9.1    0.250075  6.335735  0.9266664  4.791385
##   9.1    0.500050  5.957466  0.9504582  4.746826
##   9.1    0.750025  5.940707  0.9383797  4.809648
##   9.1    1.000000  6.393658  0.8024887  5.395651
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.1 and lambda =
0.50005.
```

Selecting tuning parameters
Fitting alpha = 0.1, lambda = 0.5 on full training set
25 samples
2 predictor
No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 21, 20, 19, 19, 21, 20, ...
Resampling results across tuning parameters:

Appendix 5

```
model_list<-list(LM =lm,Ridge = ridge,Lasso=lasso,ElasticNet = en)
res<-resamples(model_list)
summary(res)
```

Call:
summary.resamples(object = res)
Models: LM, Ridge, Lasso, ElasticNet
Number of resamples: 25

Appendix 6

```
require(ompr)
```

```
require(ompr.roi)

require(ROI.plugin.glpk)
require(magrittr)
T <- 6
wage_worker      <- 13000    # example wage cost per worker per period
cost_hire        <- 300      # example hiring cost per worker
cost_training    <- 500      # example training cost per worker
cost_layoff      <- 200      # example lay-off cost per worker
cost_subcontract <- 200      # example subcontract cost per unit
capacity_per_worker <- 1040 # units produced per worker per period
demand <- c(800, 950, 1100, 900, 1000, 1050) # units demand per period

# Regularisation weights (example)
lambda1 <- 0.1  # penalty weight for workforce changes (L1 style)
lambda2 <- 0.05 # penalty weight for workforce deviation (L2 style)

# === Model formulation ===
model <- MIPModel() %>%
  # Decision variables
  add_variable(W[t],   t = 1:T,              type = "integer", lb = 0) %>%
  add_variable(H[t],   t = 1:T,              type = "integer", lb = 0) %>%
  add_variable(Lay[t], t = 1:T,              type = "integer", lb = 0) %>%
  add_variable(S[t],   t = 1:T,              type = "continuous", lb = 0) %>%
  add_variable(P[t],   t = 1:T,              type = "continuous", lb = 0) %>%
  add_variable(D[t],   t = 2:T,              type = "continuous", lb = 0) %>%  # auxiliary for |ΔW|

  # Objective: minimise total cost including penalties
  set_objective(
    sum_expr(
      wage_worker * W[t] +
      cost_hire * H[t] +
      cost_training * H[t] +
      cost_layoff * Lay[t] +
      cost_subcontract * S[t],
      t = 1:T
    ) +
    lambda1 * sum_expr(D[t],   t = 2:T) +
    lambda2 * sum_expr(W[t] - W[1], t = 1:T),  # simplified deviation term
    sense = "min"
  ) %>%
```

```
  # Constraints
  # Starting workforce (example starting value = 50)
  add_constraint(W[1] == 50) %>%

  # Workforce evolution for t = 2..T
  add_constraint(W[t] == W[t - 1] + H[t] - Lay[t], t = 2:T) %>%

  # Production capacity constraint
  add_constraint(P[t] <= capacity_per_worker * W[t], t = 1:T) %>%

  # Demand fulfilment
  add_constraint(P[t] + S[t] >= demand[t], t = 1:T) %>%

  # Hire/lay-off constraints
  add_constraint(H[t] <= W[t],         t = 1:T) %>%
  add_constraint(Lay[t] <= W[t - 1],   t = 2:T) %>%

  # Linearisation of absolute workforce change
  add_constraint(D[t] >=  W[t] - W[t - 1], t = 2:T) %>%
  add_constraint(D[t] >= -W[t] + W[t - 1], t = 2:T)

# === Solve the model ===
result <- solve_model(model, with_ROI(solver = "glpk", verbose = TRUE))

# === Extract and print results ===
obj_val <- objective_value(result)
w_sol   <- get_solution(result, W[t])
h_sol   <- get_solution(result, H[t])
lay_sol <- get_solution(result, Lay[t])
s_sol   <- get_solution(result, S[t])
p_sol   <- get_solution(result, P[t])

cat("Total cost (objective):", obj_val, "\n")
cat("Workforce per period:\n");   print(w_sol)
cat("Hires per period:\n");       print(h_sol)
cat("Lay-offs per period:\n");    print(lay_sol)
cat("Subcontract units per period:\n"); print(s_sol)
cat("In-house production per period:\n");   print(p_sol)
```