Integrating Machine Learning with Optimization: Resource Matching

Michael Jaczynski (1), Olivier Noiret (1), Fernando Orozco (1), Juan Orozco (1), Tere Gonzalez (2), Pano Santos (1). August, 2019
Speaker Introduction

Dr. Cipriano (Pano) Santos

- Sr. Technical Content Manager at Gurobi
- Worked at Hewlett Packard Laboratories in Palo Alto, CA, for 23 years
- Retired as Distinguished Technologist
- Developed and implemented several decision support tools for Product Life-cycle Management, Customer Relationship Management, Large Data Centers Computing Resources Allocation, Professional Services Workforce Planning, Airline Dispatcher Workload Distribution Optimization, and Operating Room Planning & Scheduling
- Eighteen patents granted
- Seventeen refereed research publications
- Holds a Bachelor’s Degree in Applied Mathematics from the University of Mexico (UNAM), and a Master’s and PhD degrees in Operations Research from the University of Waterloo, Canada
MsC. Teresa (Tere) González

- Research engineer in the Industrial AI lab at the Center of Social innovation - Hitachi Labs Americas
- Projects that combine images, voice, and text to implement AI solutions in transportation and mobility domains.
- Prior, she worked in Hewlett Packard Laboratories in Palo Alto, CA.
- Holds three granted patents and more than 10 in process.
- MSc Degree in Information Technologies from Monterrey Institute of Technology.
- Experience and interests are in deep learning, machine learning, big data, and Augmented Reality (AR) that can help develop the next generation of services to improve people’s lives.
Integrating Machine Learning with Optimization: Resource Matching

Michael Jaczynski (1), Olivier Noiret (1), Fernando Orozco (1), Juan Orozco (1), Tere Gonzalez (2), Pano Santos (1). August, 2019

(1) Gurobi Optimization
(2) Hitachi Labs
Outline

Resource Management at Professional Services Industry

Resource Matching Optimization (RMO) Problem Statement 5 min

RMO Demo 15 min

Machine Learning engine 15 min

MIP engine 15 min

RMO Demo Architecture 5 min

Q&A 5 min
Resource Management
Professional Services Industry
Resource Management Problem

Motivation

• Professional service firms employ thousands of professionals making labor the industry’s **highest expense**

• Examples of industries where professional service firms are commonly used include:
  • Information technology services
  • Business process outsourcing services
  • Offering customized knowledge-based services to clients
    • Accounting
    • Legal
    • Advertising
    • Engineering
    • ... Other specialized services
Resource Management Problem

Limitations of current processes & tools

- Current labor resources assignment processes and tools are manual and present many limitations leading to
  - Poor demand fulfillment and low labor resource utilization,
  - High project delivery costs,
  - Poor customer satisfaction
The integration of ML and MIP is essential:

- Machine Learning addresses the scoring of resources and jobs, but alone cannot address the complex combinatorial nature of the assignment problem.

- MIP addresses the complex combinatorial nature of the assignment problem, but alone would require a human to provide a large number of matching scores, making the approach impractical.
Resource Matching Optimization

Problem Statement
Resource Matching Optimization

Problem Statement

Find an assignment of resources to jobs that maximizes the total matching score of resources and jobs, while satisfying the requirements of jobs and the availability of resources.

Extra constraints:

+ A resource must satisfy a minimum matching score to qualify for a job

+ There is a priority for each job

+ If job demand cannot be satisfied with available resources, then a new resource can be hired. The number of new resources that can be hired is limited.
Job-Resource Matching Score

Problem Statement

Compute a matching score to indicate how well a resource is qualified for the job.

Qualification matrix

<table>
<thead>
<tr>
<th>Job 1, resource 1, 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job 1, resource 2, 1</td>
</tr>
<tr>
<td>Job i, resource j, m_{i,j}</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Job n, resource m, m_{n,m}</td>
</tr>
</tbody>
</table>

Resource-job score

<table>
<thead>
<tr>
<th>Job 1, resource 1, 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job 1, resource 2, 0.7</td>
</tr>
<tr>
<td>Job i, resource j, m'_{i,j}</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Job n, resource m, m'_{n,m}</td>
</tr>
</tbody>
</table>

Optimal Allocation

Scores

<table>
<thead>
<tr>
<th>Job 1, resource 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job 2, resource 2</td>
</tr>
<tr>
<td>Job i, resource j</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Job n, resource m</td>
</tr>
</tbody>
</table>

Learn resource–job scores and then optimize allocations subject to problem constraints.
ML/AI and Optimization Integration

Problem Understanding
- Identify objectives, constrains and key data parameters

Data Needs

Machine Learning /AI
- Forecasting
- Classification,
- Etc.

Learned parameters

Data engineering
- Data gathering and transformation

Collected parameters

Optimization
- Objective function
- Constraints

Actionable decisions
- Dispatching,
- Scheduling,
- Routing,
- Etc.
Resource Matching Optimization

Live demo
This demo is the result of a collaboration with the Recruitology Data Science team (resumes).

https://demos.gurobi.com/rmo
Resource Matching Optimization
Machine Learning Engine
Machine learning component
Learning job and resource profiles from documents

1. Build resource – job profiles from documents.
   • Classify resources and jobs into **relevant classes**.
2. Compute matching score using weighting similarity

- **Role**: Developer or Data Scientist
- **Technology**: Web or SQL/C++
- **Domain**: Marketing, Finance, or ITOther
Corpus Description

Training set
- 167 resumes
- 132 technical documents (augmentation)

Job attributes (classes) of interest:
- Role
- Technology
- Domain

Document distribution used for resource profile classification

<table>
<thead>
<tr>
<th>Role</th>
<th>Technology</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data science</td>
<td>Developers</td>
<td>Web</td>
</tr>
<tr>
<td>64</td>
<td>103</td>
<td>15</td>
</tr>
<tr>
<td>Finance</td>
<td>SQL</td>
<td>Marketing</td>
</tr>
<tr>
<td>31</td>
<td>31</td>
<td>38</td>
</tr>
<tr>
<td>ITOther</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Bar chart showing the distribution of resumes and augmentation across different categories]
Model building - Learning job-resource profile from documents

Document classification pipeline with small datasets

- Public sources
- Data Augmentation
  - Adding relevant documents
- Text Processing
  - Tokenization
  - Stemming
  - Lemmatization
  - Stop words
- Feature Selection
  - Distributed space vectors with TF-IDF
  - Dimensionality reduction
- Classification
  - Random forest
  - SVM
  - Naïve Bayes

Classes of Interest

Resumes/Job description

Vector space models
Model Evaluation and Results

Experiments:
1. Only resumes
2. Resumes + augmentation

Datasets:
1. Testing dataset
2. Demo dataset (64 resumes)

Models:
1. Naïve Bayes
2. Random Forest
3. Support Vector Machine

Accuracy Results of Best Models on Testing Dataset

<table>
<thead>
<tr>
<th>Class</th>
<th>Model</th>
<th>Only resumes</th>
<th>Resumes + augmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td>SVM</td>
<td>90.0%</td>
<td>94.2%</td>
</tr>
<tr>
<td>Technology</td>
<td>RF</td>
<td>50.0%</td>
<td>95.0%</td>
</tr>
<tr>
<td>Domain</td>
<td>SVM/RF</td>
<td>37.0%</td>
<td>84.0%</td>
</tr>
</tbody>
</table>

Accuracy Results of Best Models on Demo Dataset

<table>
<thead>
<tr>
<th>Class</th>
<th>Model</th>
<th>Only resumes</th>
<th>Resumes + augmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td>SVM</td>
<td>84.0%</td>
<td>96.3%</td>
</tr>
<tr>
<td>Technology</td>
<td>RF</td>
<td>64.0%</td>
<td>94.3%</td>
</tr>
<tr>
<td>Domain</td>
<td>RF</td>
<td>38.8%</td>
<td>83.1%</td>
</tr>
</tbody>
</table>
**Detailed Results – Testing dataset**

**Accuracy** of job attribute classification on testing dataset

<table>
<thead>
<tr>
<th>Role</th>
<th>Technology</th>
<th>Domain</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>91.7</td>
<td>91.7</td>
<td>94.2</td>
<td>85</td>
<td>86</td>
<td>80</td>
<td>84</td>
<td>84</td>
<td></td>
</tr>
</tbody>
</table>

**Recall** of Job attribute classification on demo set

<table>
<thead>
<tr>
<th>Role</th>
<th>Technology</th>
<th>Domain</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>92</td>
<td>92</td>
<td>94</td>
<td>94</td>
<td>81</td>
<td>88</td>
<td>84</td>
<td>84</td>
<td></td>
</tr>
</tbody>
</table>


© 2019, Gurobi Optimization, LLC
Detail Results – Demo dataset

**Accuracy** of Job attribute classification on demo dataset

<table>
<thead>
<tr>
<th>Role</th>
<th>Technology</th>
<th>Domain</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>94.1</td>
<td>94.1</td>
<td>96.27</td>
<td>89.5</td>
<td>94.3</td>
<td>87.3</td>
<td>83.1</td>
<td>57.1</td>
<td>56.7</td>
</tr>
</tbody>
</table>

**Recall** of Job attribute classification on demo dataset

<table>
<thead>
<tr>
<th>Role</th>
<th>Technology</th>
<th>Domain</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
<th>Naïve</th>
<th>RF</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>96</td>
<td>96</td>
<td>97</td>
<td>96</td>
<td>97</td>
<td>93</td>
<td>94</td>
<td>88</td>
<td>88</td>
</tr>
</tbody>
</table>


© 2019, Gurobi Optimization, LLC
Resource profile example

Input resume

ResourceA is a committed developer who is experienced and has worked on several projects with different impacts. He is great working under pressure, willing to learn and share all the knowledge and strategies in order to achieve all goals. He will work hard to acquire personal and professional growth.

Software Engineer Consultant
Current Software Development for modeling POC along with the Engineering team

Software Engineer
Current Application owner. Full stack developer leading a software engineering team. Working with backend technologies as Spring Boot, MySQL, MongoDB, Rabbitmq among others. With front end primary on AngularJS with Es6.

Currently assigned as the main point of contact and technical lead on a service project within a set of microservices. My team is working tightly with other team engineers.

Software Engineer Support Specialist level Expert,
Prototype developer for Business optimization Labs/GPPO and Centers Of Excellence using asp.net, sql server technologies, java and SQLDB. Strong focus on integration, visualization and user experience in order to allow users to work with complex Mathematical Algorithms.

Develops software and algorithms for analytics services using ASP.net, Python, SqlServer, Sparql, C#, Java, Spring Framework, HSOLDB, Subsonic, Javascript, JQuery, node, React and working on the integration of the data generated from software like R or Gurobi Optimization.

Resume classification

--------Results--------

---> Classification Results <---

Document profile ----
- ID = 0
- filename = resourceA.txt
- Class = {'roles': 'developer', 'domain': 'other', 'technology': 'web'}
- MatchingScore = 1.0
- Scores = {'roles': 1, 'domain': 1, 'technology': 1}
- Relevant terms
  {'Developer', 'tool', 'javascript', 'asp net', 'project', 'optimization', 'asp', 'software engineer', 'optimization', 'angularjs', 'Development', 'engineering', 'prototype'}
Machine Learning Component

Extensions

• Document similarity between job description and resumes

• Add temporal models to consider job history

• Other classifier approaches
  • Gradient Boosting
  • Other Ensemble models
  • Deep learning with transfer learning
Resource Matching Optimization
MIP Engine
RMO Demo MIP formulation

Input data

```
# import gurobi library
from gurobipy import *
```

Decision variables

```
# Declare and Initialize model
m = Model('Matching')

# Create decision variables for transportation model
assign = m.addVars(arcs, name="allocate")

# Create gap variables for each job
gap = m.addVars(jobs, name="gap")

# Create idle variables for each resource
idle = m.addVars(resources, name="idle")

# create hire variables for each job
hire = m.addVars(jobs, name="hire")
```

Budget constraint

\[
\text{cst} \left( \sum_{j \in \text{jobs}} f_j h_j \right) \leq B
\]
RMO Demo MIP formulation .. 2

Constraints

Demand constraint: All job requirements must be filled or a gap declared:

\[
\sum_{r \in \text{resources}} assign_{r, j} + \text{gap}_j + f_j h_j = FTE_j \quad \forall j \in \text{jobs}
\]

# create satisfy job demand constraints
demand = m.addConstrs((assign.sum('.', j) + gap[j] + hireFlag[j]*hire[j] == FTE[j] for j in jobs), 'demand')

Supply constraint: The capacity of a resource must be satisfied:

\[
\sum_{j \in \text{jobs}} assign_{r, j} + \text{idle}_r = \text{CAP}_r \quad \forall r \in \text{resources}
\]

# create candidate capacity constraints
supply = m.addConstrs((assign.sum(r, '.') + idle[r] == capacity[r] for r in resources), 'supply')

Budget constraint:

\[
cst(\sum_{j \in \text{jobs}} f_j h_j) \leq B
\]

# Budget constraint
budget = m.addConstr((cst*hire.prod(hireFlag) <= B), 'budget')
Objective Function

Objective function: Maximize total score of assignments considering jobs gap and idle resources penalties

\[
\text{Max } \sum_{r \in \text{resources}} \sum_{j \in \text{jobs}} \text{score}_{r,j} \text{assign}_{r,j} - \sum_{j \in \text{jobs}} \text{gapPnlt}_{j} \text{gap}_{j} - \sum_{r \in \text{resources}} \text{idlePnlt}_{r} \text{idle}_{r}
\]

# The objective is to maximize total score considering penalties of jobs gap and idle resources
m.setObjective(assign.prod(scores) - gap.prod(gapPnlt) - idle.prod(idlePnlt), GRB.MAXIMIZE)
Consider job FTE requirements (e.g. 2.4 FTE data scientists) and resource fractional capacity (e.g. data scientist is only available 60% of its time).

There is a fixed cost of recruiting one resource of each type of job.

There is a hiring budget that limits the number of new resources that can be hired.
RMO Extension .. 2

Challenge

The new resources to be hired need to be an integer number (headcount).

The amount of effort allocated to fill job FTE requirements from new resources hired (headcount) can be fractional numbers.

We hire headcount but allocate capacity available (effort).
Addressing challenge:

Let’s define two types of decision variables for hiring.

\[ h_j \]: (effort) amount of effort allocated from new resources hired (headcount) to fill FTE requirements of job j.

\[ nh_j \]: (headcount) number of new resources hired to fill demand of job j.

The condition we will like to satisfy is: effort allocated is positive if and only if there is headcount available. That is: \( h_j \geq \varepsilon > 0 \) if and only if \( nh_j \geq 1 \).

Where \( \varepsilon \) is the minimum meaningful effort that can be allocated to perform a job, we assume \( \varepsilon = 0.01 \) (about 5 min of work).
Addressing challenge:

$h_j$: amount of effort allocated from new resources hired to fill FTE requirements of job $j$.

$nh_j$: number of new resources hired to fill demand of job $j$.

The condition we will like to satisfy is: effort allocated is positive if and only if there is headcount available

$\left( h_j \geq \epsilon > 0 \text{ if and only if } nh_j \geq 1 \right)$

To force this logical condition, we need an auxiliary binary variable $uh_j$ that:

$uh_j = 1$ whenever $h_j \geq \epsilon > 0$, then $nh_j \geq 1$ (Positive effort implies headcount hired).

And

$uh_j = 1$ whenever $nh_j \geq 1$, then $h_j \geq \epsilon > 0$ (Headcount hired implies positive effort).
Addressing challenge:

The condition we will like to satisfy is: \( h_j \geq \epsilon > 0 \) if and only if \( nh_j \geq 1 \).

(L1) \( \epsilon \times uh_j \leq h_j \leq FTE_j \times uh_j \)

(L2) \( uh_j \leq nh_j \leq \lceil FTE_j \rceil \times uh_j \)

Let’s check that constraints L1 and L2 force the if-and-only-if logical condition.

If \( h_j > 0 \) then by (L1: \( h_j \leq FTE_j \times uh_j \)), \( uh_j = 1 \). By (L2: \( uh_j \leq nh_j \)), we have \( nh_j \geq 1 \)

If \( h_j = 0 \) then by (L1: \( \epsilon \times uh_j \leq h_j \)), \( uh_j = 0 \). By (L2: \( nh_j \leq \lceil FTE_j \rceil \times uh_j \)), we have \( nh_j = 0 \)

If \( nh_j \geq 1 \) then (L2: \( nh_j \leq \lceil FTE_j \rceil \times uh_j \)), \( uh_j = 1 \). By (L1: \( \epsilon \times uh_j \leq h_j \)), we have \( h_j \geq \epsilon > 0 \)

If \( nh_j = 0 \) then (L2: \( uh_j \leq nh_j \)), \( uh_j = 0 \). By (L1: \( h_j \leq FTE_j \times uh_j \)), we have \( h_j = 0 \)

Q.E.D.
RMO Extension MIP Formulation

Decision variables

```python
# Create decision variables for transportation model
assign = m.addVars(arcs, name="allocate")

# Create gap variables for each job
gap = m.addVars(jobs, name="gap")

# Create idle variables for each resource
idle = m.addVars(resources, name="idle")

# Create hire variables for each job
hire = m.addVars(jobs, name="hire")

# Create decision variables: number of hired resources to fill FTE job demand
nh_ub = []
for j in jobs:
    nh_ub[j] = math.ceil(B/Hcost[j])

nh = m.addVars(jobs, ub=nh_ub, vtype=GRB.INTEGER, name="nh")

# Create indicator hiring variable
uh = m.addVars(jobs, vtype=GRB.BINARY, name="uh")
```

0 ≤ CAPᵣ ≤ 1

**Resources**

\[ FTE_j \geq 0 \]

**Jobs**

\[ ms (r, j) \]

Budget constraint

\[ \sum_{j \in jobs} cst_j * nh_j \leq B \]
RMO Extension MIP Formulation .. 2

Constraints

1) **Demand constraint**: Satisfy jobs FTE requirements.

   \[ \sum_{r \in \text{resources}} assign_{r,j} + gap_j + h_j = FTE_j \quad \forall j \in \text{jobs} \]

   
   # All job requirements must be filled by internal resources, hired resources, and or a gap is declared
   demand = m.addConstrs((assign.sum('r', j) + gap[j] + h[j] -- FTE[j] for j in jobs), 'demand')

   \[ FTE_j \geq 0 \]

2) **Resource constraint**: Satisfy resources capacity available.

   \[ \sum_{j \in \text{jobs}} assign_{r,j} + idle_r = CAP_r \quad \forall r \in \text{resources} \]

   
   # Do not exceed capacity available, unused capacity is considered idle
   supply = m.addConstrs((assign.sum('r', 'r') + idle[r] -- capacity[r] for r in resources), 'supply')

   \[ 0 \leq CAP_r \leq 1 \]
3) **Budget constraint**: The total cost of hiring new resources should be less or equal than the hiring budget.

\[
\sum_{j \in \text{jobs}} \text{cst}_j \times \text{nh}_j \leq B
\]

```python
# budget constraint
budget = m.addConstr((nh.prod(Hcst) <= B), 'budget')
```

**Auxiliary constraints**: Ensure that when new resources are allocated to fill job FTE requirements, the proper number of new resources are hired.

5) \(c \times uh_j \leq h_j \leq FTE_j \times uh_j\).

6) \(uh_j \leq nh_j \leq \lceil FTE_j \rceil \times uh_j\).

```python
# Create constraints 5)
eps = 0.81
Constrlb = m.addConstrs((eps*uh[j] - h[j] <= 0 for j in jobs), 'Constrlb')
Constriub = m.addConstrs((h[j] - FTE[j]*uh[j] <= 0 for j in jobs), 'Constriub')

# Create constraints 6)
Constrlub = m.addConstrs((uh[j] - nh[j] <= 0 for j in jobs), 'Constrlub')
Constriub = m.addConstrs((nh[j] - math.ceil(FTE[j])*uh[j] <= 0 for j in jobs), 'Constriub')
```
Objective Function

**Objective function:** Maximize total score of assignments considering jobs gap and idle resources penalties

\[
\text{Max} \sum_{r \in \text{resources}} \sum_{j \in \text{jobs}} \text{score}_{r, j}\text{assign}_{r, j} - \sum_{j \in \text{jobs}} \text{gapPnlt}_{j}\text{gap}_{j} - \sum_{r \in \text{resources}} \text{idlePnlt}_{r}\text{idle}_{r}
\]

```python
# The objective is to maximize total score considering penalties of jobs gap and idle resources
m.setObjective(assign.prod(scores) - gap.prod(gapPnlt) - idle.prod(idlePnlt), GRB.MAXIMIZE)
```
Conclusions: Integrating Machine Learning with Optimization

Core Message:

- Predicting is not enough.
- You need to react to your predictions and decide the best course of action.

An upper bound on the number of possible assignment to explore is the combinations of assigning 62 resources among the 2,620 positive matching scores.

$$\binom{2,620}{62} = 1.3208313298e+126$$

This is a daunting number, a MIP approach is required to efficiently navigate this solution space and identify the Global optimal solution, or declare the problem infeasible.

Typically, people use a ranking approach based on the matching scores to assign resources to jobs. This heuristic may lead to very poor suboptimal assignments.
Resource Matching Optimization

Architecture
Build optimization model using the Gurobi Python API. Solve using Gurobi Instant Cloud. Compute KPIs.

Machine Learning (Training)

- Step 1: Deploy classification model in Docker container
- Step 2: Compute job/resource matching scores

Machine Learning (Document Classification)

Step 1

- Build document classification model based on training corpus

RMO Web UI (React)

- HTTP
- WebSockets

RMO Server (Node.js)

Message Queue (Redis)

Scenario Database (MongoDB)

RMO Worker (Python)

Step 2

- Store jobs, resources, matching scores and optimized assignment plan

Optimization (Gurobi)
Thank You

- Questions?
Next Steps

• If you haven’t already done so, please register for an account at www.gurobi.com.
• Try the demo yourself! https://demos.gurobi.com/rmo
• For questions about Gurobi pricing contact sales@gurobi.com.
• Additional resources:
  • HP Enterprise Services Uses Optimization for Resource Planning
    • https://doi.org/10.1287/inte.1110.0621
  • Adaptive Employee Profile Classification for Resource Planning Tool
    • https://www.researchgate.net/publication/261308627_Adaptive_Employee_Profile_Classification_for_Resource_Planning_Tool