GAMS

MODEL – SOLVE – DEPLOY

An Introduction To Modelling with GAMSPy

Workshop

Frederik Fiand
ffiand@gams.com

Lutz Westermann
lwestermann@gams.com

Outline

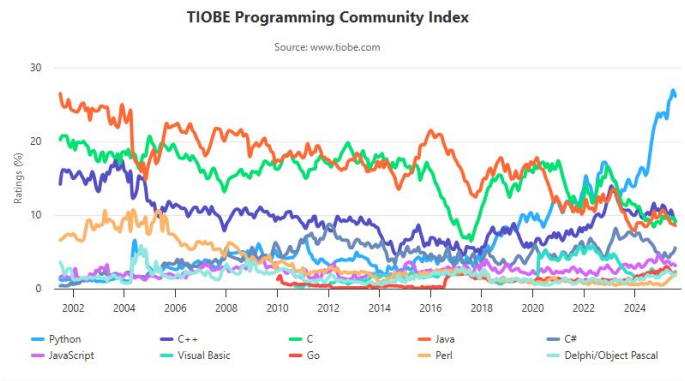
GAMS
MODEL – SOLVE – DEPLOY

- Motivation
- GAMSPy Design
- GAMSPy Features
- From GAMS to GAMSPy

Solving Real-World Problems

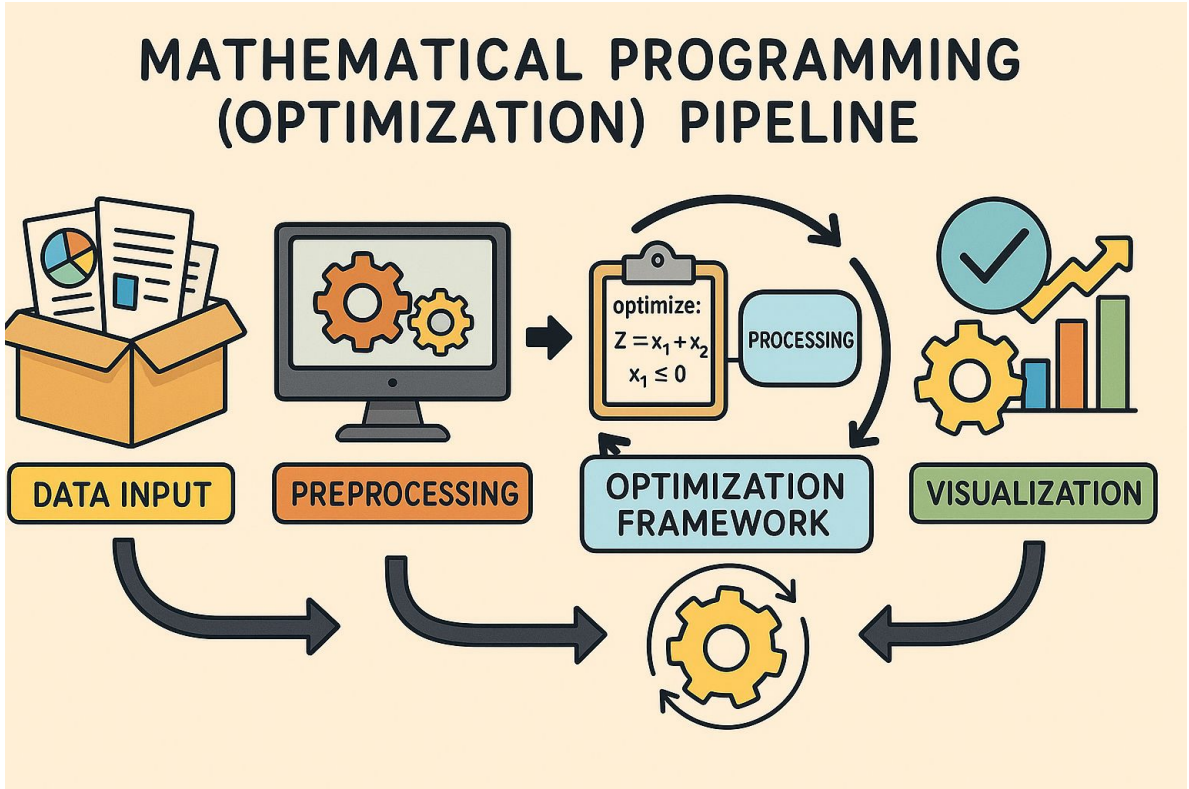
GAMS
MODEL – SOLVE – DEPLOY

- Not *all* about the model
- Data cleaning
- Data manipulation
- Data visualization



Optimization Pipeline

GAMS
MODEL – SOLVE – DEPLOY



python™
Cython
R
jupyter
NumPy
pandas
NetworkX
Network Analysis in Python

GAMS Strengths

GAMS
MODEL – SOLVE – DEPLOY

- Algebraic modeling language (AML)
- Backward Compatibility - *stable!*
- Active development since 1987 - *stable!*
- Familiar syntax - *human readable!*
- Data is held in GAMS structures - *fast!*
- GAMS Structures - *very large models!*
- *Solver Independence – use what you need!*

GAMSPy

GAMS
MODEL – SOLVE – DEPLOY

- Streamlined data pipeline
- Huge variety of third-party packages
- GAMS-like performance

Ease of installation

GAMS
MODEL – SOLVE – DEPLOY

```
pip install gamspy
```

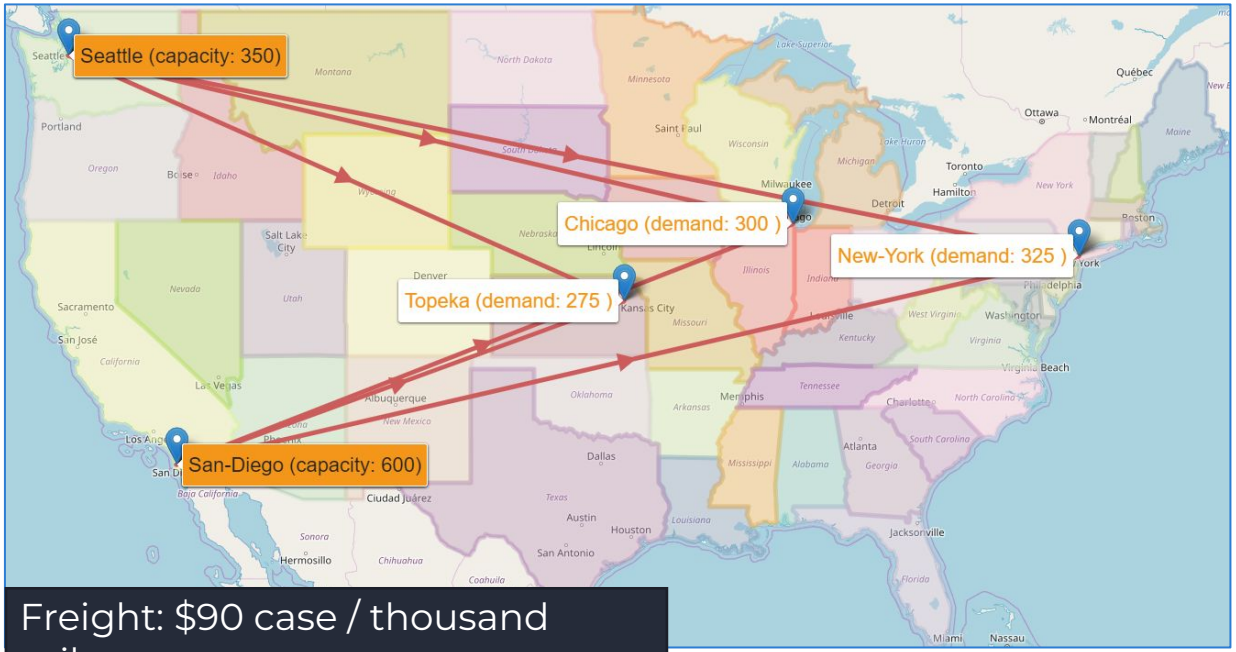
- Stable version released: https://pypi.org/project/gamspy/
- Documentation: https://gamspy.readthedocs.io/en/latest/
- Model library (100+ models): https://github.com/GAMS-dev/gamspy-examples
- Google Colab Notebooks: https://gamspy.readthedocs.io/en/latest/user/examples.html

GAMSPy Design

A Simple Transportation Problem

GAMS
MODEL – SOLVE – DEPLOY

Canning Plants (supply) → shipments (Number of cases) → **Markets** (demand)



Mathematical Model Formulation

GAMS
MODEL – SOLVE – DEPLOY

Indices: i (Canning plants)
j (Markets)

Decision variables: x_{ij} (Number of cases to ship)

Data: c_{ij} (Transport cost per case)
a_i (Capacity in cases)
b_j (Demand in cases)

$\min \quad \sum_i \sum_j c_{ij} \cdot x_{ij}$ (Minimize total transportation cost)

subject to

$\sum_j x_{ij} \leq a_i \quad \forall\, i$ (Shipments from each plant $\leq$ supply capacity)
$\sum_j x_{ij} \geq b_j \quad \forall\, j$ (Shipments to each market $\geq$ demand)

$x_{ij} \geq 0 \quad \forall i,j$ (Do not ship from market to plant)
$i,j \in \mathbb{N}$

Mathematical Model Formulation

GAMS
MODEL – SOLVE – DEPLOY

Indices: i (Canning plants)
j (Markets)

Sets

Decision variables: x_{ij} (Number of cases to ship)

Variables

Data: c_{ij} (Transport cost per case)
a_i (Capacity in cases)
b_j (Demand in cases)

Parameters

$\min \quad \sum_i \sum_j c_{ij} \cdot x_{ij}$ (Minimize total transportation cost)

subject to

$\sum_j x_{ij} \leq a_i \quad \forall\, i$ (Shipments from each plant $\leq$ supply capacity)
$\sum_j x_{ij} \geq b_j \quad \forall\, j$ (Shipments to each market $\geq$ demand)

$x_{ij} \geq 0 \quad \forall i,j$ (Do not ship from market to plant)
$i,j \in \mathbb{N}$

Equations

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```python
from gamspy import Container, Equation, Model, Parameter, Sense, Set,
Sum, Variable

m = Container()

i = Set(m, "i")
j = Set(m, "j")
a = Parameter(m, "a", domain=i)
b = Parameter(m, "b", domain=j)
c = Parameter(m, "c", domain=[i, j])
x = Variable(m, "x", type="positive", domain=[i, j])
supply = Equation(m, "supply", domain=i)
demand = Equation(m, "demand", domain=j)

supply[i] = Sum(j, x[i, j]) <= a[i]
demand[j] = Sum(i, x[i, j]) >= b[j]
```

Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```python
from gamspy import Container, Equation, Model, Parameter, Sense, Set,
Sum, Variable

m = Container()

i = Set(m)
j = Set(m)
a = Parameter(m, domain=i)
b = Parameter(m, domain=j)
c = Parameter(m, domain=[i, j])
x = Variable(m, type="positive", domain=[i, j])
supply = Equation(m, domain=i)
demand = Equation(m, domain=j)

supply[i] = Sum(j, x[i, j]) <= a[i]
demand[j] = Sum(i, x[i, j]) >= b[j]
```

Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```
transport = Model(
    m,
    name="transport",
    equations=m.getEquations(),
    problem="LP",
    sense=Sense.MIN,
    objective=Sum((i, j), c[i, j] * x[i, j]),
)



transport.solve()
```

Model objects need a Container, but live outside

Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```python
z = Variable(m, "z")
objective = Equation(m, "objective")
objective[...] = z == Sum((i, j), c[i, j] * x[i, j])
transport = Model(
    m,
    name="transport",
    equations=m.getEquations(),
    problem="LP",
    sense=Sense.MIN,
    objective=z,
)



transport.solve()
```

Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```
[...]

transport = Model(
    m,
    name="transport",
    equations=m.getEquations(),
    problem="LP",
    sense=Sense.MIN,
    objective=Sum((i, j), c[i, j] * x[i, j]),
)



transport.solve()
```

Is there anything missing?

Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```python
transport = Model(
    m,
    name="transport",
    equations=m.getEquations(),
    problem="LP",
    sense=Sense.MIN,
    objective=Sum((i, j), c[i, j] * x[i, j]),
)

# load data
m.loadRecordsFromGdx("trnsport_input_data.gdx")

transport.solve()
```

Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```python
transport = Model(
    m,
    name="transport",
    equations=m.getEquations(),
    problem="LP",
    sense=Sense.MIN,
    objective=Sum((i, j), c[i, j] * x[i, j]),
)

# load data
m.loadRecordsFromGdx("trnsport_input_data.gdx")

transport.solve()
```

```python
# Different ways of loading/defining data

# data assignment upon declaration
i = Set(m, name="i", records=["seattle", "san-diego"])
j = Set(m, name="j", records=["new-york", "chicago", "topeka"])

# Set declaration and data assignment can also be done separately:
i = Set(m, "i", description="plants")
i.setRecords(["seattle", "san-diego"])

# Loading particular symbol from GDX
m.loadRecordsFromGdx("trnsport_input_data.gdx", symbol_names=["i","j"])

...
```

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

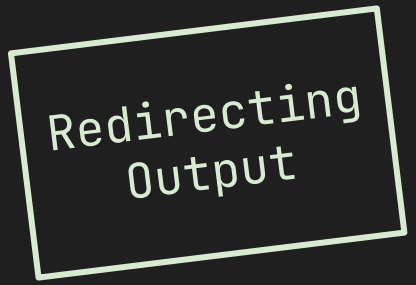
```python
transport = Model(
    m,
    name="transport",
    equations=m.getEquations(),
    problem="LP",
    sense=Sense.MIN,
    objective=Sum((i, j), c[i, j] * x[i, j]),
)

# load data
m.loadRecordsFromGdx("trnsport_input_data.gdx")

transport.solve()
```

```python
# Different ways of loading/defining data

# getting data from data frames
distances_df = pd.DataFrame(
    [
        ["seattle", "new-york", 2.5],
        ["seattle", "chicago", 1.7],
        ["seattle", "topeka", 1.8],
        ["san-diego", "new-york", 2.5],
        ["san-diego", "chicago", 1.8],
        ["san-diego", "topeka", 1.4],
    ],
    columns=["plant", "market", "distance"],
)
i = Set(m, name="i", records=capacities_df["plant"].tolist())
j = Set(m, name="j", records=demands_df["market"].tolist())
```

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```
[...]

# load data
m.loadRecordsFromGdx("trnsport_input_data.gdx")

# redirect output to stdout
transport.solve(output=sys.stdout)

# redirect output to a file
with open("my_out_file.log", "w") as file:
    transport.solve(output=file)
```



Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```
[...]

# load data
m.loadRecordsFromGdx("trnsport_input_data.gdx")

# redirect output to stdout
transport.solve(solver="highs", output=sys.stdout)

# redirect output to a file
with open("my_out_file.log", "w") as file:
    transport.solve(solver="cbc", output=file)
```

Changing solvers

Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```python
[...]

# load data
m.loadRecordsFromGdx("trnsport_input_data.gdx")

# redirect output to stdout
transport.solve(solver="highs", output=sys.stdou

# redirect output to a file
with open("my_out_file.log", "w") as file:
    transport.solve(solver="cbc", output=file)
```

```
--- Job _SDEkpCCdTTmBZwIT9mtl7g.gms Start 08/27/25 15:44:57 50.4.0 c55df396 WEX-WEI x86 64bit/MS
Windows
--- GAMS Parameters defined
    LP highs
[...]
--- Generating LP model transport
---   6 rows  7 columns  19 non-zeroes
--- Range statistics (absolute non-zero finite values)
--- RHS       [min, max] : [ 2.750E+02, 6.000E+02] - Zero values observed as well
--- Bound     [min, max] : [        NA,        NA] - Zero values observed as well
--- Matrix    [min, max] : [ 1.260E-01, 1.000E+00]
--- Executing HIGHS (Solvelink=2): elapsed 0:00:00.039
[...]
Model status        : Optimal
Objective value     :  1.5367500000e+02
Relative P-D gap    :  0.0000000000e+00
HiGHS run time      :          0.00
--- Reading solution for model transport
--- Executing after solve: elapsed 0:00:00.097
--- _SDEkpCCdTTmBZwIT9mtl7g.gms(188) 4 Mb
--- GDX File C:\Users\ffian\AppData\Local\Temp\tmp18kvqkxv\_SDEkpCCdTTmBZwIT9mtl7gout.gdx
*** Status: Normal completion
--- Job _SDEkpCCdTTmBZwIT9mtl7g.gms Stop 08/27/25 15:44:57 elapsed 0:00:00.117
```

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```
i = Set(m, "i")
j = Set(m, "j")
[...]
supply = Equation(m, "supply", domain=i)
demand = Equation(m, "demand", domain=j)

supply[i] = Sum(j, x[i, j]) <= a[i]
demand[i] = Sum(i, x[i, j]) >= b[j]
[...]
```

Early Errors

Abstract Model

Declared/defined over sets

Logically consistent (no domain violations, uncontrolled sets)

Compact

Like “writing on paper”

Completely abstract (no data)

Leverages operator overloading

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```
i = Set(m, "i")
j = Set(m, "j")
[...]
supply = Equation(m, "supply", domain=i)
demand = Equation(m, "demand", domain=j)

supply[i] = Sum(j, x[i, j]) <= a[i]
demand[i] = Sum(i, x[i, j]) >= b[j] #wrong equation domain!
[...]
------------------------------------------------------------
Traceback (most recent call last):
  File "myFile.py", line 18, in <module>
    demand[i] = Sum(i, x[i, j]) >= b[j]
[...]
gamspy.exceptions.ValidationError: `Given set `i` is not a
valid domain for declared domain `j`
```

Early Errors

GAMS Philosophy

Tight syntax leads to better modeling

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```
from rich.traceback import install
install()
[...]
supply = Equation(m, "supply", domain=i)
demand = Equation(m, "demand", domain=j)

supply[i] = Sum(j, x[i, j]) <= a[i]
demand[i] = Sum(i, x[i, j]) >= b[j] #wrong equation domain!
```

Early Errors

```
Traceback (most recent call last)
c:\Users\ffian\Documents\projects\gamspy-examples\models\trnsport\4_transport_domainviol.py:18
in <module>

  15 demand = Equation(m, "demand", domain=j)
  16
  17 supply[i] = Sum(j, x[i, j]) <= a[i]
❱ 18 demand[i] = Sum(i, x[i, j]) >= b[j] #wrong equation domain!
  19

ValidationError: `Given set `i` is not a valid domain for declared domain `j`
```

GAMS Philosophy

Tight syntax leads to better modeling

GAMSPy Design

GAMS
MODEL – SOLVE – DEPLOY

```
i = Set(m)
j = Set(m)
[...]
supply = Equation(m, domain=i)
demand = Equation(m, domain=j)

supply[i] = Sum(j, x[i, j]) <= a[i]
demand[i] = Sum(i, x[i, j]) >= b[j] #wrong equation domain!
```

Early Errors

Nameless Syntax!

```
Traceback (most recent call last)
c:\Users\ffian\Documents\projects\gamspy-examples\models\trnsport\5_transport_domainviol_nameless.py:18 in <module>

  15 demand = Equation(m, domain=j)
  16
  17 supply[i] = Sum(j, x[i, j]) <= a[i]
❱ 18 demand[i] = Sum(i, x[i, j]) >= b[j] #wrong equation domain!
  19

ValidationError: `Given set `swZIYTYDVRzmbeKoVmHYFcggpauto` is not a valid domain for declared domain `sTKcGVCl7RuqOqtbBWEZOywgpauto`
```

GAMS Philosophy

Tight syntax leads to better modeling

GAMSPy Design

```
from gamspy import Container, Equation, Model, Parameter, Sense, Set,
Sum, Variable, set_options
set_options({"USE_PY_VAR_NAME": "yes"})
i = Set(m)
j = Set(m)
[...]
supply = Equation(m, domain=i)
demand = Equation(m, domain=j)

supply[i] = Sum(j, x[i, j]) <= a[i]
demand[i] = Sum(i, x[i, j]) >= b[j] #wrong equation domain!
```

```
Traceback (most recent call last)
c:\Users\ffian\Documents\projects\gamspy-examples\models\trnsport\5_transport_domainviol_nameles
s.py:18 in <module>

  15 demand = Equation(m, domain=j)
  16
  17 supply[i] = Sum(j, x[i, j]) <= a[i]
❱ 18 demand[i] = Sum(i, x[i, j]) >= b[j] #wrong equation domain!
  19

ValidationError: `Given set `i` is not a valid domain for declared domain `j`
```

Early Errors

Nameless Syntax!

GAMS

MODEL – SOLVE – DEPLOY

GAMS Philosophy

Tight syntax leads to better modeling

GAMS

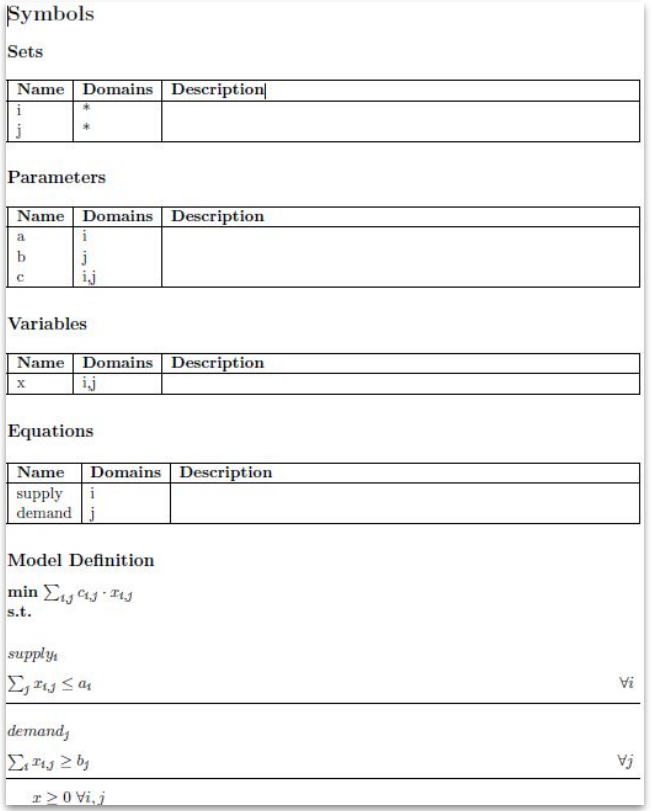
MODEL – SOLVE – DEPLOY

GAMSPy Features

Interoperability (.toLatex)

GAMS
MODEL – SOLVE – DEPLOY

```
transport.toLatex("latex_out_path")

-----------------------------------

[CONVERTER - INFO] LaTeX (.tex) file has been generated
under latex_out_path\transport.tex
```



Symbols

Sets

Name	Domains	Description
i	*	
j	*	

Parameters

Name	Domains	Description
a	i	
b	j	
c	i,j	

Variables

Name	Domains	Description
x	i,j	

Equations

Name	Domains	Description
supply	i	
demand	j	

Model Definition

$\min \sum_{i,j} c_{i,j} \cdot x_{i,j}$
s.t.

$supply_i$

$\sum_j x_{i,j} \leq a_i \quad \forall i$

$demand_j$

$\sum_i x_{i,j} \geq b_j \quad \forall j$

$x \geq 0 \; \forall i, j$

Interoperability (.toGams)

GAMS
MODEL – SOLVE – DEPLOY

```
transport.toGams("gams_out_path")

-----------------------------------

[CONVERTER - INFO] GAMS (.gms) file has been generated
under gams_out_path\transport.gms
```

transport.gms

```
1  Set i(*) / /;
2  Set j(*) / /;
3  Parameter a(i) / /;
4  Parameter b(j) / /;
5  Parameter c(i,j) / /;
6  positive Variable x(i,j) / /;
7  Equation supply(i) / /;
8  Equation demand(j) / /;
9  free Variable transport_objective_variable / /;
10 Equation transport_objective / /;
11 Model transport / supply,demand,transport_objective /;
12 $onMultiR
13 $gdxLoadAll gams_out_path\transport_data.gdx
14 $offMulti
15 supply(i) .. sum(j,x(i,j)) =l= a(i);
16 demand(j) .. sum(i,x(i,j)) =g= b(j);
17 transport_objective .. sum((i,j),c(i,j) * x(i,j)) =e= transport_objective_variable;
18 solve transport using LP MIN transport_objective_variable;
```

transport_data.gdx

Filter ... Export

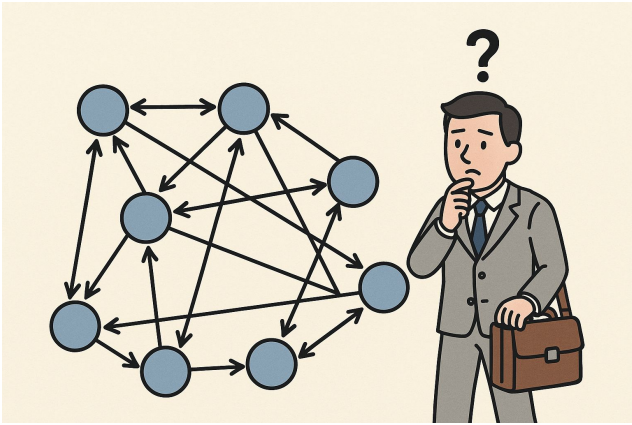
Entry	Name	Type	Dim	Records	Text
1	i	Set	1	0	
2	j	Set	1	0	
3	a	Parameter	1	0	
4	b	Parameter	1	0	
5	c	Parameter	2	0	
6	x	Variable (Positive)	2	0	
7	supply	Equation (=e=)	1	0	
8	demand	Equation (=e=)	1	0	
9	transport_objective_variable	Variable (Free)	0	1	
10	transport_objective	Equation (=e=)	0	1	

Interoperability (advanced)

GAMS
MODEL – SOLVE – DEPLOY

Travelling Salesperson Problem

- Visit all cities
- Visit each city only once
- Minimize cost



GAMS Py → Modelling

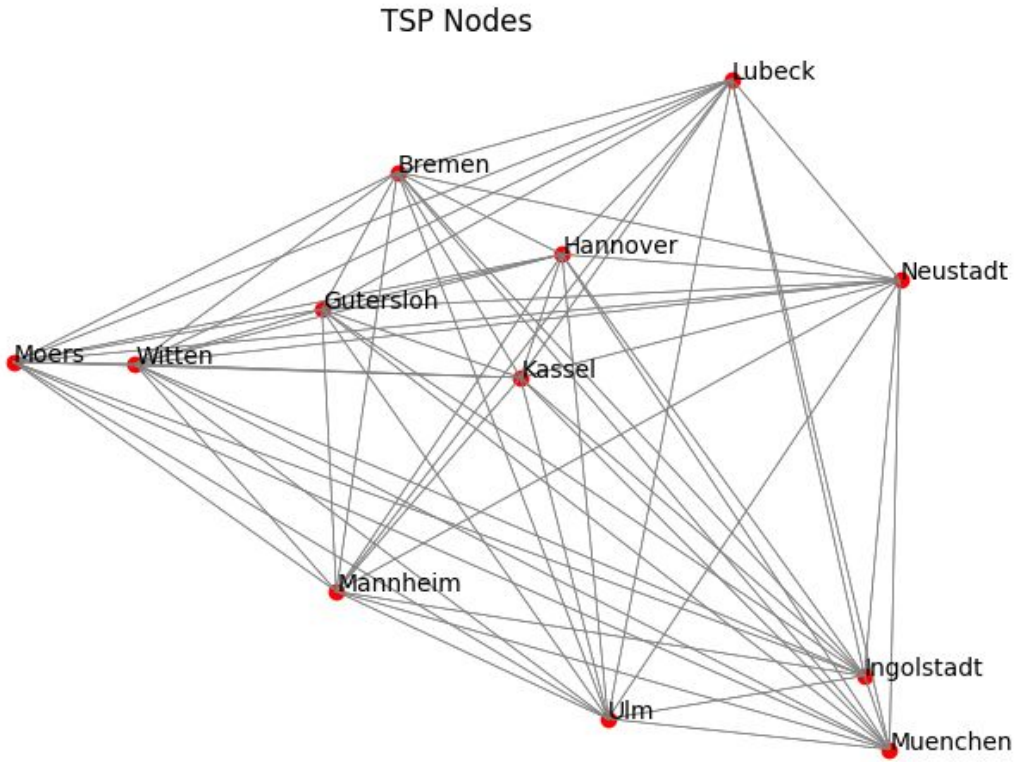
NetworkX
Network Analysis in Python → Subtour Detection & Plotting

Traveling Salesman Problem

GAMS
MODEL – SOLVE – DEPLOY

Data Input:

- Nodes
- Edges
- Edge weights



Traveling Salesman Problem

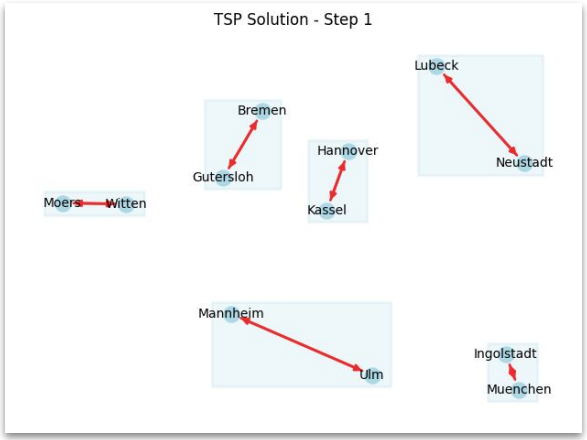
GAMS
MODEL – SOLVE – DEPLOY

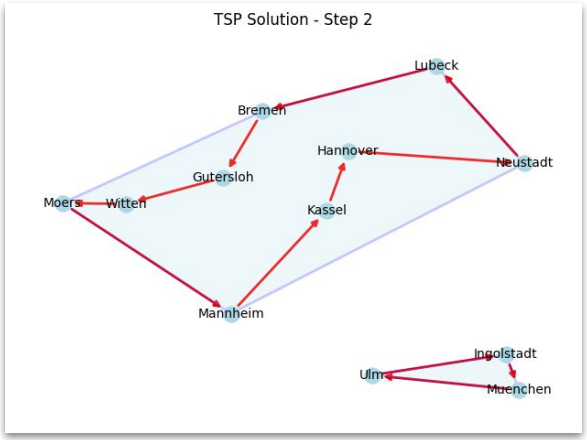
```
[...]
rowsum = Equation(m, name="rowsum", domain=ii, description="leave each city only once")
colsum = Equation(m, name="colsum", domain=jj, description="arrive at each city only once")

# the assignment problem is a relaxation of the TSP
rowsum[i] = Sum(j, x[i, j]) == 1
colsum[j] = Sum(i, x[i, j]) == 1

# Objective Function
objective = Sum([i, j], c[i, j] * x[i, j])

# exclude diagonal
x.fx[ii, ii] = 0
[...]

# Cuts to be added dynamically
cut = Equation(m, name="cut", domain=cut_ids, description="dynamic cuts")
cut[cut_active] = (Sum([i, j], cutcoeff[cut_active, i, j] * x[i, j]) >= 1)
[...]
```

Cuts will be added iteratively for subtours detected via nx.strongly_connected_components

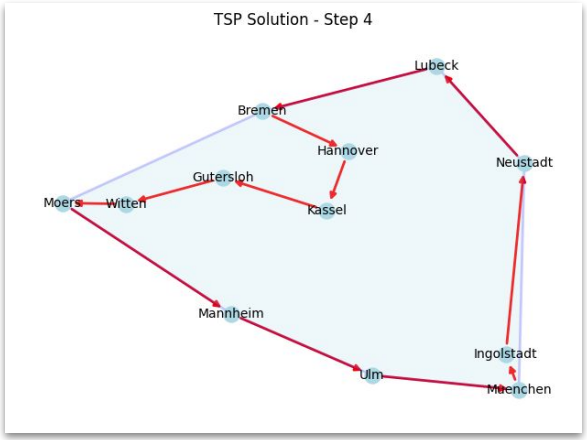
Traveling Salesman Problem

GAMS
MODEL – SOLVE – DEPLOY





...



Corresponding example available at
https://github.com/GAMS-dev/gamspy-examples/blob/master/models/tsp/tsp.py

MIRO (from GAMSPy model to Web App)

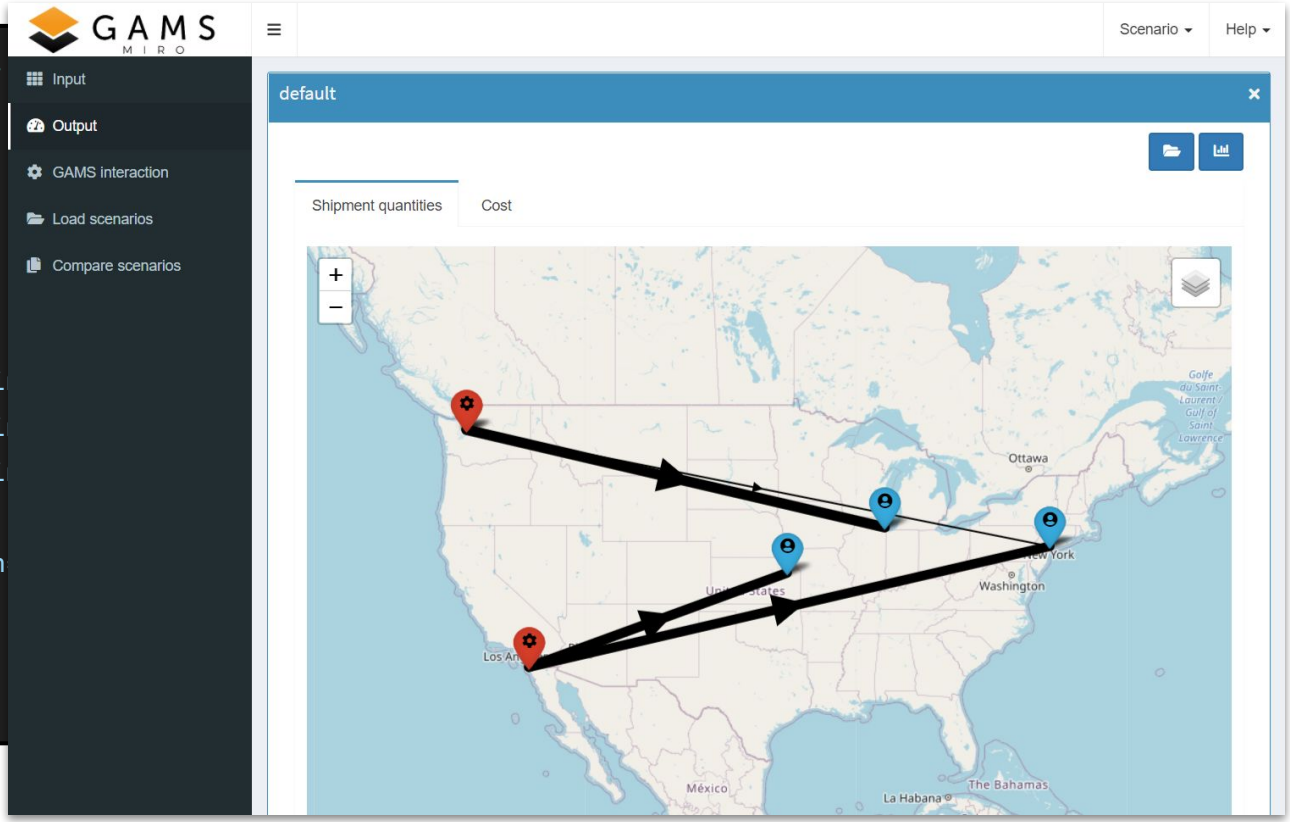
GAMS
MODEL – SOLVE – DEPLOY

```
from gamspy import Container, ...
...
...
data preparation
...
...

a = Parameter(m, name="a", domain=[i], records=capacities)
b = Parameter(m, name="b", domain=[j], records=demands)
d = Parameter(m, name="d", domain=[i, j], records=distances, is_miro_input=True)

x = Variable(m, name="x", domain=[i, j], type="Positive", is_miro_output=True)
...
...
model.solve()
```

MIRO (from GAMSPy model to Web App)

GAMS
MODEL – SOLVE – DEPLOY

```
from gamspy import Container, ..
...
...
data preparation
...
...

a = Parameter(m, name="a", domai
b = Parameter(m, name="b", domai
d = Parameter(m, name="d", domai

x = Variable(m, name="x", domain
...
...
model.solve()
```



Remote Solve (NEOS)

GAMS
MODEL – SOLVE – DEPLOY

```
from gamspy import Container, Equation, Model, NeosClient, Parameter, Sense, Set,
Sum, Variable
import sys
import os

m = Container()
[...]
client = NeosClient(
    email=os.environ["NEOS_EMAIL"],
    username=os.environ["NEOS_USER"],
    password=os.environ["NEOS_PASSWORD"],
)

transport.solve(solver="highs", output=sys.stdout, backend="neos", client=client)
```

Remote Solve (NEOS)

GAMS
MODEL – SOLVE – DEPLOY

```python
from gamspy import Container, Equation, Model, NeosClient, Parameter, Sense, Set,
Sum, Variable
import sys
import os

m = Container()
[...]
client = NeosClient(
    email=os.environ["NEOS_EMAIL"],
    username=os.environ["NEOS_USER"],
    password=os.environ["NEOS_PASSWORD"],
)

transport.solve(solver="highs", output=sys.stdout, ba
```

```
[NEOS - INFO] Job Number: 17342076, Job Password: hYgsnIRH
--- Job MODEL.gms Start 08/25/25 04:48:47 49.6.1 55d34574 LEX-LEG x86 64bit/Linux
[...]
--- Executing HIGHS (Solvelink=2): elapsed 0:00:00.002[LST:22]

HIGHS            49.6.1 55d34574 May 28, 2025          LEG x86 64bit/Linux
[...]
Model status        : Optimal
Objective value     :  1.5367500000e+02
Relative P-D gap    :  0.0000000000e+00
HiGHS run time      :          0.00
[...]
*** Status: Normal completion[LST:73]
--- Job MODEL.gms Stop 08/25/25 04:48:47 elapsed 0:00:00.012
```

Remote Solve (NEOS)

GAMS
MODEL – SOLVE – DEPLOY

```python
from gamspy import Container, Equation, Model, NeosClient, Parameter, Sense, Set,
Sum, Variable
import sys
import os

m = Container()
[...]
client = NeosClient(
    email=os.environ["NEOS_EMAIL"],
    username=os.environ["NEOS_USER"],
    password=os.environ["NEOS_PASSWORD"],
)

transport.solve(solver="highs", output=sys.stdout, ba
```

optional

```
[NEOS - INFO] Job Number: 17342076, Job Password: hYgsnIRH
--- Job MODEL.gms Start 08/25/25 04:48:47 49.6.1 55d34574 LEX-LEG x86 64bit/Linux
[...]
--- Executing HIGHS (Solvelink=2): elapsed 0:00:00.002[LST:22]

HIGHS            49.6.1 55d34574 May 28, 2025          LEG x86 64bit/Linux
[...]
Model status        : Optimal
Objective value     :  1.5367500000e+02
Relative P-D gap    :  0.0000000000e+00
HiGHS run time      :          0.00
[...]
*** Status: Normal completion[LST:73]
--- Job MODEL.gms Stop 08/25/25 04:48:47 elapsed 0:00:00.012
```

Remote Solve (GAMS Engine)

GAMS
MODEL – SOLVE – DEPLOY

```
from gamspy import Container, EngineClient, Equation, Model, Parameter, Sense, Set,
Sum, Variable, Options
import sys
import os

m = Container()
[...]
client = EngineClient(
    host=os.environ["ENGINE_URL"],
    username=os.environ["ENGINE_USER"],
    password=os.environ["ENGINE_PASSWORD"],
    namespace=os.environ["ENGINE_NAMESPACE"],
)

transport.solve(solver="highs", output=sys.stdout, backend="engine", client=client)
```

Remote Solve (GAMS Engine)

GAMS
MODEL – SOLVE – DEPLOY

```python
from gamspy import Container, EngineClient, Equation, Model, Parameter, Sense, Set,
Sum, Variable, Options
import sys
import os

m = Container()
[...]
client = EngineClient(
    host=os.environ["ENGINE_URL"],
    username=os.environ["ENGINE_USER"],
    password=os.environ["ENGINE_PASSWORD"],
    namespace=os.environ["ENGINE_NAMESPACE"],
)

transport.solve(solver="highs", output=sys.stdout, backend="engine", client=client)
```

```
(gamspy) PS C:...\trnsport> & ... python.exe 8_transport_engine.py
[ENGINE - INFO] Job status is queued...
[ENGINE - INFO] Job status is queued...
--- Job _h9ODwTG8QxW_gCq7UTbZXw.gms Start 08/30/25 05:01:55 50.4.1 84b10359
LEX-LEG x86 64bit/Linux
[...]
--- Executing HIGHS (Solvelink=2): elapsed 0:00:00.003
[...]
Model status        : Optimal
Objective value     :  1.5367500000e+02
Relative P-D gap    :  0.0000000000e+00
HiGHS run time      :          0.00
[...]
*** Status: Normal completion
--- Job _X48ZOZpwSQuuKxEW7iBNTA.gms Stop 08/31/25 19:22:45 elapsed 0:00:00.496
[ENGINE - INFO] Results have been extracted to your working directory:
C:\Users\ffian\AppData\Local\Temp\tmpt18c6oa4.
```

Frozen Solve

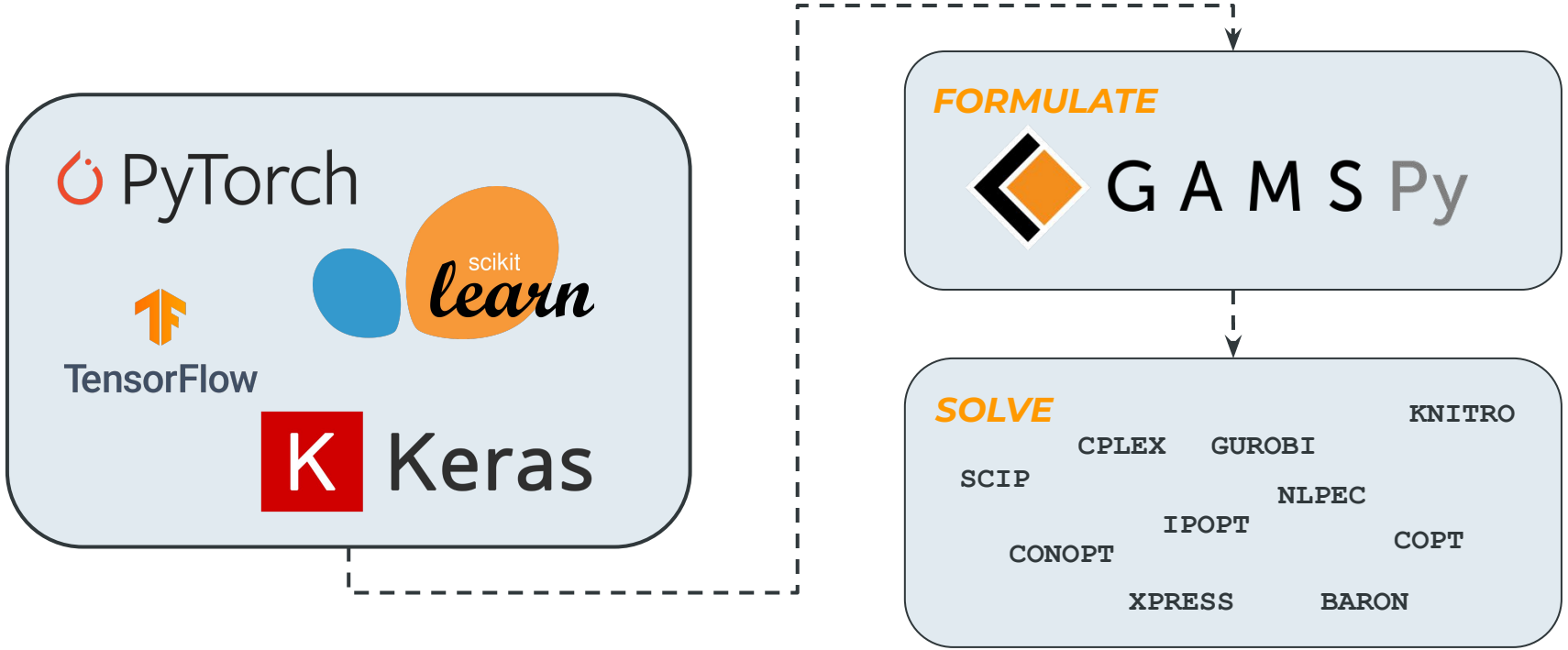
GAMS
MODEL – SOLVE – DEPLOY

```
# define the model
...
# parameter definition
bmult = Parameter(m, name="bmult", records=1)
demand[j] = Sum(i, x[i, j]) >= bmult * b[j]
# parameter values
bmult_list = [0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3]

transport.freeze(modifiables=[bmult])
for b_value in bmult_list:
    bmult.setRecords(b_value)
    transport.solve(solver="conopt")
transport.unfreeze()
```

New Opportunities

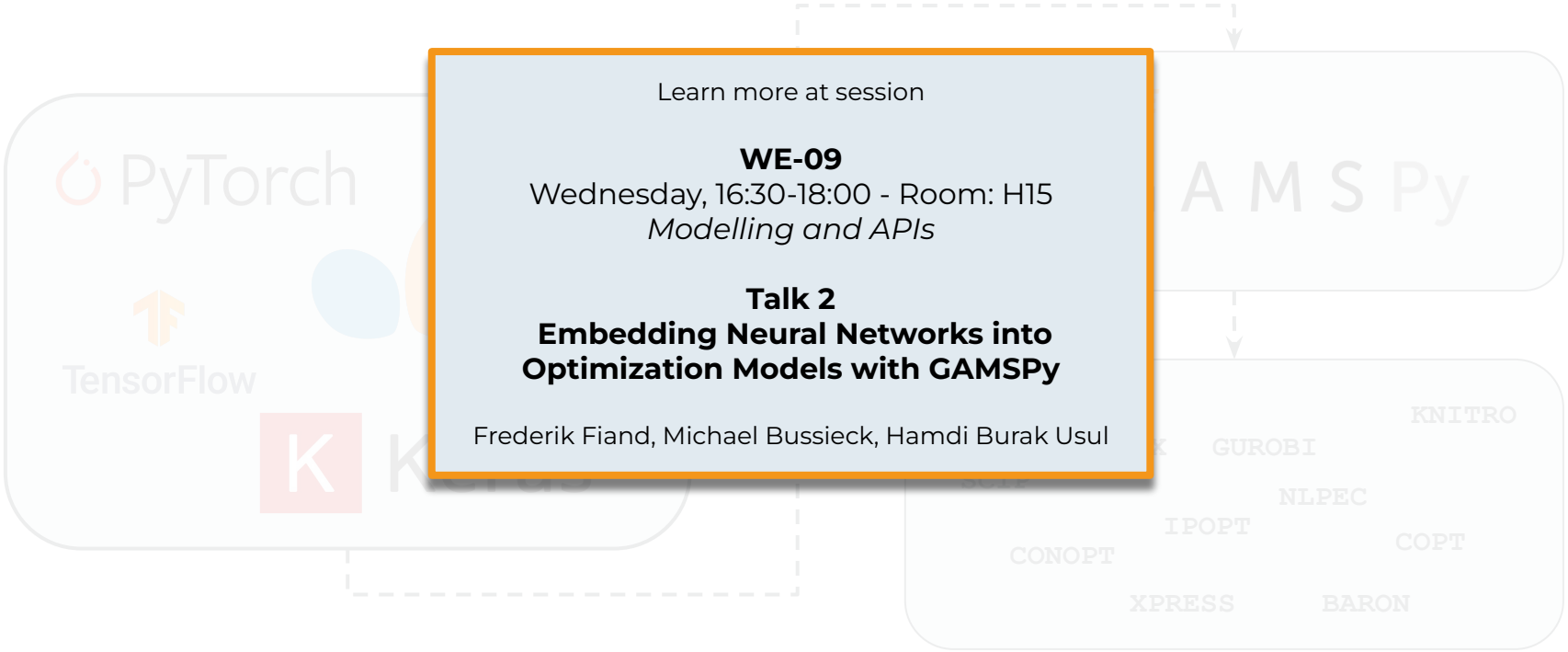
GAMS
MODEL – SOLVE – DEPLOY

Python is THE ML Language



New Opportunities

GAMS
MODEL – SOLVE – DEPLOY



Learn more at session

WE-09
Wednesday, 16:30-18:00 - Room: H15
Modelling and APIs

Talk 2
Embedding Neural Networks into Optimization Models with GAMSPy

Frederik Fiand, Michael Bussieck, Hamdi Burak Usul

Summary

GAMS
MODEL – SOLVE – DEPLOY

- GAMSPy leverages GAMS backend to generate and solve models
- State-of-the-art optimization solvers
- Unique and streamlined way to complete pre/post-processing and visualization – all in a single environment
- Seamless integration with GAMS MIRO, GAMS Engine, and NEOS (local machines vs. cloud/AWS machines)
- Installed with one line:

```
pip install gamspy
```

GAMS
MODEL – SOLVE – DEPLOY

From GAMS to GAMSPy

Basics

Translating Symbols

GAMS
MODEL – SOLVE – DEPLOY

```
Set       i    / i1, i2 /;
Alias     (i, a);
Parameter p(i) / i1 1, i2 2 /;
Variable  v(i), z;
Equation  e(i);
e(i) .. v(i) + p(i) =l= z;
Model my_model / e /;
solve my_model using LP min z;
```

```
import gamspy as gp
gp.set_options({"USE_PY_VAR_NAME": "yes"})

m = gp.Container()

i = gp.Set(m, records=['i1','i2'])
a = gp.Alias(m, alias_with=i)
p = gp.Parameter(m, domain=i, records=[['i1','1'], ['i2','2']])
v = gp.Variable(m, domain=i)
z = gp.Variable(m)
e = gp.Equation(m, domain=i)
e[i] = v[i] + p[i] <= z
my_model = gp.Model(m, equations=[e], problem="lp", sense="min",
objective=z)
my_model.solve()
```

Translating Operations: Sum, Prod , ...

GAMS
MODEL – SOLVE – DEPLOY

Ellipsis Literal
→ Scalar Equation

```
Set       i    / i1,   i2 /;
Parameter a(i) / i1 1, i2 2 /;
Variable  z;
Equation  eq;
eq .. sum(i, a(i)) =l= z;
```

```
import gamspy as gp
from gamspy import Sum, Product, Smin, Smax
gp.set_options({"USE_PY_VAR_NAME": "yes"})

m = gp.Container()
i = gp.Set(m, records=['i1','i2'])
a = gp.Parameter(m, domain=i, records=[['i1','1'], ['i2','2']])
z = gp.Variable(m)

eq = gp.Equation(m)
eq[...] = Sum(i, a[i]) <= z
```

Card/Ord

GAMS
MODEL – SOLVE – DEPLOY

```
Set i / i0*i180 /;
Parameter step;
step = pi / 180;
Parameter omega(i);
omega(i) = (ord(i) - 1) * step;
```

```
import gamspy as gp
import math
gp.set_options({"USE_PY_VAR_NAME": "yes"})

m = gp.Container()
i = gp.Set(m, records=[f'i{i}' for i in range(181)])
step = gp.Parameter(m, records=math.pi / 180)
omega = gp.Parameter(m, domain=i)
omega[i] = (gp.Ord(i) - 1) * step
```

Condition

GAMS
MODEL – SOLVE – DEPLOY

```
Set i / i1*i5 /;
Alias (i, j);
Set adjacent(i, j);
adjacent(i, j) = ord(i)=ord(j)-1;

Parameter p(i,j);
p(i,j)$adjacent(i, j) = 42;
```

```
import gamspy as gp
gp.set_options({"USE_PY_VAR_NAME": "yes"})

m = gp.Container()
i = gp.Set(m, records=[f"i{b}" for b in range(1, 6)])
j = gp.Alias(m, alias_with=i)
adjacent = gp.Set(m,domain=[i, j])
adjacent[i, j] = gp.Ord(i)==gp.Ord(j)-1

p = gp.Parameter(m,"cntAdj", [i, j])
p[i ,j].where[adjacent[i, j]] = 42
```

Condition - Domain

GAMS
MODEL – SOLVE – DEPLOY

```
Set i / i1*i5 /;
Alias (i, j);
Set adjacent(i, j);
adjacent(i, j) = ord(i)=ord(j)-1;

Parameter cntAdj;
cntAdj = sum((i,j)$adjacent(i, j), 1);
```

```
import gamspy as gp
gp.set_options({"USE_PY_VAR_NAME": "yes"})

m = gp.Container()
i = gp.Set(m, records=[f"i{b}" for b in range(1, 6)])
j = gp.Alias(m, alias_with=i)
adjacent = gp.Set(m,domain=[i, j])
adjacent[i, j] = gp.Ord(i)==gp.Ord(j)-1

cntAdj = gp.Parameter(m)
cntAdj[...] = gp.Sum(gp.Domain(i, j).where[adjacent[i, j]], 1 )
```

Condition - Number

GAMS
MODEL – SOLVE – DEPLOY

```
Set k / "1964-i", "1964-ii",
        "1964-iii", "1964-iv"/,
    ki(k) "initial period";
ki(k) = ord(k) = 1;
```

```
import gamspy as gp
gp.set_options({"USE_PY_VAR_NAME": "yes"})

m = gp.Container()
k = gp.Set(m,
records=["1964-i","1964-ii","1964-iii","1964-iv"])
ki = gp.Set(m, domain=k, description="initial period")
ki[k] = gp.Number(1).where[gp.Ord(k) == 1]
```

Math package

GAMS
MODEL – SOLVE – DEPLOY

```
Set i / i1, i2 /;
Set k / k1, k2 /;
Parameter sigma(i,k);
sigma(i,k) = uniform(0.1, 1);
display sigma;
```

```
import gamspy as gp
import gamspy.math as gams_math
import random
gp.set_options({"USE_PY_VAR_NAME": "yes"})

m = gp.Container()
i = gp.Set(m, records=['i1', 'i2'])
k = gp.Set(m, records=['k1', 'k2'])
sigma = gp.Parameter(m, domain=[i, k])
sigma[i, k] = gams_math.uniform(0.1, 1) # Generates a different
value from uniform distribution for each element of the domain.
sigma[i, k] = random.uniform(0.1, 1) # This is not equivalent
to the statement above. This generates only one value for the
whole domain.
```

Logical Operations

GAMS
MODEL – SOLVE – DEPLOY

Since it is not possible in Python to overload keywords such as **and**, **or**, and **not**, one needs to use bitwise operators **&**, **|**, and **~**.

Mapping:

- **and** → &
- **or** → |
- **not** → ~

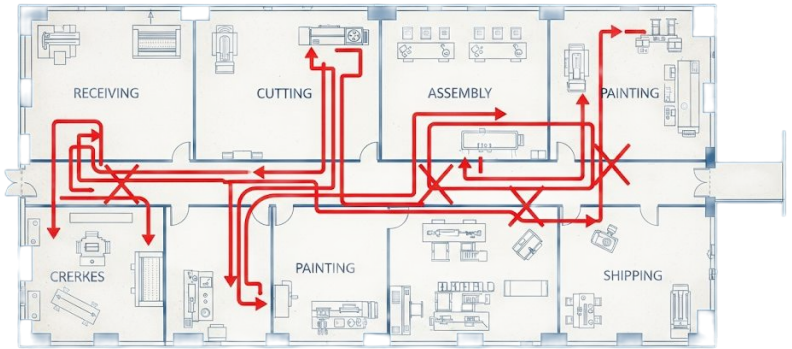
```
error01(s1,s2) = rt(s1,s2) and not lfr(s1,s2) or not rt(s1,s2) and lfr(s1,s2);
```

```
error01[s1, s2] = rt[s1, s2] & (~lfr[s1, s2]) | ((~rt[s1, s2]) & lfr[s1, s2])
```

Example: Layout planning

GAMS
MODEL – SOLVE – DEPLOY

- **Goal:** Figure out the best way to place a number of objects into available locations to make things run as smoothly as possible
- **Input:**
  - List of locations with given space capacity
  - List of objects to be placed, each with space requirement
  - Number of trips between every pair of objects
  - Distance between every pair of locations
  - Placement requirements



Example: Layout planning

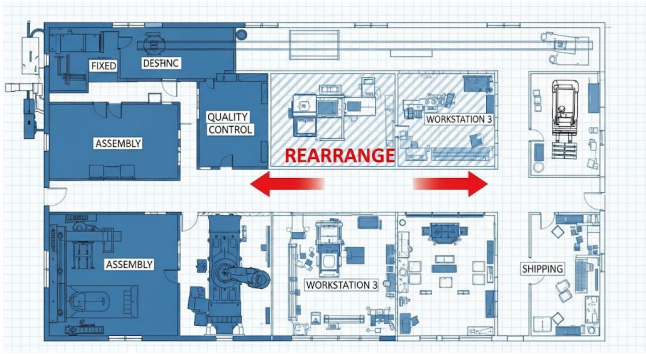
GAMS
MODEL – SOLVE – DEPLOY

- **Objective**
  - Minimize Transportation Cost: Sum up the flow between every pair of objects multiplied by the distance between their final locations
- **Decision Variables**
  - Decide if a specific object will be placed in a specific location for every possible pairing of object and location
- **Constraints**
  - The total space of all objects placed in a single location cannot be more than that location's available space
  - Each object must be assigned to exactly one location
  - Respects placement requirements

Solution Approach: Fix and Optimize

GAMS
MODEL – SOLVE – DEPLOY

- Iterative process that repeatedly solves a series of smaller sub-problems of the original problem
- **Start:** A feasible solution
- **The "Fix":** Fixes most of the decision variables to their current values
- **The "Optimize":** Optimizes a small, unfixed subset of the variables to find a better solution
- **Iteration:** If the optimization step improves the overall solution, that new solution becomes the starting point for the next round
- **End:** No further improvements can be found.



GAMS

MODEL – SOLVE – DEPLOY

From GAMS to GAMSPy

Example from:
https://www.operations-management-online.de/

Author:
Prof. Dr. Stefan Helber
Leibniz Universität
Hannover

Online Course
“Effortless Modeling in Python with GAMSPy”

GAMS
MODEL – SOLVE – DEPLOY

4 + 1 Modules to achieve:

- Tailor software engineering best practises to optimization software
- Implement models like in LaTeX
- Maintainable optimization software

Course Instructor: **Tim Varelmann**

Bluebird Optimization

- GAMS user since 2018
- RWTH, UQ Brisbane, UT Austin & MIT
- Independent optimization consultant

Registration opens 10th of September

GAMS
MODEL – SOLVE – DEPLOY

Sign up to stay in the loop!

- Don't miss out on additional information
- Get exclusive bonuses
- Sweet discounts

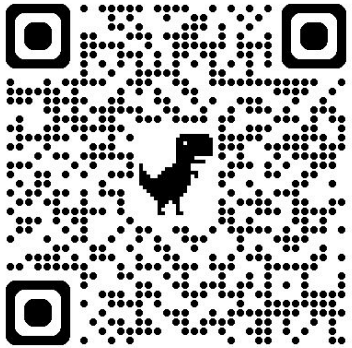
Try before you buy

- 30-minute course intro when you ride home again
- Required prerequisites & instructor introduction
- Goals, content & structure of the course
- Leave your comment!

🎓Academic Program🎓

GAMS
MODEL – SOLVE – DEPLOY

- Free GAMS and GAMSPy licenses
  - No model size limit on GAMSPy
  - Access to all open-source solvers + commercial solvers such as CPLEX, CONOPT, COPT, MOSEK, XPRESS, and GUROBI-link
  - Sign up at 🔗https://academic.gams.com/🔗



- Other resources
  - 💬Community FORUM💬 at https://forum.gams.com/
  - 🎥YouTube channel🎥 at https://www.youtube.com/@GAMSLessons
  - Ready-to-run notebooks, case studies, blog posts, and more!

MODEL – SOLVE – DEPLOY