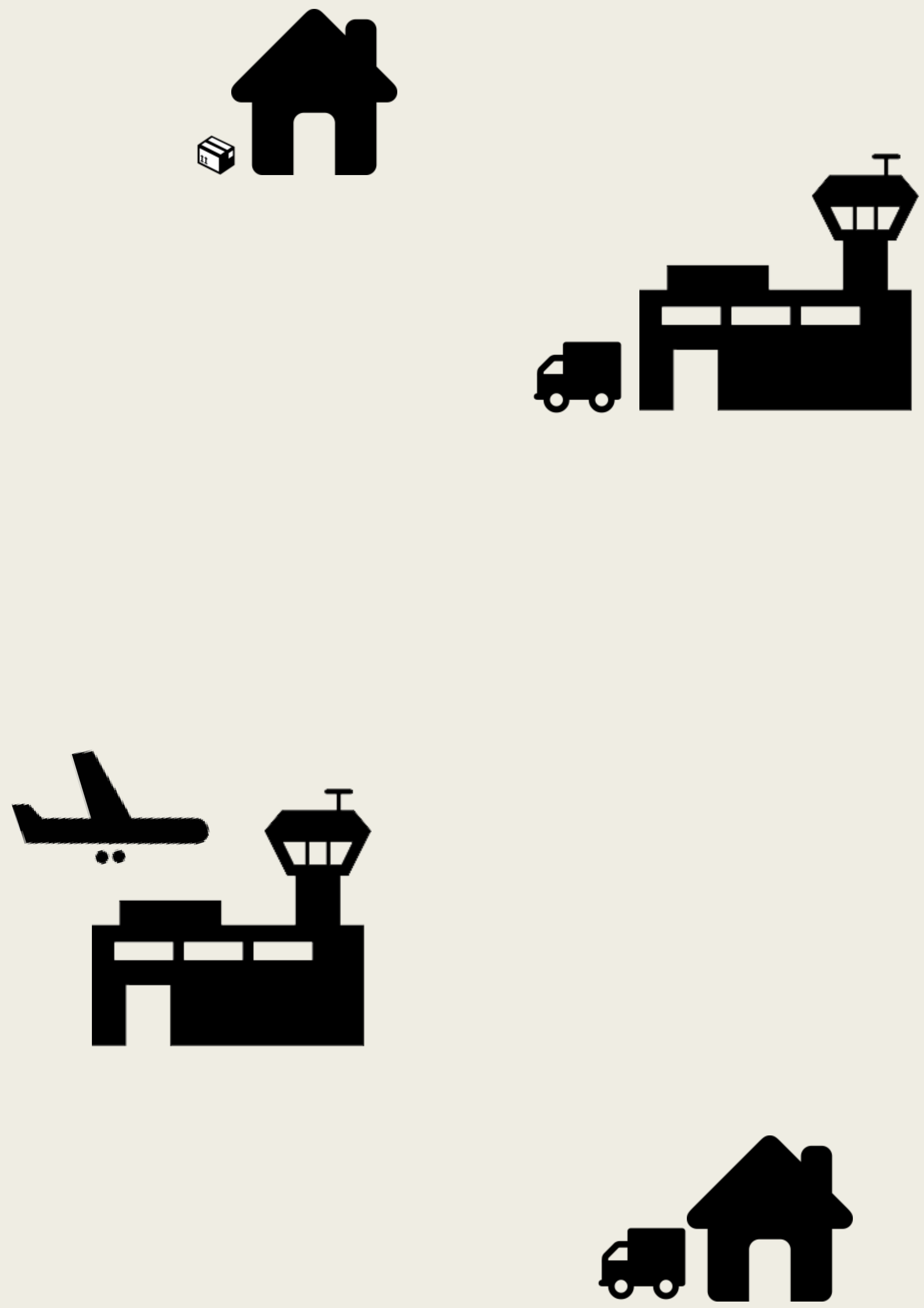
CONCEPT LANGUAGES AS EXPERT INPUT FOR GENERALIZED PLANNING

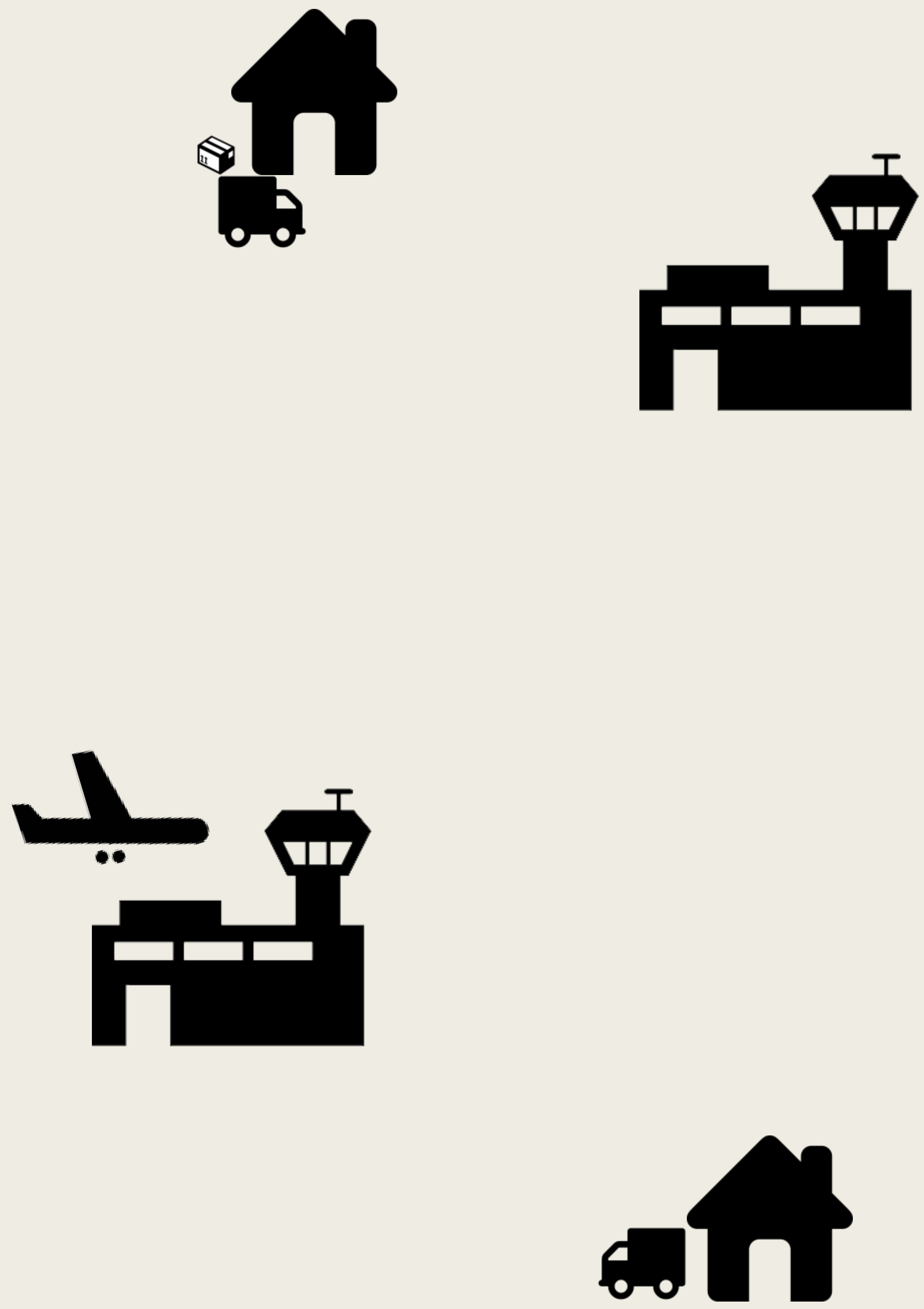
BSc Thesis Presentation

Rik de Graaff

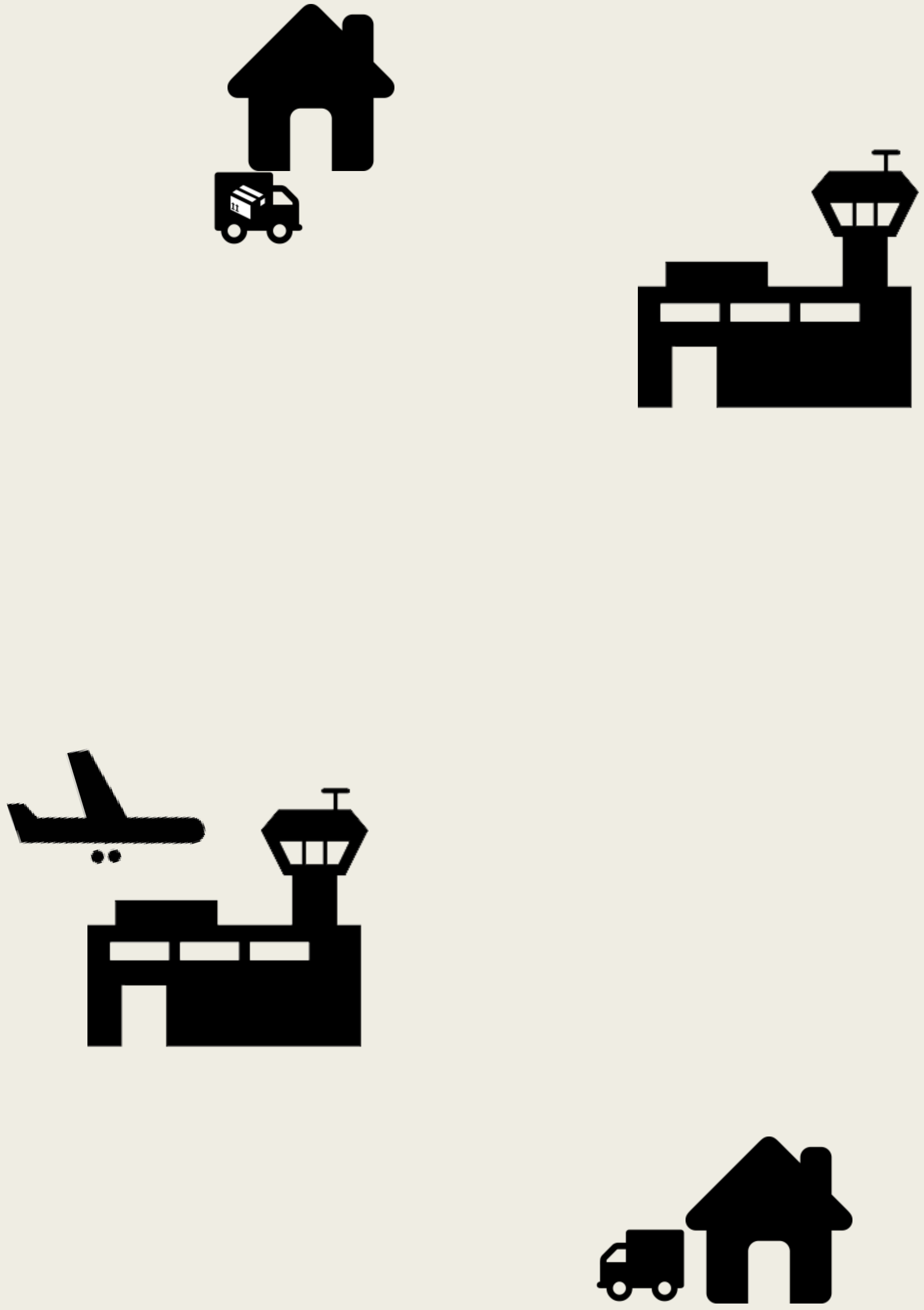
The Logistics Problem



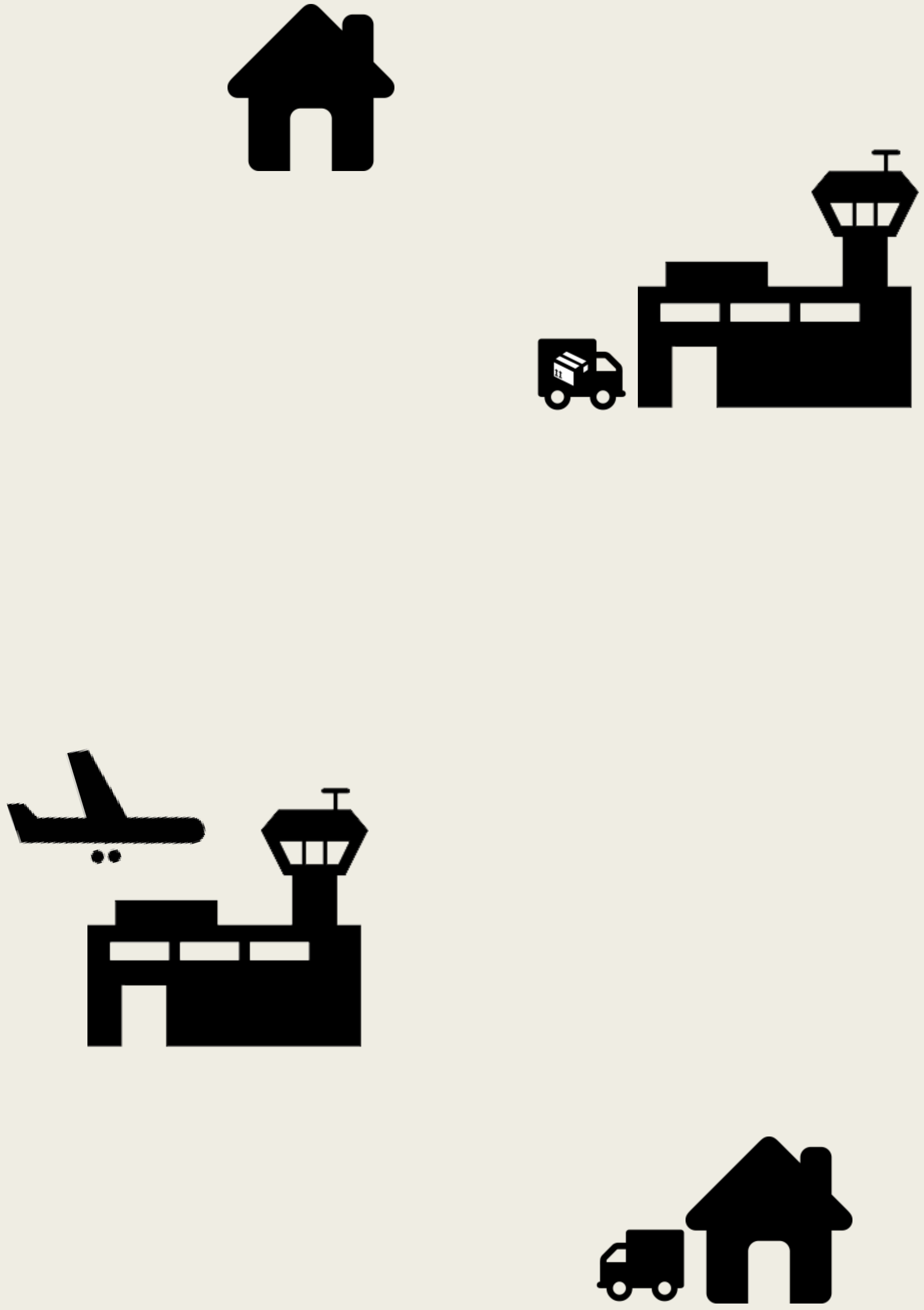
The Logistics Problem



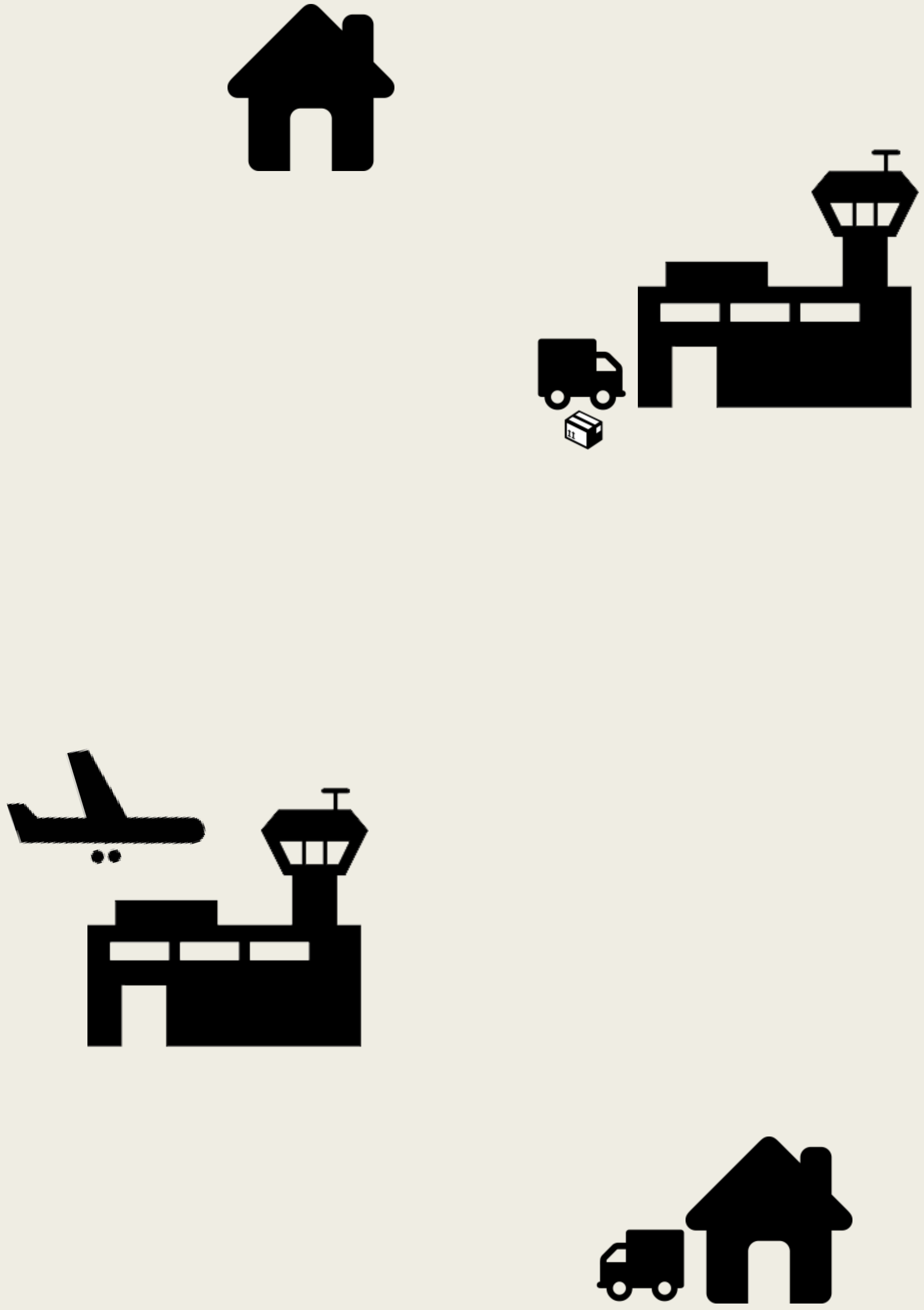
The Logistics Problem



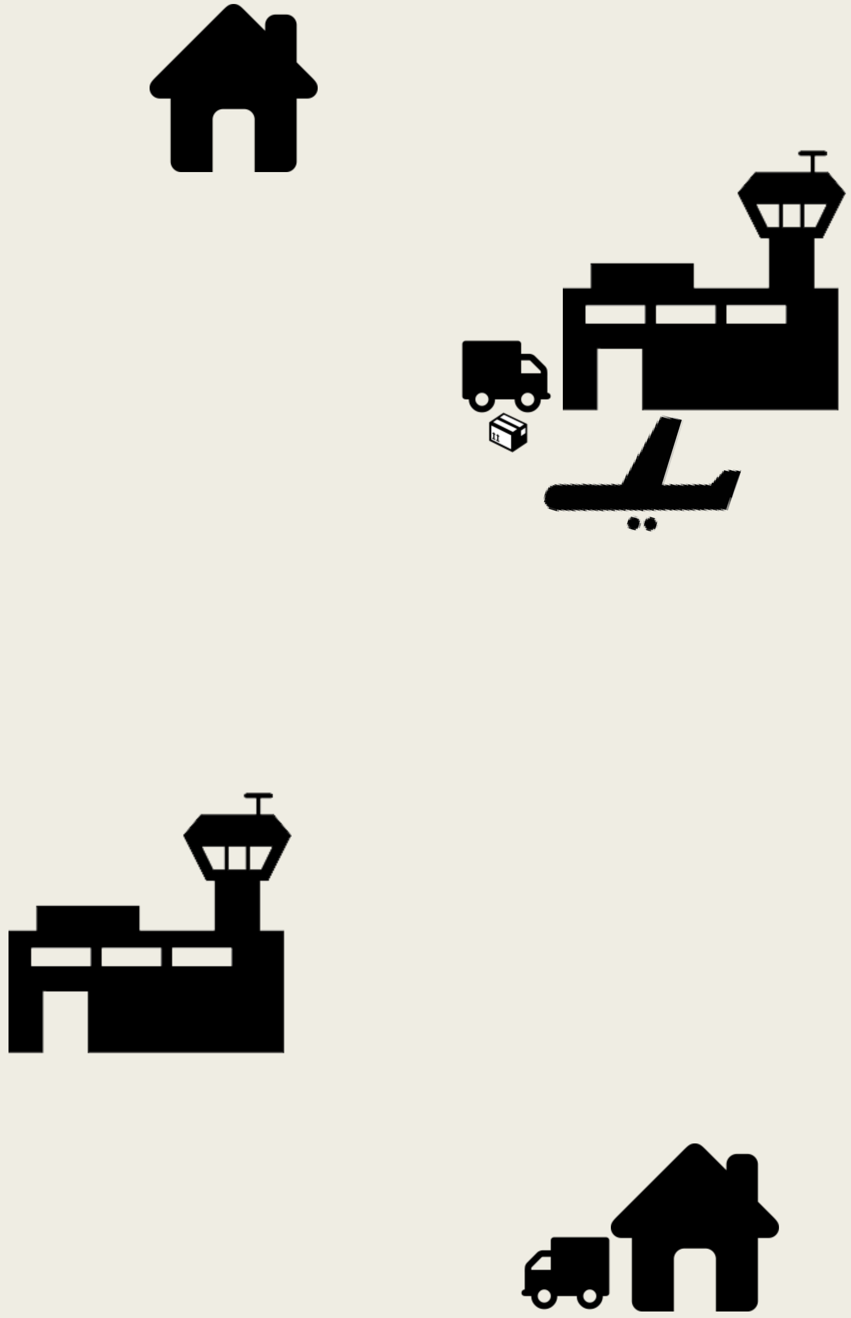
The Logistics Problem



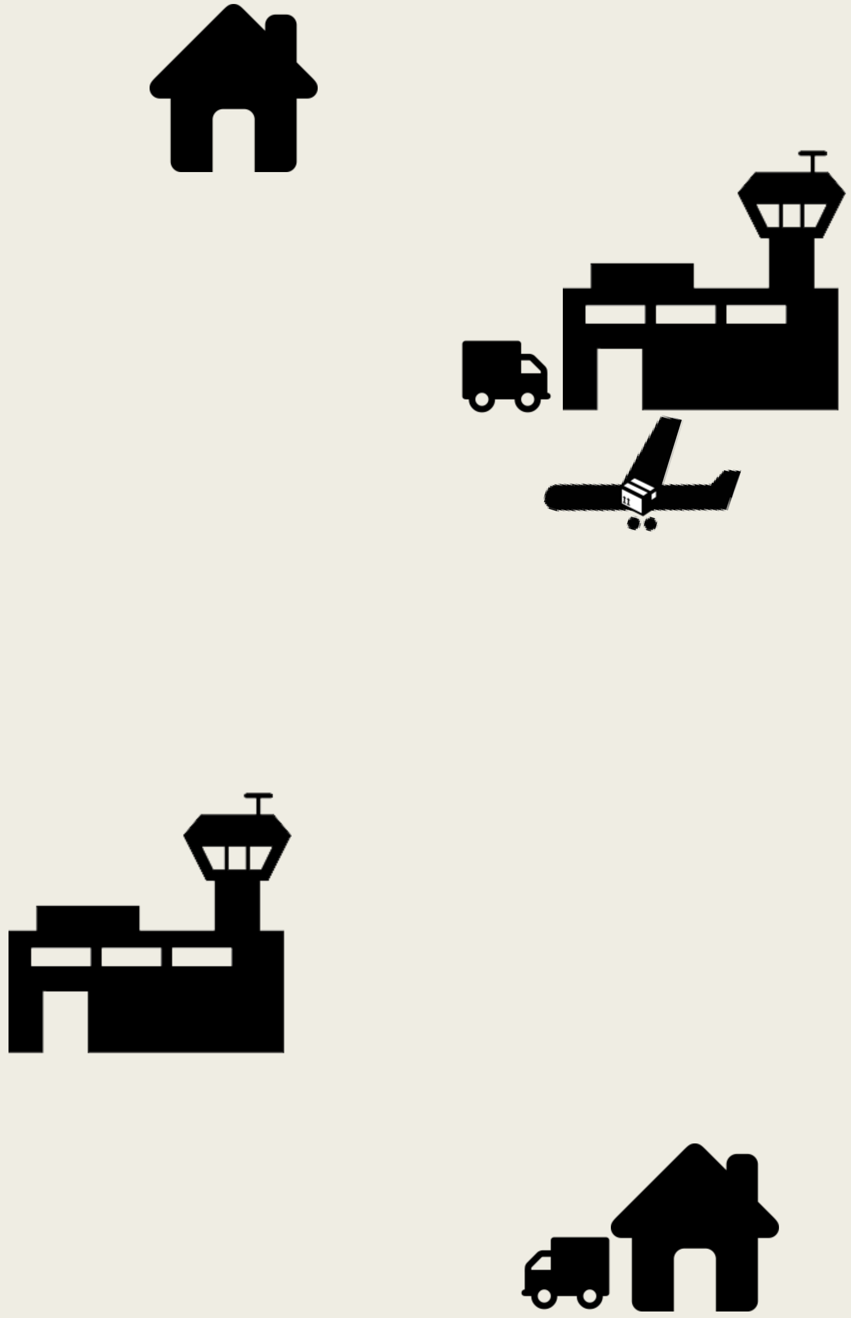
The Logistics Problem



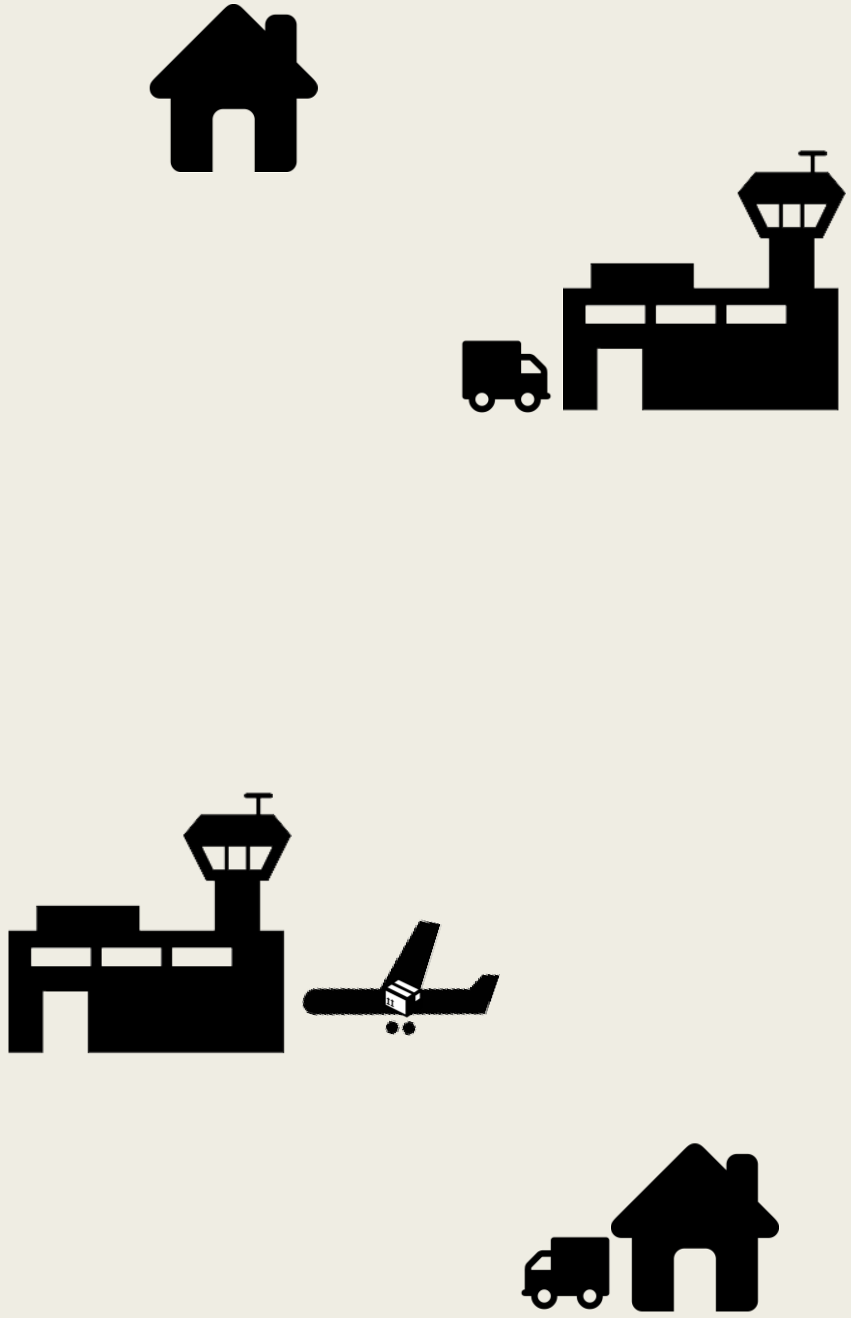
The Logistics Problem



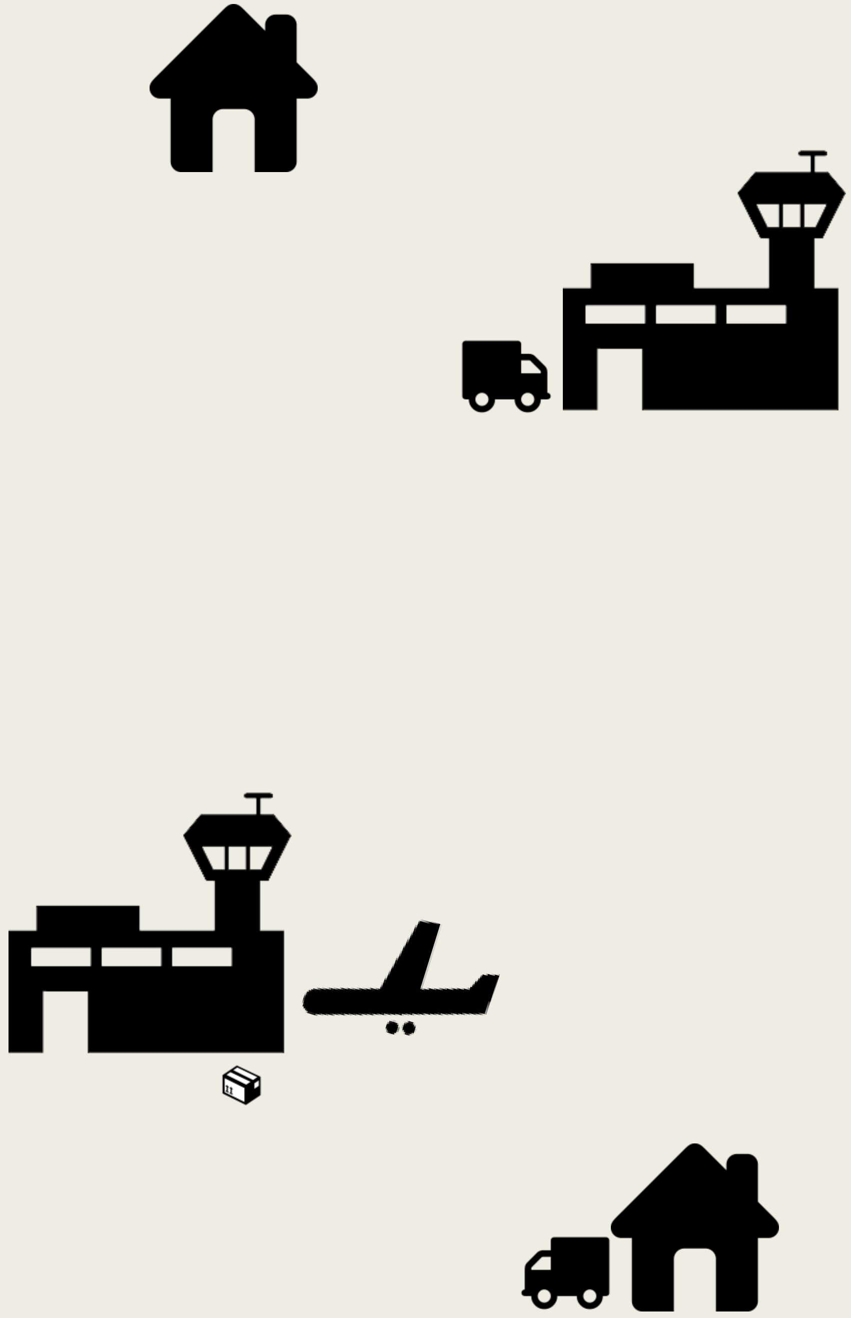
The Logistics Problem



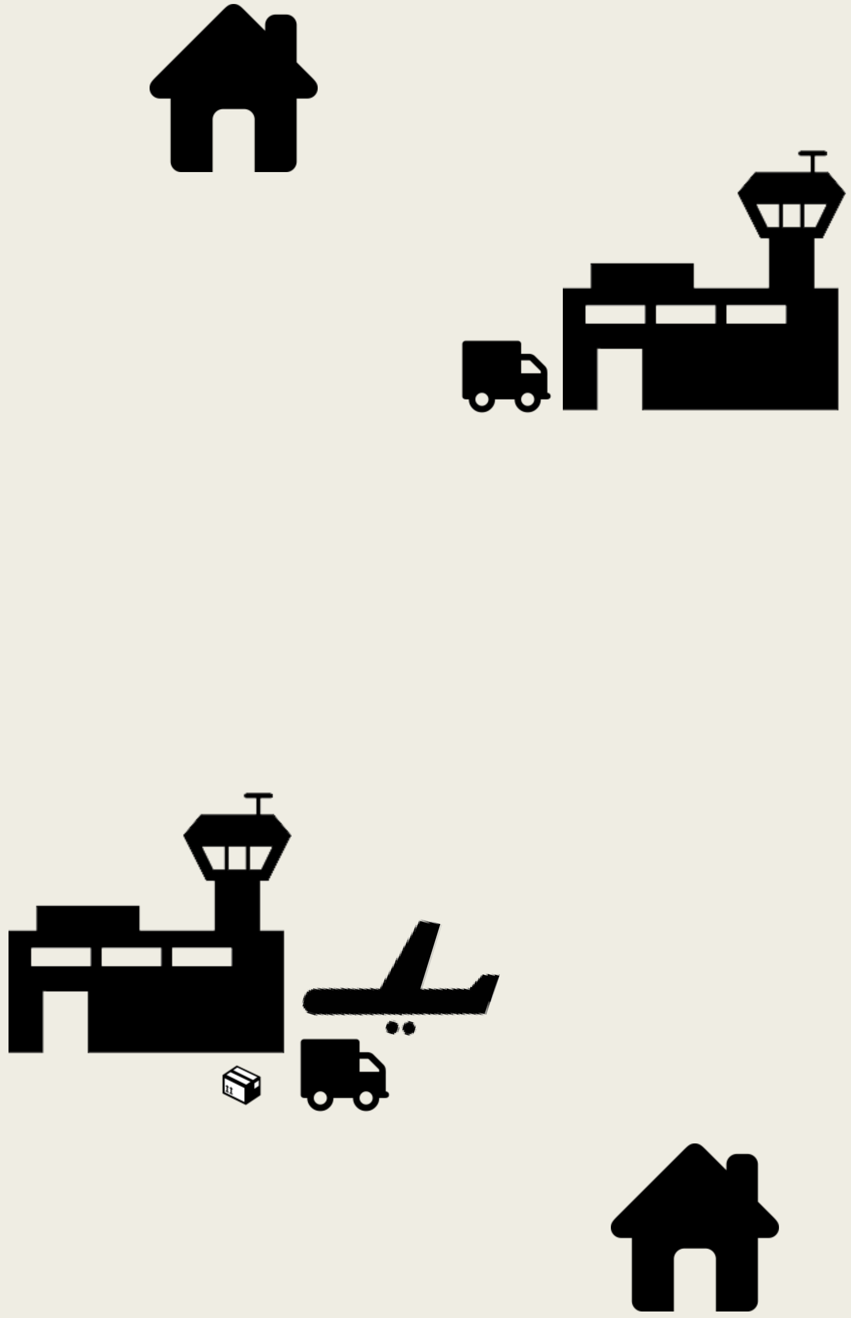
The Logistics Problem



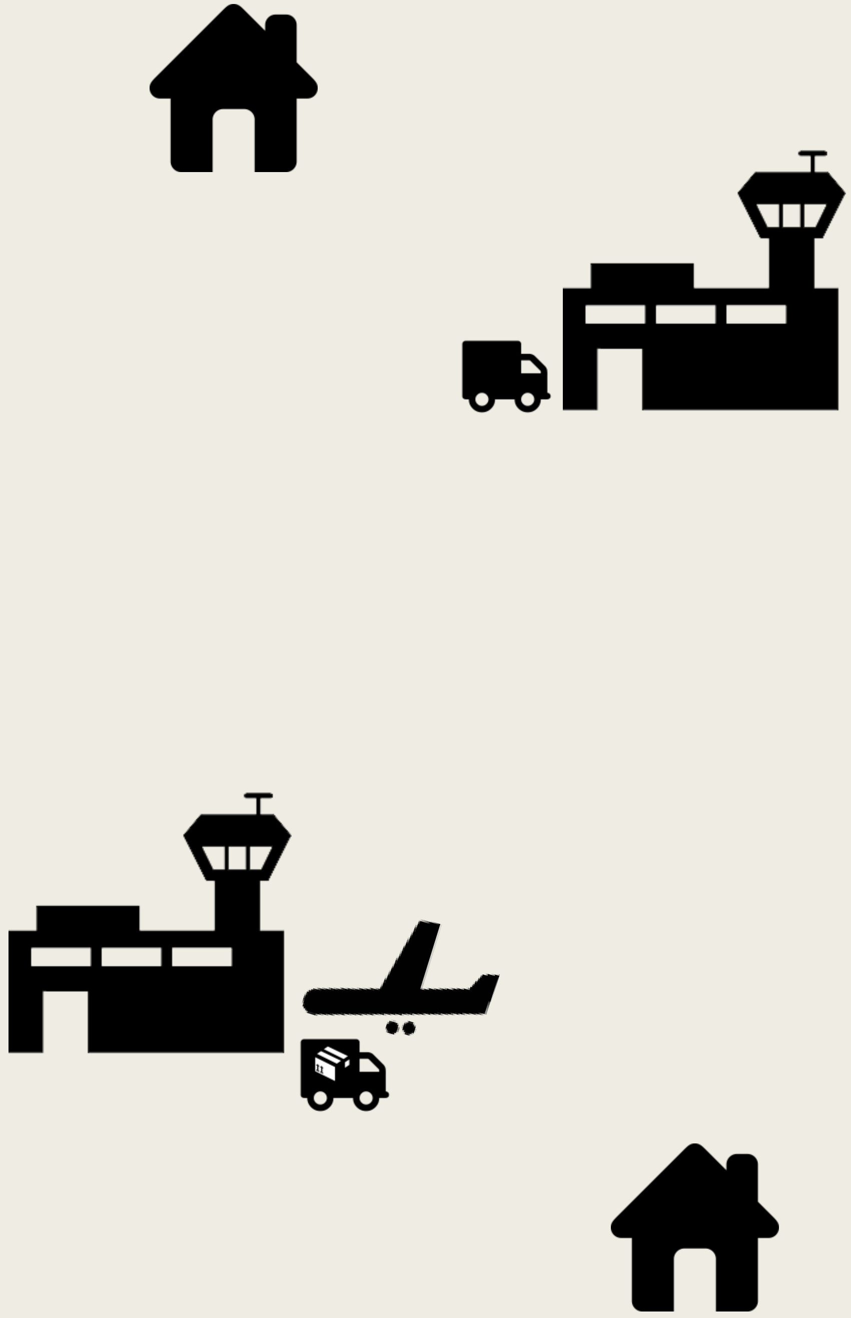
The Logistics Problem



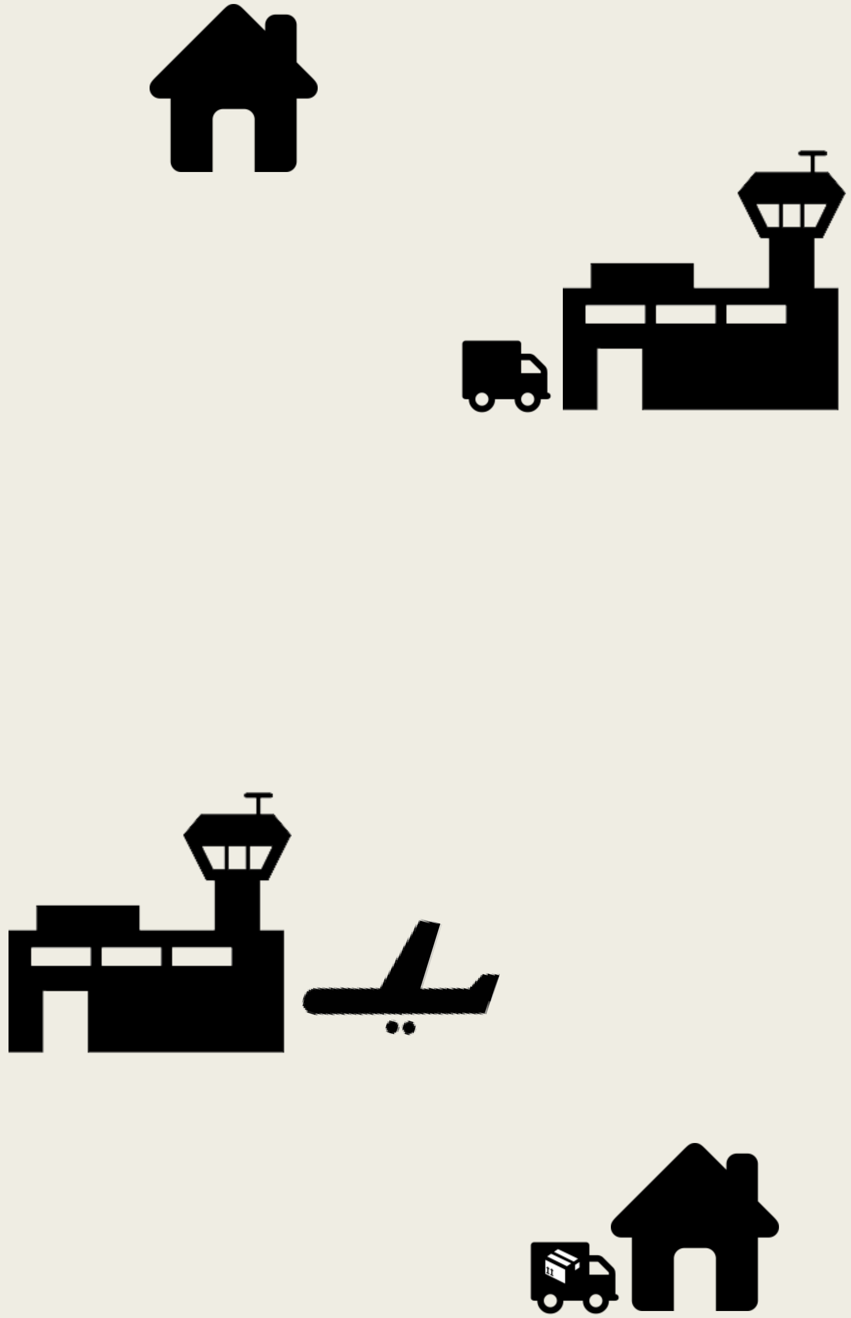
The Logistics Problem



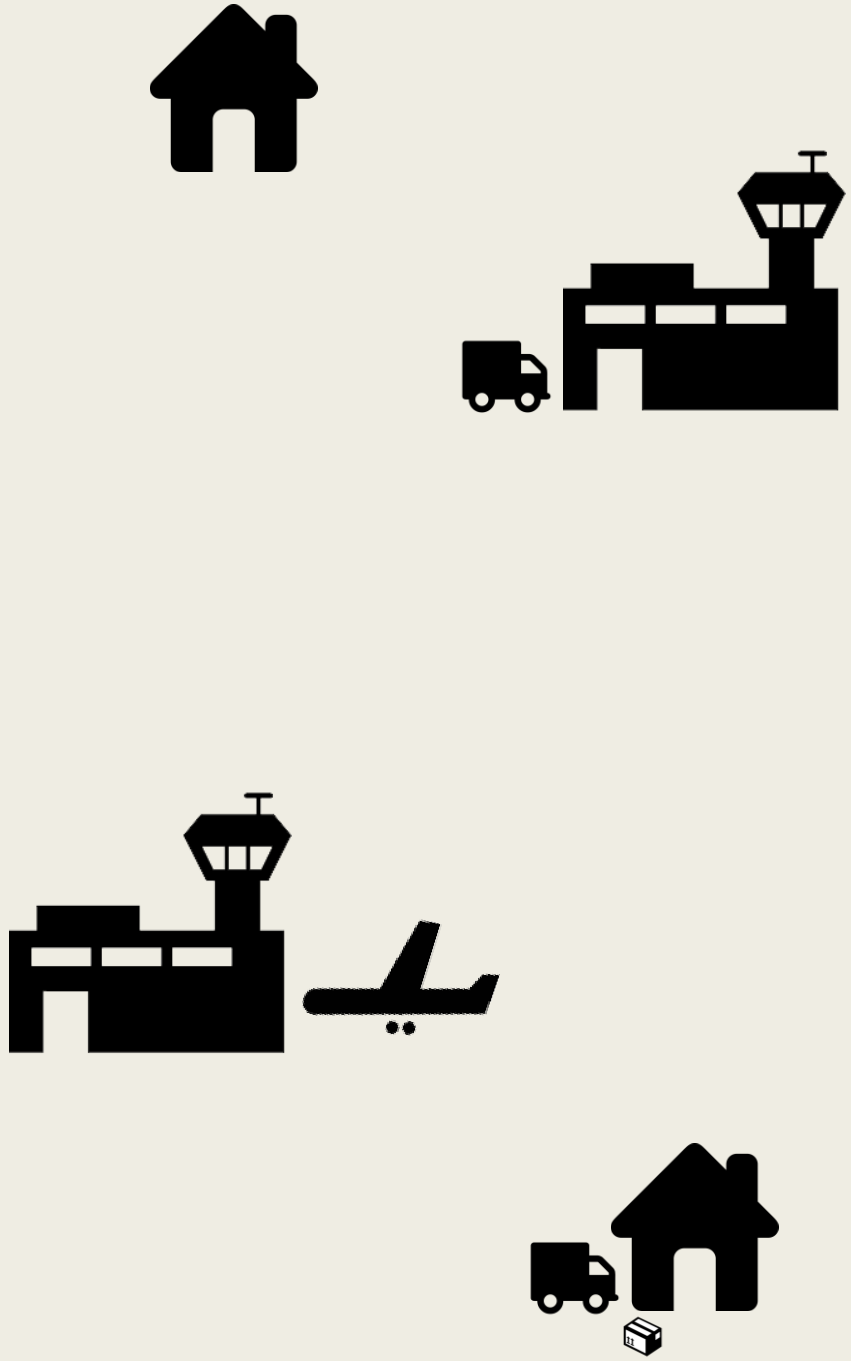
The Logistics Problem



The Logistics Problem



The Logistics Problem



Solving your problem

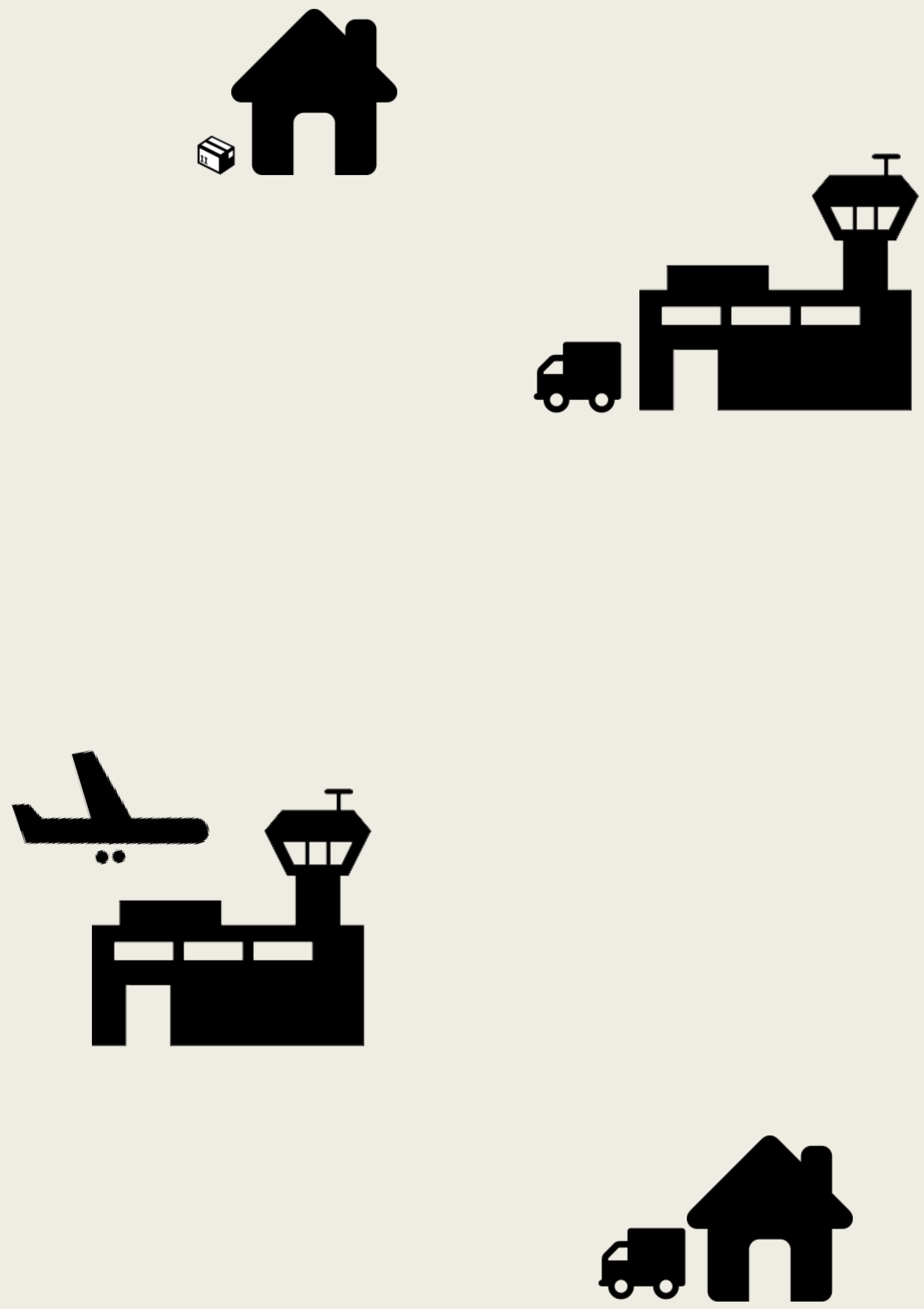
Domain specific solution

- Fast
- Takes development time
- Hard to get right

General solver (planning)

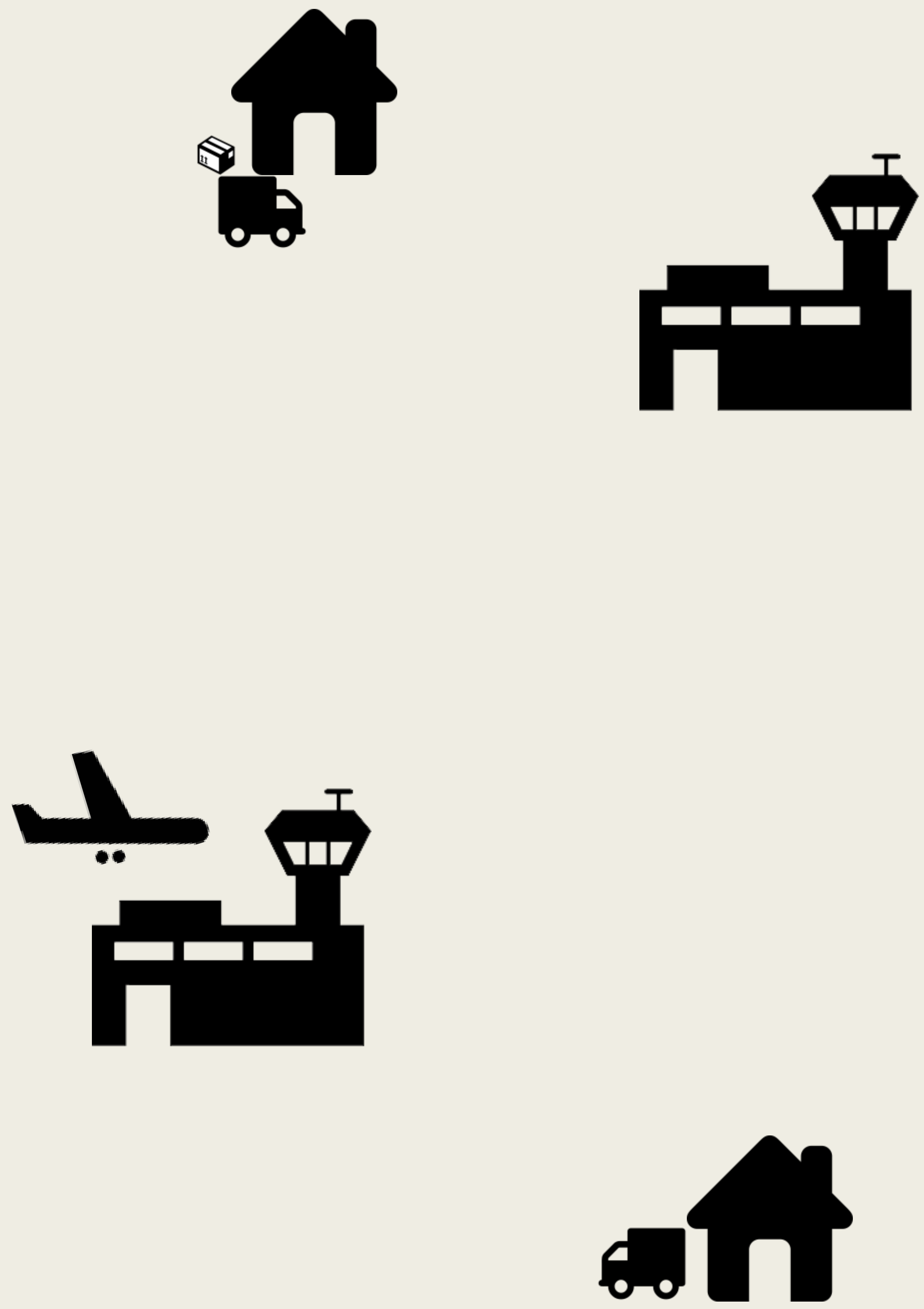
- Easy
- Highly optimized
- Requires just the right setup

A middle way?



A middle way?

- Identify interesting features

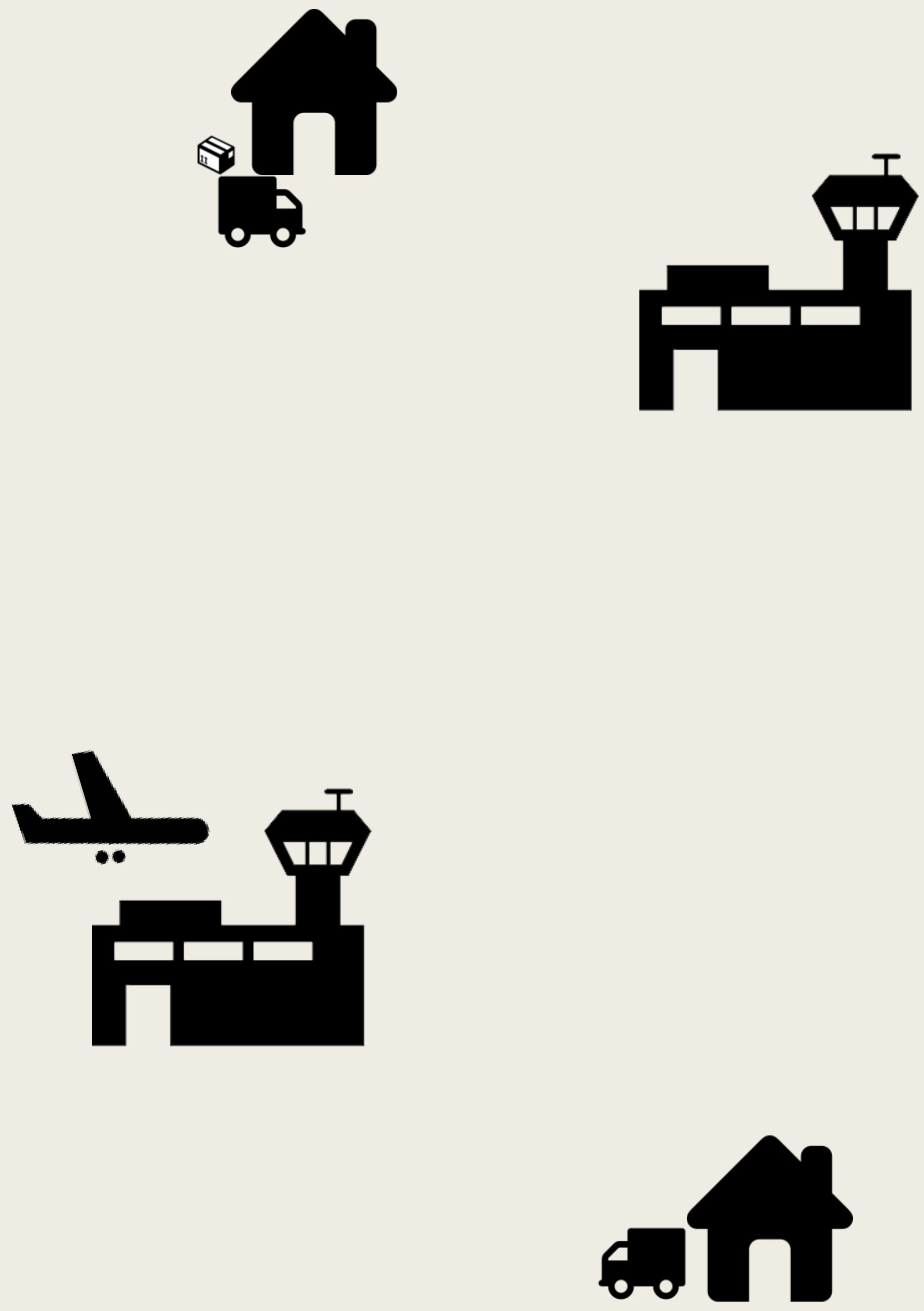


PDDL

package(package1)

at(package1, position1)
at(truck1, position1)
at(truck2, position2)
at(airplane, airport2)

in-city(position1, city1)
in-city(airport1, city1)
in-city(position2, city2)
in-city(airport2, city2)

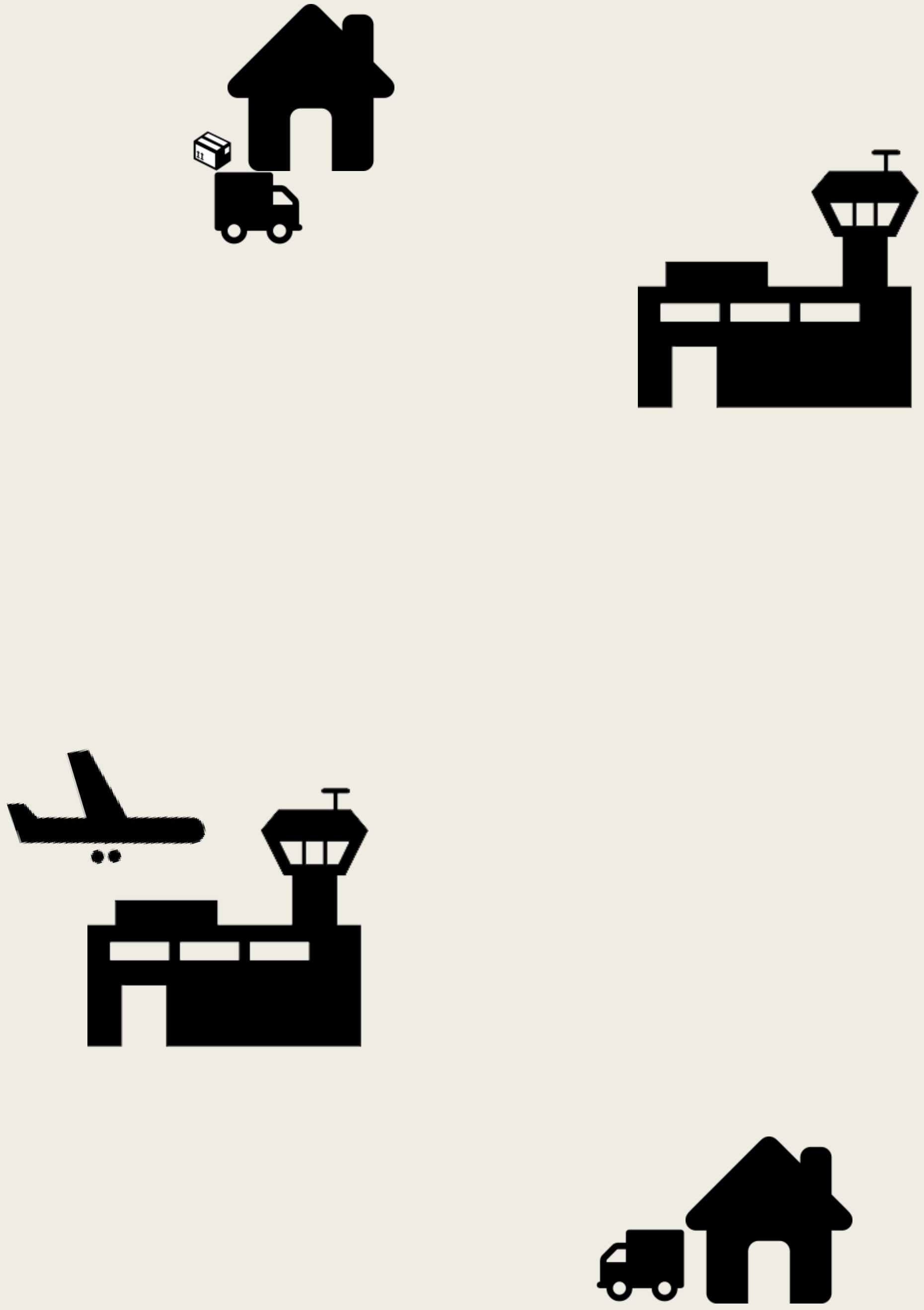


Concept languages

package → { package1 }

at → { (package1, position1),
(truck1, position1),
(truck2, position2),
(airplane, airport2) }

in-city → { (position1, city1),
(airport1, city1),
(position2, city2),
(airport2, city2) }

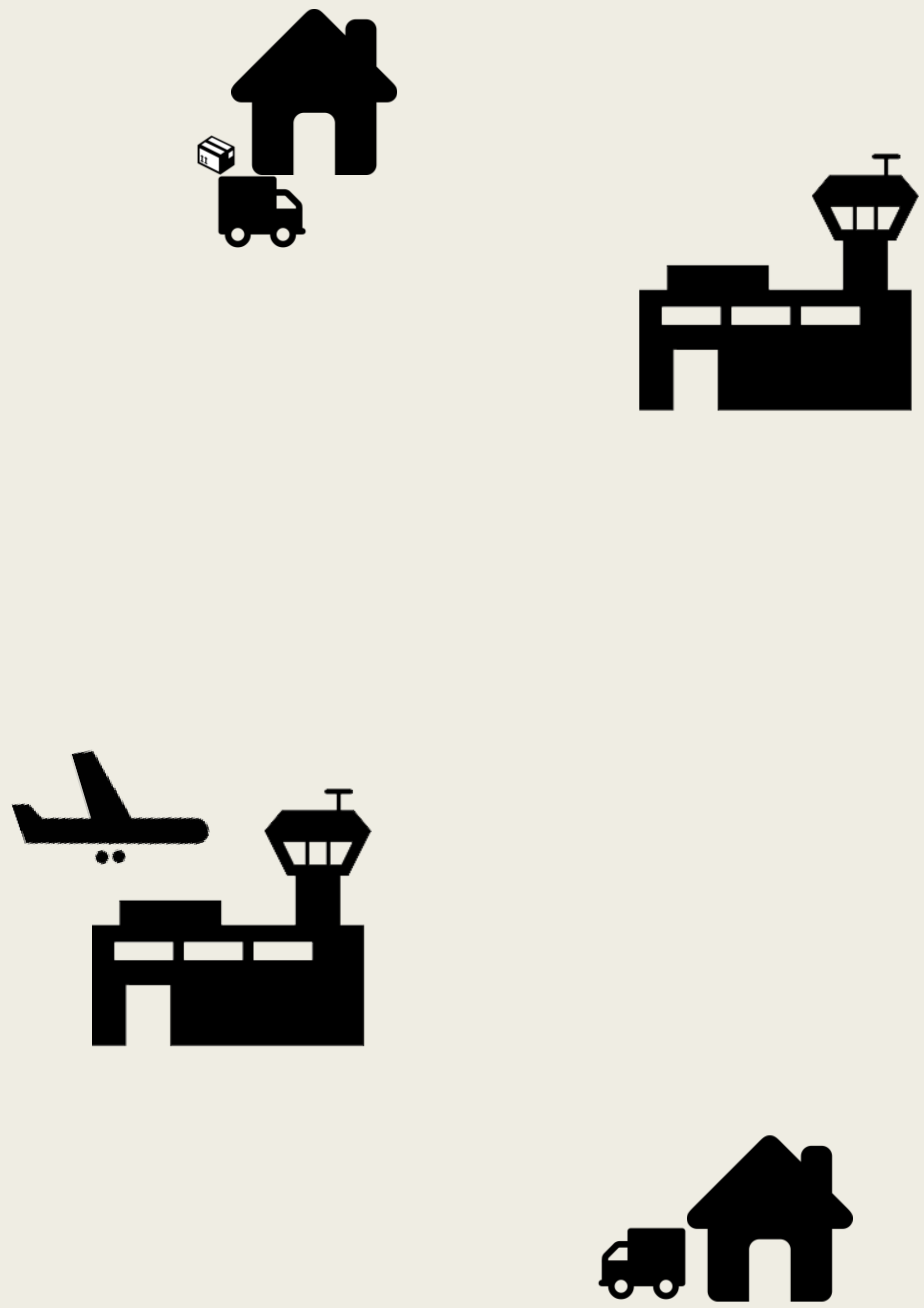


Concept languages

$\mathrm{truck} \sqcup airplane$
$\mathrm{truck} \sqcap airplane$
$\neg truck$

$\mathrm{in} \circ at$
$at = at_G$

$\exists at.airport$
$\forall in.truck$



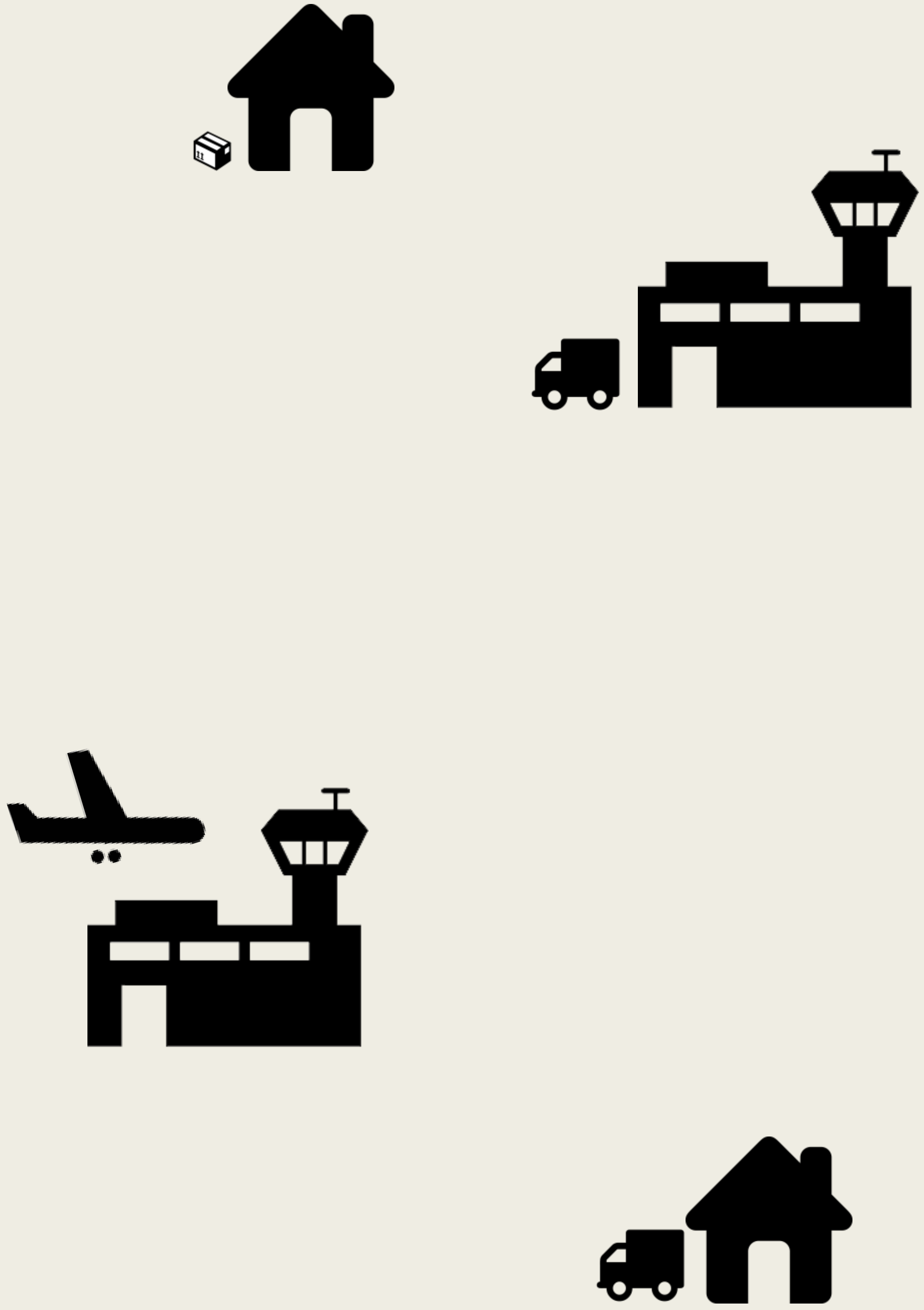
Features

- Cardinality: $|C|$
- Distance: $dist(C, R, C')$
- Multiplication: $f_1 \cdot f_2$

Features

#packages at wrong location with truck

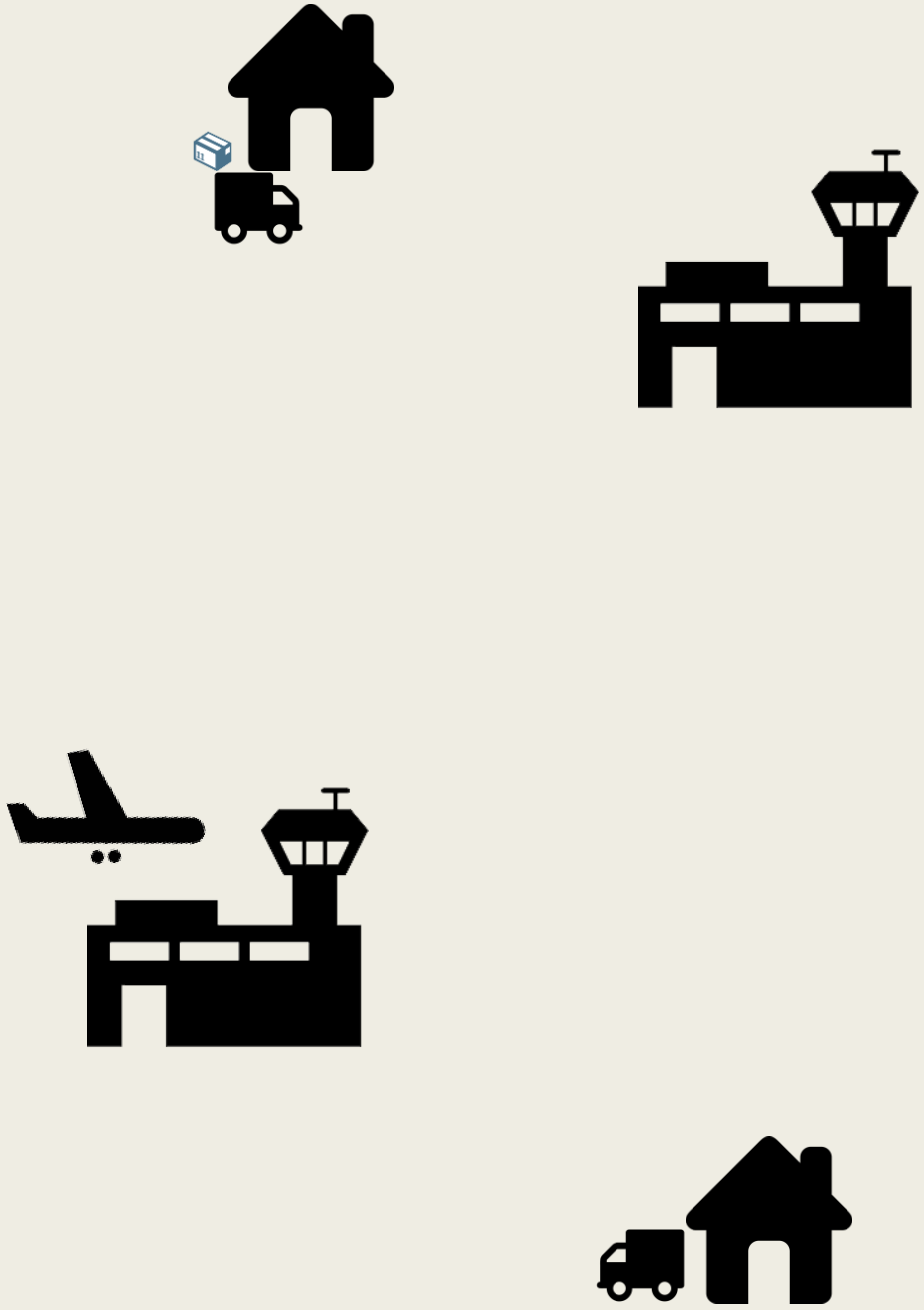
$$|package \sqcap \neg(at = at_G) \sqcap \exists at.truck| = 0$$



Features

#packages at wrong location with truck

$$|package \sqcap \neg(at = at_G) \sqcap \exists at.truck| = 1$$



Using features

- Fully explore small instances
- Learn a heuristic

Approach a descending heuristic

Minimize $\sum slack$

subject to $h(s) + slack \leq h(s') + 1$ for all states s and some successor s'

Locally approach h*

Minimize $\sum |slack|$

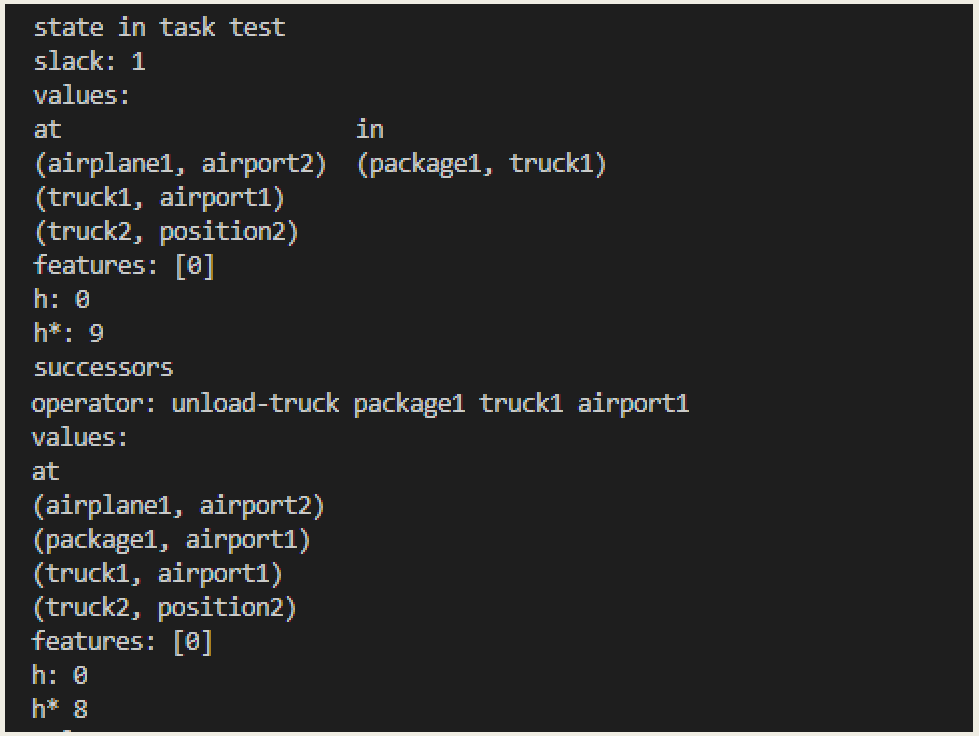
subject to $h(s) - h(s') + slack = h^*(s) - h^*(s')$ for all states s and all successors s'

Implementation

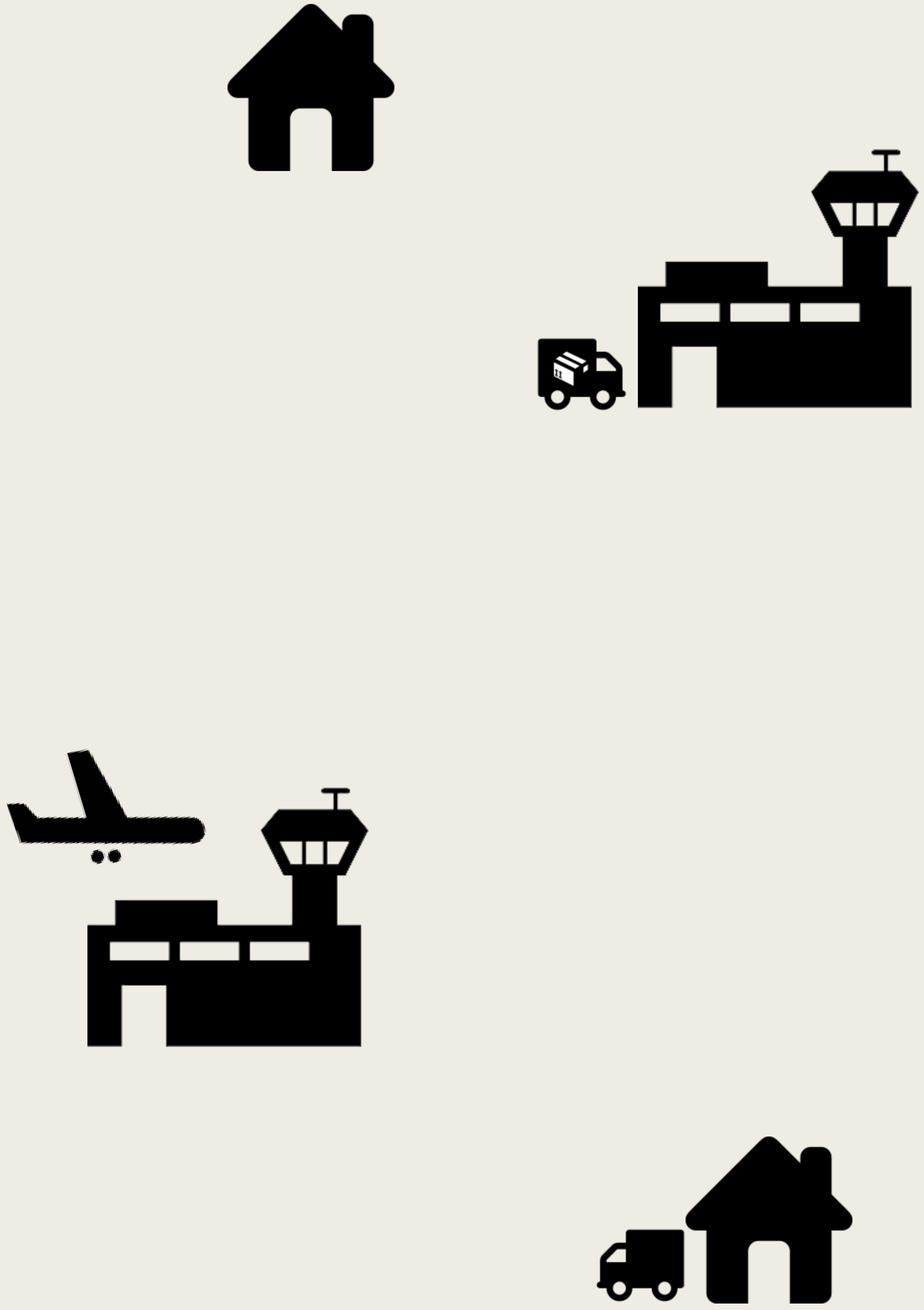
- Python command line tool
- Fast Downward
- CPLEX

Example

$|package \sqcap \neg(at = at_G) \sqcap \exists at.truck|$

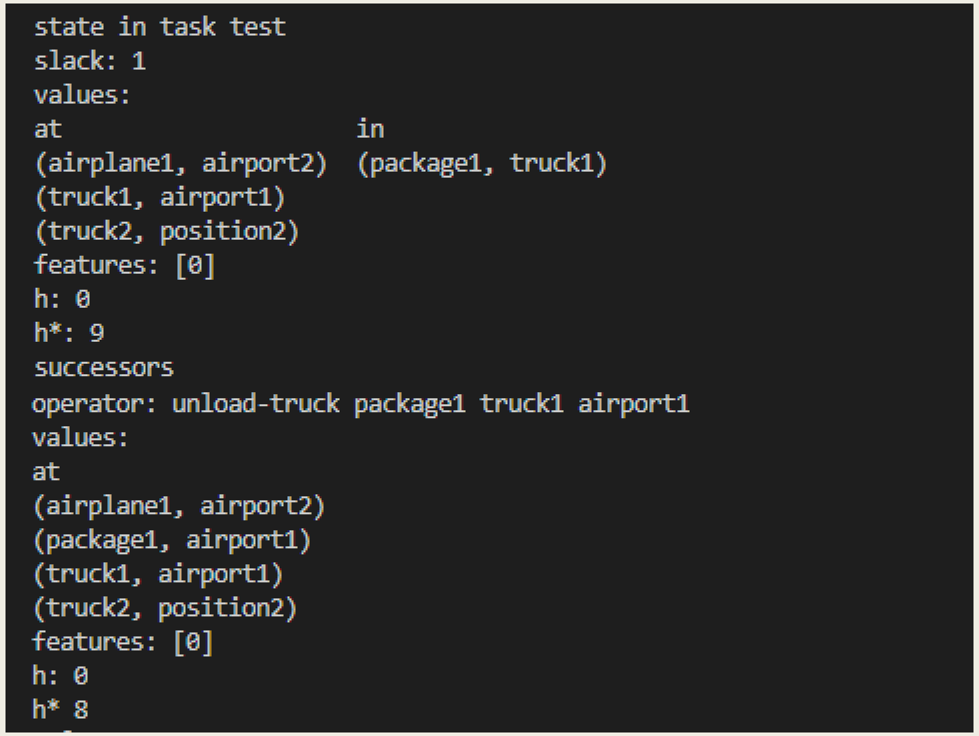


```
state in task test
slack: 1
values:
at                     in
(airplane1, airport2)  (package1, truck1)
(truck1, airport1)
(truck2, position2)
features: [0]
h: 0
h*: 9
successors
operator: unload-truck package1 truck1 airport1
values:
at
(airplane1, airport2)
(package1, airport1)
(truck1, airport1)
(truck2, position2)
features: [0]
h: 0
h* 8
```

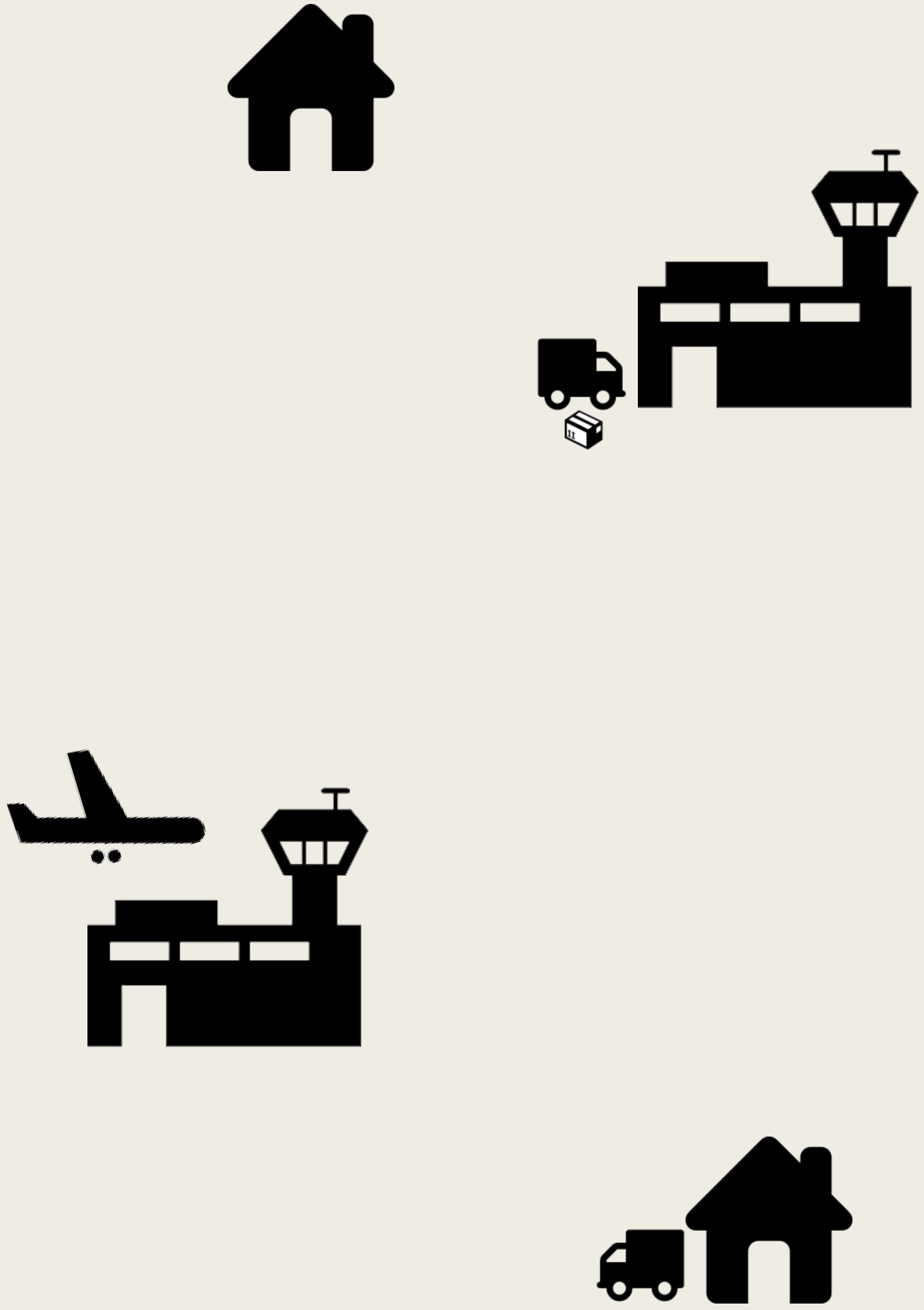


Example

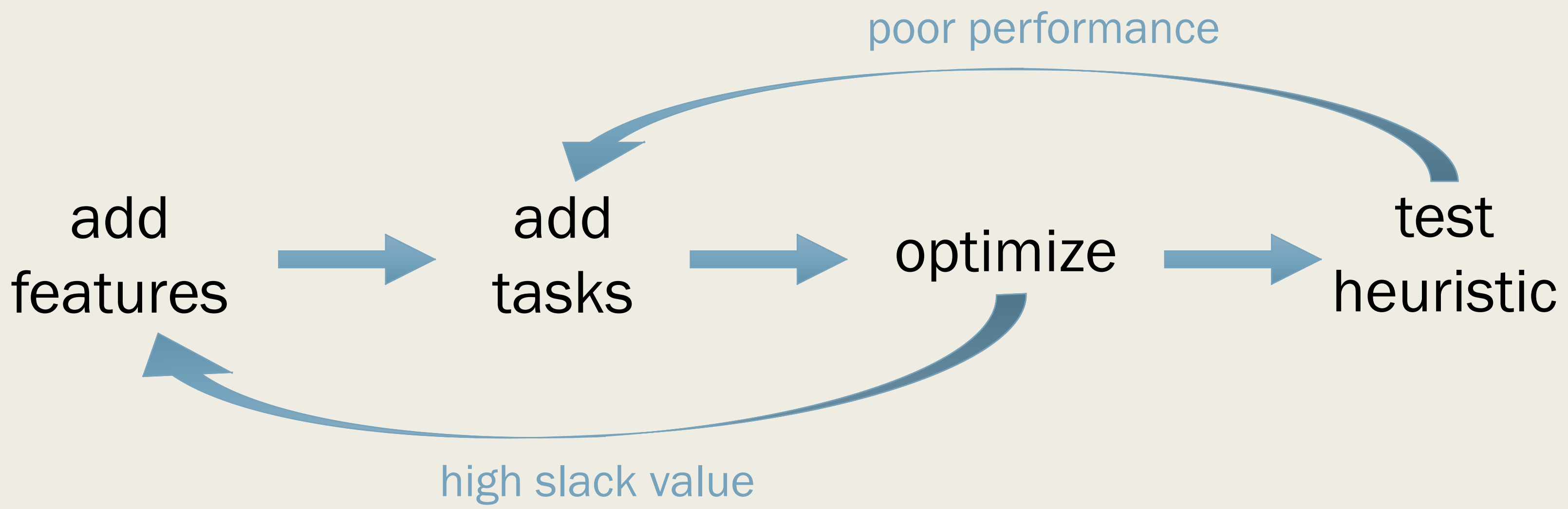
$$|package \sqcap \neg(at = at_G) \sqcap \exists at.truck|$$

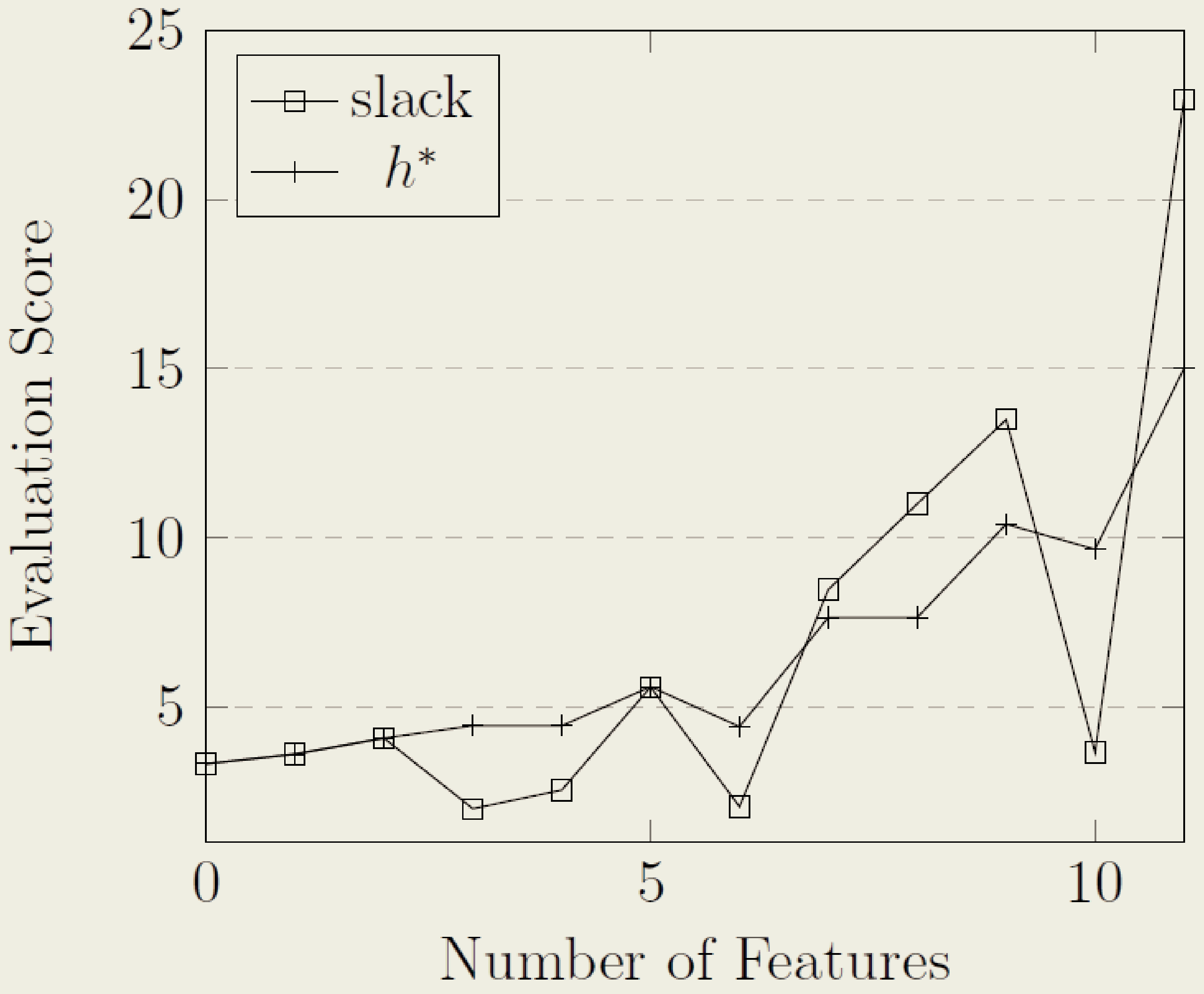


```
state in task test
slack: 1
values:
at                      in
(airplane1, airport2)  (package1, truck1)
(truck1, airport1)
(truck2, position2)
features: [0]
h: 0
h*: 9
successors
operator: unload-truck package1 truck1 airport1
values:
at
(airplane1, airport2)
(package1, airport1)
(truck1, airport1)
(truck2, position2)
features: [0]
h: 0
h* 8
```



Workflow


poor performance
add features
add tasks
optimize
test heuristic
high slack value

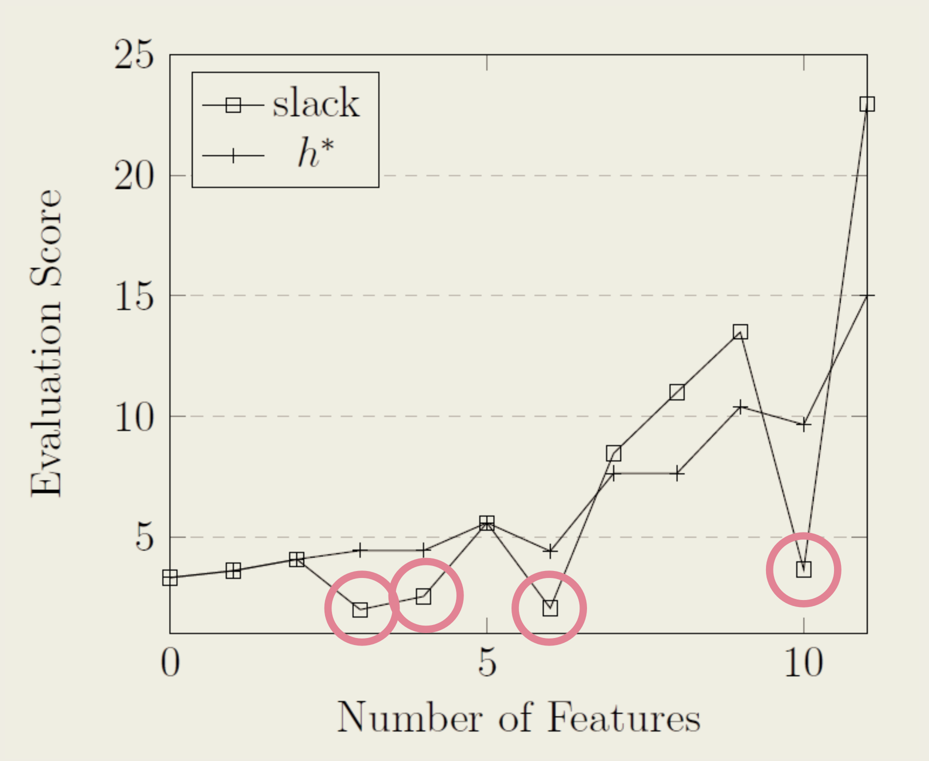


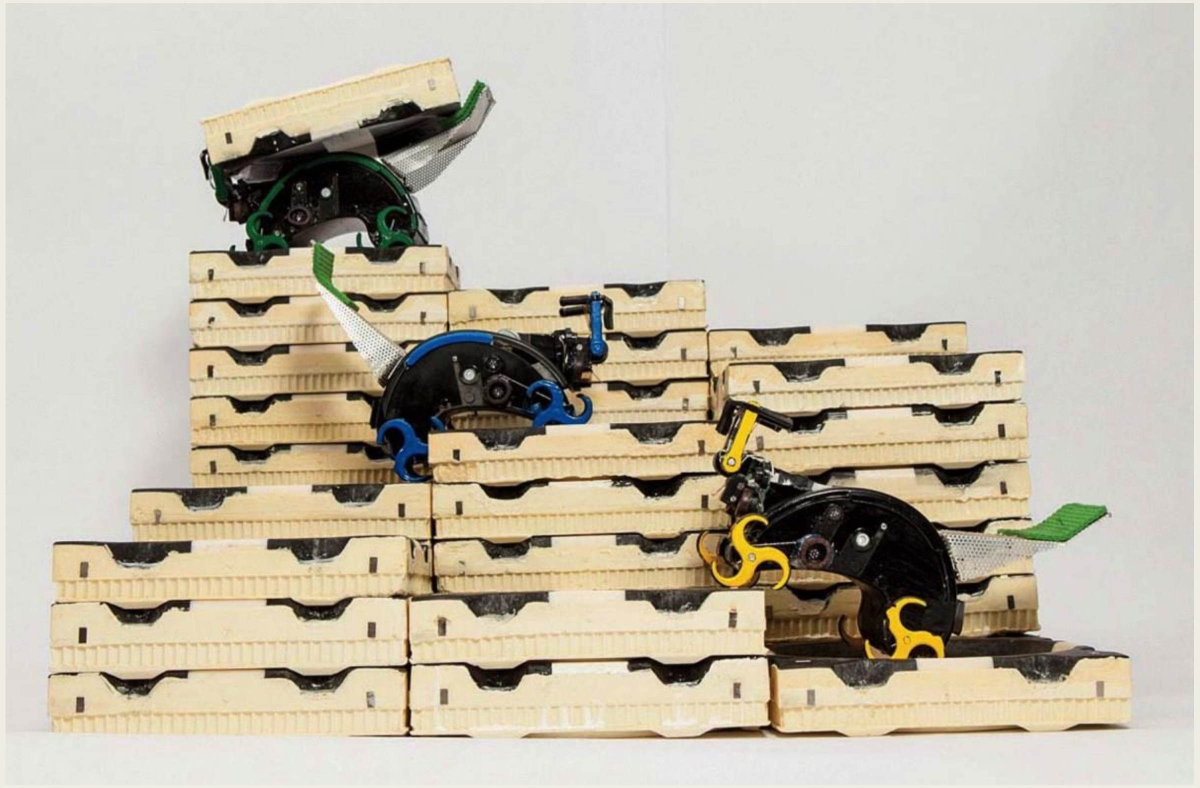
Results

- General upward trend
- h* more stable

Logistics problems

- With truck at airport ⟶ -6
- In truck at airport ⟶ -5
- In truck at destination ⟶ -4



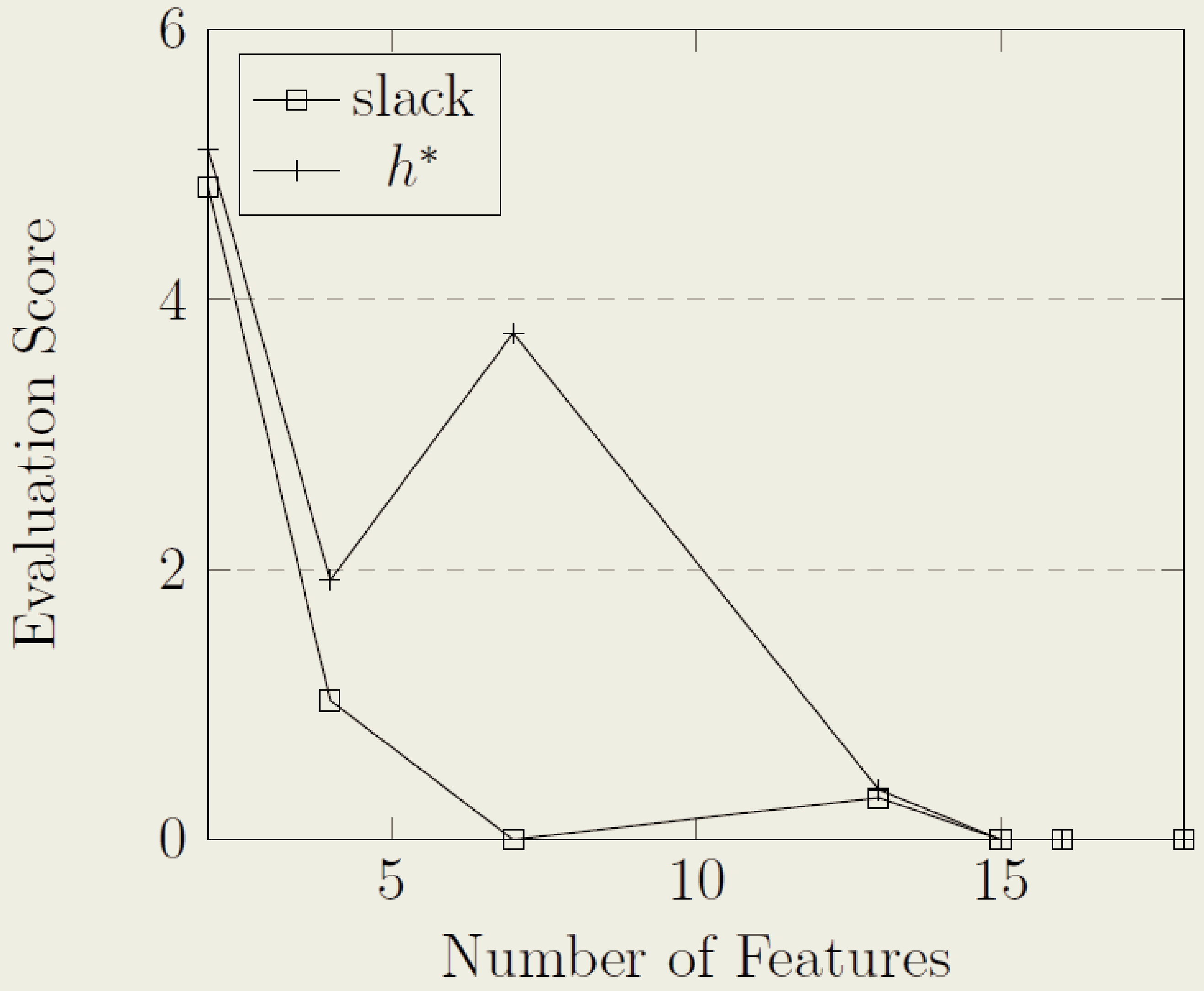


source: https://www.harvardmagazine.com/2017/11/robobee-harvard

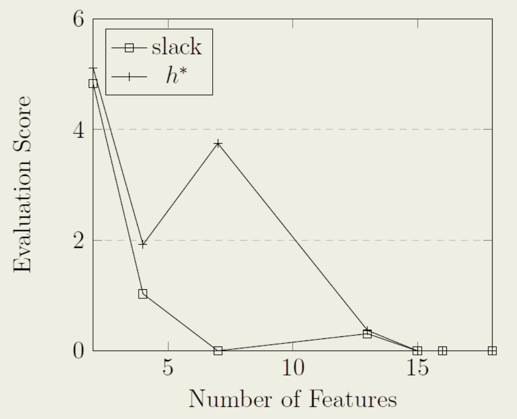
TERMES

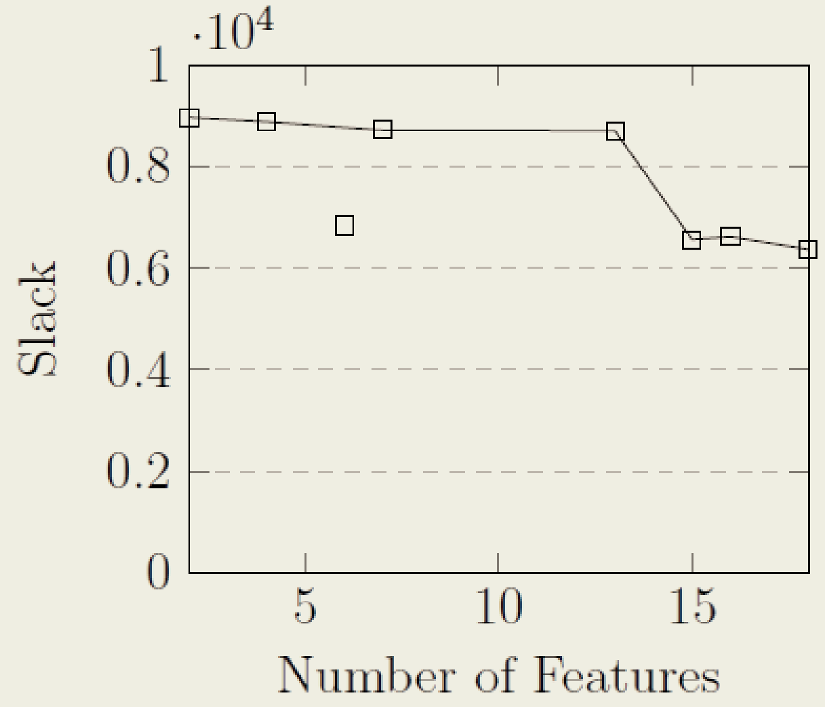
TERMES

- Downward trend
- h* more robust

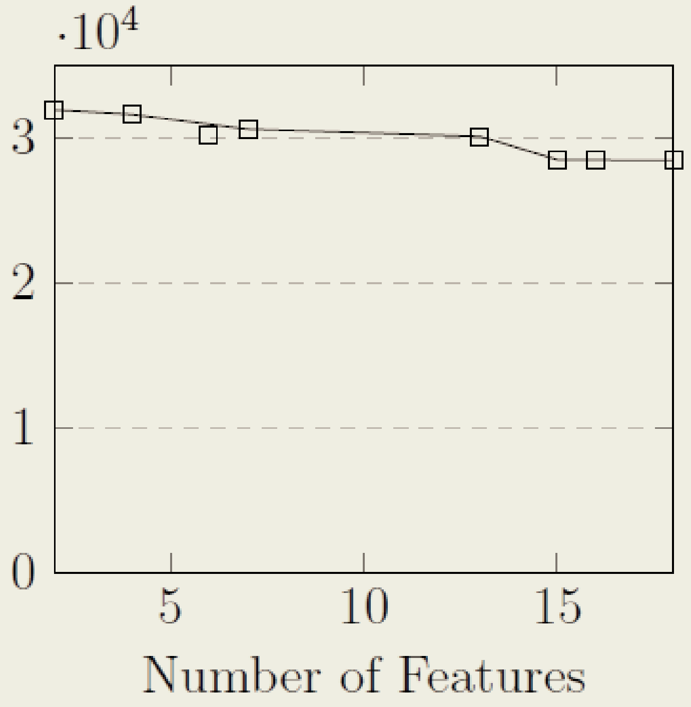


TERMES SLACK





(a) Heuristics found by $\mathcal{M}_{slack}$.



(b) Heuristics found by $\mathcal{M}^*$.

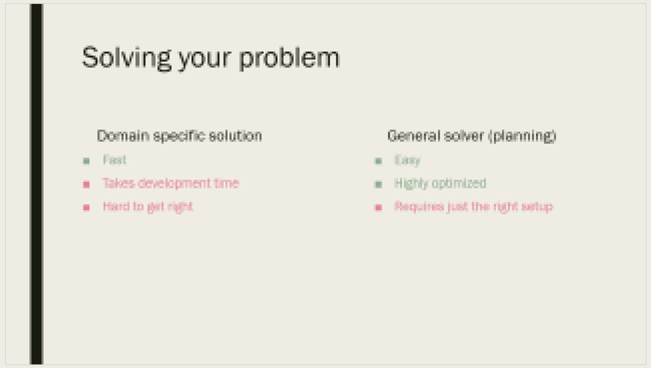
Conclusion

- Reasonably efficient and highly usable implementation
- Specifying domain knowledge
- Incorporating domain knowledge

Future work

- Automatically augment feature set
- Different learning objective
  - *Goal Distance Rank Correlation*
- Neural net for learning
- Interactive mode
- Scripting language

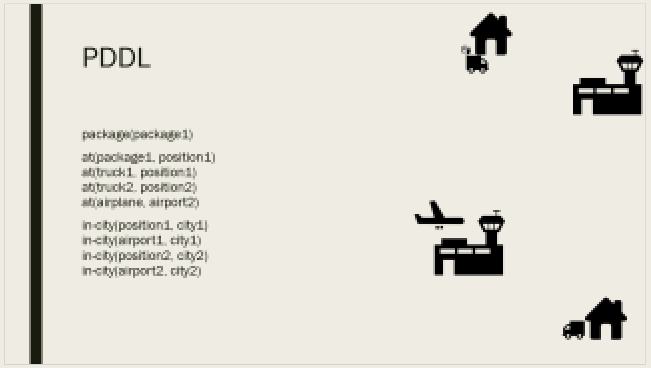
The Logistics Problem



Solving your problem

Domain specific solution
- Fast
- Takes development time
- Hard to get right

General solver (planning)
- Easy
- Highly optimized
- Requires just the right setup

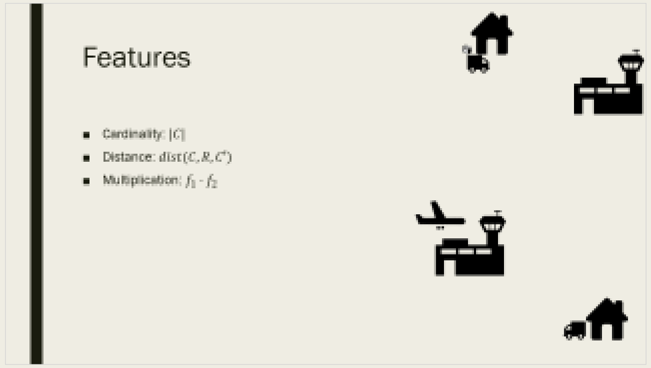


PDDL

package(package1)

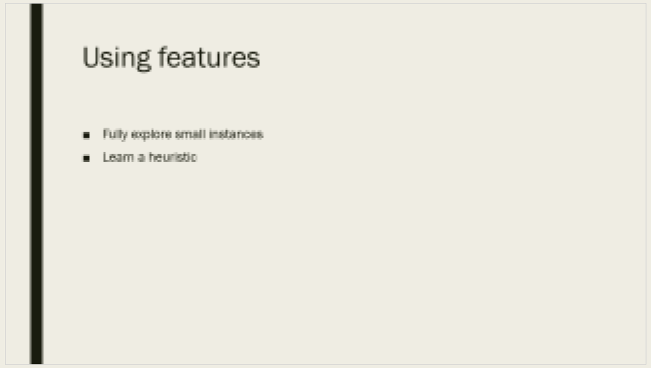
at(package1, position1)
at(truck1, position1)
at(truck2, position2)
at(airplane, airport2)

in-city(position1, city1)
in-city(airport1, city1)
in-city(position2, city2)
in-city(airport2, city2)



Features

- Cardinality: $|C|$
- Distance: $dist(C, R, C')$
- Multiplication: $f_1 \cdot f_2$

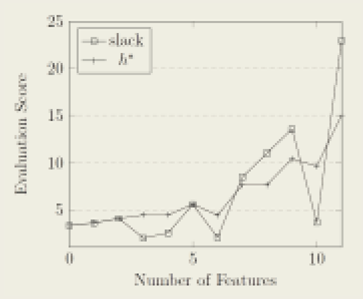


Using features

- Fully explore small instances
- Learn a heuristic

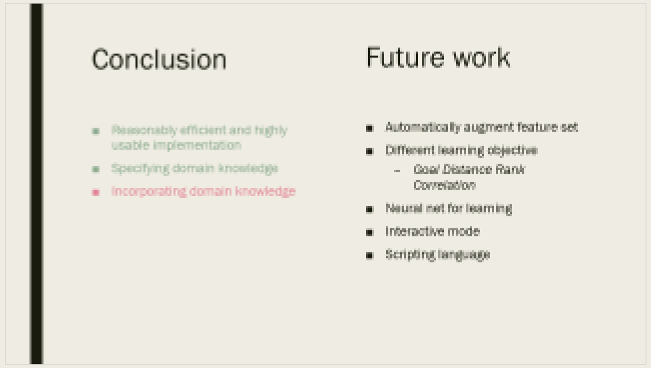
Implementation

- Python command line tool
- Fast Downward
- CPLEX



Results

- General upward trend
- h^* more stable



Conclusion

- Reasonably efficient and highly usable implementation
- Specifying domain knowledge
- Incorporating domain knowledge

Future work

- Automatically augment feature set
- Different learning objective
  - *Goal Distance Rank Correlation*
- Neural net for learning
- Interactive mode
- Scripting language

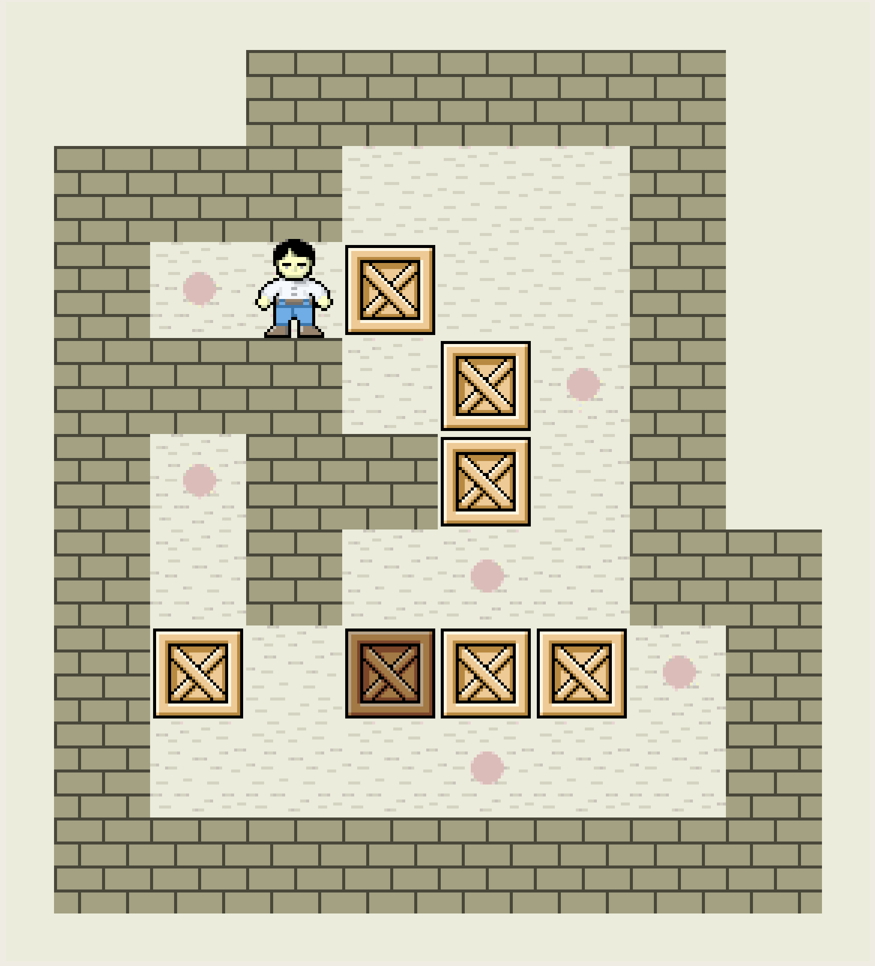
Extensions of Concept Languages

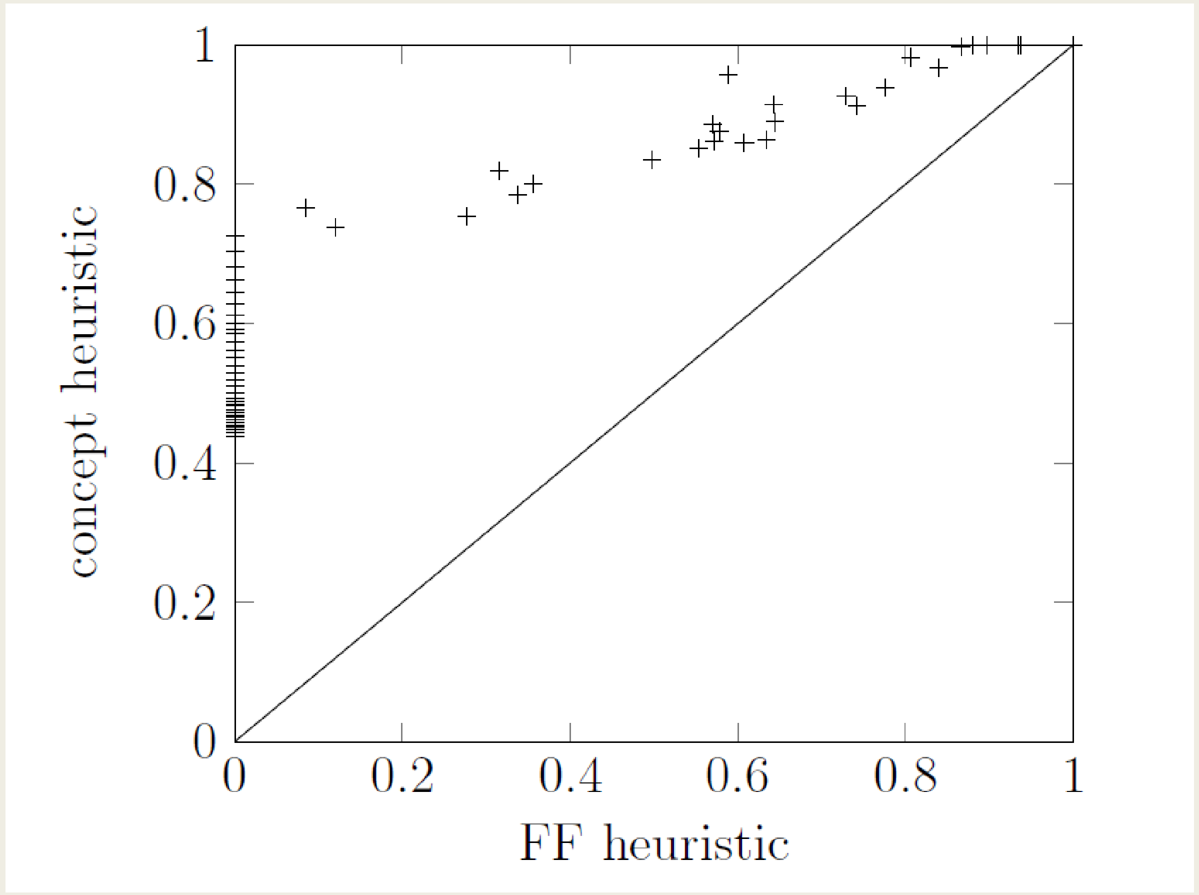
- n-ary roles
  - *selection + projection*
  - *existential and universal quantifier*
- role disjunction + conjunction
- Qualified cardinality restrictions
  - $> 2\ connected.clear$
- Heuristic feature

n-ary roles

MOVE-DIR location location direction

$connected = \exists x \in dir.\mathrm{MOVE{-}DIR}[\top, \top, x]$





VISITALL