# Constraint Satisfaction Problems 2

**CS 4300**
**Artificial Intelligence**

Prof. Alan Kuntz
alan.kuntz@utah.edu

Many slides adapted from Berkeley AI slides

# Announcements

- P2 due Friday 10/6 BUT
  - The entirety of Fall break will count as 1 penalty free late day.

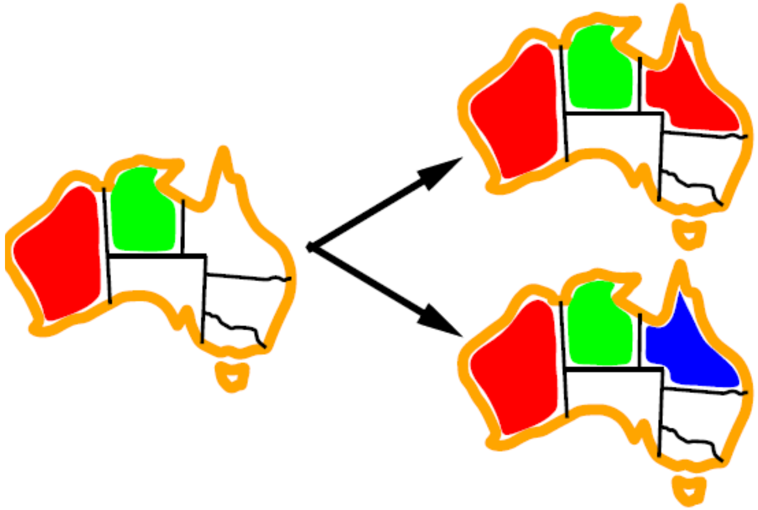
# Ordering: Minimum Remaining Values

- Variable Ordering: Minimum remaining values (MRV):
  - Choose the variable with the fewest legal left values in its domain

- Why min rather than max?
- Also called “most constrained variable”
- “Fail-fast” ordering

# Ordering: Least Constraining Value

- Value Ordering: Least Constraining Value
  - Given a choice of variable, choose the *least constraining value*
  - I.e., the one that rules out the fewest values in the remaining variables
  - Note that it may take some computation to determine this! (E.g., rerunning filtering)

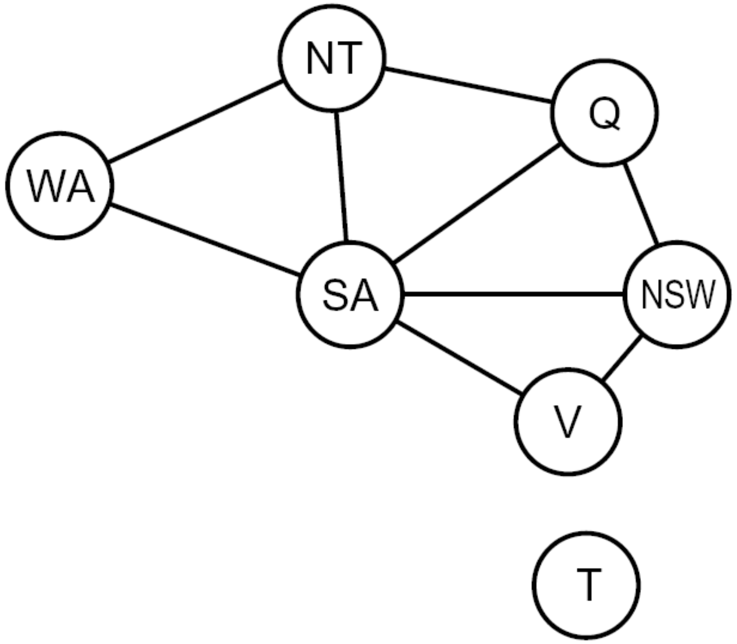


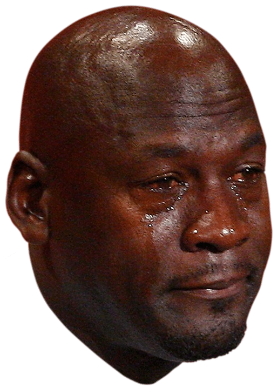
- Why least rather than most?

- Combining these ordering ideas makes 1000 queens feasible

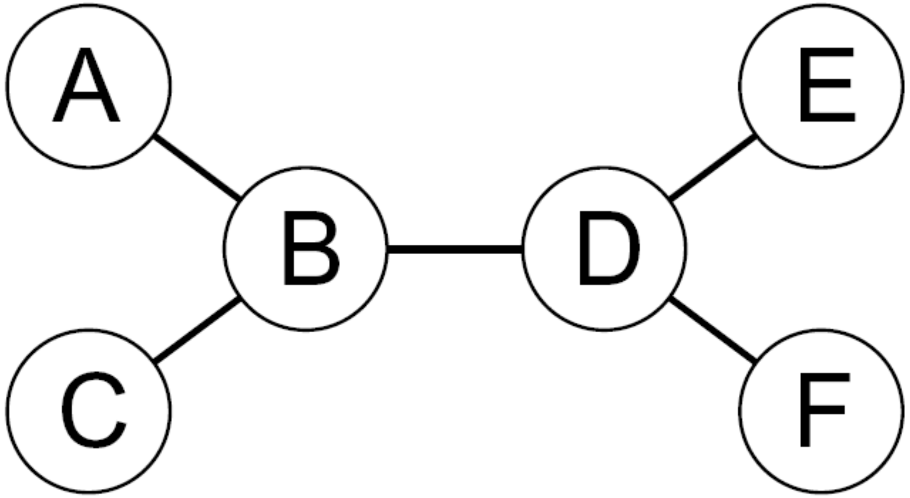
# Problem Structure

- Extreme case: independent subproblems
  - Example: Tasmania and mainland do not interact
- Independent subproblems are identifiable as connected components of constraint graph
- Suppose a graph of n variables can be broken into subproblems of only c variables:
  - Worst-case solution cost is $O((n/c)(d^c))$, much better than $O(d^n)$!
  - E.g., n = 80, d = 2, c =20
  - $2^{80}$ = 4 billion years at 10 million nodes/sec
  - $(4)(2^{20})$ = 0.4 seconds at 10 million nodes/sec
- Sad news: Doesn't come up that often



# Tree-Structured CSPs

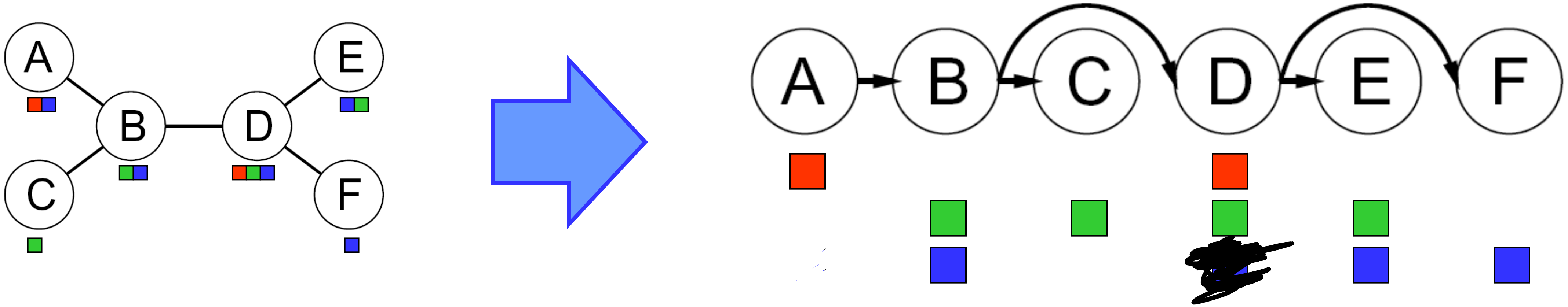



- Theorem: if the constraint graph has no cycles, the CSP can be solved in $O(n\ d^2)$ time
  - Compare to general CSPs, where worst-case time is $O(d^n)$

- This property also applies to probabilistic reasoning (later)

# Tree-Structured CSPs

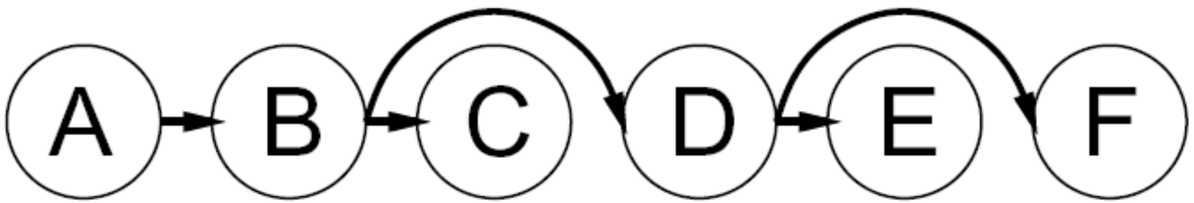
- Algorithm for tree-structured CSPs:
  - Order: Choose a root variable, order variables so that parents precede children




  - Remove backward: For i = n : 2, apply RemoveInconsistent(Parent($X_i$),$X_i$)
  - Assign forward: For i = 1 : n, assign $X_i$ consistently with Parent($X_i$)
- Runtime: $O(n\ d^2)$ (why?)

# Tree-Structured CSPs

- Claim 1: After backward pass, all root-to-leaf arcs are consistent
- Proof: Each X→Y was made consistent at one point and Y's domain could not have been reduced thereafter (because Y's children were processed before Y)

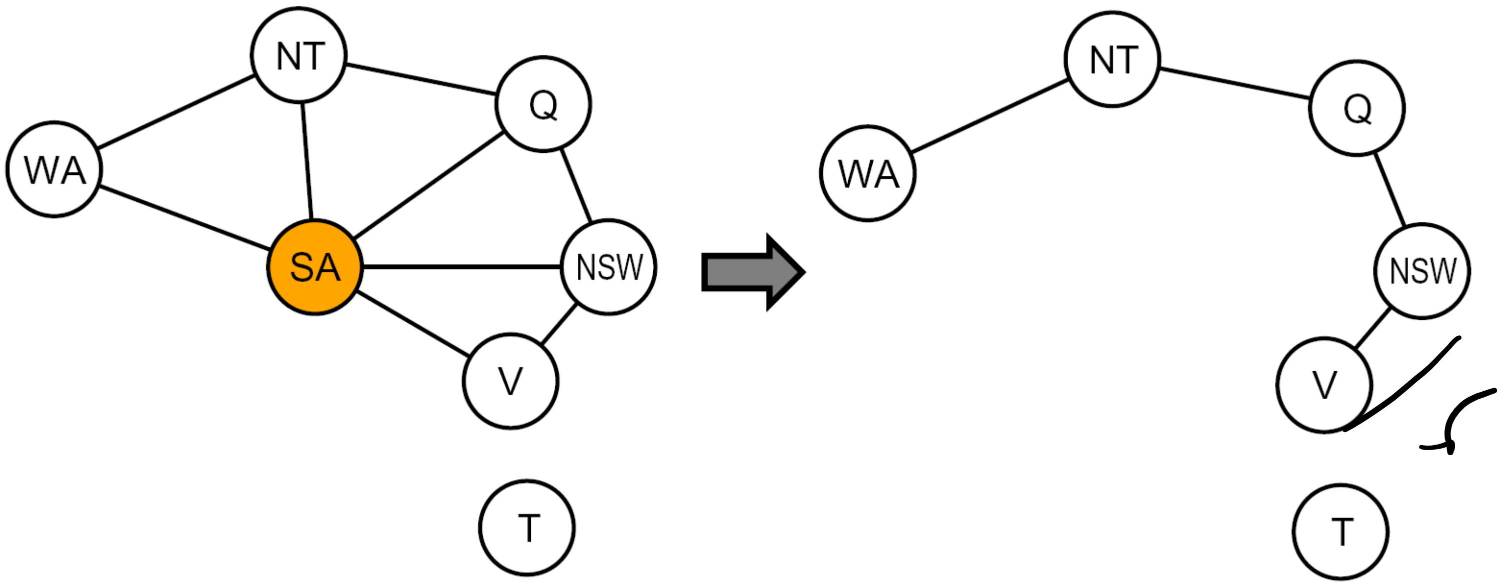



- Claim 2: If root-to-leaf arcs are consistent, forward assignment will not backtrack
- Proof: Induction on position

- Why doesn't this algorithm work with cycles in the constraint graph?

- Note: we'll see this basic idea again with Bayes' nets

# Nearly Tree-Structured CSPs




- Conditioning: instantiate a variable, prune its neighbors' domains
- Cutset conditioning: instantiate (***<u>in all ways</u>***) a set of variables such that the remaining constraint graph is a tree
- Cutset size c gives runtime $O( (d^c) (n-c) d^2 )$, very fast for small c

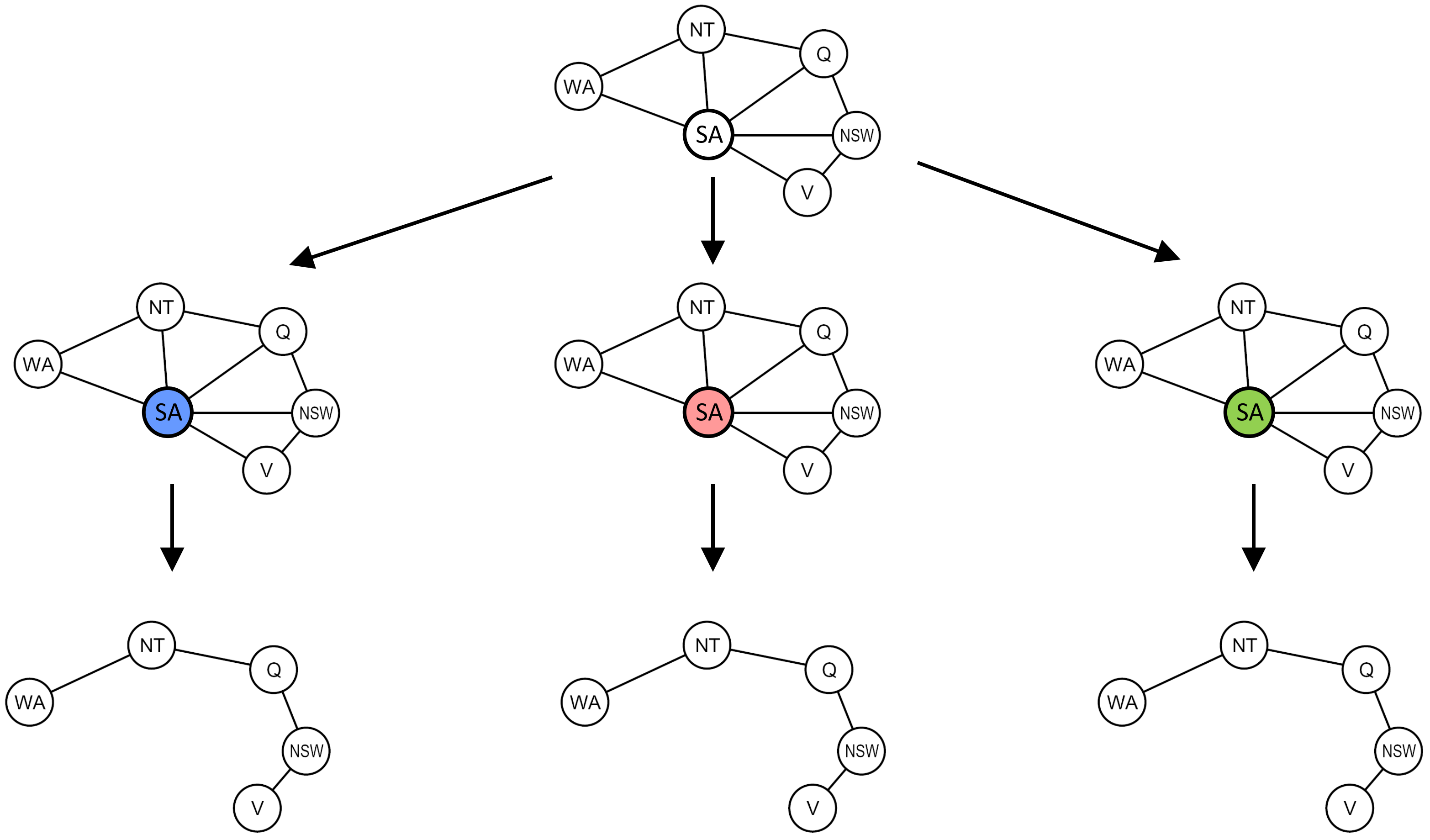
# Cutset Conditioning

Choose a cutset

Instantiate the cutset (all possible ways)

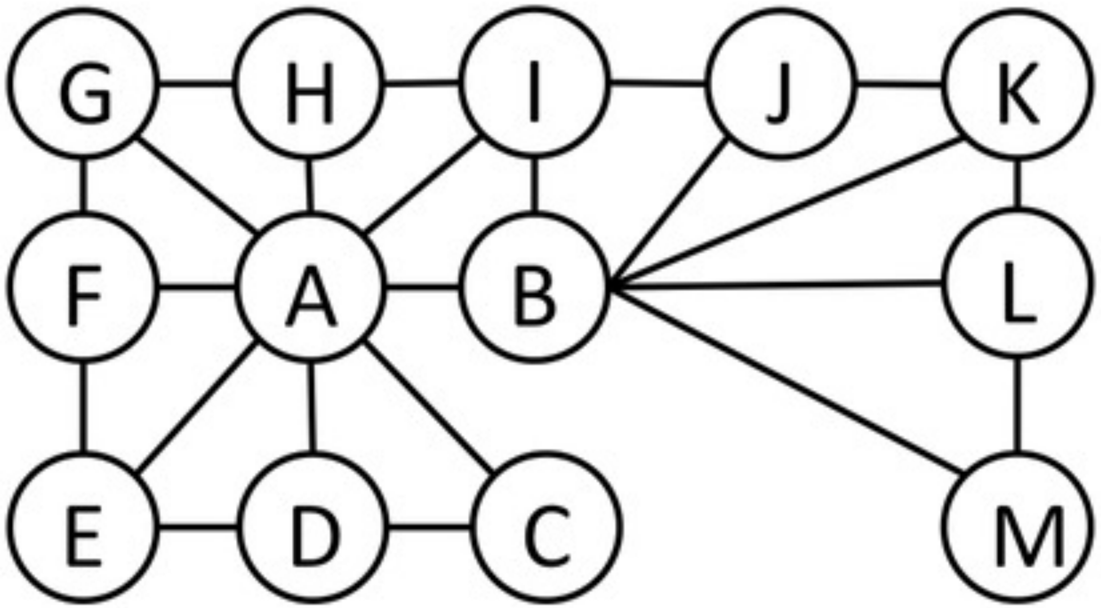
Compute residual CSP for each assignment

Solve the residual CSPs (tree structured)



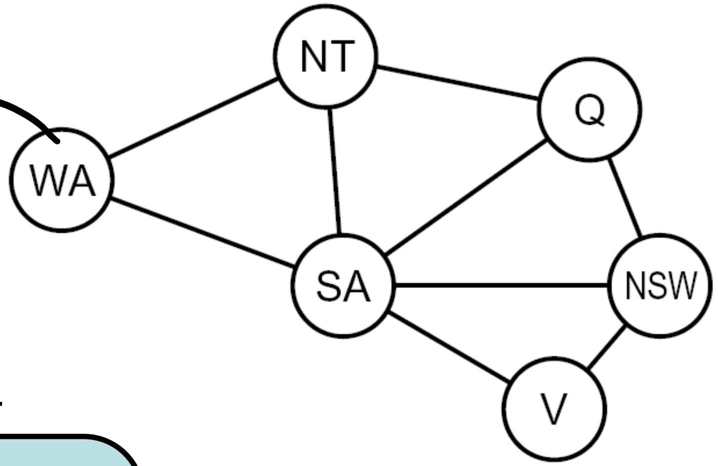
# Cutset Quiz

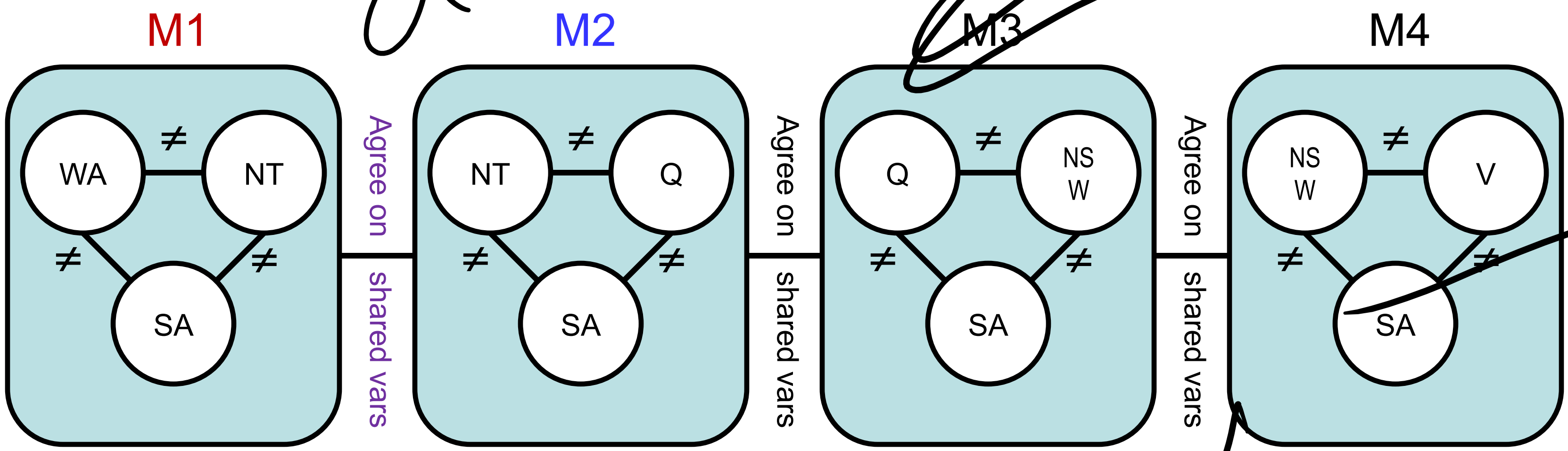
- Find the smallest cutset that results in a tree for the graph below.



# Tree Decomposition

- Idea: create a tree-structured graph of mega-variables
- Each mega-variable encodes part of the original CSP
- Subproblems overlap to ensure consistent solutions




{(WA=r,SA=g,NT=b), (WA=b,SA=r,NT=g), …}

{(NT=r,SA=g,Q=b), (NT=b,SA=g,Q=r), …}

Agree: (M1,M2) ∈ {((WA=g,SA=g,NT=g), (NT=g,SA=g,Q=g)), …}

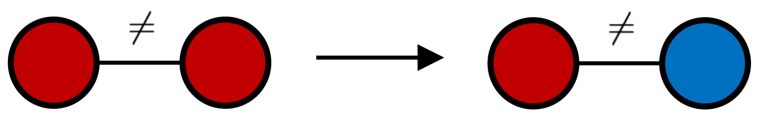
$O(nd^2)$

# Iterative Improvement

# Iterative Algorithms for CSPs

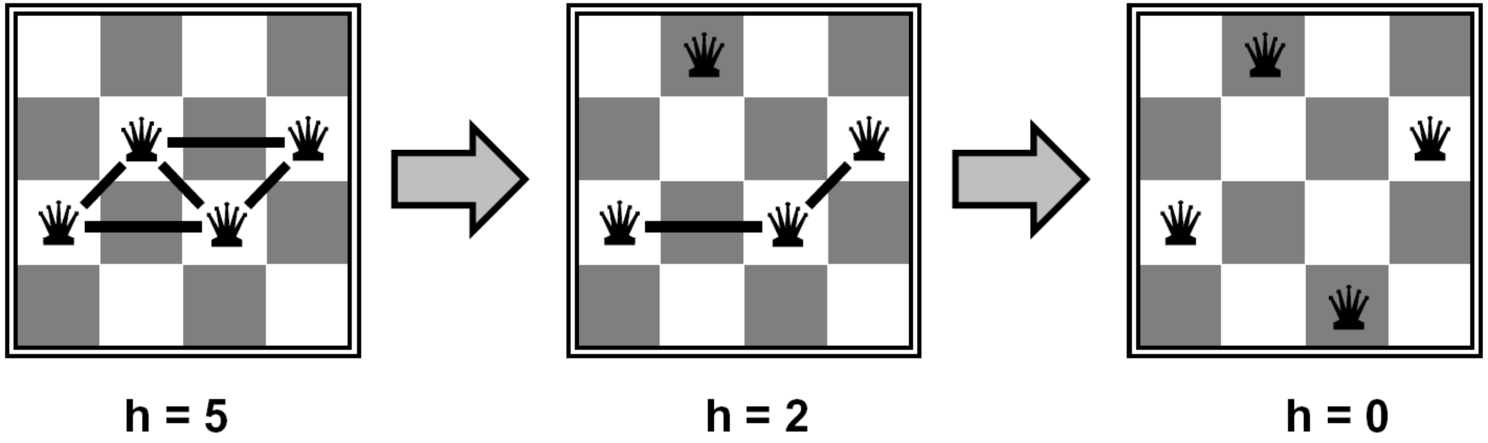
- Local search methods typically work with “complete” states, i.e., all variables assigned

- To apply to CSPs:
  - Take an assignment with unsatisfied constraints
  - Operators *reassign* variable values
  - No fringe! Live on the edge.




- Algorithm: While not solved,
  - Variable selection: randomly select any conflicted variable
  - Value selection: min-conflicts heuristic:
    - Choose a value that violates the fewest constraints
    - I.e., hill climb with h(n) = total number of violated constraints

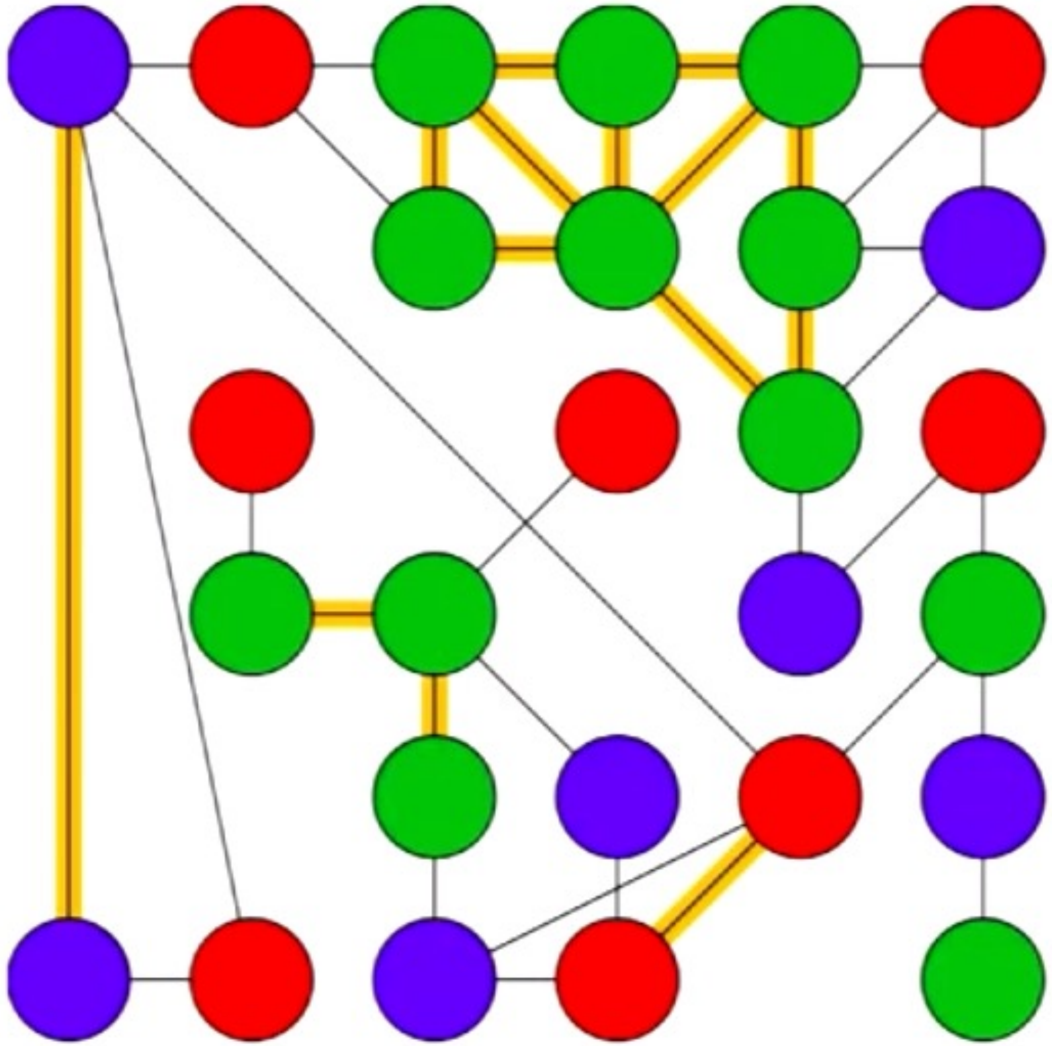
# Example: 4-Queens




- States: 4 queens in 4 columns ($4^4$ = 256 states)
- Operators: move queen in column
- Goal test: no attacks
- Evaluation: c(n) = number of attacks

[Demo: coloring – iterative improvement]

# Iterative Improvement – Coloring

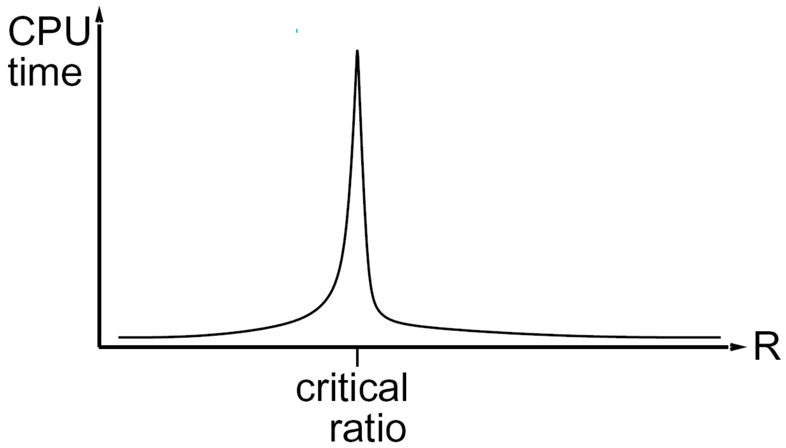


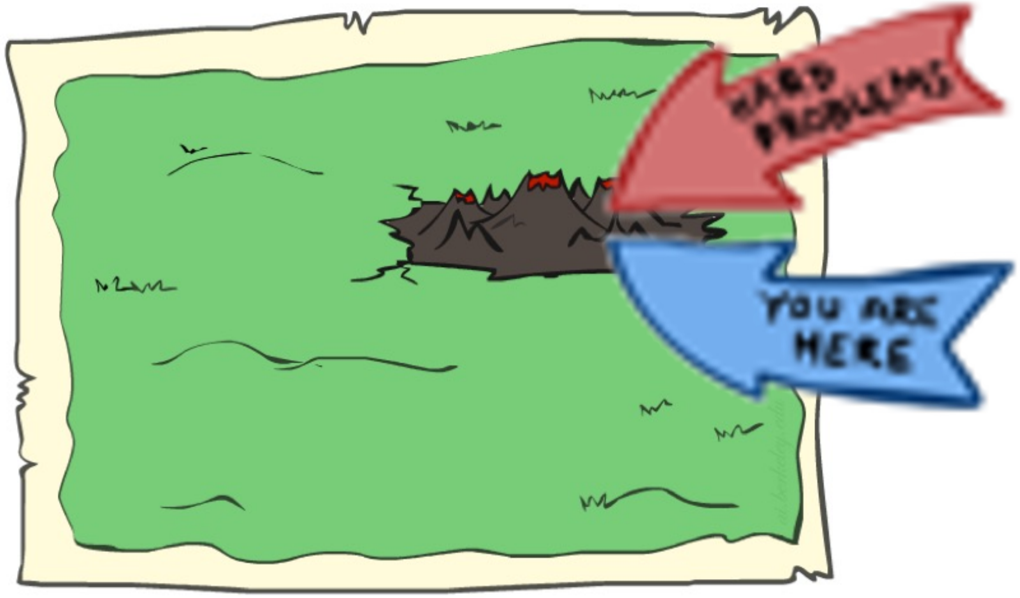
# Performance of Min-Conflicts

- Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)!

- The same appears to be true for any randomly-generated CSP *except* in a narrow range of the ratio

$$R = \frac{\text{number of constraints}}{\text{number of variables}}$$



# Summary: CSPs

- CSPs are a special kind of search problem:
  - States are partial assignments
  - Goal test defined by constraints
- Basic solution: backtracking search
- Speed-ups:
  - Ordering
  - Filtering
  - Structure
- Iterative min-conflicts is often effective in practice

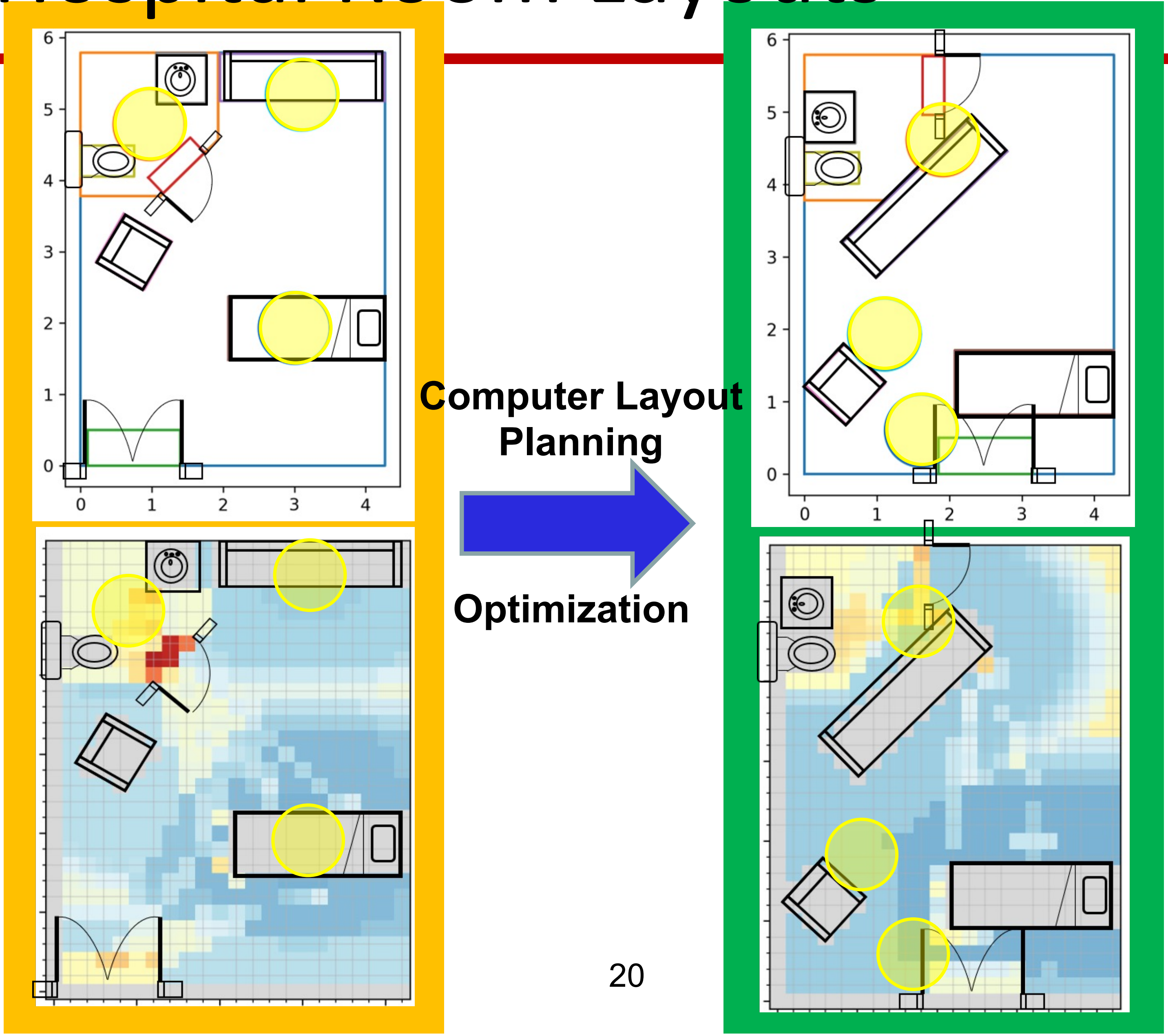
# Example from my Research

**Optimizing Hospital Room Layout to Reduce the Risk of Patient Falls**

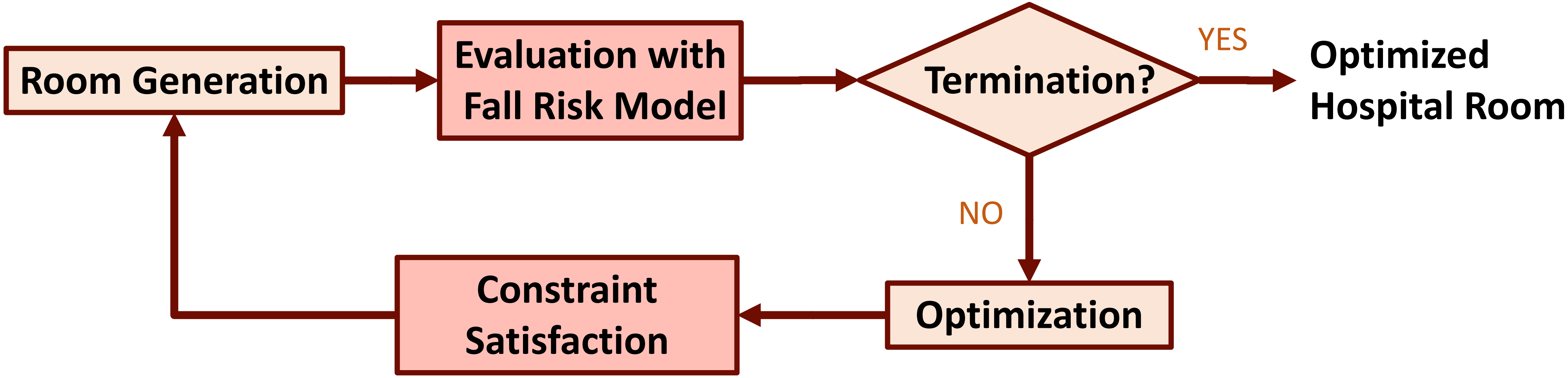
Sarvenaz Chaeibakhsh[1], Roya Sabbagh Novin[1], Tucker Hermans[2],
Andrew Merryweather[1], and Alan Kuntz[2]
[1] *Department of Mechanical Engineering, University of Utah, UT, USA*
[2] *School of Computing, University of Utah, UT, USA*
*sarvenaz.chaeibakhsh@utah.edu*

# Optimizing Hospital Room Layouts

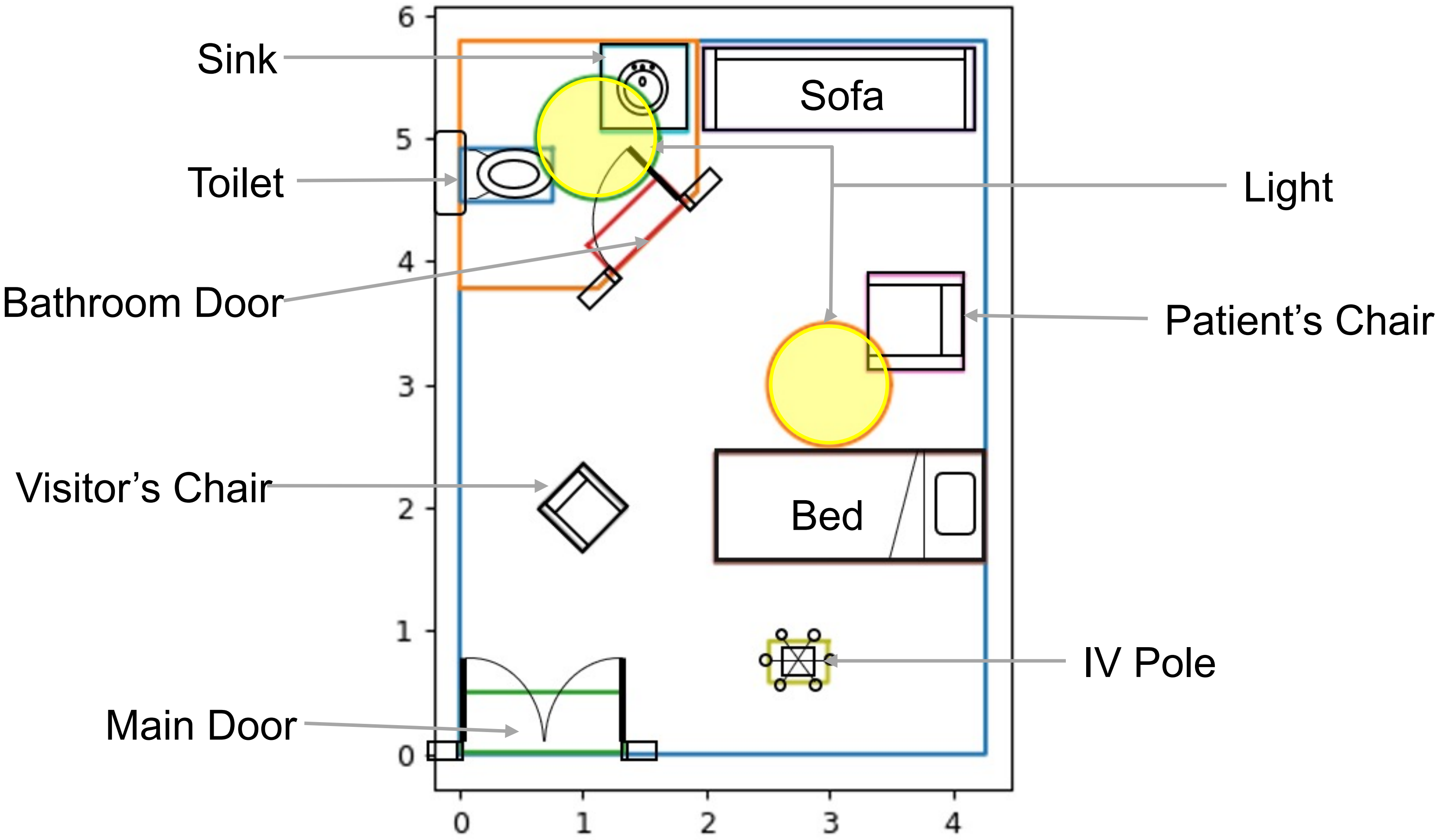
- Reconfiguring hospital room object placement layout to reduce the risk of fall.
  - Use of **computer layout planning** and **optimization** models to provide a safer room for patient's ambulation



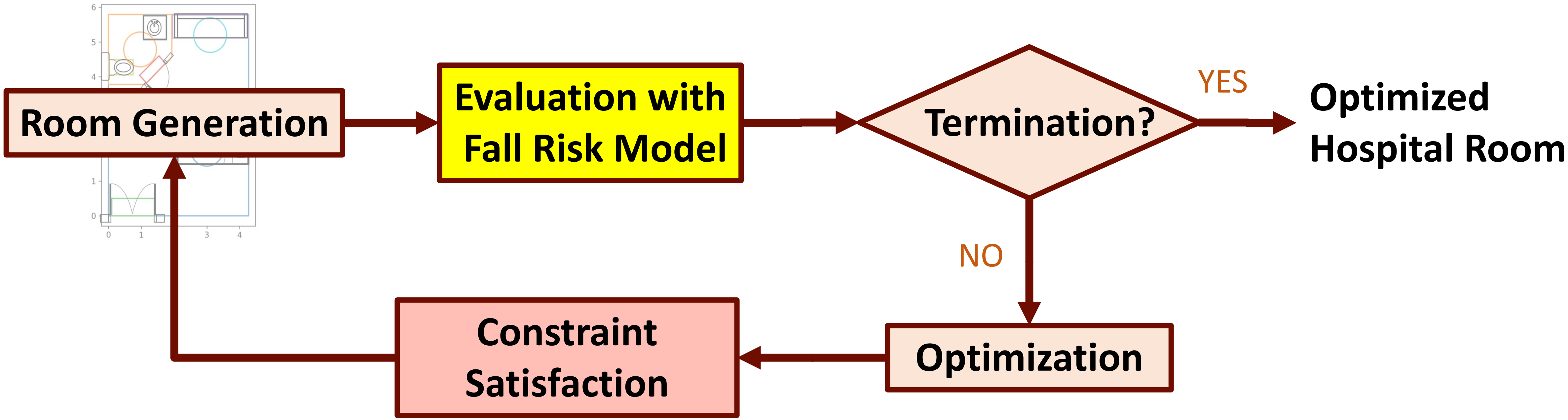
# General Road Map



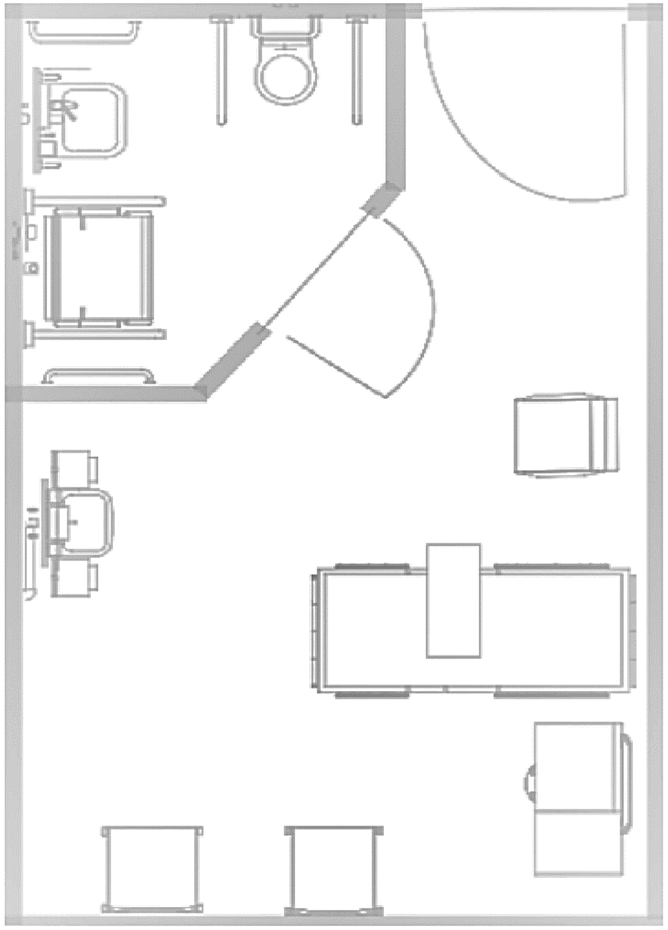
# Initial Room

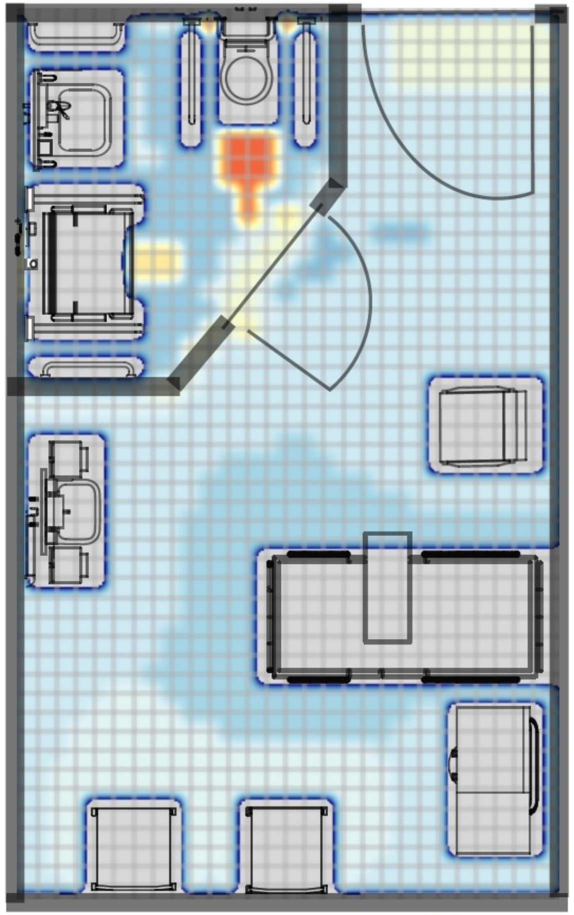


# General Road Map

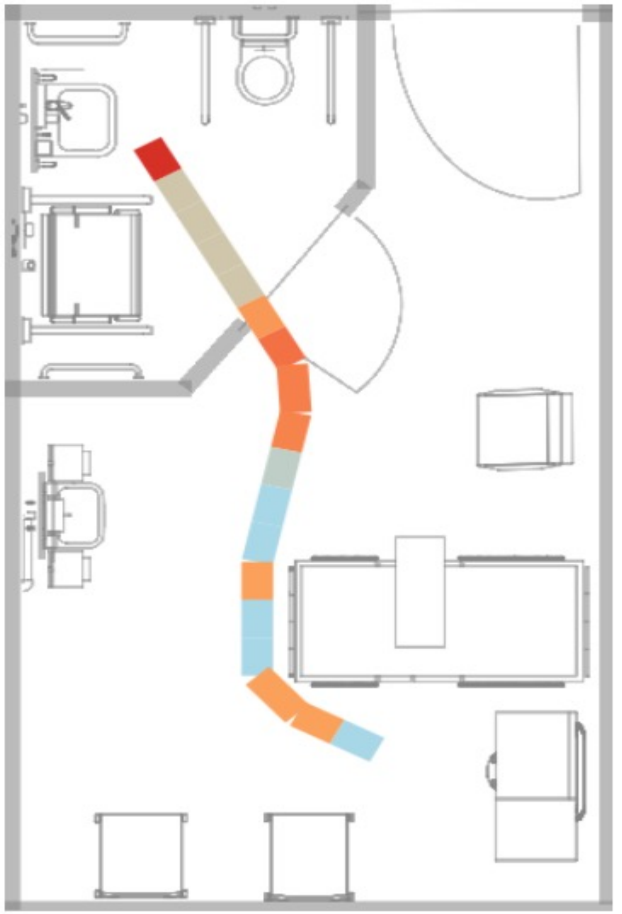


# Fall Model (By: Roya Sabbaghnovin)



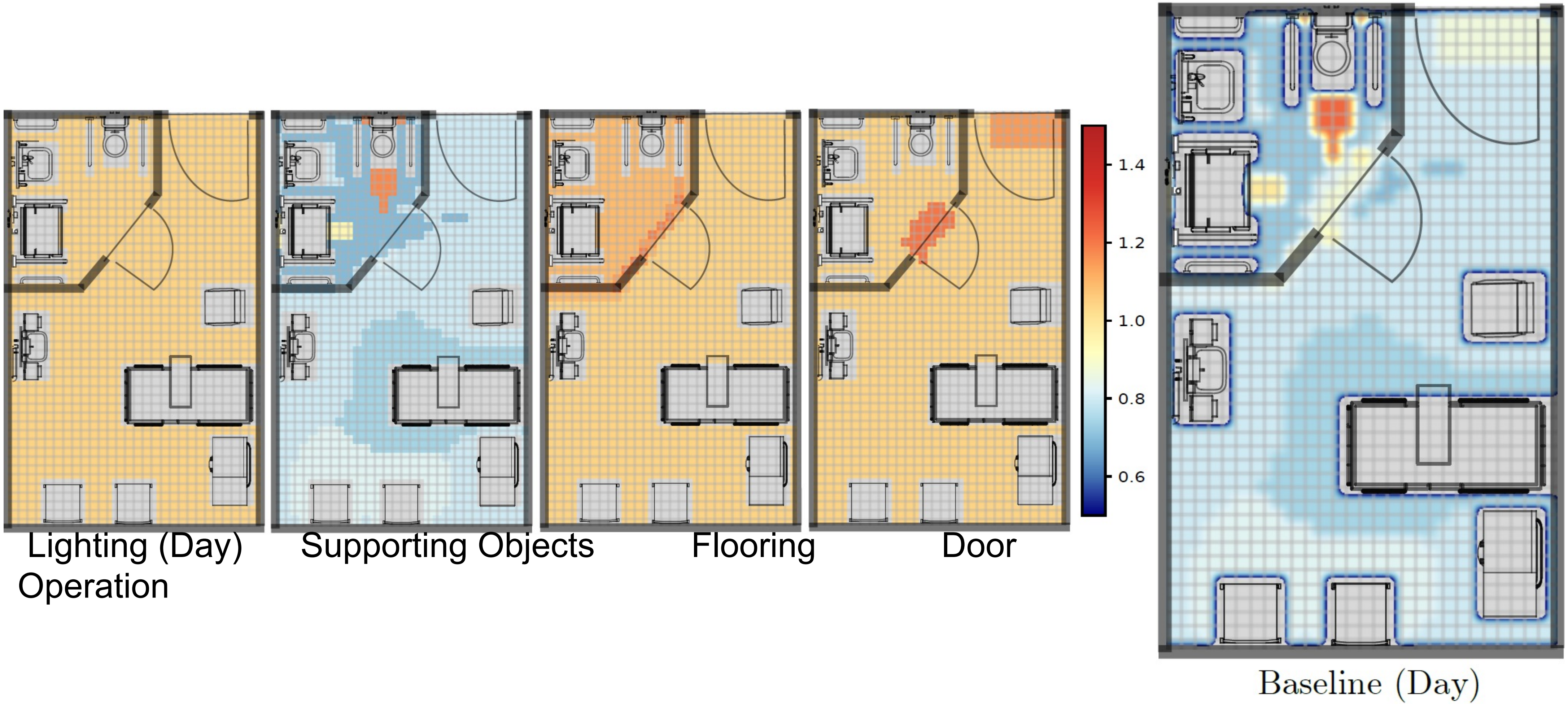


Baseline (Day)



Figures from: R. S. Novin, E. Taylor, T. Hermans, A. Merryweather, Development of a novel computational model for evaluating fall risk in patient room design., HERD: Health Environments Research & Design Journal (2020)

# Fall Model: Baseline



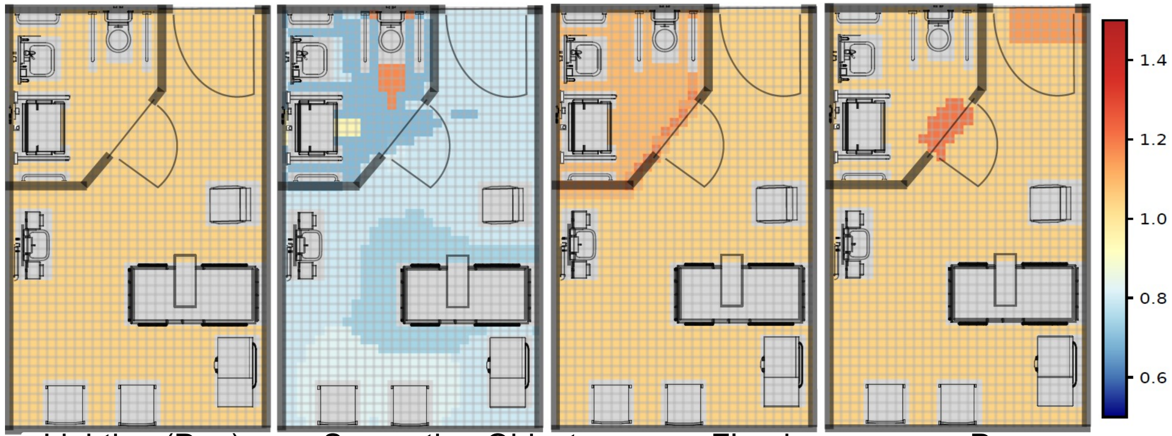

Figures from: R. S. Novin, E. Taylor, T. Hermans, A. Merryweather, Development of a novel computational model for evaluating fall risk in patient room design., HERD: Health Environments Research & Design Journal (2020)

# Fall Model: Trajectory


Sampled Trajectories

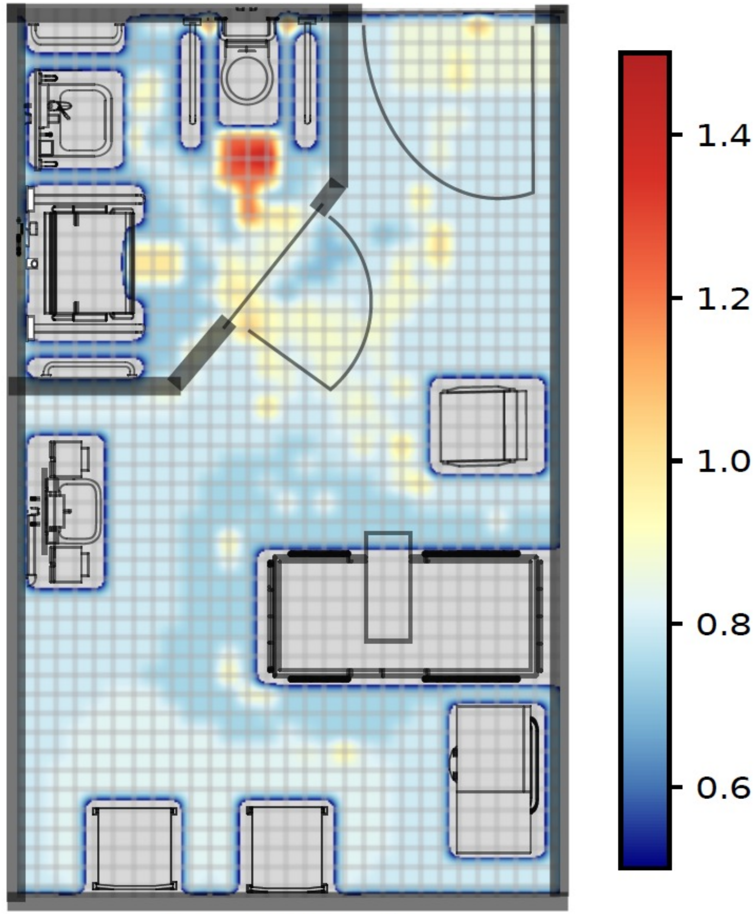
Figures from: R. S. Novin, E. Taylor, T. Hermans, A. Merryweather, Development of a novel computational model for evaluating fall risk in patient room design., HERD: Health Environments Research & Design Journal (2020)

# Fall Model




Lighting (Day) Operation | Supporting Objects | Flooring | Door

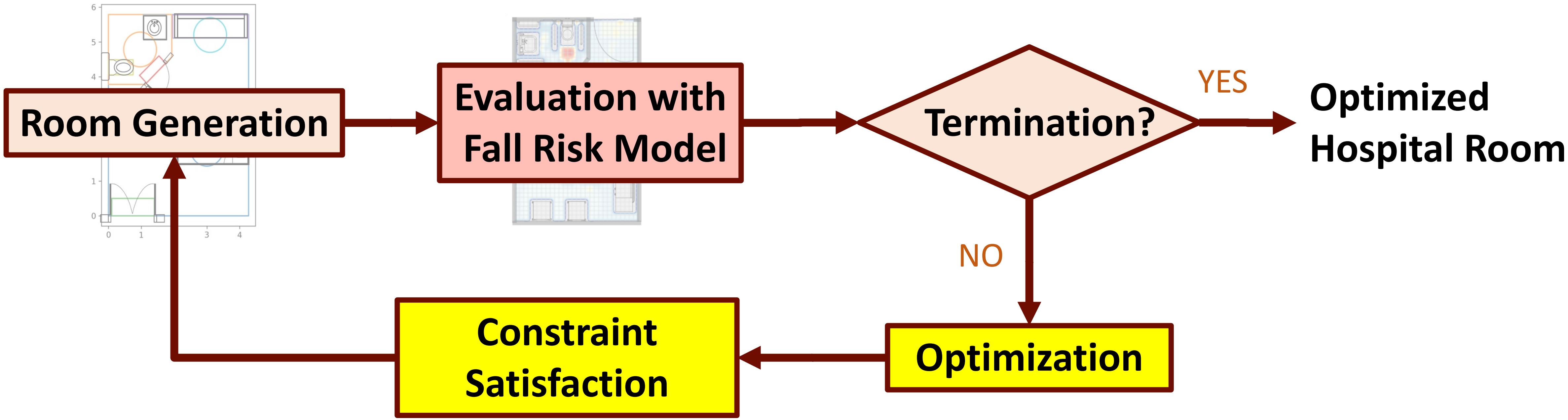

Sampled Trajectories




Final (Day)

Figures from: R. S. Novin, E. Taylor, T. Hermans, A. Merryweather, Development of a novel computational model for evaluating fall risk in patient room design., HERD: Health Environments Research & Design Journal (2020)

# General Road Map



# Constraint Satisfaction

- Set of variables: $X$
- Domain: $D$
- Set of Constraints: $C$

# Constraint Satisfaction

- Set of variables: $X = \{F_0, \dots, F_n,\ L_0, \dots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
- Set of Constraints: $C$

# Constraint Satisfaction

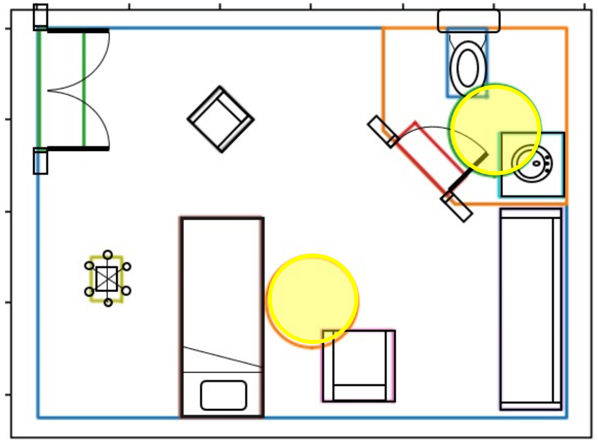
- Set of variables: $X = \{F_0, \ldots, F_n,\ L_0, \ldots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
  - $\overline{conf_{X_i}} = \{x, y, \theta\}$
  - $D_{X_i} = \left[\min\left(x_{room_{X_i}}\right), \max\left(x_{room_{X_i}}\right)\right], \left[min\left(y_{room_{X_i}}\right), \max\left(y_{room_{X_i}}\right)\right], [0,2\pi)\}$
- Set of Constraints: $C$

# Constraint Satisfaction

- Set of variables: $X = \{F_0, \dots, F_n,\ L_0, \dots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
  - $\overline{conf_{X_i}} = \{x, y, \theta\}$
  - $D_{X_i} = \left[\min\left(x_{room_{X_i}}\right), \max\left(x_{room_{X_i}}\right)\right], \left[min\left(y_{room_{X_i}}\right), \max\left(y_{room_{X_i}}\right)\right], [0,2\pi]\}$
- Set of Constraints: $C$
  - Inside the room
  - Collision avoidance
  - Standing against a wall
  - Clearance
  - Door placement

# Constraint Satisfaction

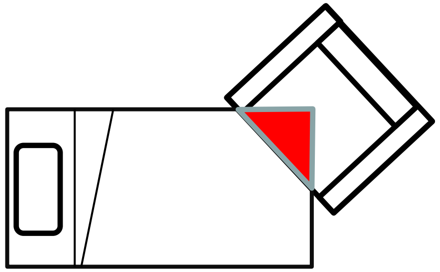
- Set of variables: $X = \{F_0, \dots, F_n,\ L_0, \dots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
  - $\overline{conf_{X_i}} = \{x, y, \theta\}$
  - $D_{X_i} = \left[\min\left(x_{room_{X_i}}\right), \max\left(x_{room_{X_i}}\right)\right], \left[\min\left(y_{room_{X_i}}\right), \max\left(y_{room_{X_i}}\right)\right], [0{,}2\pi]\}$
- Set of Constraints: $C$
  - Inside the room
  - Collision avoidance
  - Standing against a wall
  - Clearance
  - Door placement



# Constraint Satisfaction

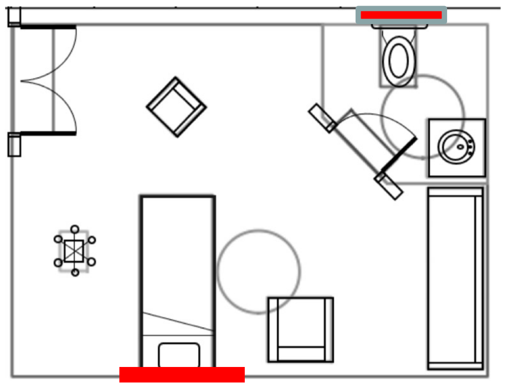
**Constraint Satisfaction Problem**

- Set of variables: $X = \{F_0, \dots, F_n,\ L_0, \dots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
  - $\overline{conf_{X_i}} = \{x, y, \theta\}$
  - $D_{X_i} = \left[\min\left(x_{room_{X_i}}\right), \max\left(x_{room_{X_i}}\right)\right], \left[min\left(y_{room_{X_i}}\right), \max\left(y_{room_{X_i}}\right)\right], [0{,}2\pi]\}$
- Set of Constraints: $C$
  - Inside the room
  - Collision avoidance
  - Standing against a wall
  - Clearance
  - Door placement



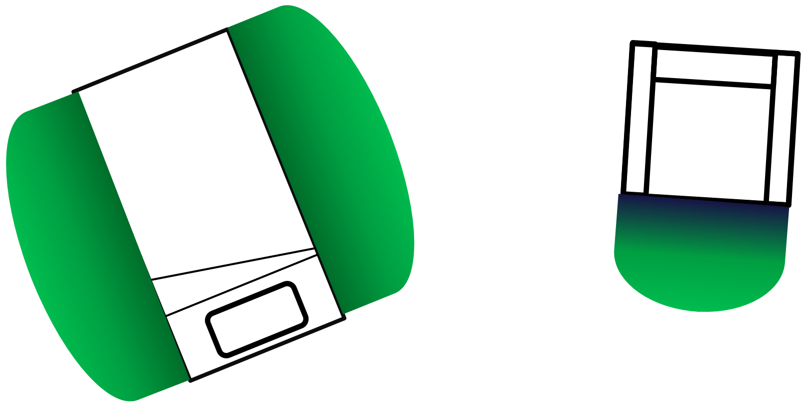
# Constraint Satisfaction

- Set of variables: $X = \{F_0, \dots, F_n,\ L_0, \dots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
  - $\overline{conf_{X_i}} = \{x, y, \theta\}$
  - $D_{X_i} = \left[\min\left(x_{room_{X_i}}\right), \max\left(x_{room_{X_i}}\right)\right], \left[\min\left(y_{room_{X_i}}\right), \max\left(y_{room_{X_i}}\right)\right], [0,2\pi]\}$
- Set of Constraints: $C$
  - Inside the room
  - Collision avoidance
  - Standing against a wall
  - Clearance
  - Door placement



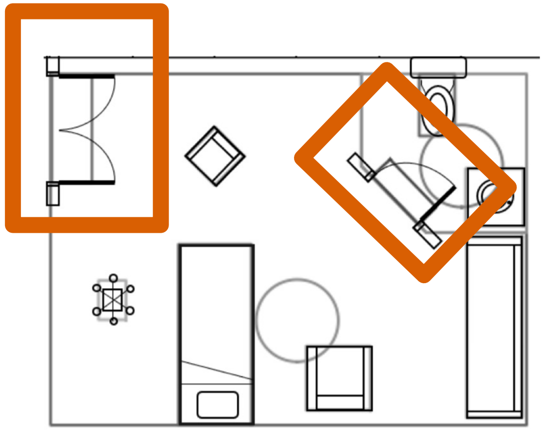
# Constraint Satisfaction

- Set of variables: $X = \{F_0, \dots, F_n,\ L_0, \dots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
  - $\overline{conf_{X_i}} = \{x, y, \theta\}$
  - $D_{X_i} = \left[\min\left(x_{room_{X_i}}\right), \max\left(x_{room_{X_i}}\right)\right], \left[\min\left(y_{room_{X_i}}\right), \max\left(y_{room_{X_i}}\right)\right], [0{,}2\pi]\}$
- Set of Constraints: $C$
  - Inside the room
  - Collision avoidance
  - Standing against a wall
  - Clearance
  - Door placement



# Constraint Satisfaction

- Set of variables: $X = \{F_0, \dots, F_n,\ L_0, \dots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
  - $\overline{conf_{X_i}} = \{x, y, \theta\}$
  - $D_{X_i} = \left[\min\left(x_{room_{X_i}}\right), \max\left(x_{room_{X_i}}\right)\right], \left[\min\left(y_{room_{X_i}}\right), \max\left(y_{room_{X_i}}\right)\right], [0,2\pi]\}$
- Set of Constraints: $C$
  - Inside the room
  - Collision avoidance
  - Standing against a wall
  - Clearance
  - Door placement



# Constraint Satisfaction

## Constraint Satisfaction Problem

- Set of variables: $X = \{F_0, \dots, F_n,\ L_0, \dots, L_m, D_{main}, D_{bath}\}$
- Domain: $D$
  - $\overline{conf_{X_i}} = \{x, y, \theta\}$
  - $D_{X_i} = \left[\min\left(x_{room_{X_i}}\right), \max\left(x_{room_{X_i}}\right)\right], \left[min\left(y_{room_{X_i}}\right), \max\left(y_{room_{X_i}}\right)\right], [0,2\pi]\}$
- Set of Constraints: $C$
  - Inside the room
  - Collision avoidance
  - Standing against a wall
  - Clearance
  - Door placement

**What if the set of constraints cannot be satisfied?** → Backtracking