# Finetuning Randomized Heuristic Search For 2D Path Planning: Finding The Best Input Parameters For R* Algorithm Through Series Of Experiments

**Konstantin Yakovlev, Egor Baskin, Ivan Hramoin**

Lab "Dynamic Intelligent Systems"
Institute for Systems Analysis of
Russian Academy of Sciences

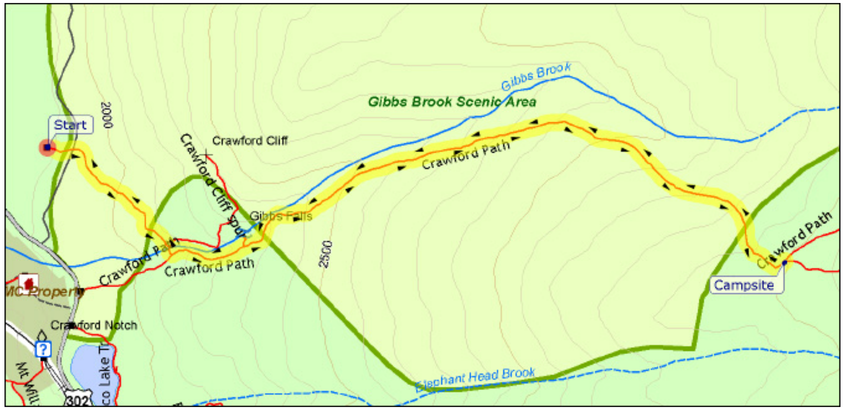
yakovlev@isa.ru

AIMSA'2014

# My name is Konstantin Yakovlev

- PhD in Theoretical Computer Sciences (2011)
- Live and work in Moscow, Russia
- Institute for Systems Analysis of Russian Academy of Sciences
- Research interests: Artificial Intelligence, Heuristic Search, Path Planning, Robotics
- Current research focus: computationally effective control system for small-scale unmanned aerial vehicles

# 2D path planning



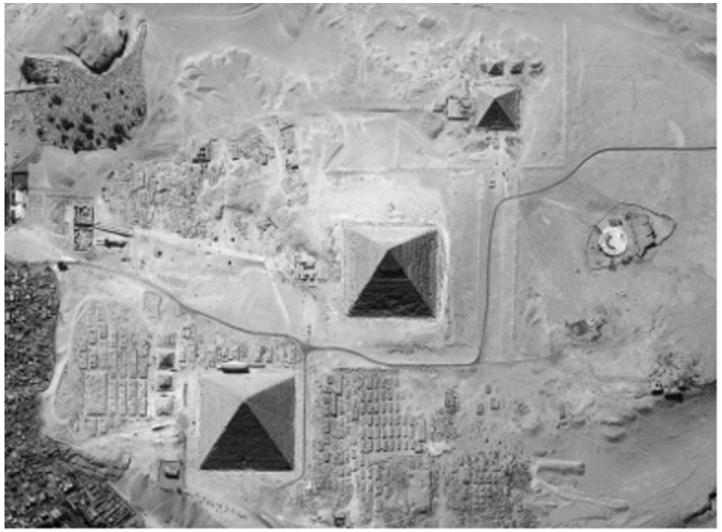

**(1)** Goal: autonomous intelligent agents (AI agents) – pathfinding is a must

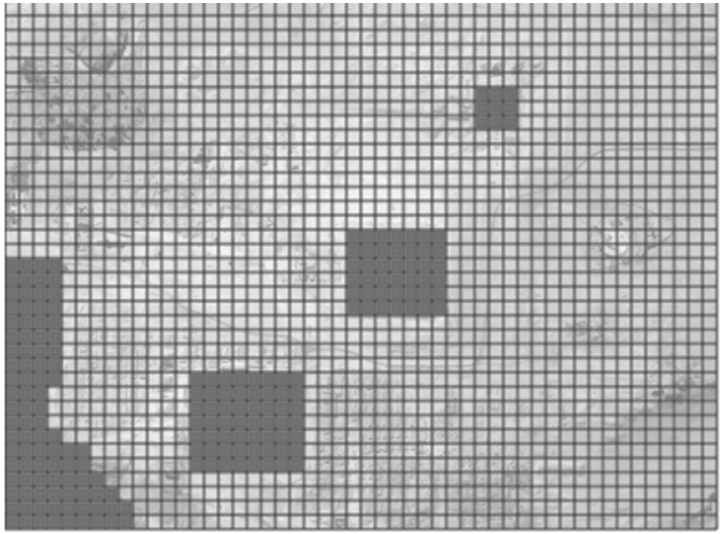
**(2)** Problem: existing methods *can not scale well to large problems* and can not solve problems (even "trivial" ones) *under tough resource constraints*

**(3)** Solution: modification of existing methods and development of new methods and algorithms

# 2D path planning as searching for a path o grid

## **Grid** – simple yet powerful 2D terrain model





**Elfes, A. 1989**. Using occupancy grids for mobile robot perception and navigation. Computer, 22(6), 46-57.

**Yap, P. 2002**. Grid-based path-finding. In Proceedings of 15th Conference of the Canadian Society for Computational Studies of Intelligence, 44-55. Springer Berlin Heidelberg.

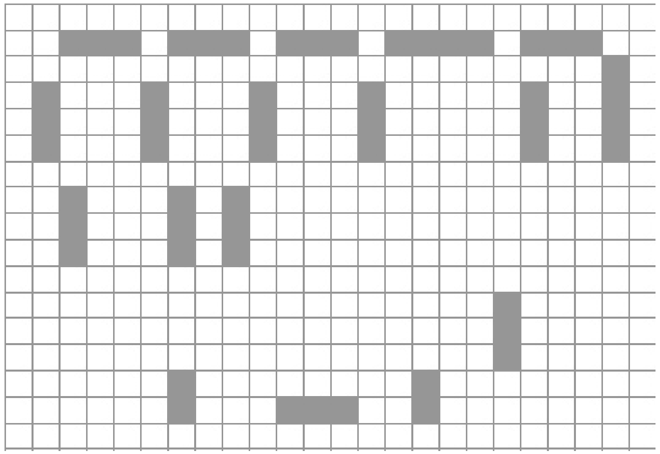
**Tozour, P. 2004**. Search space representations. In Rabin, S. (Ed.), AI Game Programming Wisdom 2, 85–102. Charles River Media.

# Grid – formal definitions

**GR=<*A, dist, c*>**

*A* – **set of cells** which can be represented in matrix-form:

$A_{m\times n}=\{a_{ij}\}$: $a_{ij}=0\vee a_{ij}=1$, $i, j$: $0\leq i<m$, $0\leq j<n$, $m, n \in \mathbf{N}\setminus\{0\}$.



***dist*** – **metrics** on the set of $A^{+}=\{a_{ij}|a_{ij}\in A, a_{ij}=0\}$

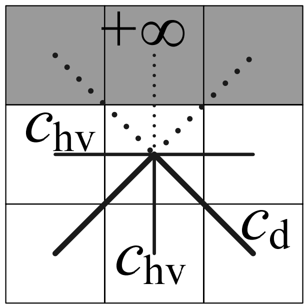
$d(a_{ij}, a_{lk})=\{\Delta_i(a_{ij}, a_{lk})=|i-l|, \Delta_j(a_{ij}, a_{lk})=|j-k|\}=$

$$\begin{cases} c_{hv}\Delta_j + c_d(\Delta_i - \Delta_j), & if\ \Delta_i \geq \Delta_j \\ c_{hv}\Delta_i + c_{hv}(\Delta_j - \Delta_i), & if\ \Delta_i < \Delta_j \end{cases}$$

***c***: $ADJ \rightarrow \{c_{hv}, c_d, +\infty\}$ – **weight function**:

$ADJ\subset A\times A$ – set of all pairs of adjacent cells

$c_{hv}\in R^{+}$, $c_d=k\cdot c_{hv}$ $1\leq k<2$




$c(a_{ij}, a_{lk})=c_{hv}\vee c_d\vee +\infty$

$c_{hv}=10$ $c_d=14$

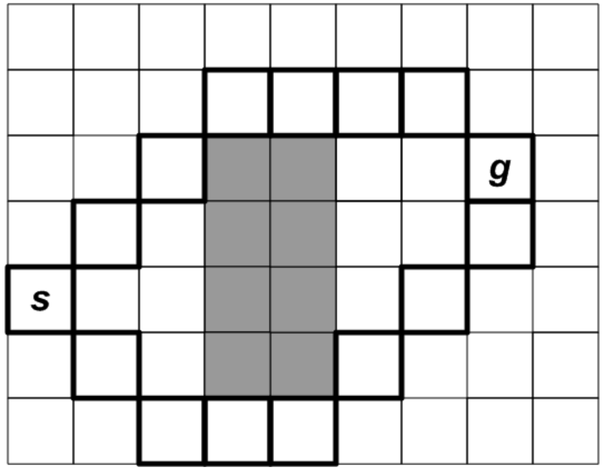
# Path planning task – formal definitions

**Path**

$$\pi = \pi(s,g) = \{a_{i_0 j_0}, a_{i_1 j_1}, \ldots, a_{i_{s-1} j_{s-1}}, a_{i_s j_s}\}$$

$$a_{i_0 j_0} = s, \quad a_{i_s j_s} = g, \quad a_{i_v j_v}, a_{i_{v+1} j_{v+1}} \in ADJ \ \forall v : 0 < v < s$$

$$a_{i_v j_v} = 0 \ \forall v : 0 \le v \le s$$




**Path weight (length)**

$$L(\pi) = \sum_{v=1}^{s} c(a_{i_{v-1} j_{v-1}}, a_{i_v j_v})$$

---

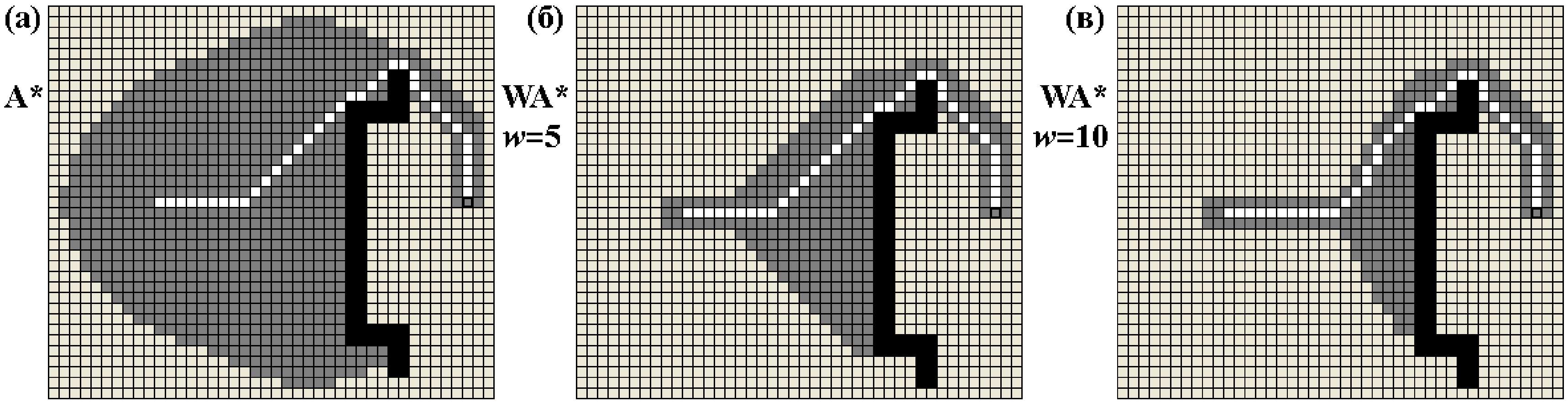
**Path planning task** **PTask**=$\langle Gr, s, g \rangle$

**Solution** $\boldsymbol{\pi}$ (path on Gr from s to g)

***Solution depth*** $\boldsymbol{D}$=$max\{|goalI–startI|, |goalJ–startJ|\}$

# Traditional approach – iterative algorithms (Dijkstra, A* etc)

- Well Studied ($O(D^2)$)
- Optimal solutions (**Dijkstra**, **A*** with metrics as heuristic function)
- Limited number of techniques of reducing the search effort (at the cost of finding suboptimal solutions), e.g. weighted heuristics (**WA***), iterative deepening (**IDA***), forced limitation of search space (**beam search**).

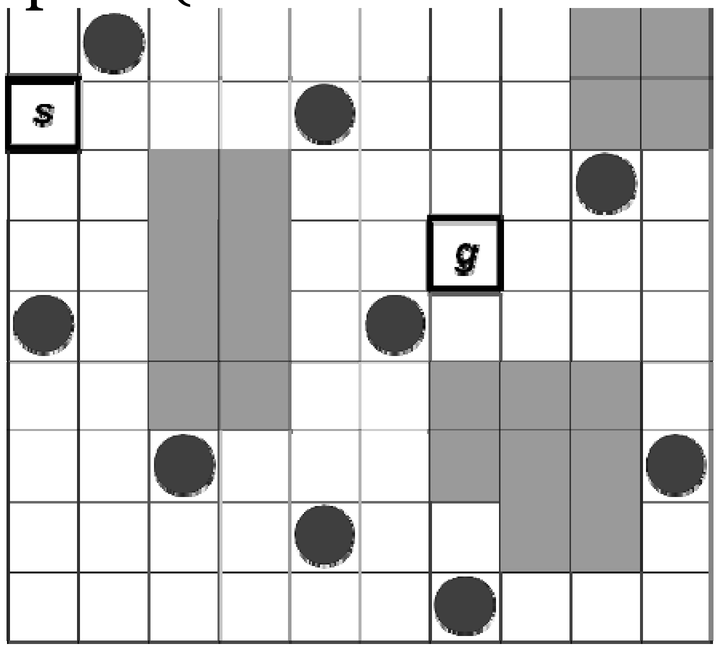



**Computationally ineffective**
**Can not scale well to large problems**

# Finding a path: decomposition approach

Idea:

1. Choose (somehow) a set of cells (*base cells*)
2. Find (somehow) paths between pairs of set cells (*sections*)
3. Compose final path (with the sections found)



# One of decomposition approach: Parameterized random choice

**R*** algorithm
[Likhachev, M., & Stentz, A. 2008, R* Search. In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence. Menlo Park, Calif.: AAAI press.]

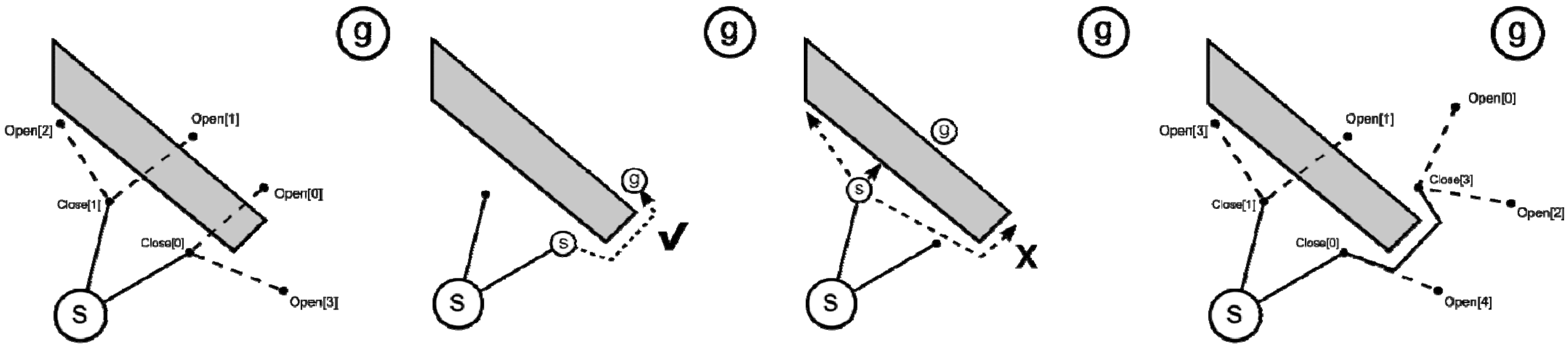
**Problems:**
(1) algorithm performance depends ***heavily*** on them
(2) initial values of algorithm parameters **are unknown**

# R* working principles

(1) choose the most promising cell *c* from OPEN list (initially containing only start cell)

(2) select $K$ traversable cells residing at the distance $\Delta$ from $c$: $SUCC(c)=\{b_1, \ldots, b_K: dist(b_i, c)=\Delta\}$ called successors of $c$ and inserts them into OPEN. If $d(g, c) \leq \Delta$ then goal cell is also added to OPEN.

(3) Try to find a local path $\pi(pred(c), c)$ with **WA*** algorithm. If the path is not found after the $m$ steps of **WA*** the cell $c$ is labeled AVOID and kept in OPEN. If the path is found the cell is removed from OPEN and is inserted into CLOSED list.



# Finetuning R*: Finding The Best Input Parameters

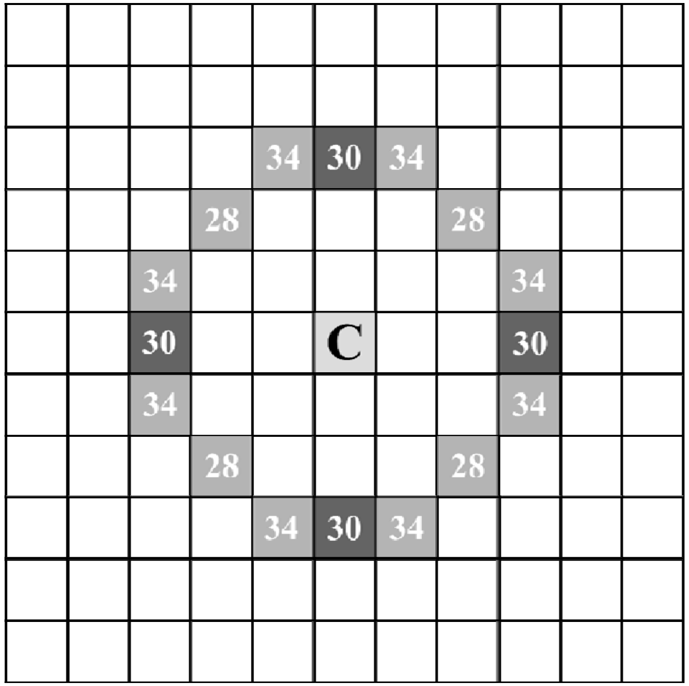
(1) Formulate a range of assumptions concerning possible upper and lower bounds of R* parameters, their interdependency and their influence on R* performance.

(2) Evaluate the assumptions by running a large number of experiments.

(3) Formulate set of heuristic rules which can be used to set the values of all R* parameters in a way that leads to algorithm's best performance when solving 2D path planning tasks.

# R* implementation note: generating successors

select ***K*** traversable cells residing at the distance **Δ** from *c*: $SUCC(c)=\{b_1, \ldots, b_K: dist(b_i, c)=\Delta\}$ called successors of *c* and inserts them into OPEN. If $d(g, c) \le \Delta$ then goal cell is also added to OPEN.



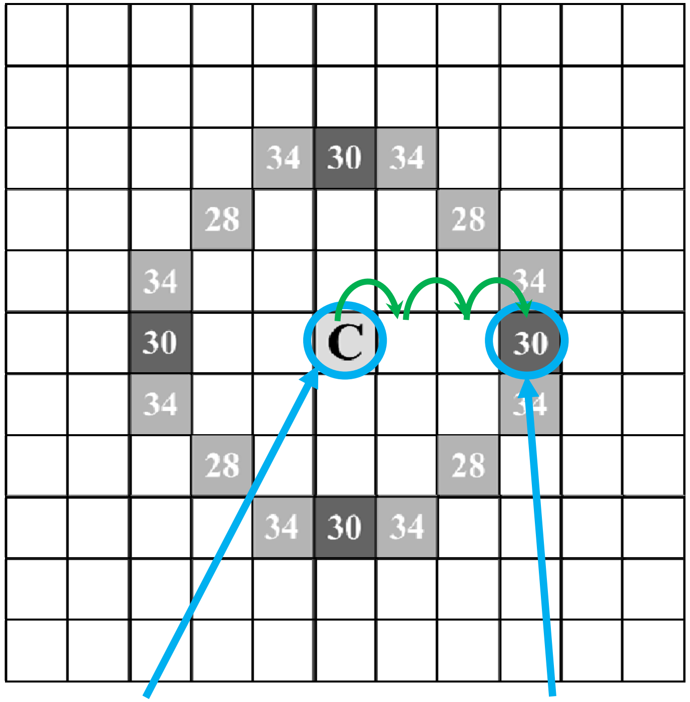

Midpoint circle algorithm is used to generate successors

**Pitteway, M. L. V. 1985.** Algorithms of conic generation. In Fundamental algorithms for computer graphics, 219-237. Springer Berlin Heidelberg.

$$\Delta = k \cdot c_{hv} = 10k$$

*k* – radius of circumference measured “in cells”

# R* Parameters Influence on Algorithm Performance: *m* – step threshold for the local planner




local start (***s***) local goal (***g***)

$m^{*}=\max(|i(s)-i(g)|, |j(s)-j(g)|)$

$\Delta = k \cdot c_{hv}$

$m_{min} = \Delta / c_{hv} = \Delta / 10$

$m_{max} = \alpha \cdot m_{min}$

α – koef saying how much harder the local pathfinding task than the trivial one is

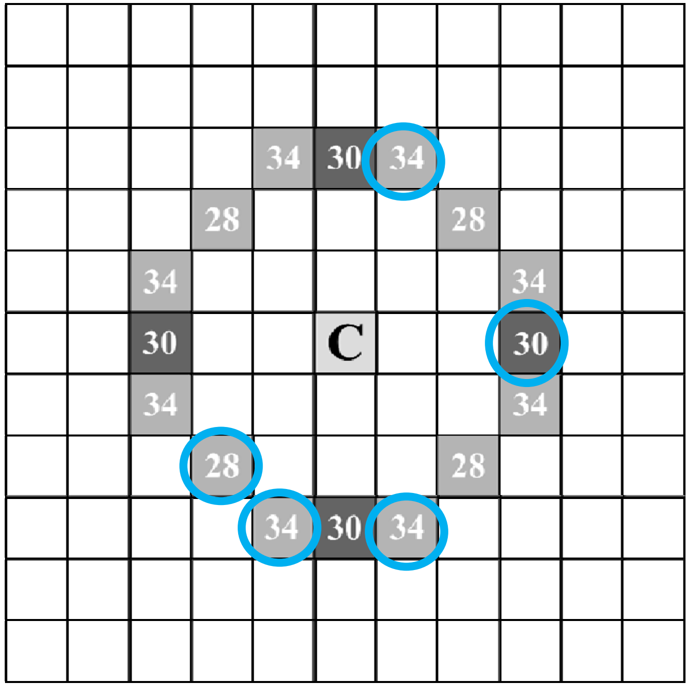
***processing time*** – influence is **high**, low values recommended

*memory consumption* – influence is low, low values recommended

*solution quality* – parameter's influence is unevident

**Assumption 1 - low values recommended**

# R* Parameters Influence on Algorithm Performance: *K* – Number of Successors for Each Expanded Cell




$K = 5$

$K_{max} = 2 \cdot \pi \cdot \Delta / c_{hv} \approx 6 \cdot \Delta / 10$
$K_{min} = 3$

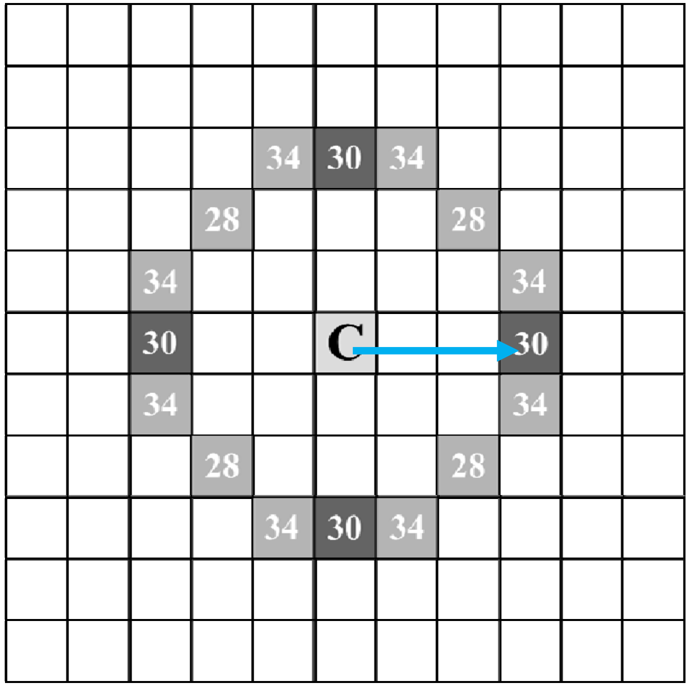
*processing time* – influence is low, low values recommended

***memory consumption*** – influence is high, low values recommended

***solution quality*** – influence is high, high values potentially lead to better solution

**Assumption 2 – mid values recommended**

# R* Parameters Influence on Algorithm Performance: Δ – Distance Between Expanding Cell and it's Successors




$\Delta = 30$

small Δ ≈WA* (bad)
big Δ ≈again WA* (bad)

$\Delta_{min} = c_{hv} = 10$
$\Delta_{max} = dist(s, g)$

***processing time*** – influence is high
***memory consumption*** – influence is high
***solution quality*** – influence is high

**Assumption 3 – mid values recommended**

# R* Parameters Influence on Algorithm Performance

***m*** **– Maximum Number of Steps for the Local Planner**
***processing time*** – parameter's influence is high, low values recommended
*memory consumption* – parameter's influence is low, low values recommended
*solution quality* – parameter's influence is unevident
Assumption 1 - **low values recommended**

***K*** **– Number of Successors for Each Expanded Cell**
*processing time* – parameter's influence is low, low values recommended
***memory consumption*** – degree of impact is high, low values recommended
***solution quality*** – degree of impact is high, high values potentially lead to better solution
Assumption 2 – **mid values recommended**

**Δ – Distance Between Expanding Cell and it's Successors**
**processing time** – parameter's influence is high, mid values recommended
**memory consumption** – parameter's influence is high, mid values recommended
**solution quality** – parameter's influence is high, mid values recommended
Assumption 3 – **mid values recommended**

# R* Parameters Lower and Upper Bounds

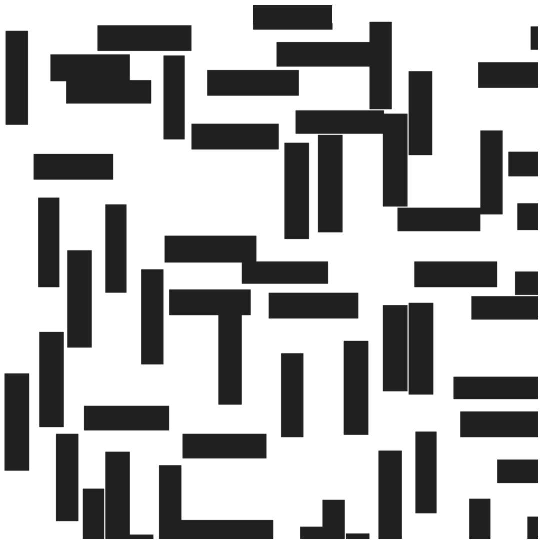
| Parameter | Theoretical lower/upper bounds | "Common sense" lower/upper bounds |
|---|---|---|
| $m$ | [Δ/10; T], T – number of grid cells | [Δ/10; 2Δ] |
| $K$ | [1; 6·Δ/10 ] | [3; Δ/5] |
| Δ | $[c_h, dist(s, g)]$ | $[dist(s, g)/k_1; dist(s, g)/k_2]$, $3<k_1<10$, $k_2\geq 100$ |

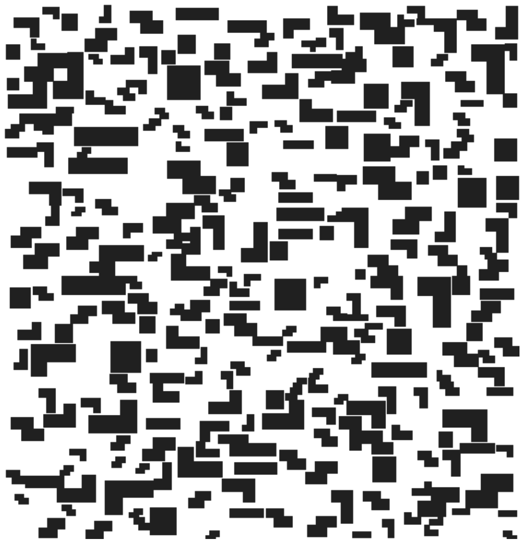
**(1)** ***m*** and **K** can be expressed as linear functions of **Δ** (**Δ** – is the key parameter)
**(2)** **Δ** can be expressed as linear function of start and goal locations.

# Experimental Analysis

**5250** of experiments on 3 types of grids:

- **randomly generated grids** containing **rectangle shaped** obstacles of different sizes (1750 experiments);
- **randomly generated grids** containing **tetris-shaped** obstacles of different sizes (1750 experiments);
- **grids** which model **city landscape** (1750 experiments).

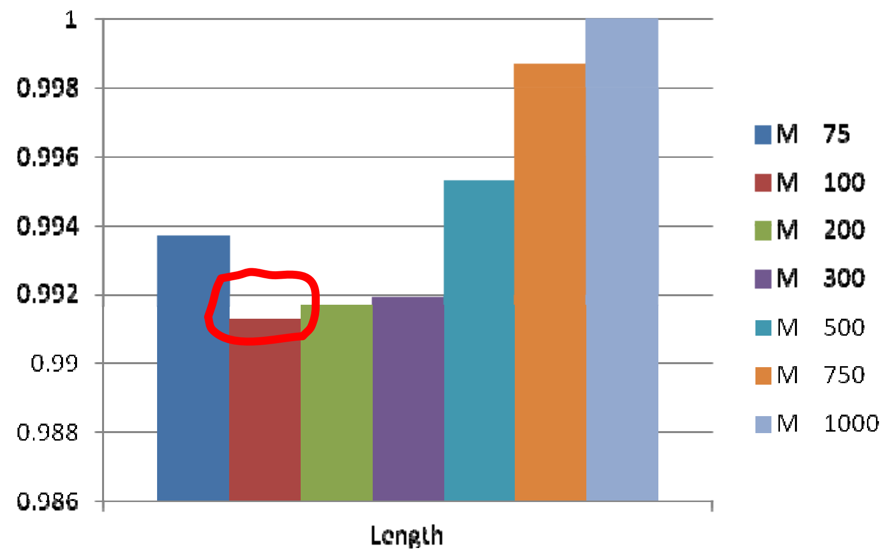


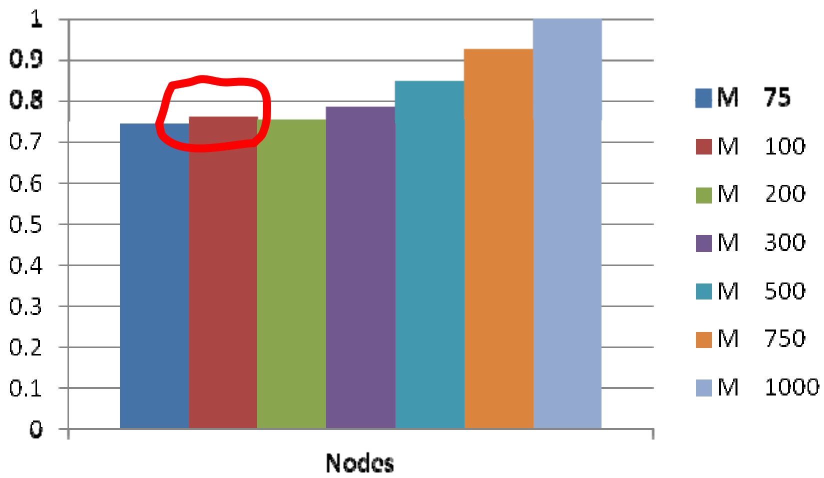


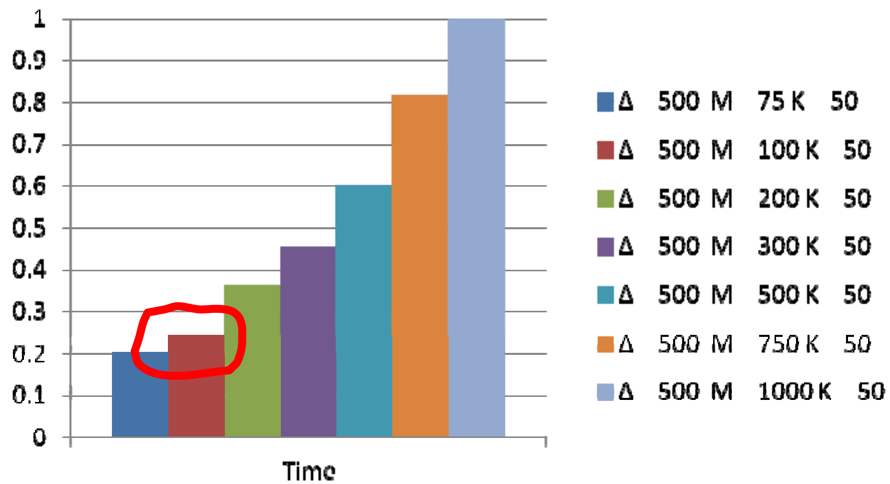
# *m* influence

**$m = [\Delta/10..2\Delta] = [75..\ 1000]$**
$K = \Delta/10 = 50$
$\Delta = \text{dist}(s, g)/10 = 500$

**$m = 100 = \Delta/5$**



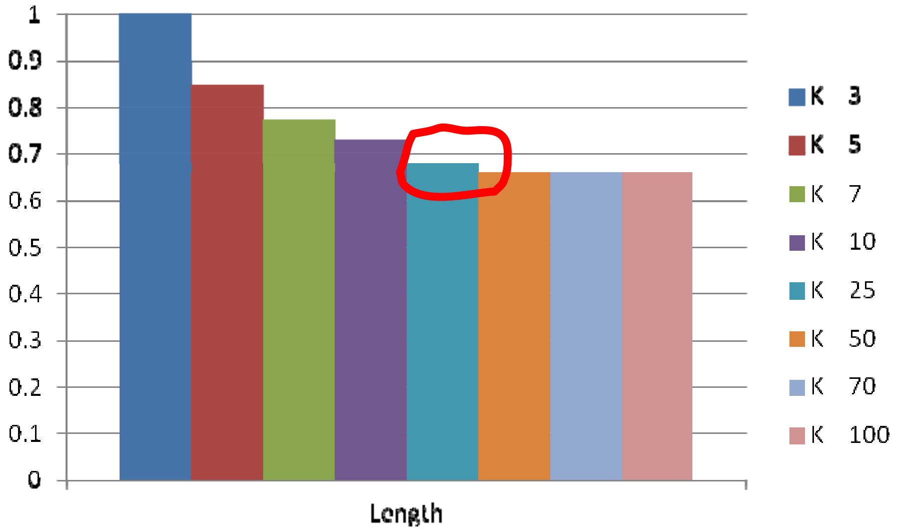



# *K* influence

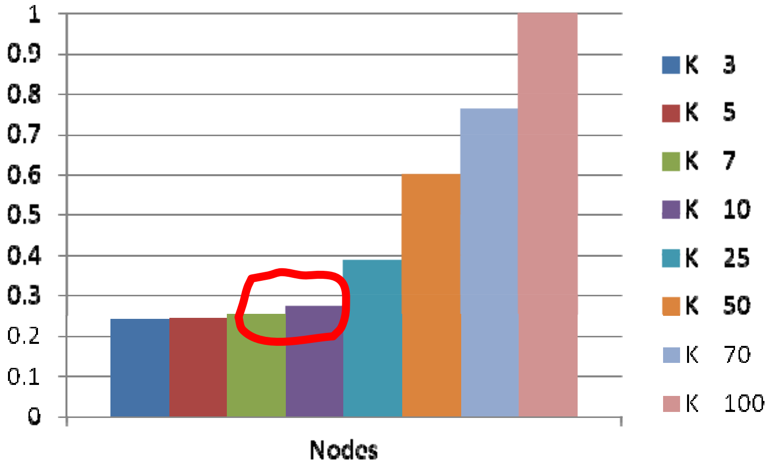
$m = \Delta/5 = 100$

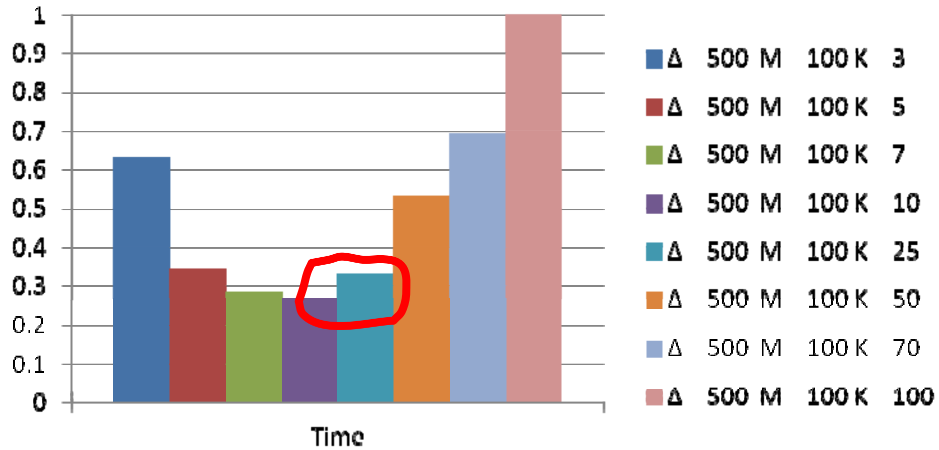
$\boldsymbol{K = [3..\Delta/5] = 3..100}$

$\Delta = \text{dist}(s, g)/10 = 500$



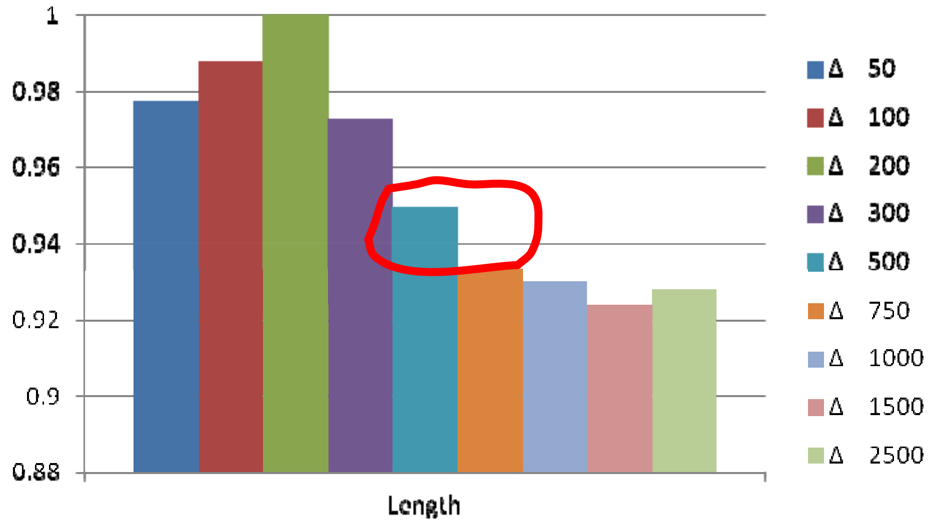

$\boldsymbol{K = 25 = \Delta/20}$



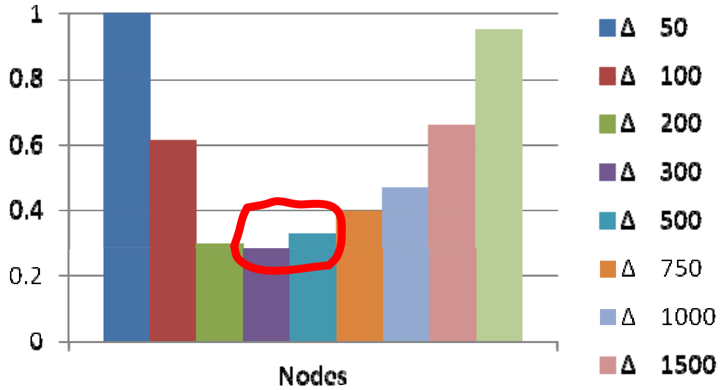



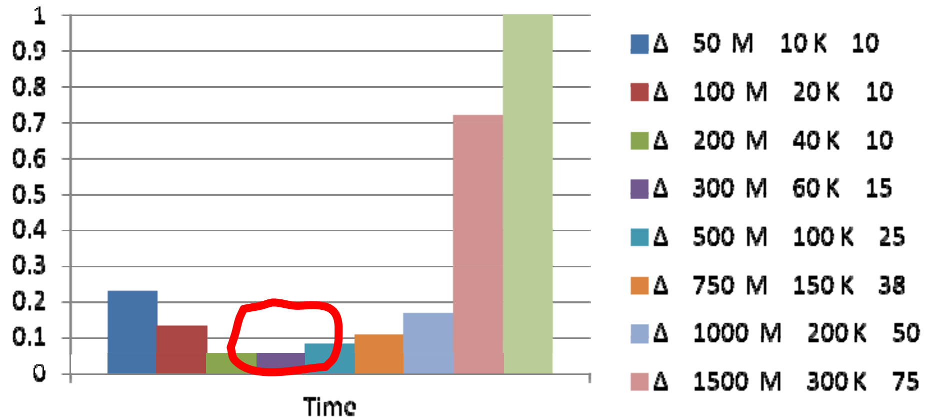
# Δ influence

*m* = Δ/5 = 100
*K* = Δ/20 = 25
**Δ = [dist(s, g)/100..dist(s, g)/2] = [50..2500]**




**Δ = 500 = dist(s, g)/10**

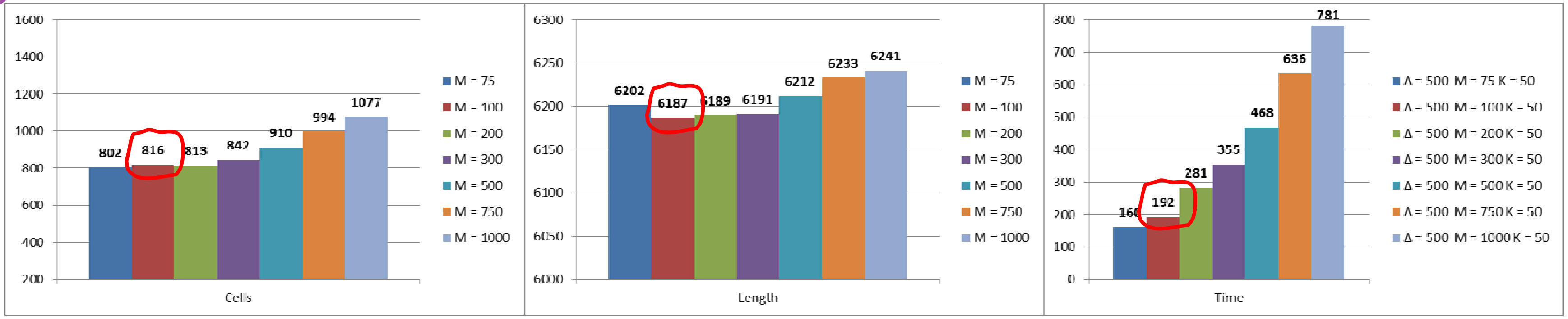


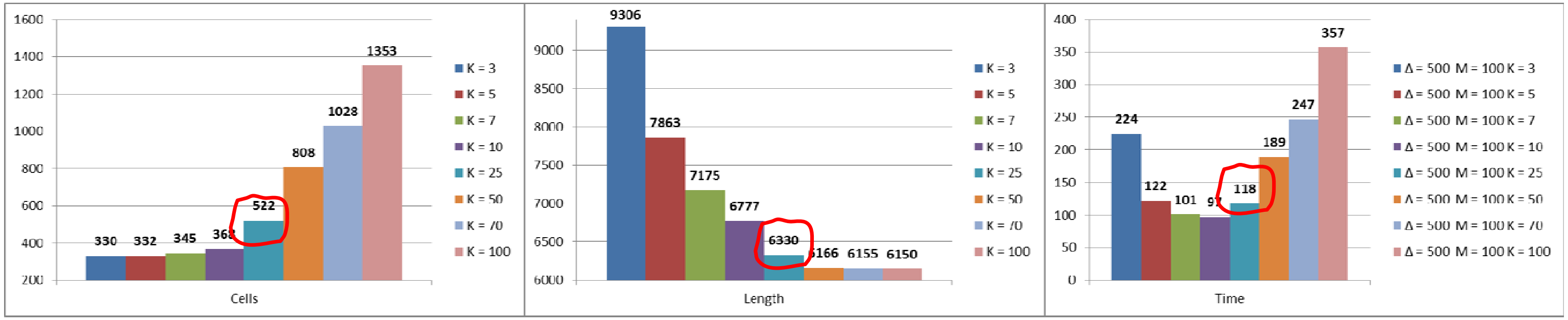
# Experimental Analysis



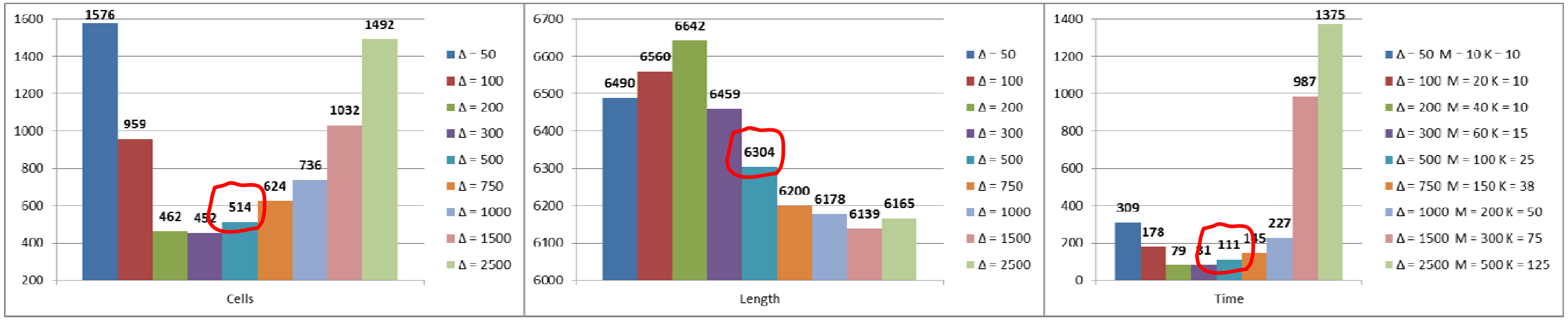
## 1) *m* influence




## 2) *K* influence




## 3) Δ influence



# Summary – set of heuristic rules for **automatic** parameterization of R*

1. R* performance depends ***heavily*** on the values of its parameters
2. Assigning them in a wrong way can lead to dramatic (more than **10 times**) fall in computational efficiency.
3. There exist a set of rules which can be used to **automatically initialize R*** input parameters in such a way that leads to best performance:

$$\Delta = dist(s, g)/10;$$
$$K = \max(10, \Delta/20);$$
$$m = \Delta/5.$$

4. Presented bindings are **dependent only of start and goal locations** which are known a priori.

# Questions?

**Konstantin Yakovlev**

Lab “Dynamic Intelligent Systems”
Institute for Systems Analysis of Russian Academy of Sciences

yakovlev@isa.ru