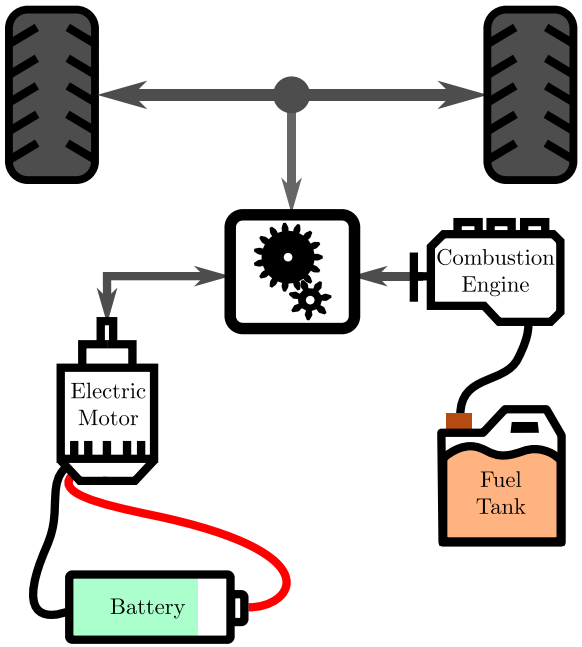
Scalable FPGA Implementation of Dynamic Programming for Optimal Control of Hybrid Electrical Vehicles

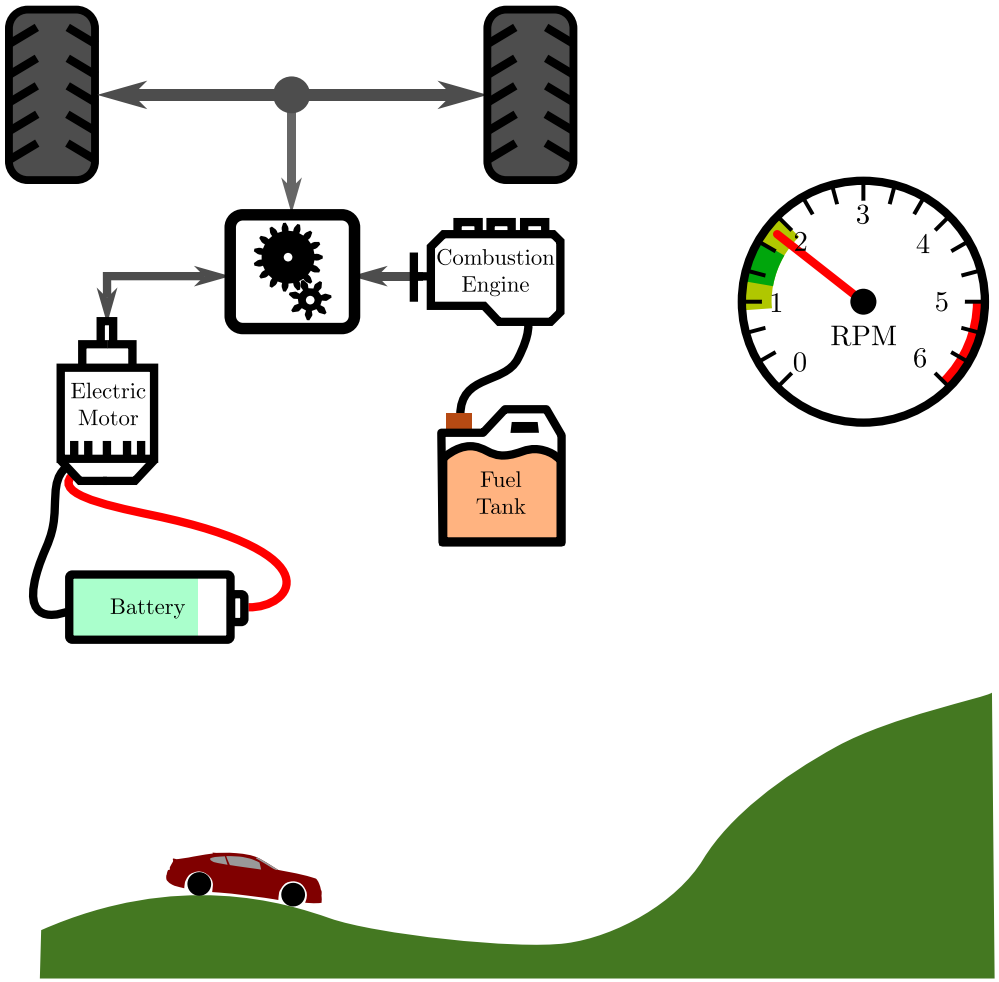
Frans Skarman, Oscar Gustafsson

Linköping University

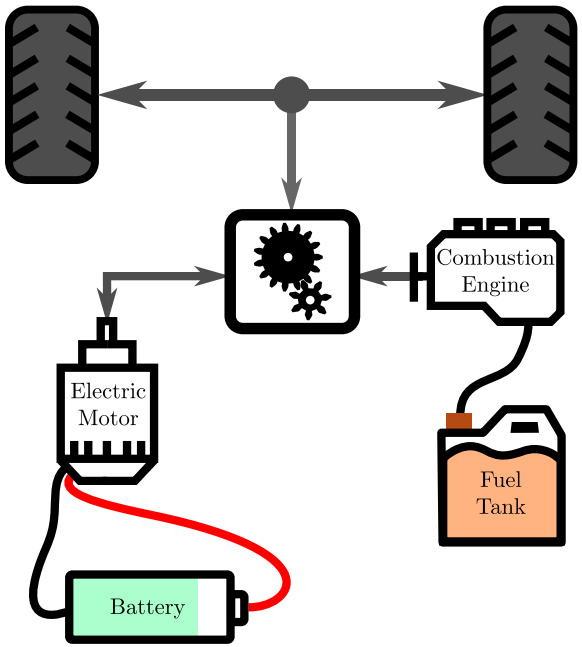
Determine optimal split between electric motor and combustion engine

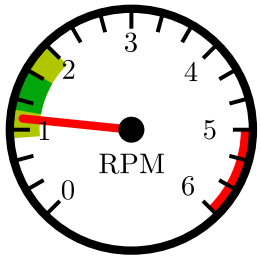


Determine optimal split between electric motor and combustion engine

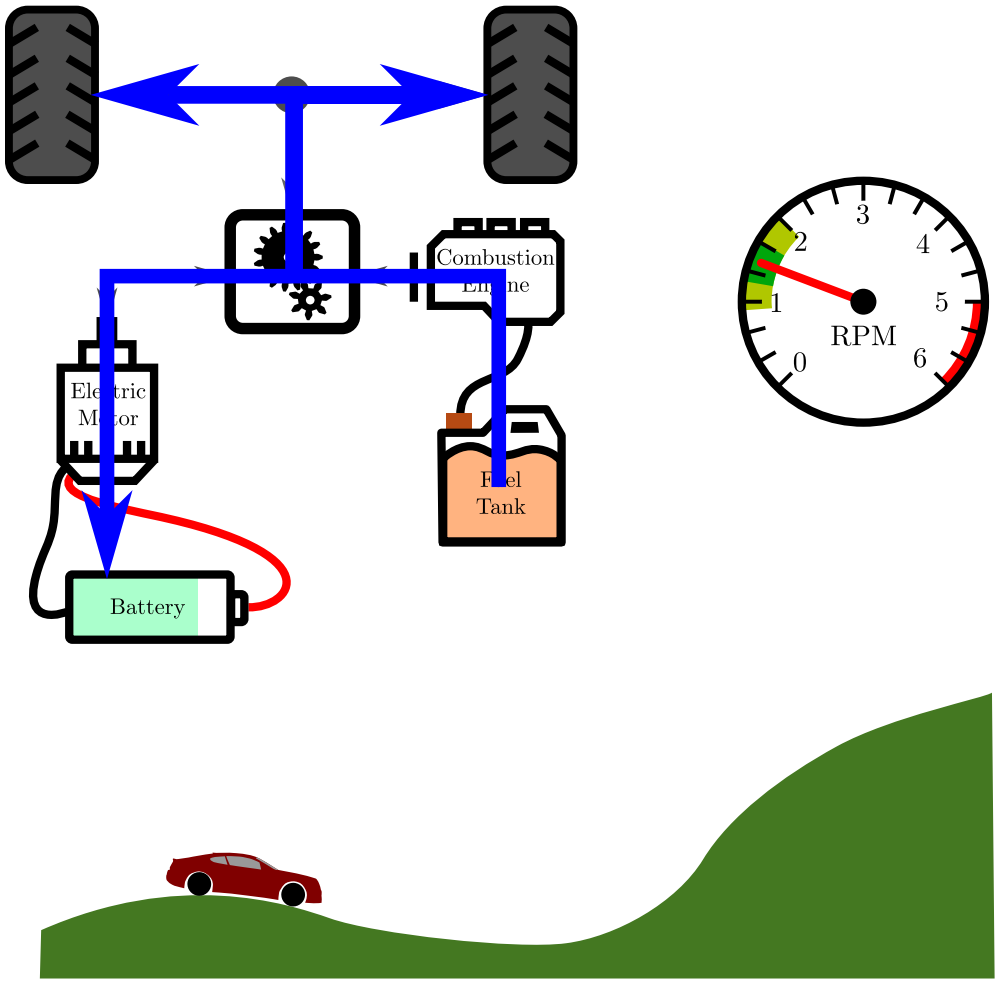


Determine optimal split between electric motor and combustion engine

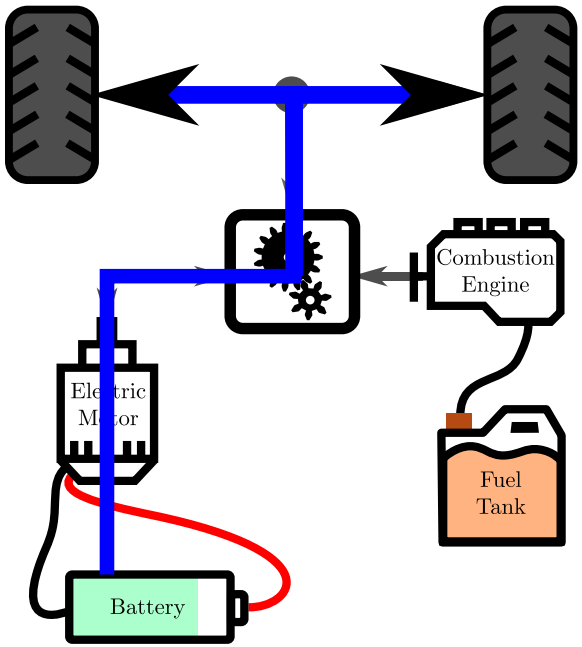


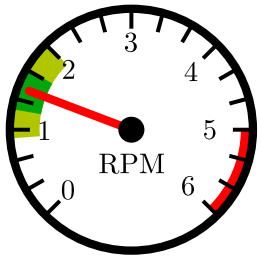


Determine optimal split between electric motor and combustion engine



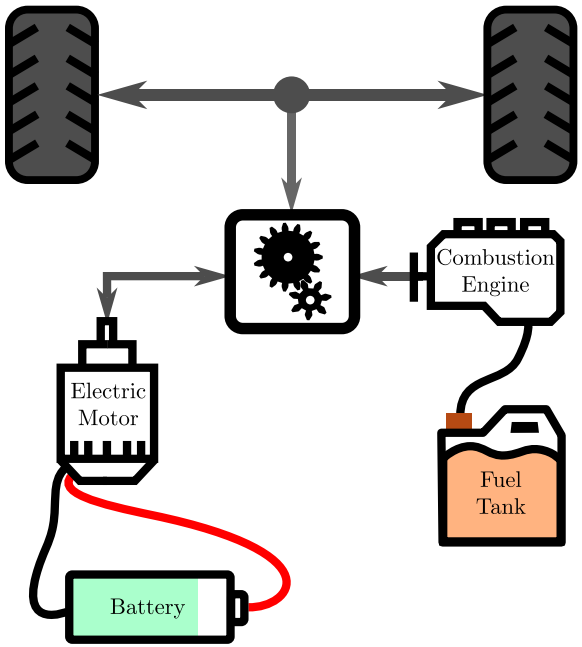
Determine optimal split between electric motor and combustion engine



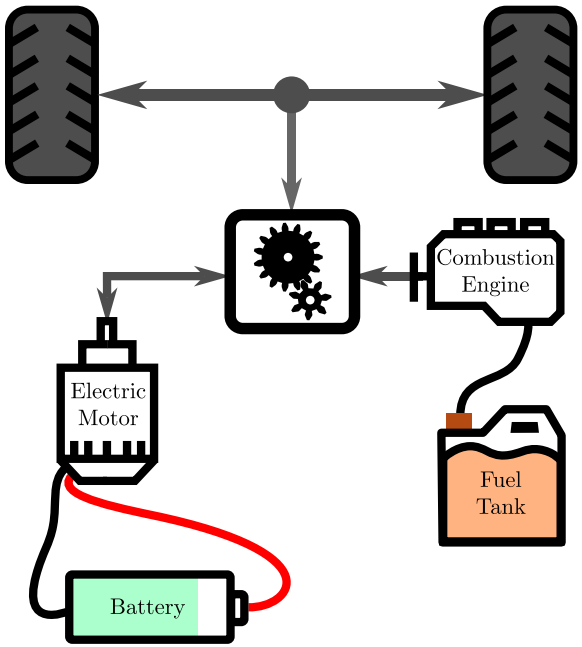


Determine optimal split between electric motor and combustion engine

Dynamic programming is not fast enough on CPUs



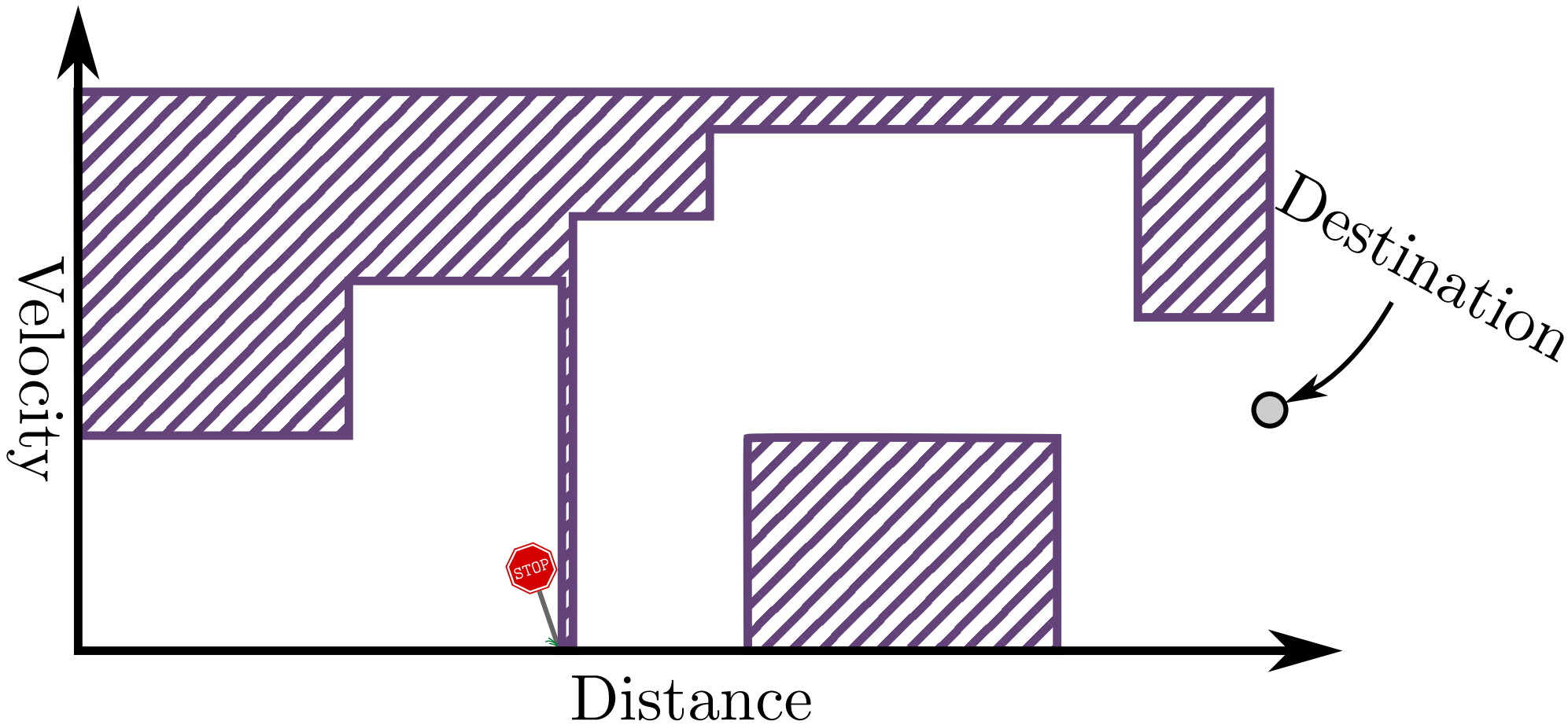
Determine optimal split between electric motor and combustion engine



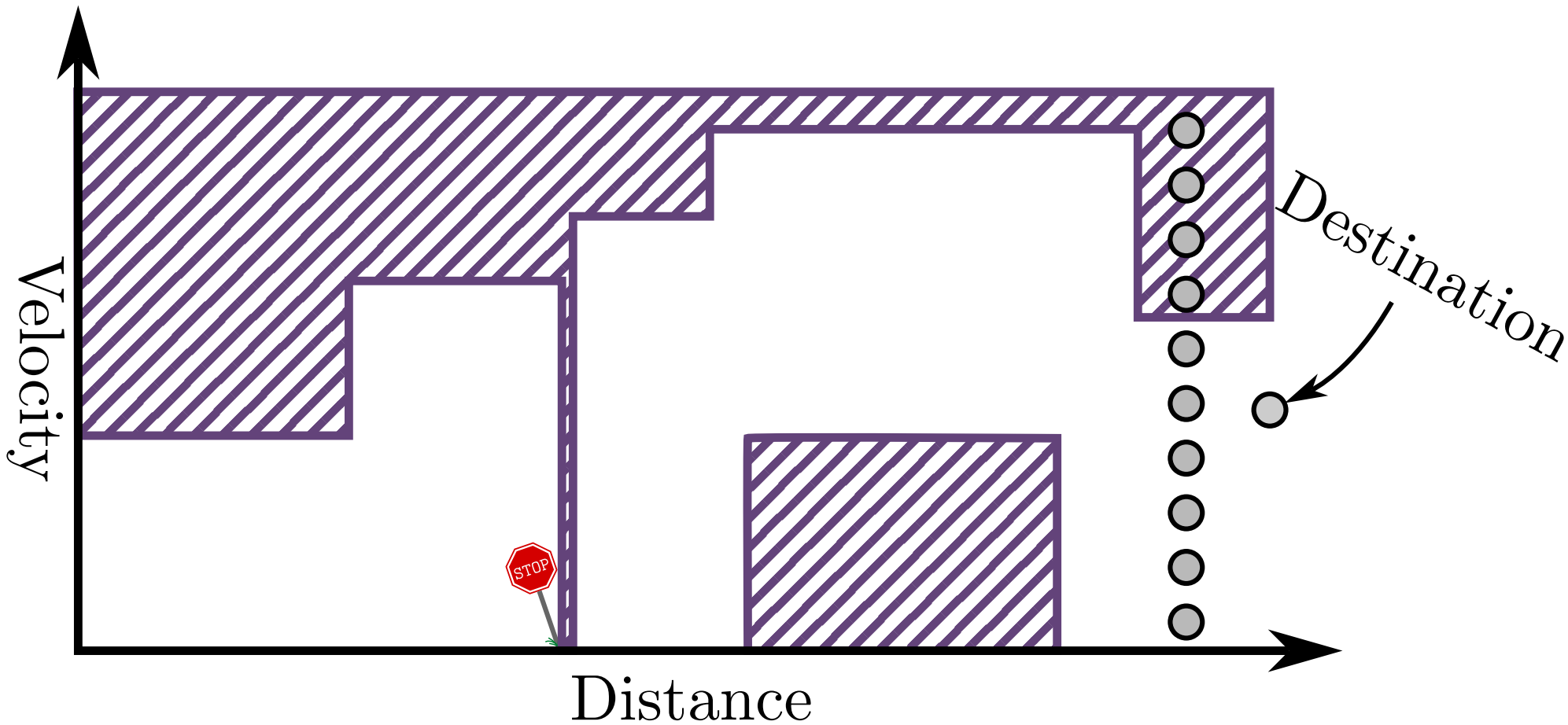
Dynamic programming is not fast enough on CPUs

A scalable FPGA architecture that makes real-time use possible

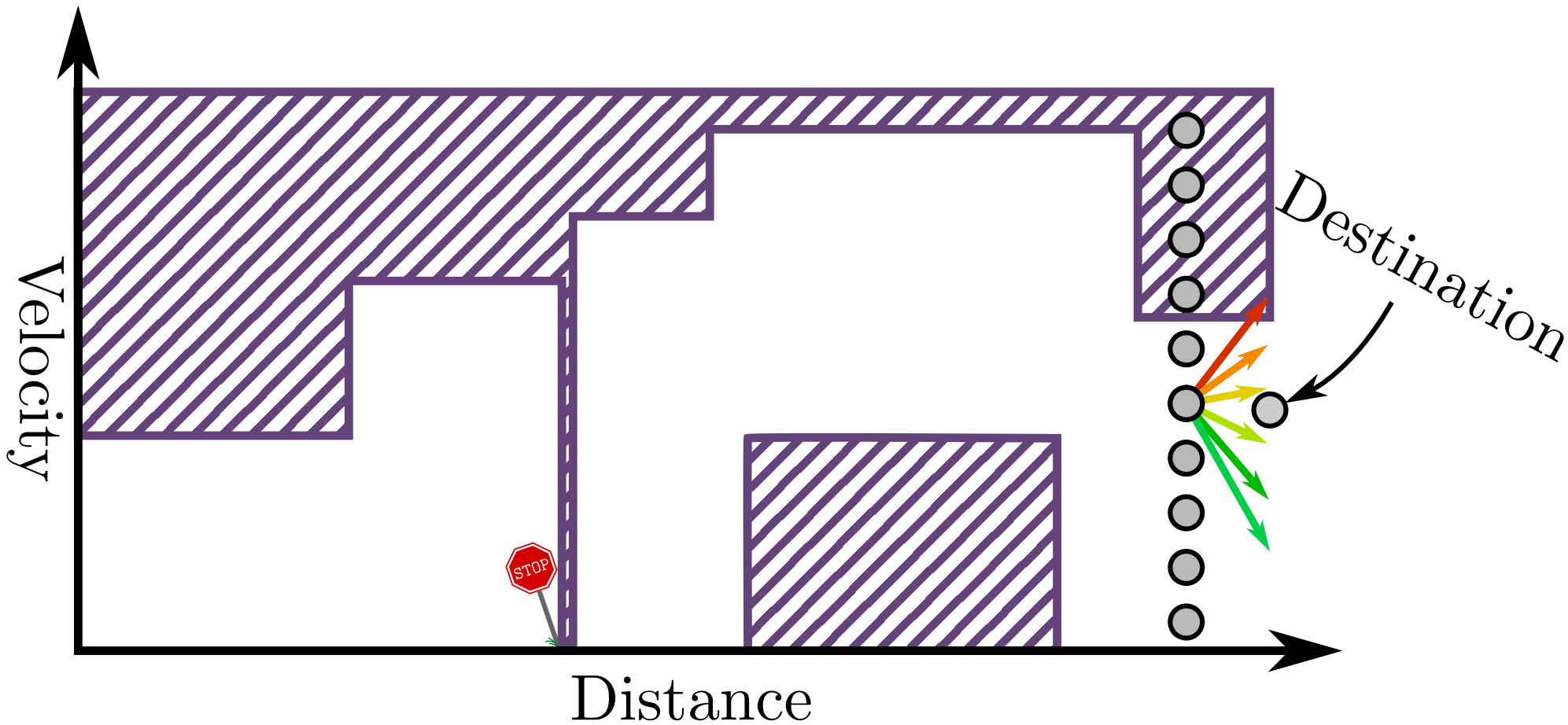
The dynamic programming algorithm



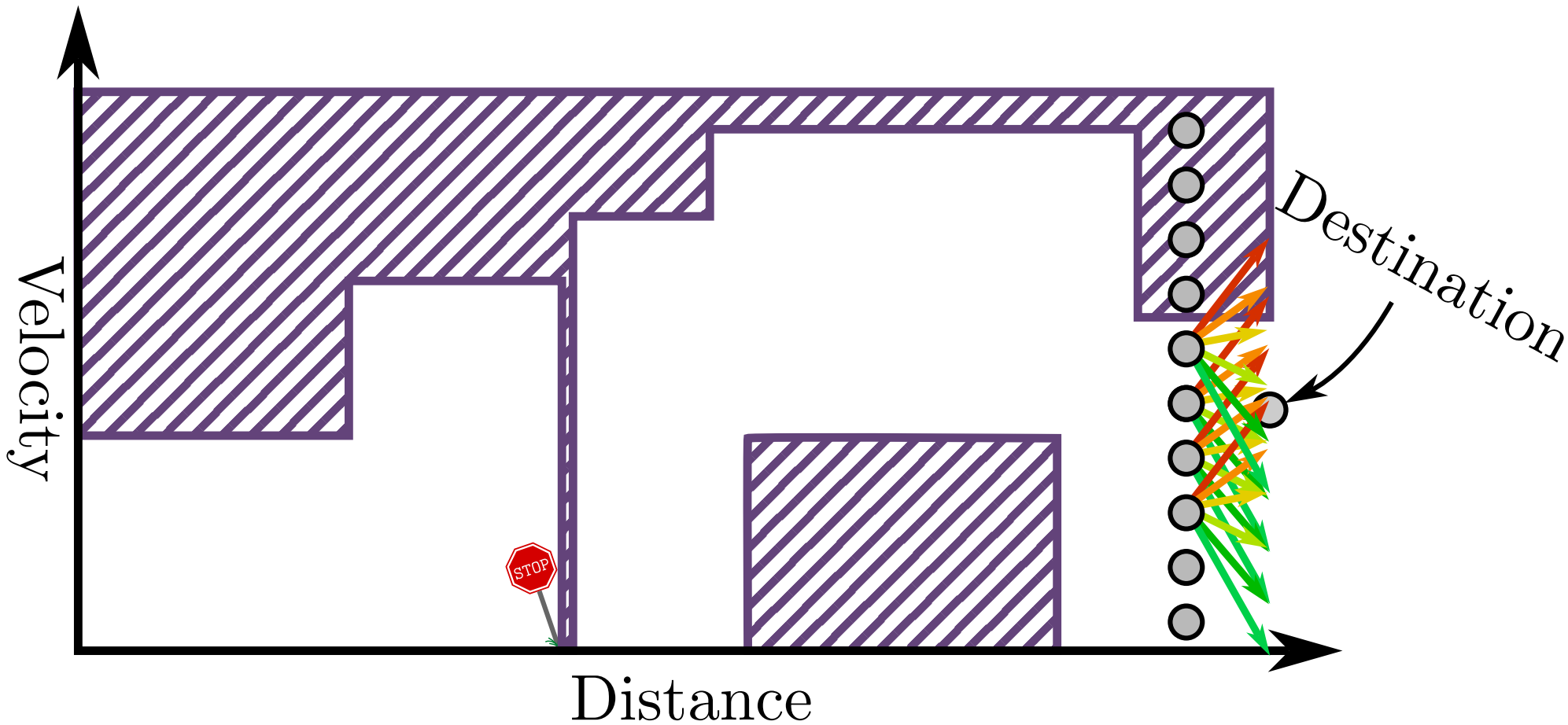
The dynamic programming algorithm



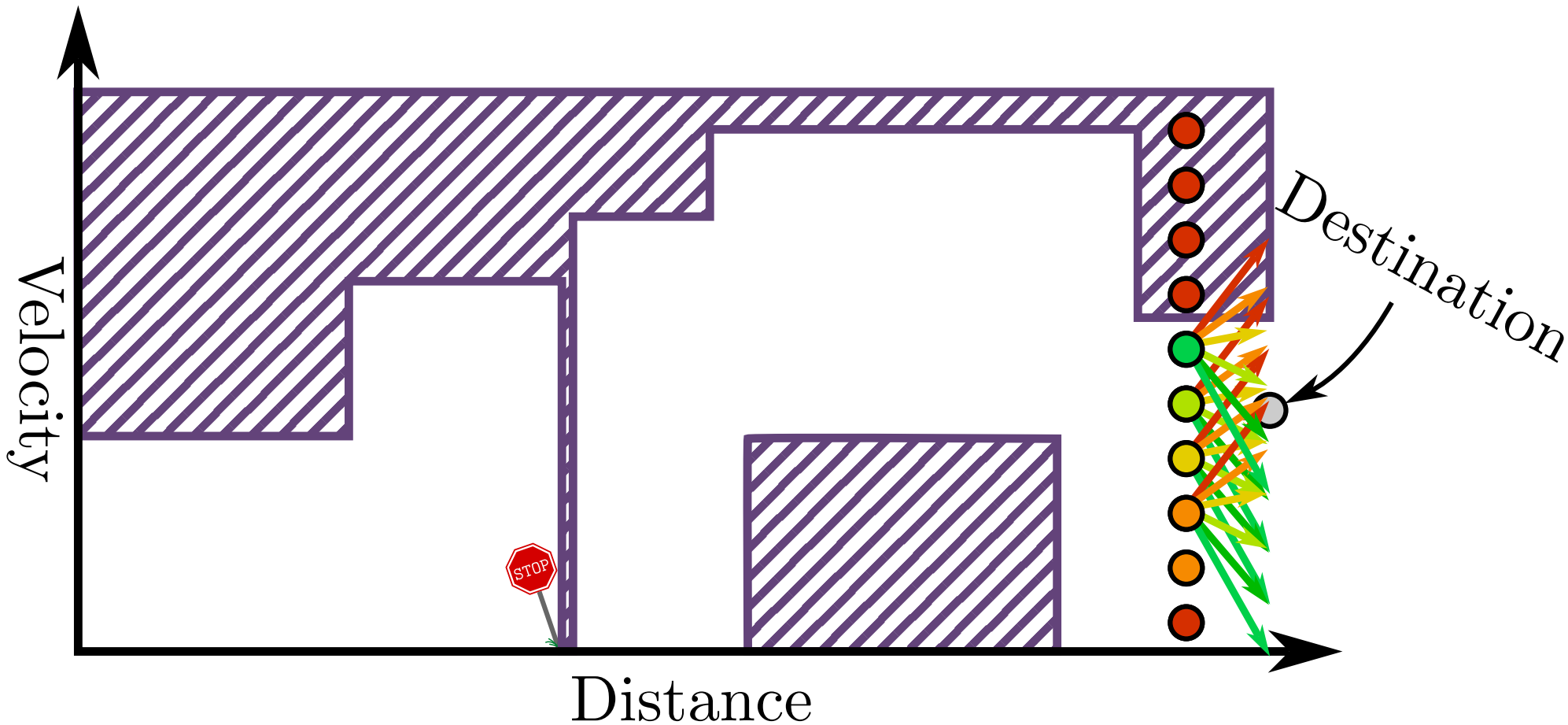
The dynamic programming algorithm



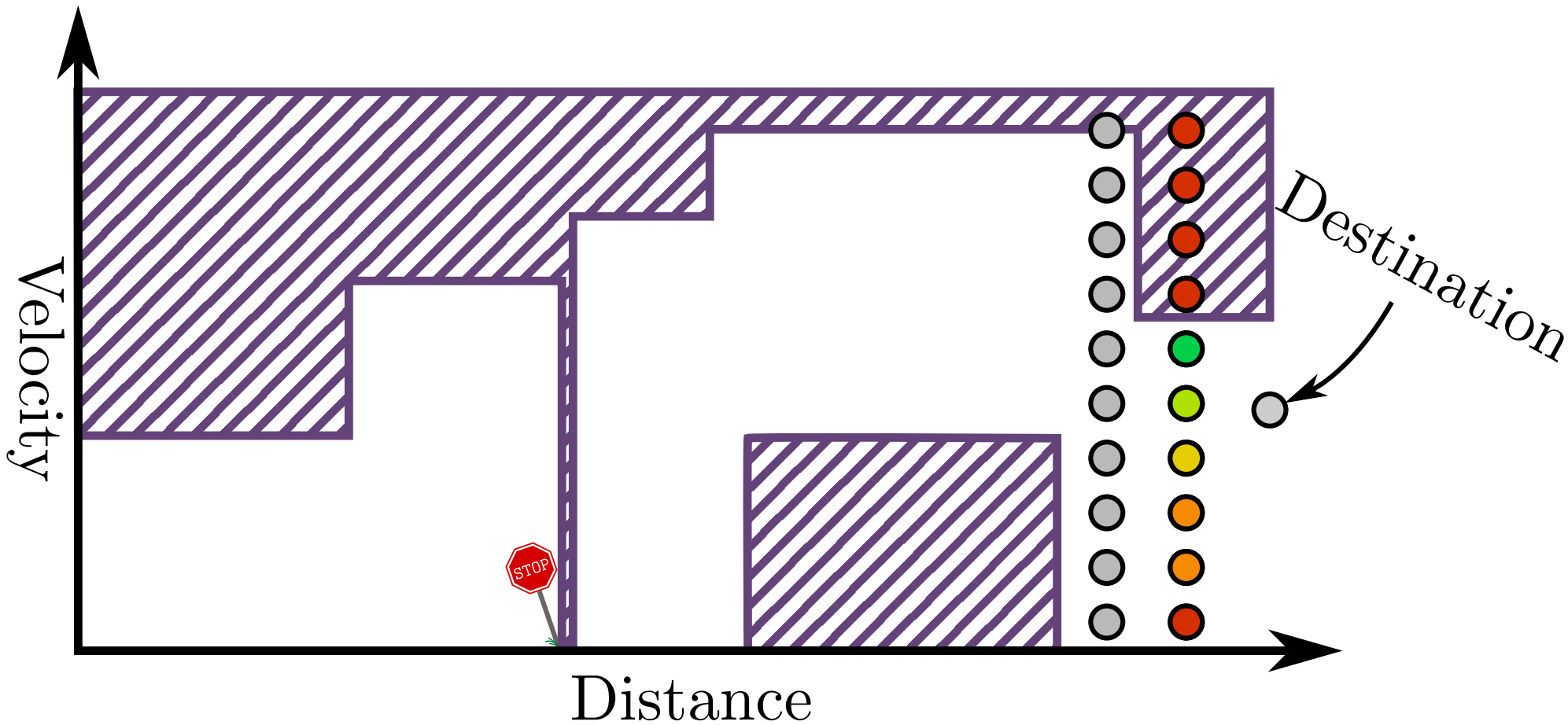
The dynamic programming algorithm



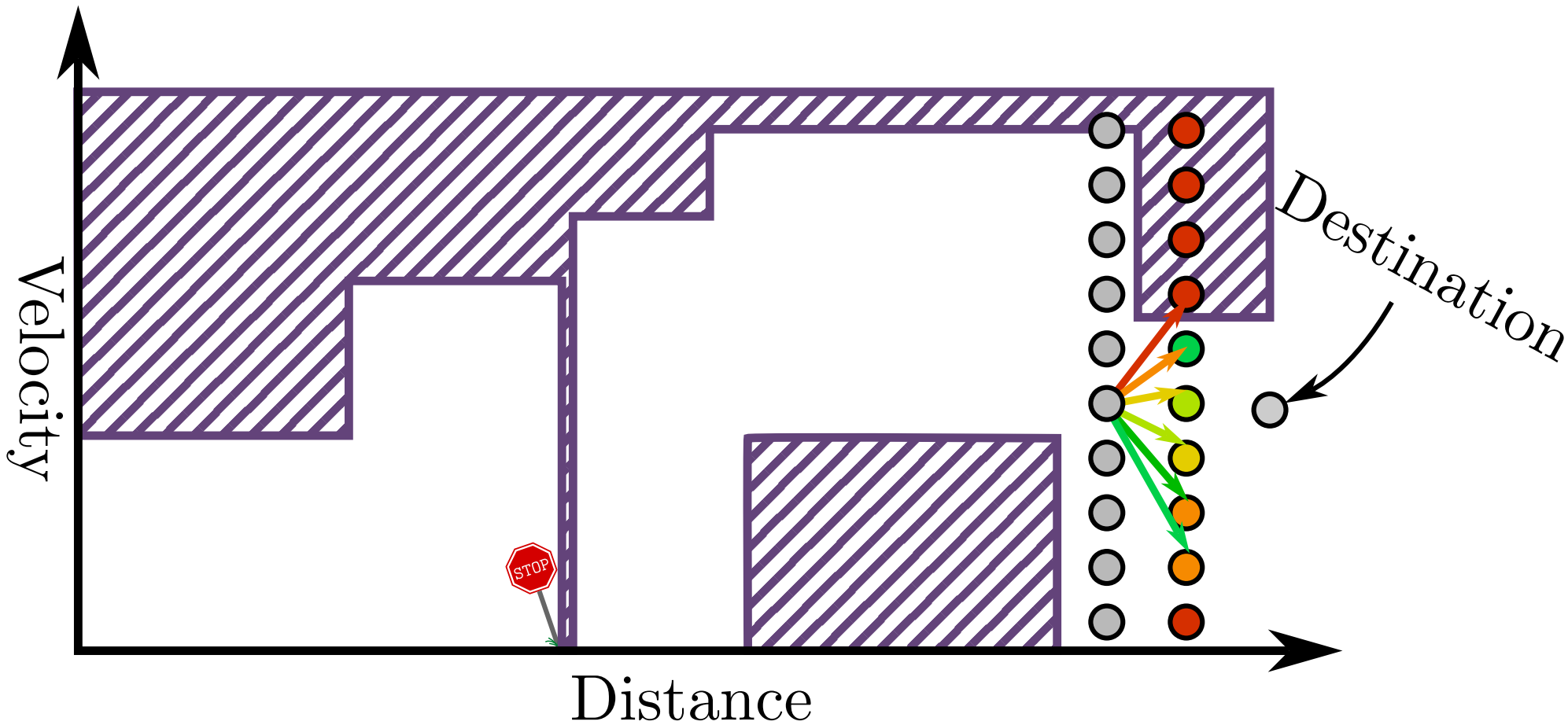
The dynamic programming algorithm



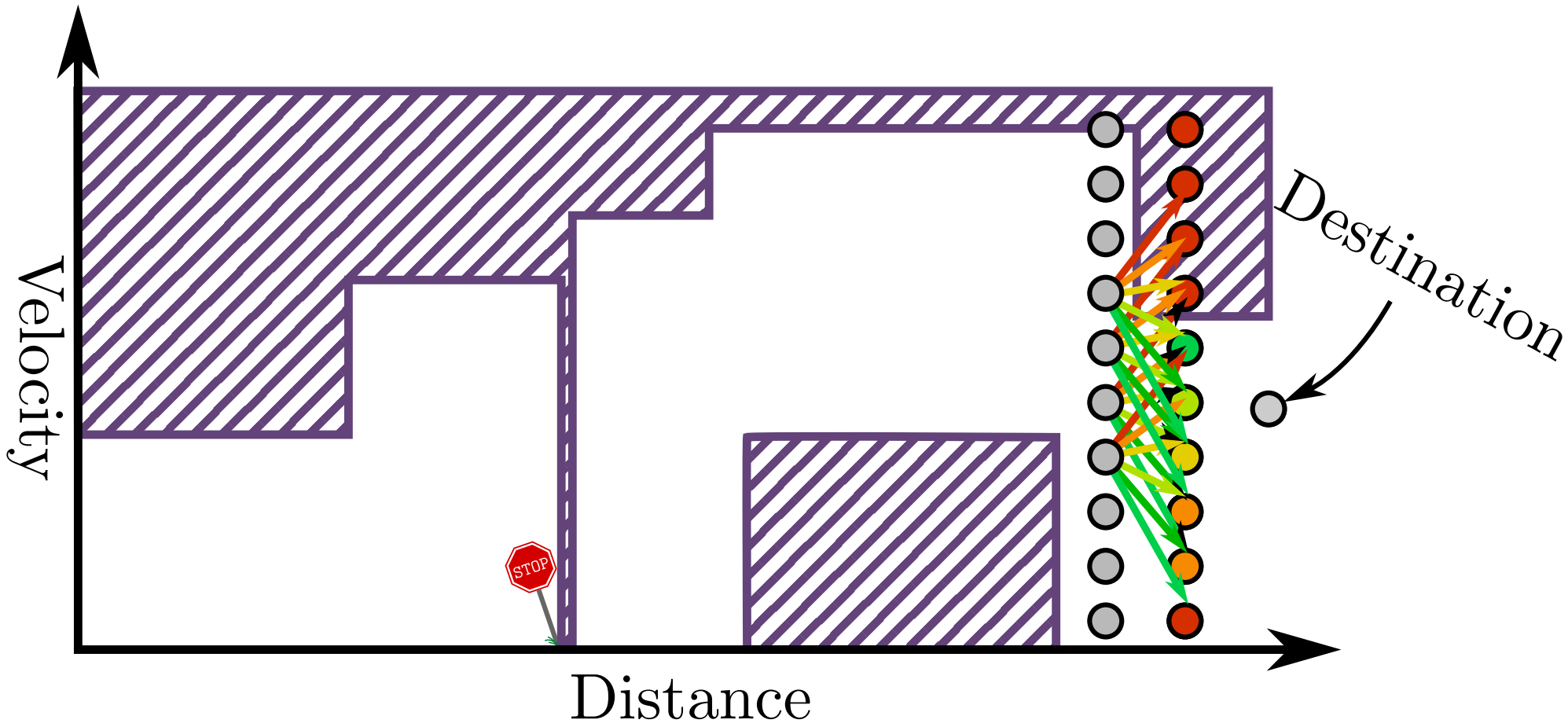
The dynamic programming algorithm



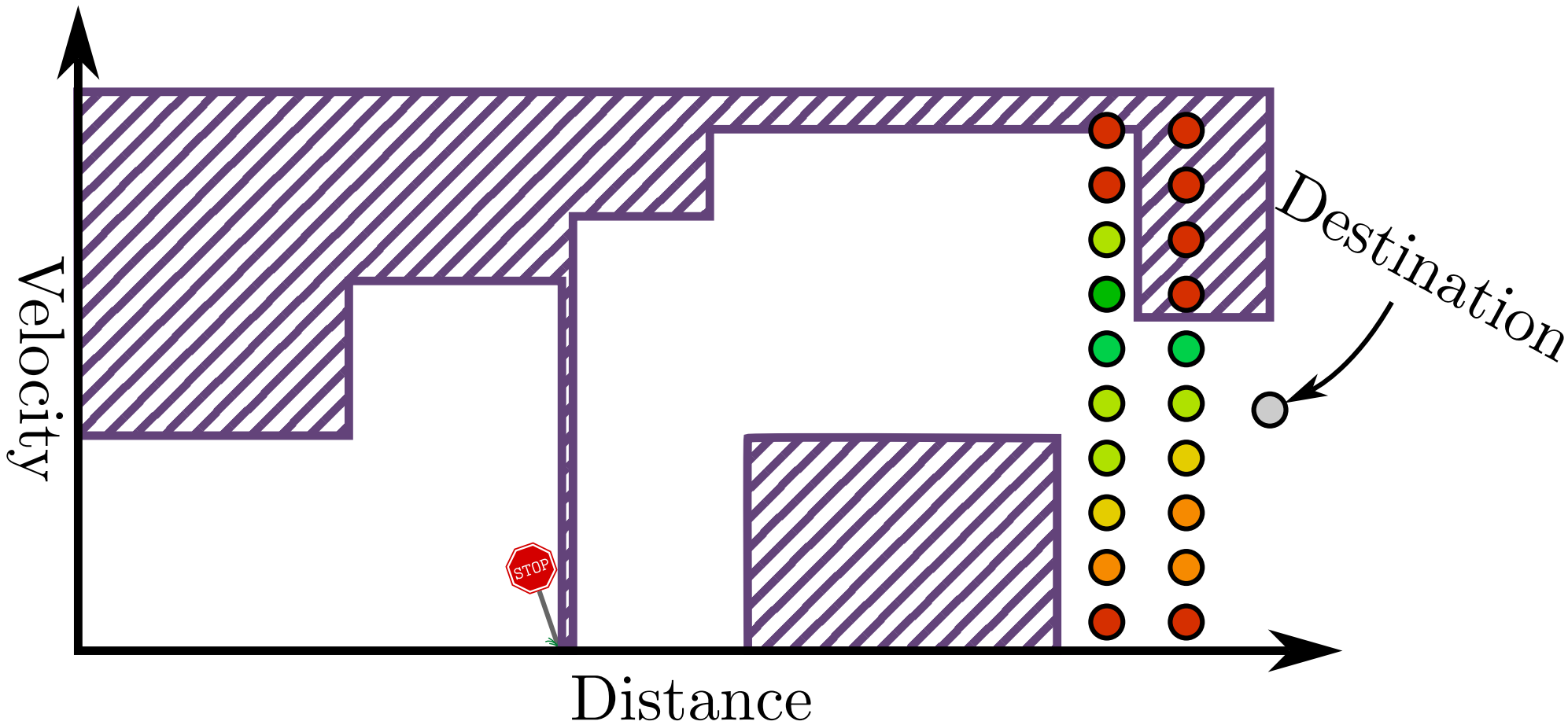
The dynamic programming algorithm



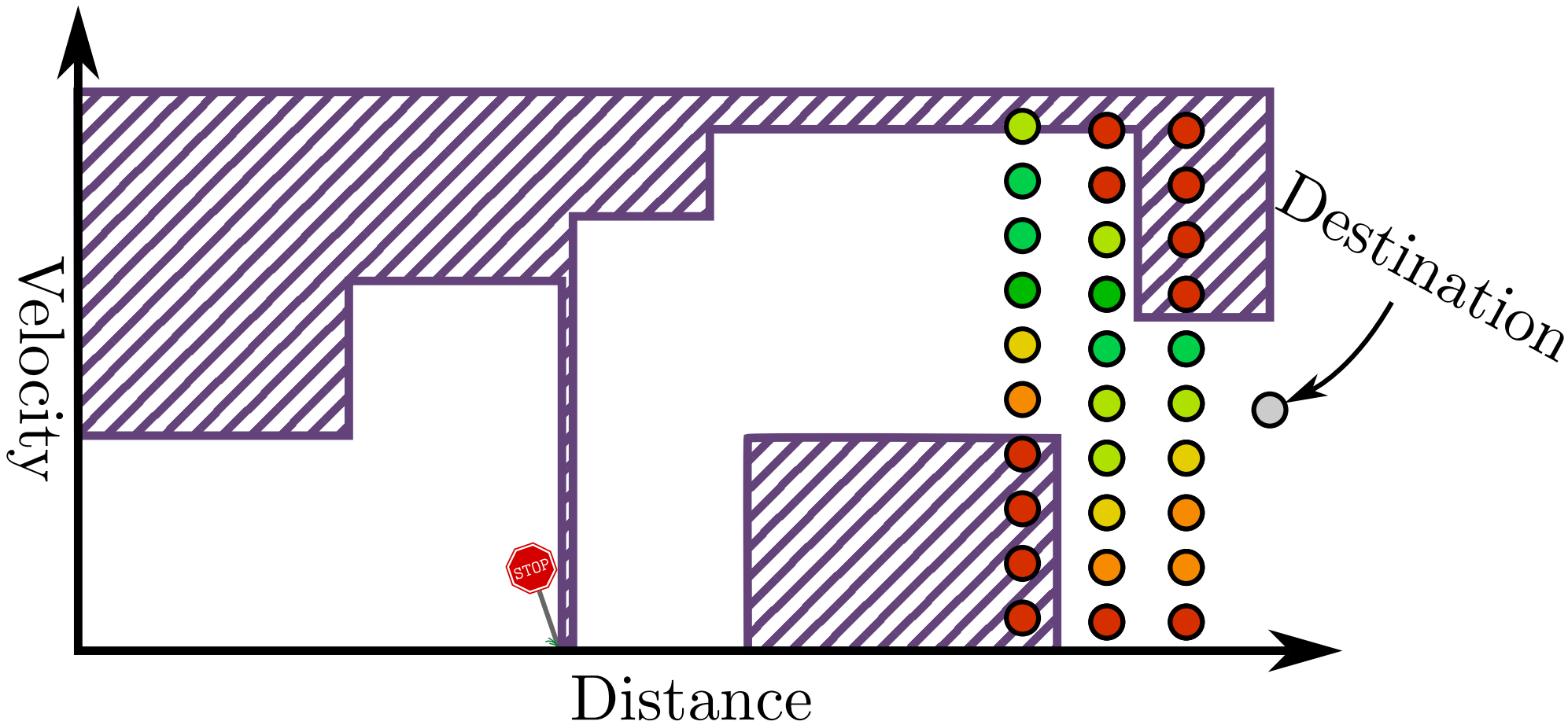
The dynamic programming algorithm



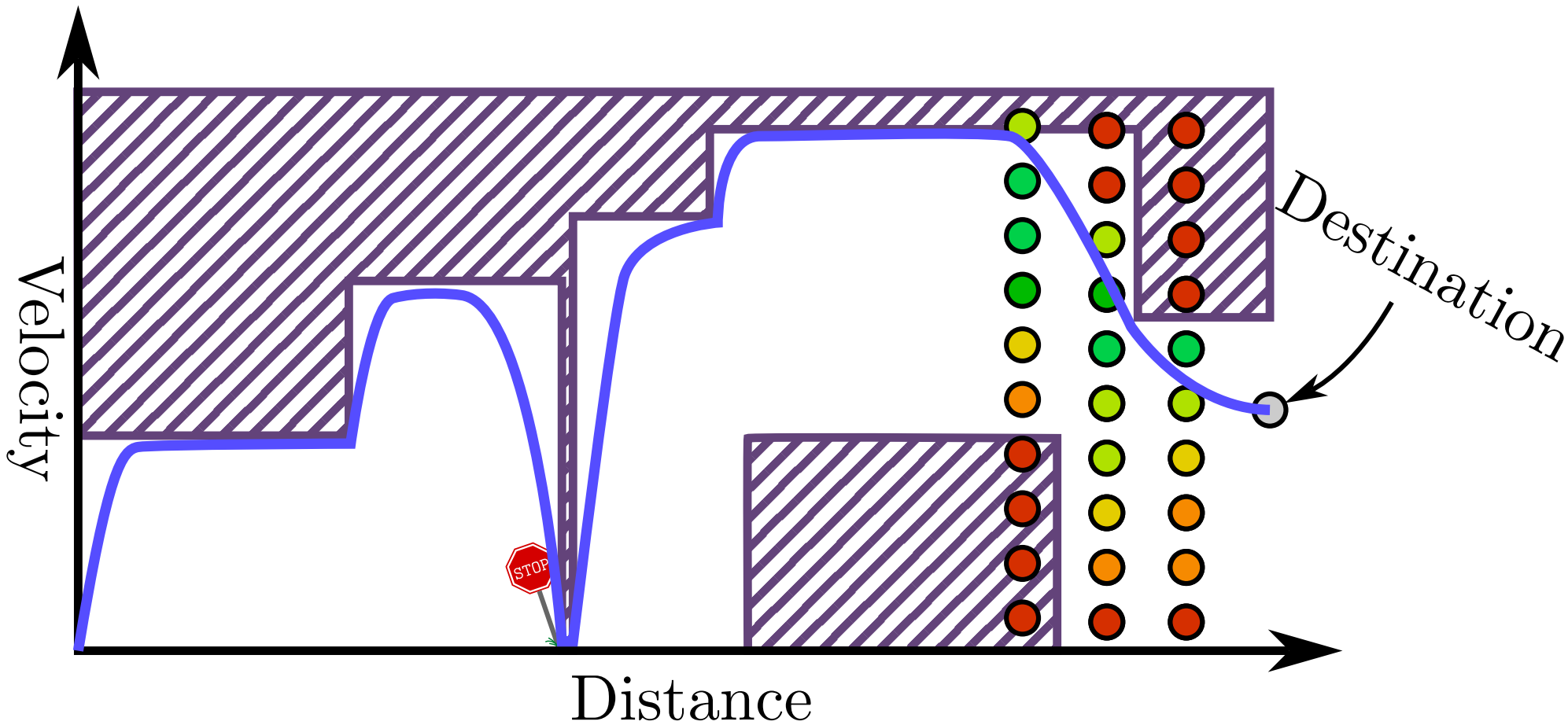
The dynamic programming algorithm



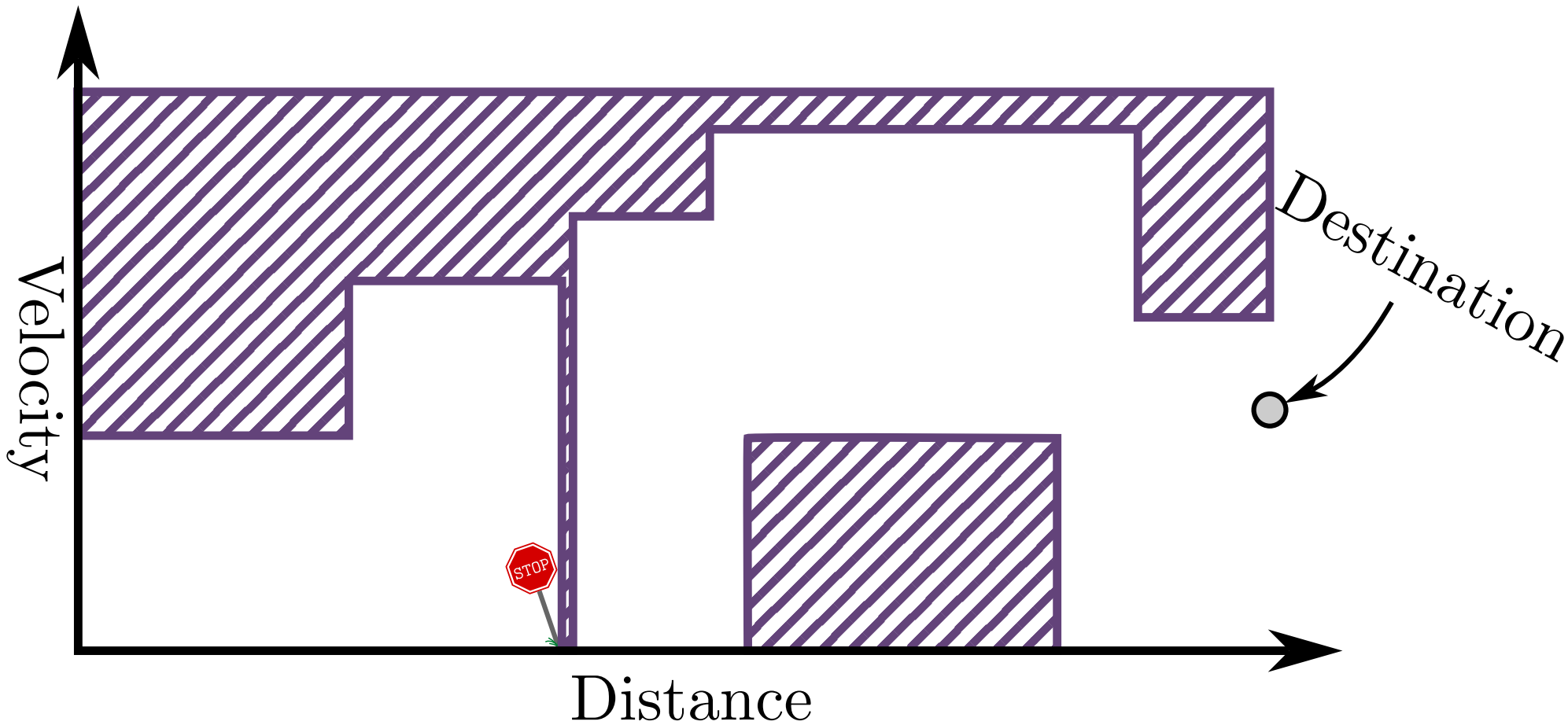
The dynamic programming algorithm



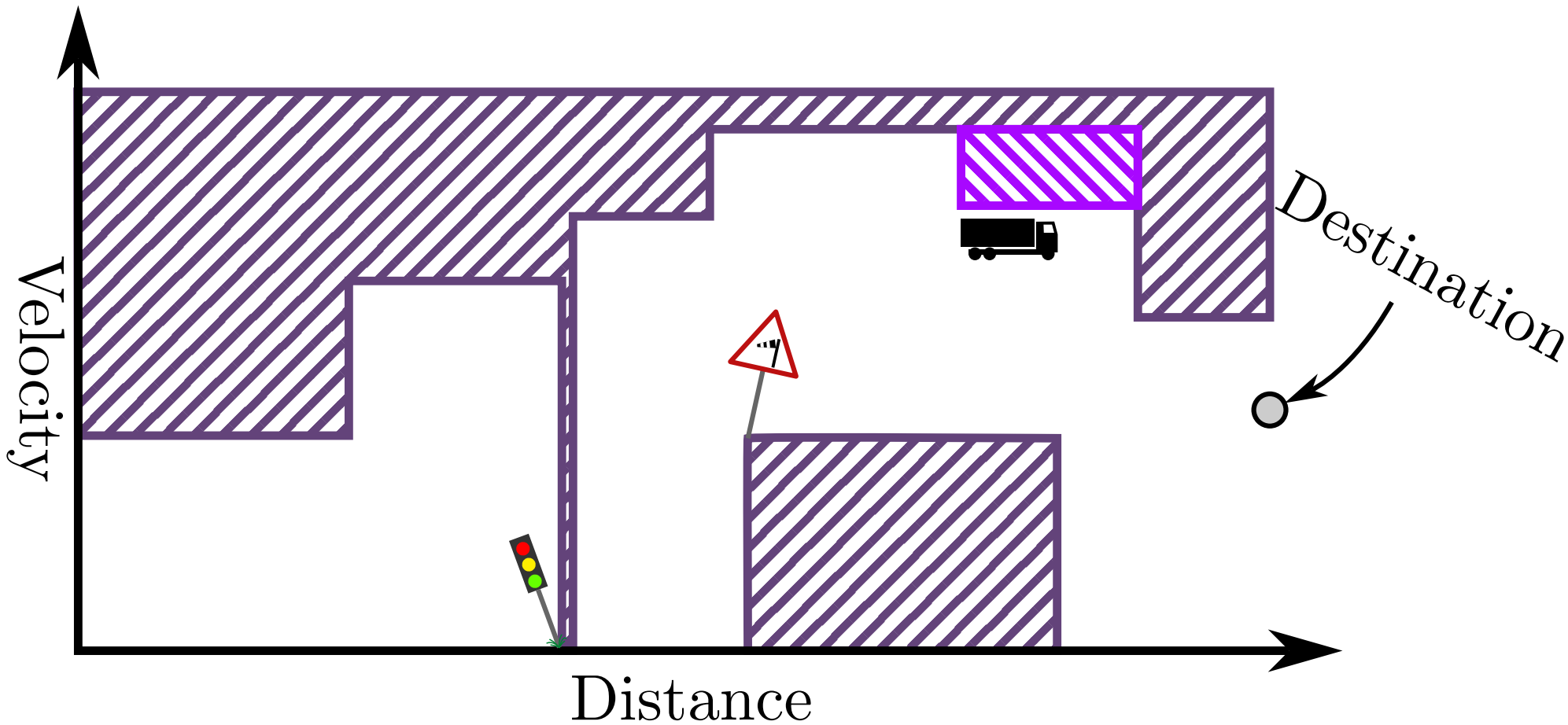
The dynamic programming algorithm



The dynamic programming algorithm



The dynamic programming algorithm



Compute Requirements

- 5.12 km Search horizon with 10 m steps

Compute Requirements

- 5.12 km Search horizon with 10 m steps

States

- 30 velocity steps
- 30 State of Charge (SOC) steps

Compute Requirements

- 5.12 km Search horizon with 10 m steps

States

- 30 velocity steps
- 30 State of Charge (SOC) steps

Inputs

- 30 steps of electric torque
- 30 steps of conbustion torque
- 6 gears

Compute Requirements

$$\underbrace{512}_{\text{Distance Steps}} \times \underbrace{30 \times 30}_{\text{States}} \times \underbrace{30 \times 30 \times 6}_{\text{Inputs}} \approx \underbrace{2.4 \text{ billion}}_{\text{Model evaluations}}$$

Compute Requirements

Distance Steps

Inputs

$$512 \times 30 \times 30 \times 30 \times 30 \times 6 \approx 2.4 \text{ billion}$$

States

Model evaluations

≈ 2 seconds for real time

Compute Requirements

$$\underbrace{512}_{\text{Distance Steps}} \times \underbrace{30 \times 30}_{\text{States}} \times \underbrace{30 \times 30 \times 6}_{\text{Inputs}} \approx \underbrace{2.4 \text{ billion}}_{\text{Model evaluations}}$$

≈ 2 seconds for real time

> 1 model execution every clock cycle

Vehicle Model

State

Next state

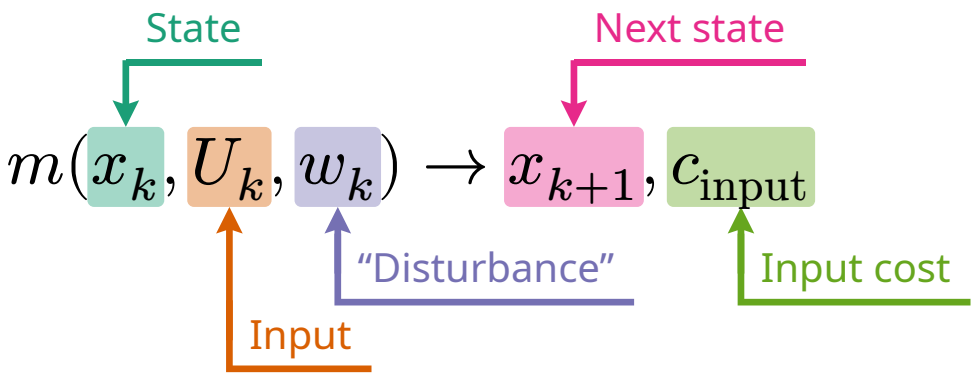
$$m(x_k, U_k, w_k) \rightarrow x_{k+1}, c_{\text{input}}$$

"Disturbance"

Input cost

Input

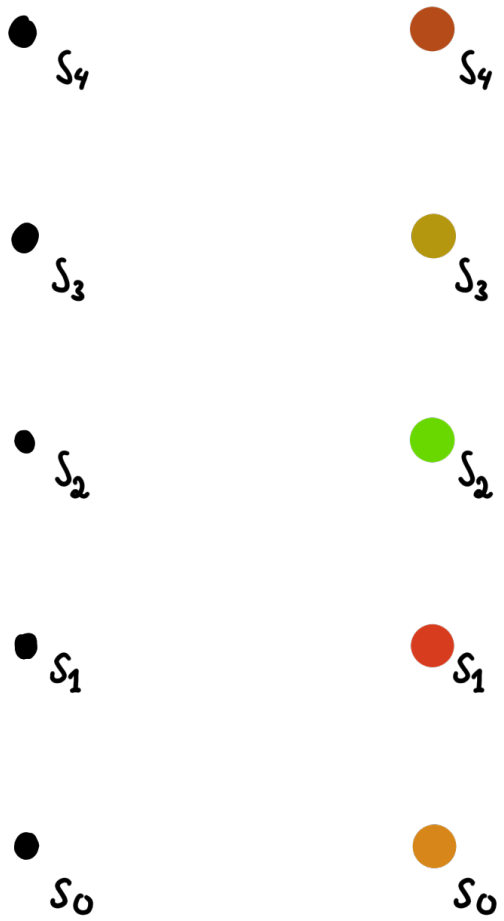
Vehicle Model



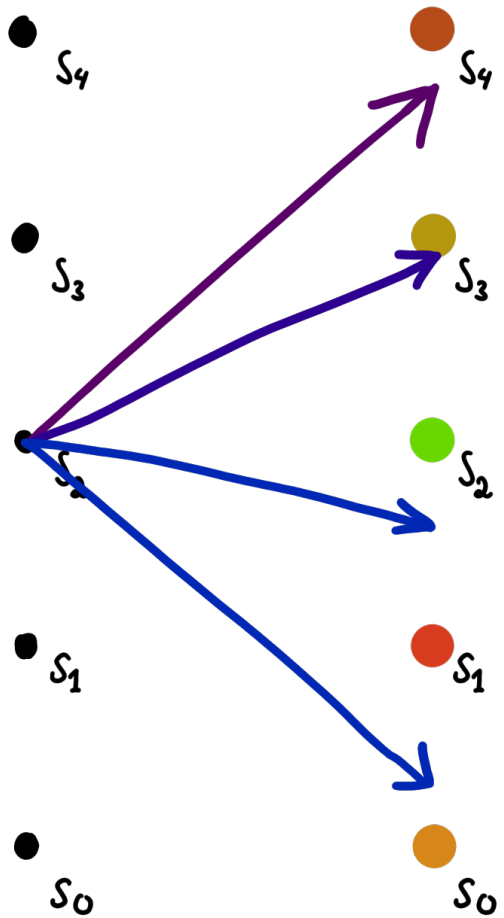
C++ model converted with HLS

Pipelined with Initiation Interval = 1

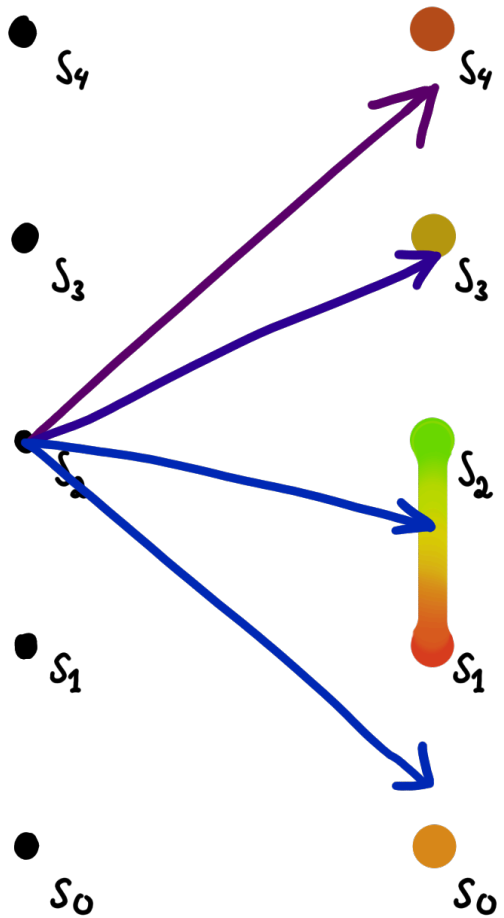
Interpolation



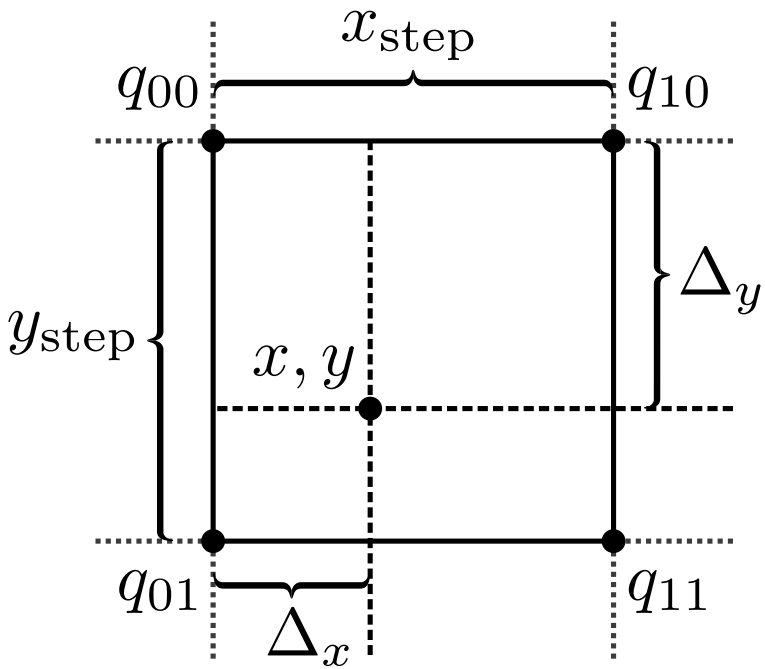
Interpolation



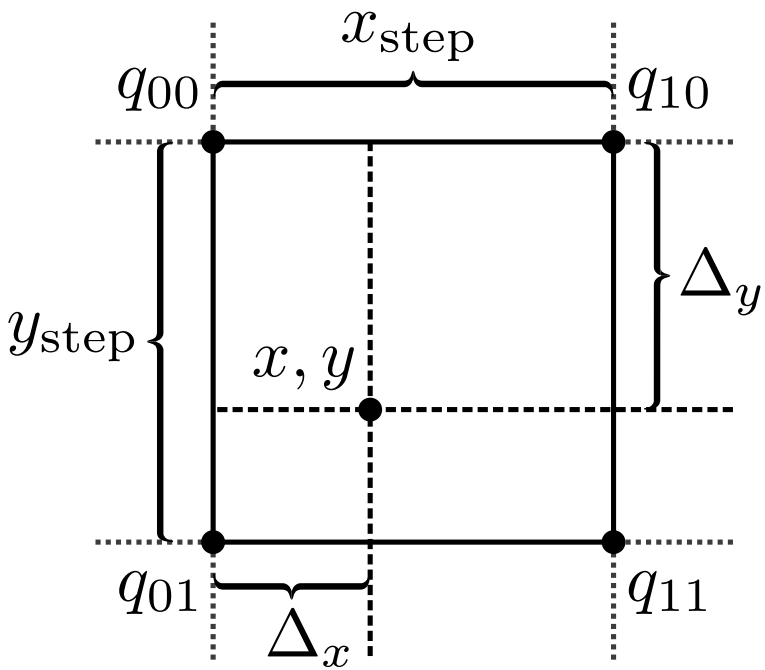
Interpolation



Interpolation

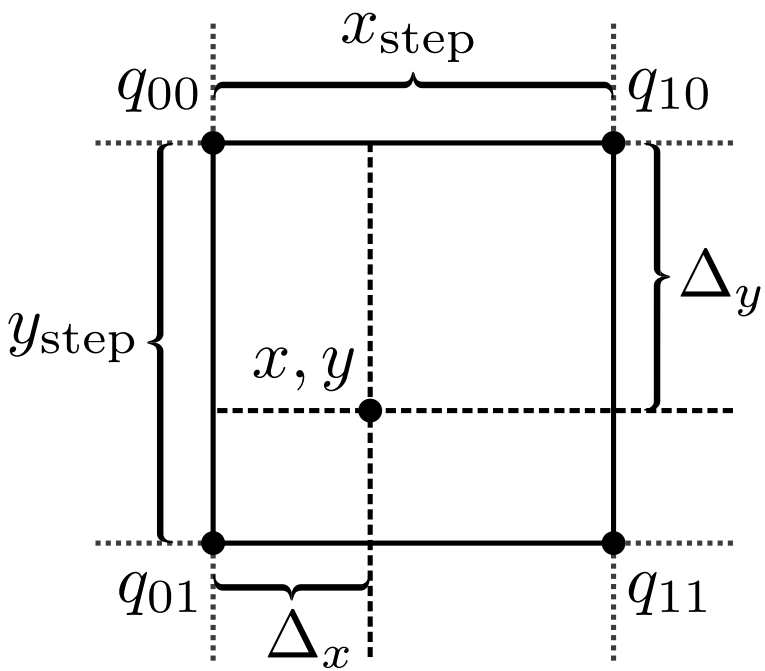


Interpolation



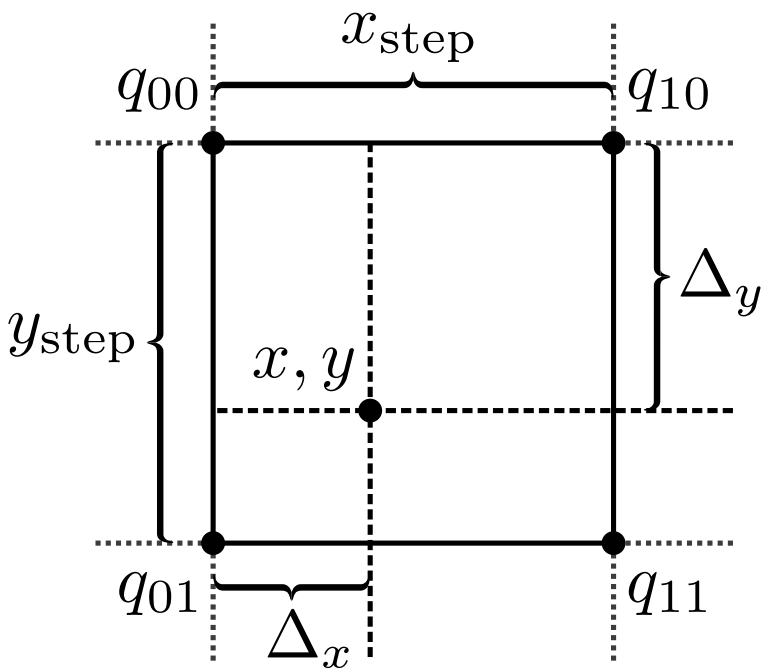
- 2D linear interpolation

Interpolation



- 2D linear interpolation
- Requires 4 memory accesses per value

Interpolation

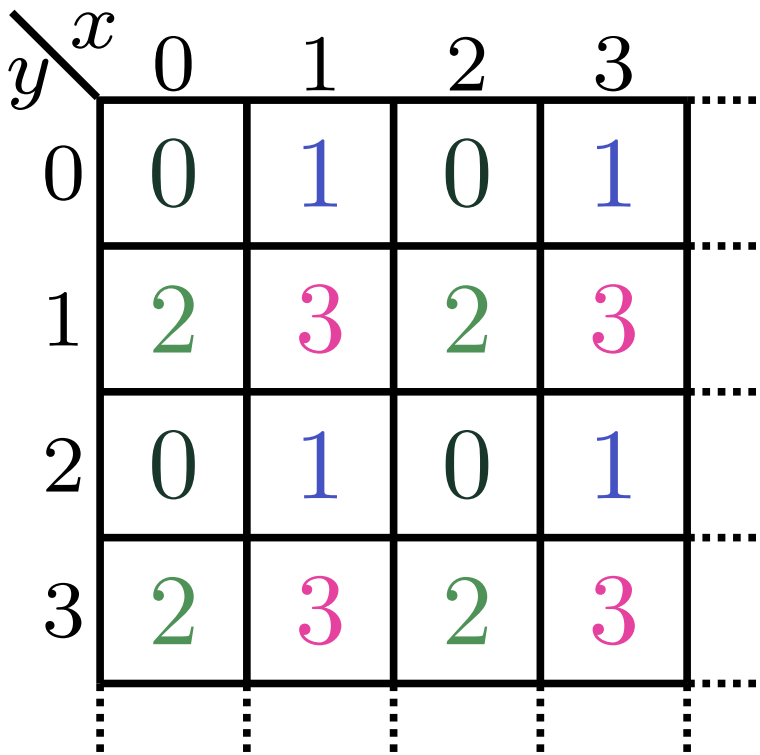


- 2D linear interpolation
- Requires 4 memory accesses per value
- Simultaneous writeback is required

Memory Read Partitioning

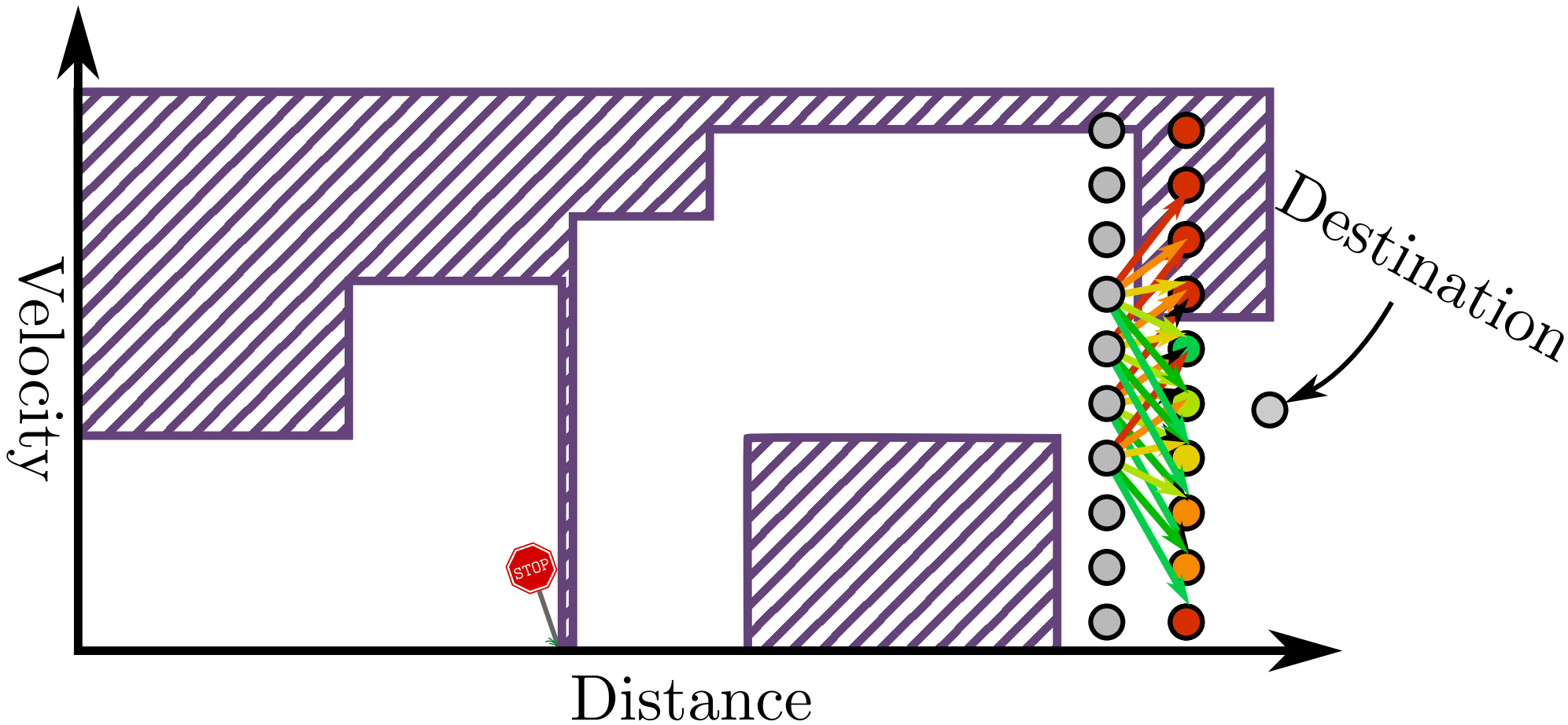
$y \backslash x$	0	1	2	3
0	0	1	0	1
1	2	3	2	3
2	0	1	0	1
3	2	3	2	3

Memory Read Partitioning

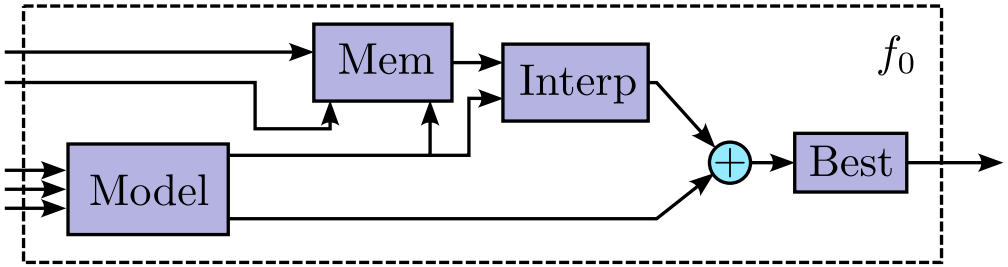


- 4 separate memories
- x, y evenness determines indexing

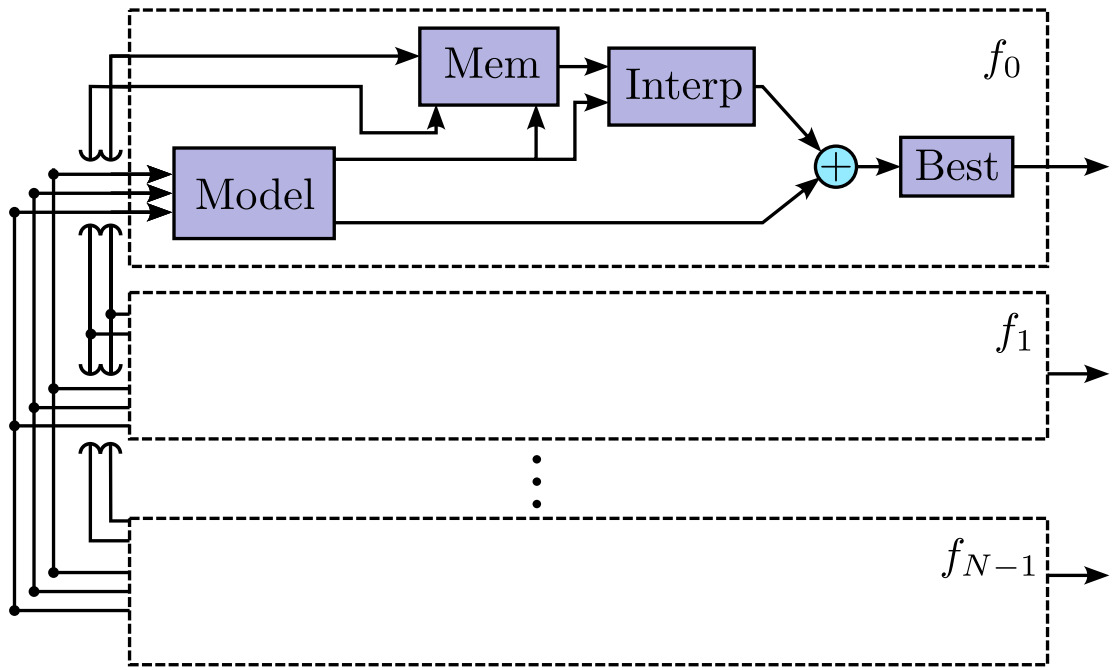
Memory Write Partitioning



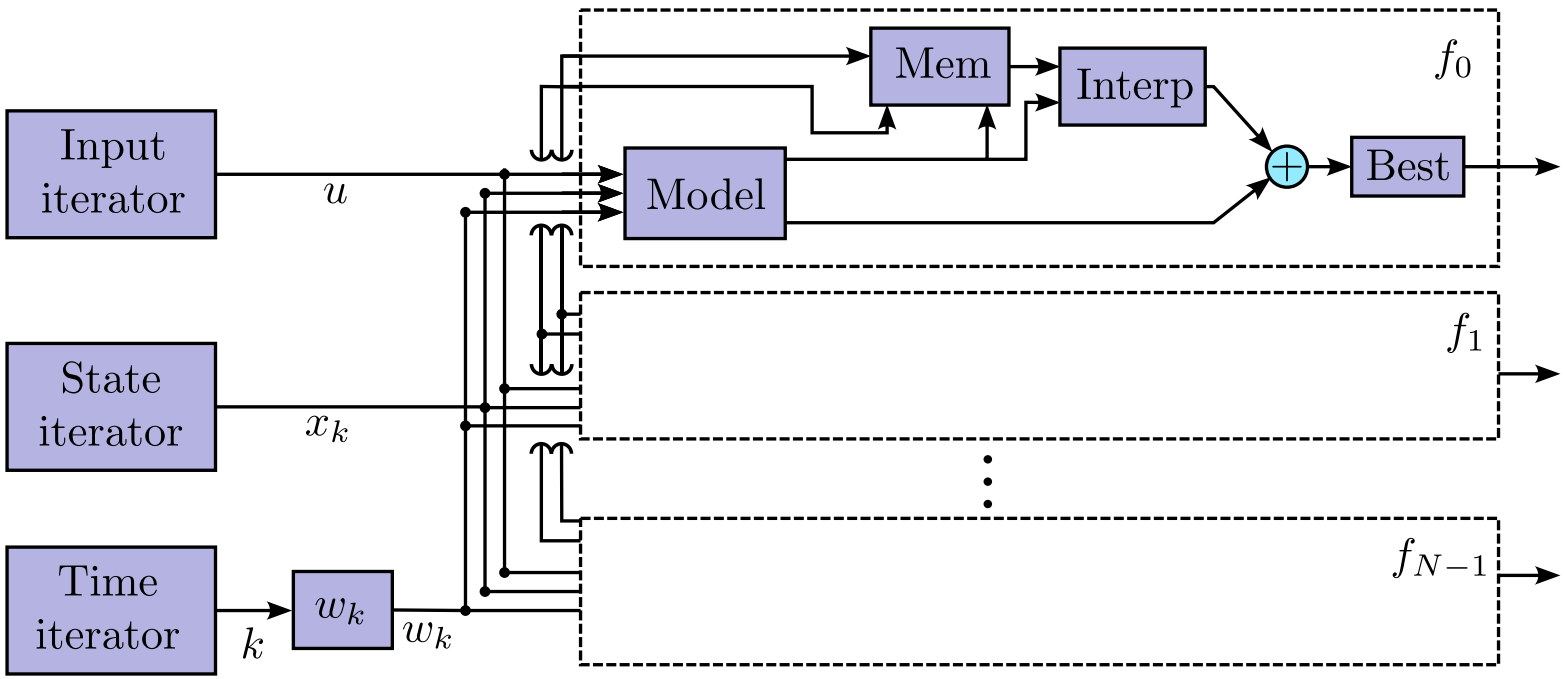
Architecture Overview



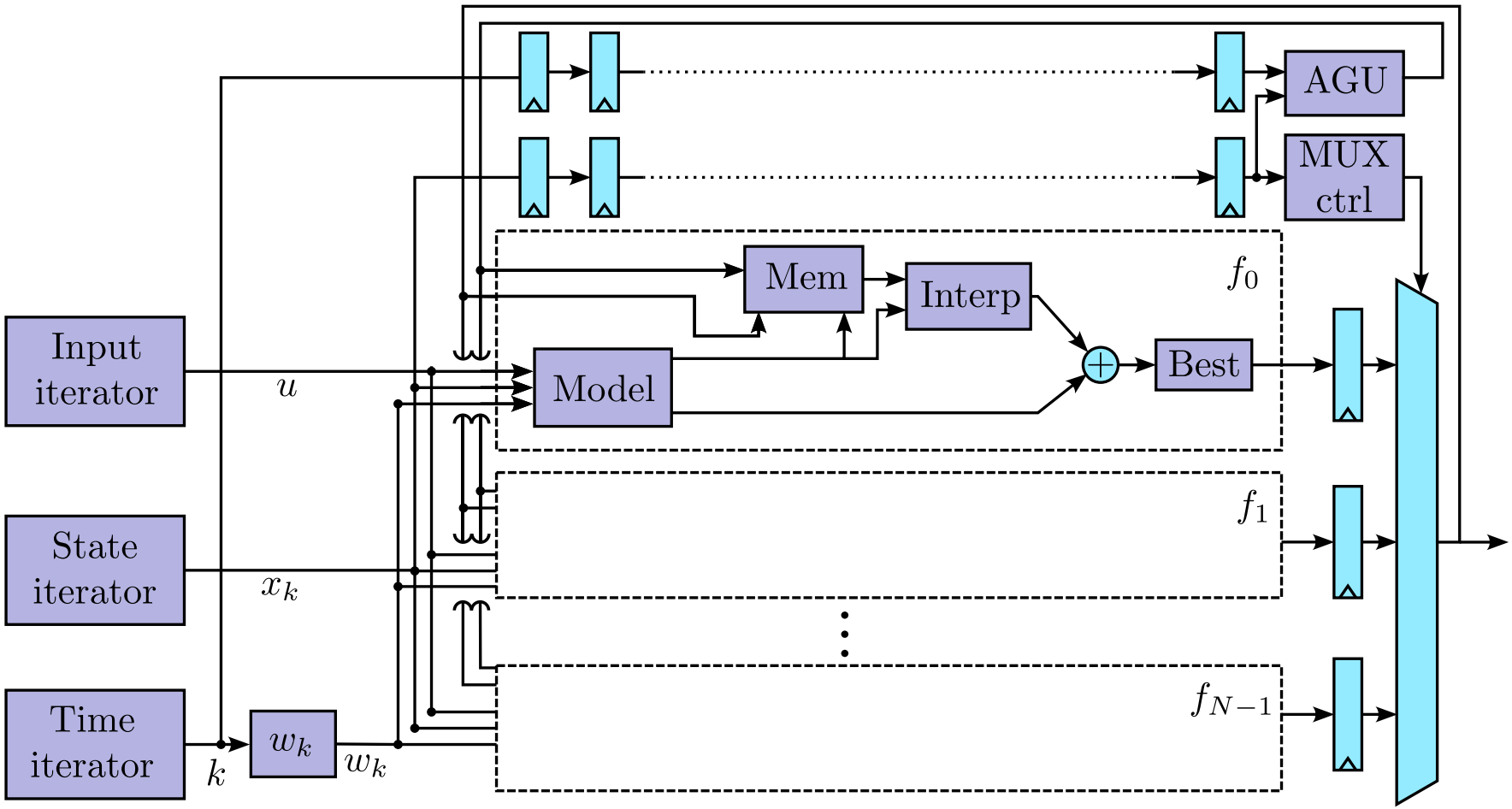
Architecture Overview



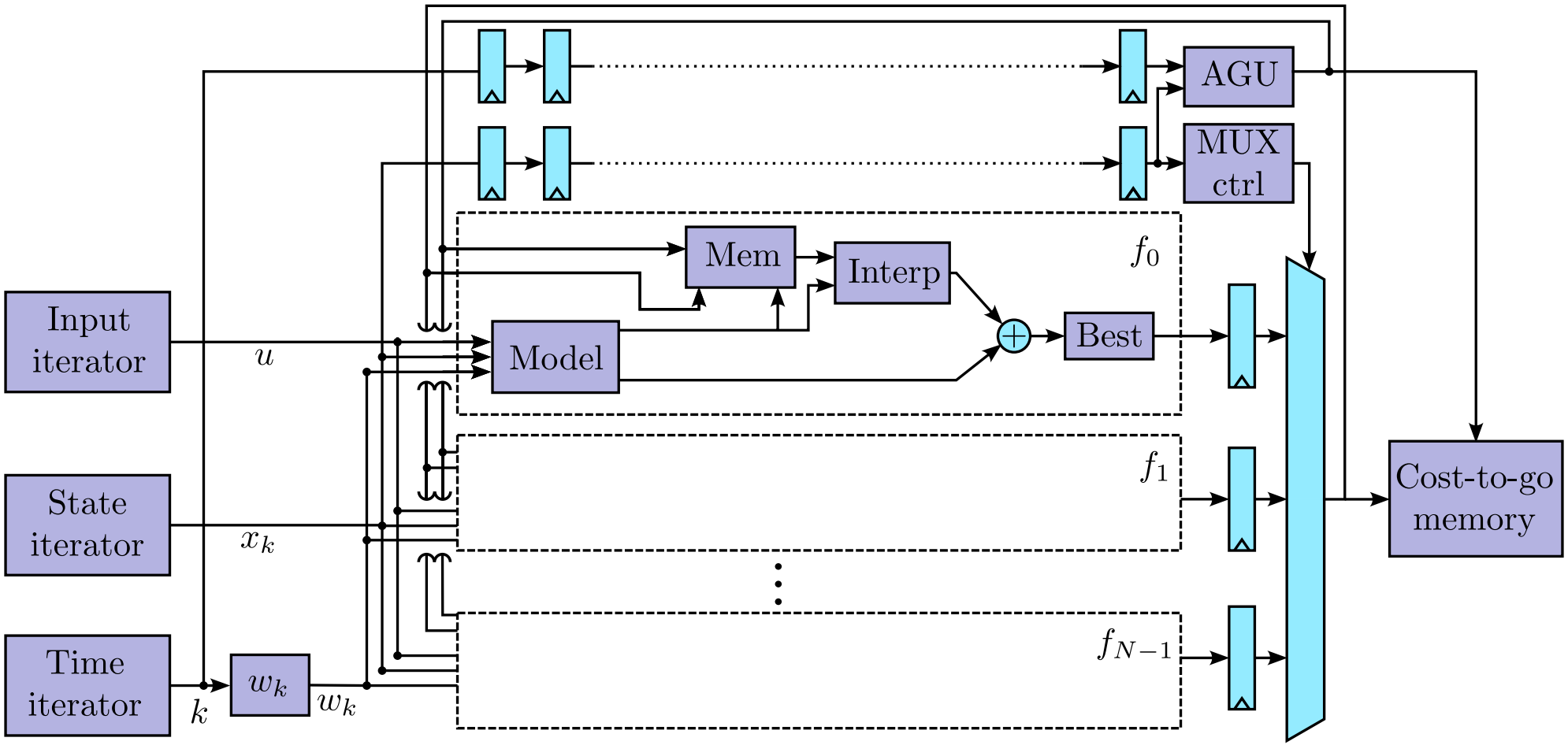
Architecture Overview



Architecture Overview



Architecture Overview



Schedule and Performance

Inputs

States

Interpolation latency

Model latency

$$N\left(U\cdot\left\lceil\frac{S}{P}\right\rceil+P+I-1\right)+D=C$$

Exeuction units

Clock cycles

Distance steps

Schedule and Performance

Inputs

States

Interpolation latency

Model latency

$$512\left(5400\cdot\left\lceil\frac{900}{4}\right\rceil+4+10-1\right)+894=6.220\cdot10^{8}$$

Exeuction units

Clock cycles

Distance steps

Resource Usage and Runtime

- Implemented in Spade HDL
- Tools:
  - Vitis HLS 2022.1
  - Vivado 2022.1
  - AMD Virtex UltraScale+ `xcvu13pfhga2104-3-e`

Resource Usage and Runtime

EUs	$f_{\max}$ (MHz)	CCs required	Run time (s)	Speedup	CLB	DSP	BRAM	URAM
1	370	$2.488 \cdot 10^9$	6.73	$18\times$	10196	515	238	154
2	336	$1.244 \cdot 10^9$	3.70	$34\times$	21384	1030	476	154
4	335	$6.220 \cdot 10^8$	1.85	$68\times$	41654	2060	952	154
6	288	$4.147 \cdot 10^8$	1.44	$87\times$	62627	3090	1428	154
9	272	$2.764 \cdot 10^8$	1.02	$123\times$	94874	4635	2142	154

Resource Usage and Runtime

EUs	$f_{\max}$ (MHz)	CCs required	Run time (s)	Speedup	CLB	DSP	BRAM	URAM
1	370	$2.488 \cdot 10^9$	6.73	18 ×	10196	515	238	154
2	336	$1.244 \cdot 10^9$	3.70	34 ×	21384	1030	476	154
4	335	$6.220 \cdot 10^8$	1.85	68 ×	41654	2060	952	154
6	288	$4.147 \cdot 10^8$	1.44	87 ×	62627	3090	1428	154
9	272	$2.764 \cdot 10^8$	1.02	123 ×	94874	4635	2142	154

Resource Usage and Runtime

EUs	$f_{\max}$ (MHz)	CCs required	Run time (s)	Speedup	CLB	DSP	BRAM	URAM
1	370	$2.488 \cdot 10^9$	6.73	18 ×	10196	515	238	154
2	336	$1.244 \cdot 10^9$	3.70	34 ×	21384	1030	476	154
4	335	$6.220 \cdot 10^8$	1.85	68 ×	41654	2060	952	154
6	288	$4.147 \cdot 10^8$	1.44	87 ×	62627	3090	1428	154
9	272	$2.764 \cdot 10^8$	1.02	123 ×	94874	4635	2142	154

Resource Usage and Runtime

EUs	$f_{\max}$ (MHz)	CCs required	Run time (s)	Speedup	CLB	DSP	BRAM	URAM
1	370	$2.488 \cdot 10^9$	6.73	18 ×	10196	515	238	154
2	336	$1.244 \cdot 10^9$	3.70	34 ×	21384	1030	476	154
4	335	$6.220 \cdot 10^8$	1.85	68 ×	41654	2060	952	154
6	288	$4.147 \cdot 10^8$	1.44	87 ×	62627	3090	1428	154
9	272	$2.764 \cdot 10^8$	1.02	123 ×	94874	4635	2142	154

Single threaded Xeon W-1250: **126 seconds**

Resource Usage and Runtime

EUs	$f_{\max}$ (MHz)	CCs required	Run time (s)	Speedup	CLB	DSP	BRAM	URAM
1	370	$2.488 \cdot 10^9$	6.73	18 ×	10196	515	238	154
2	336	$1.244 \cdot 10^9$	3.70	34 ×	21384	1030	476	154
4	335	$6.220 \cdot 10^8$	1.85	68 ×	41654	2060	952	154
6	288	$4.147 \cdot 10^8$	1.44	87 ×	62627	3090	1428	154
9	272	$2.764 \cdot 10^8$	1.02	123 ×	94874	4635	2142	154

Single threaded Xeon W-1250: **126 seconds**

Resource Usage and Runtime

EUs	$f_{\max}$ (MHz)	CCs required	Run time (s)	Speedup	CLB	DSP	BRAM	URAM
1	370	$2.488 \cdot 10^9$	6.73	18 ×	10196	515	238	154
2	336	$1.244 \cdot 10^9$	3.70	34 ×	21384	1030	476	154
4	335	$6.220 \cdot 10^8$	1.85	68 ×	41654	2060	952	154
6	288	$4.147 \cdot 10^8$	1.44	87 ×	62627	3090	1428	154
9	272	$2.764 \cdot 10^8$	1.02	123 ×	94874	4635	2142	154

- CLB, DSP, BRAM increase linearly
- URAM only stores final cost-to-go map
- Vehicle model dominates resource usage and $f_{\max}$

Conclusions & Contributions

- **Scalable** architecture for hybrid electric vehicle optimization
- Enabling **real-time** use
- $> 100 \times$ **speedup** over CPU implementation

frans.skarman@liu.se

LINKÖPING UNIVERSITY