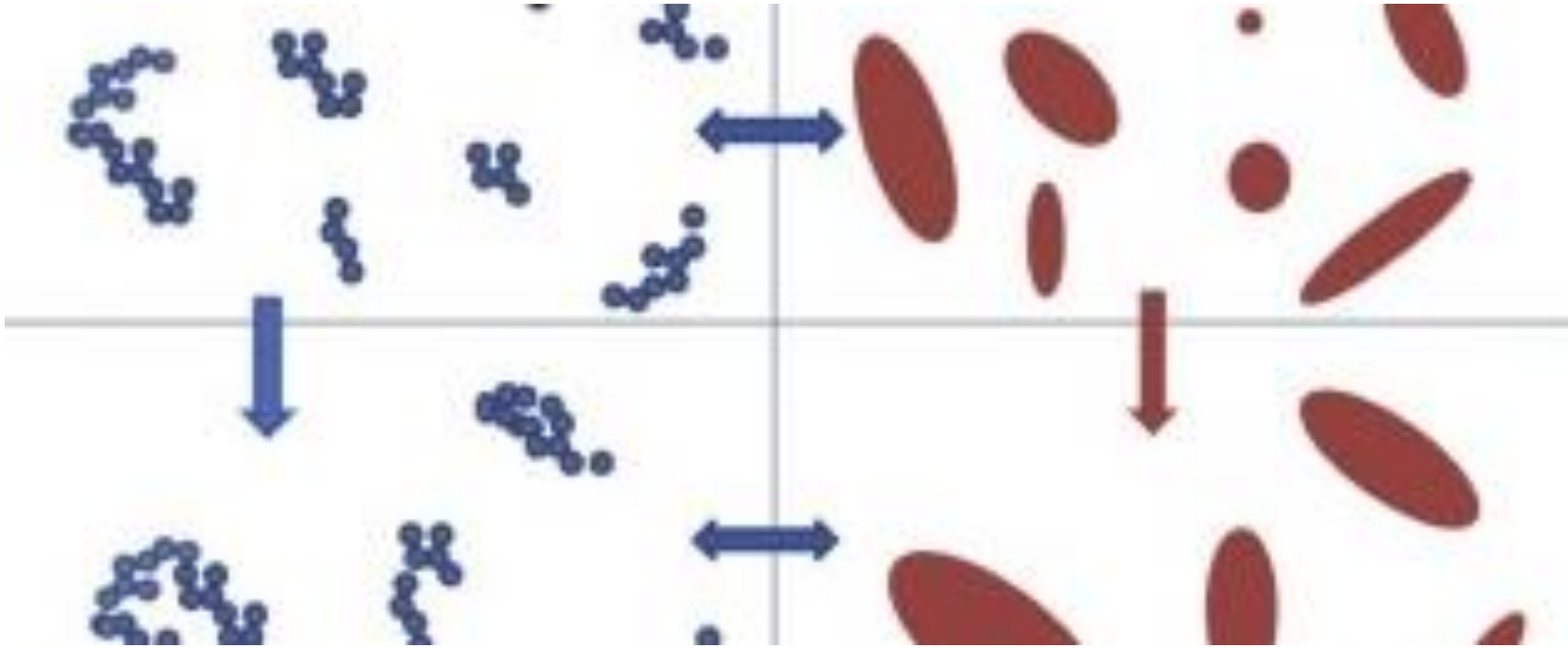
Cuarta Parte:
Clase 5 – VFA – *Lookup Tables* 2

Optimización Dinámica - ICS



Mathias Klapp

Desafíos de AVI

1. ¿Cómo **calibar el paso α_n** del suavizamiento exponencial?

2. Como fue presentado, **no garantiza convergencia al Q óptimo**

3. Aceleración de ajuste (puede ser lento calibrando)
   - Backward-pass

3. Mantiene la maldición de decisiones y la de estados
   - Agregación del espacio de estados.
   - Heuristicas de decisión

4. ¿Problemas con estados continuos o una grilla muy fina?
   - Dynamic *look-up* table.

- Backward-pass
- Agregación de estados
- Dynamic *look-up* tables

Algoritmo *AVI* (version estándar)

1. Iniciar: $n \leftarrow 1$, $\bar{Q}_t^0(y_t)$ para todo t y y_t
2. Mientras $n \leq N$:
   a. Iniciar corrida: $s_1^n \leftarrow s_1$
   b. Para cada $t = 1, \ldots, T$:

      Optimizar:

      $$x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{r_t(s_t^n, x) + \bar{Q}_t^{n-1}(y_t(s_t^n, x))\},$$
      $$y_t^n \leftarrow y_t(s_t^n, x_t^n),$$
      $$q_t^n \leftarrow r_t(s_t^n, x_t^n) + \bar{Q}_t^{n-1}(y_t^n).$$

      Actualizar $\boldsymbol{Q}$:

      $$\bar{Q}_{t-1}^n(y_{t-1}^n) \leftarrow (1 - \alpha_n)\bar{Q}_{t-1}^{n-1}(y_{t-1}^n) + \alpha_n q_t^n$$

      Simular transición a próximo estado:

      $$s_{t+1}^n \leftarrow f(y_t^n, \omega_t^n)$$

   c. $n = n + 1$
3. Retornar $\bar{Q}_t^N$

Optimización determinística

Estadística

Simulación

AVI equipado con *backward-pass*

AVI puede ser lento y local para renovar sus estimaciones.

Consideremos la recalibración en la corrida n y etapa t:

$$\bar{Q}_{t-1}^{n}(y_{t-1}^{n}) \leftarrow (1-\alpha_n)\bar{Q}_{t-1}^{n-1}(y_{t-1}^{n}) + \alpha_n q_t^n$$

La aproximación del estado de post-decisión y_{t-1}^{n} no se propaga a etapas anteriores (``hacia atrás'') y require de varias iteraciones para afectar el *value-to-go* de las etapas $k < t$.

Arreglo: *backpropagation through time.* Hacer actualización ``hacia atrás'' por la trayectoria simulada hasta $t = 1$.

Algoritmo AVI equipado con *backward-pass*

1. Iniciar: $n \leftarrow 1$, $\bar{Q}_t^0(y_t)$ para todo t y y_t
2. Mientras $n \leq N$:
   a. Iniciar corrida: $s_1^n \leftarrow s_1$
   b. **Forward-pass:** Para cada $t = 1, \dots, T$:
   $$x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{r_t(s_t^n, x) + \bar{Q}_t^{n-1}(y_t(s_t^n, x))\},$$
   $$y_t^n \leftarrow y_t(s_t^n, x_t^n),$$
   $$s_{t+1}^n \leftarrow f(y_t^n, \omega_t^n).$$
   c. **Backward-pass:**
   $$q_{T+1}^n = 0$$
   Para cada $t = T, T-1 \dots, 2$:
   $$q_t^n \leftarrow r_t(s_t^n, x_t^n) + q_{t+1}^n$$
   $$\bar{Q}_{t-1}^n(y_{t-1}^n) \leftarrow (1 - \alpha_n)\bar{Q}_{t-1}^{n-1}(y_{t-1}^n) + \alpha_n q_t^n$$
   d. $n = n + 1$
3. Retornar $\bar{Q}_t^N$

- Backward-pass
- Agregación de estados
- Dynamic *look-up* tables

Agregación de estados

Si hay maldición de estados, se requiere un número explosivo de simulaciones para recorrerlos todos varias veces y hacer estadística.

Arreglo: Estimar $\bar{Q}$ sobre **super-estados** que representen varios estados de similar value-*to-go*.

Pasos:

1. Definir una función $G_t: \mathbb{Y}_t \rightarrow \mathbb{G}_t$ que mapea cada estado de postdecisión a un espacio de super-estados con $|\mathbb{G}_t| \ll |\mathbb{Y}_t|$ estados postdecisión.
2. Luego, estimar valor $Q_t^G(g_t)$ para cada super-estado $g_t \in \mathbb{G}_t$.

Ejemplo: Gestión de recursos móviles:

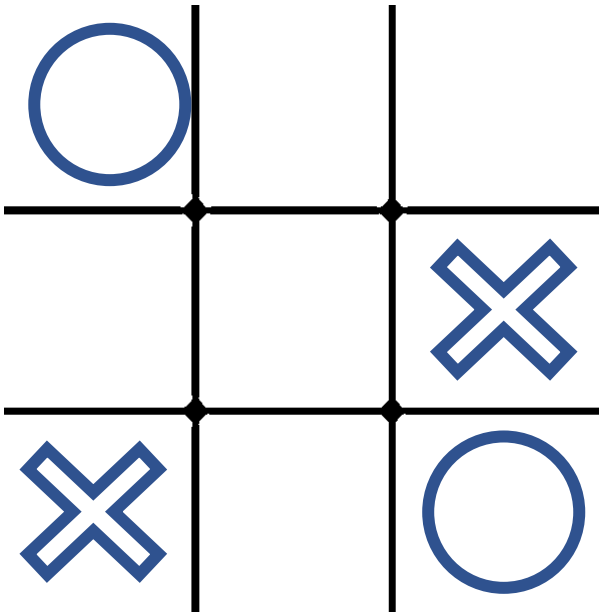
Posibles agregaciones:

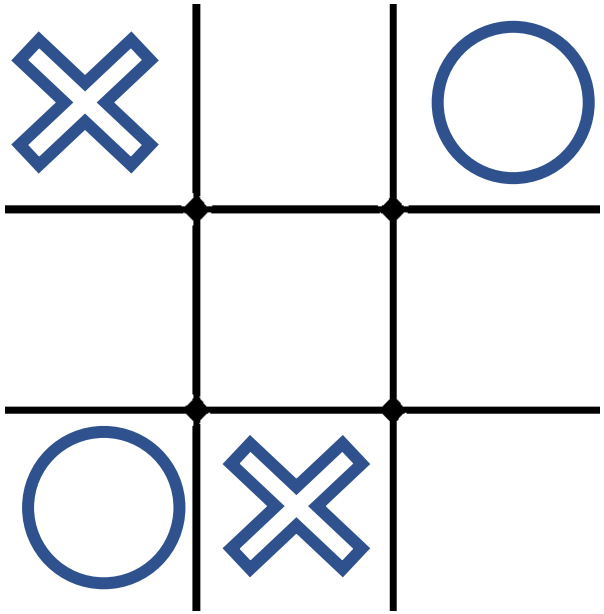
Aggregation Level	Location	Fleet Type	Domicile	Size of State Space
0	Sub-region	Fleet	Region	$400 \times 5 \times 100 = 200{,}000$
1	Region	Fleet	Region	$100 \times 5 \times 100 = 50{,}000$
2	Region	Fleet	Zone	$100 \times 5 \times 10 = 5{,}000$
3	Region	Fleet	—	$100 \times 5 \times 1 = 500$
4	Zone	—	—	$10 \times 1 \times 1 = 10$

Referencia: Warren Powell

Ejemplo: TIC-TAC-TOE (Gato)

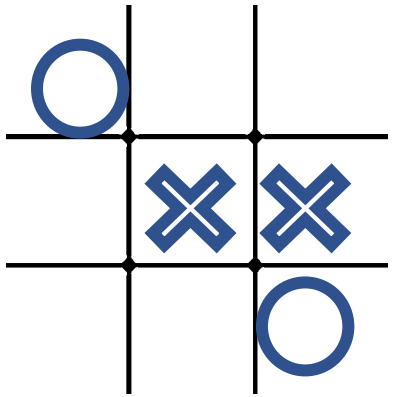
- El tablero posee $3^9 = 19{,}683$ estados.
- Uno decide las X. El juego del contrincante (O) se modela como una transición estocástica a otro estado dependiente del estado de postdecisión.
- Se calibra Q mediante simulación (``jugando'').
- Problema: Hay simetría. Estos 2 estados poseen mismo value-to-go:



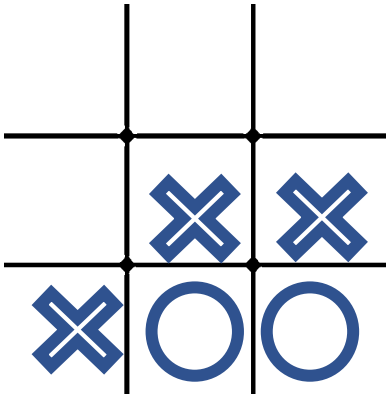


Una forma de agregación a 30 estado

- $:x \in \{0,1\}$: si centro está ocupado por X.
- $y \in \{0,1,2,3,4\}$: # de X en celdas esquina.
- $z \in \{0,1,\geq 2\}$: cantidad de opciones para hacer 3 en línea.

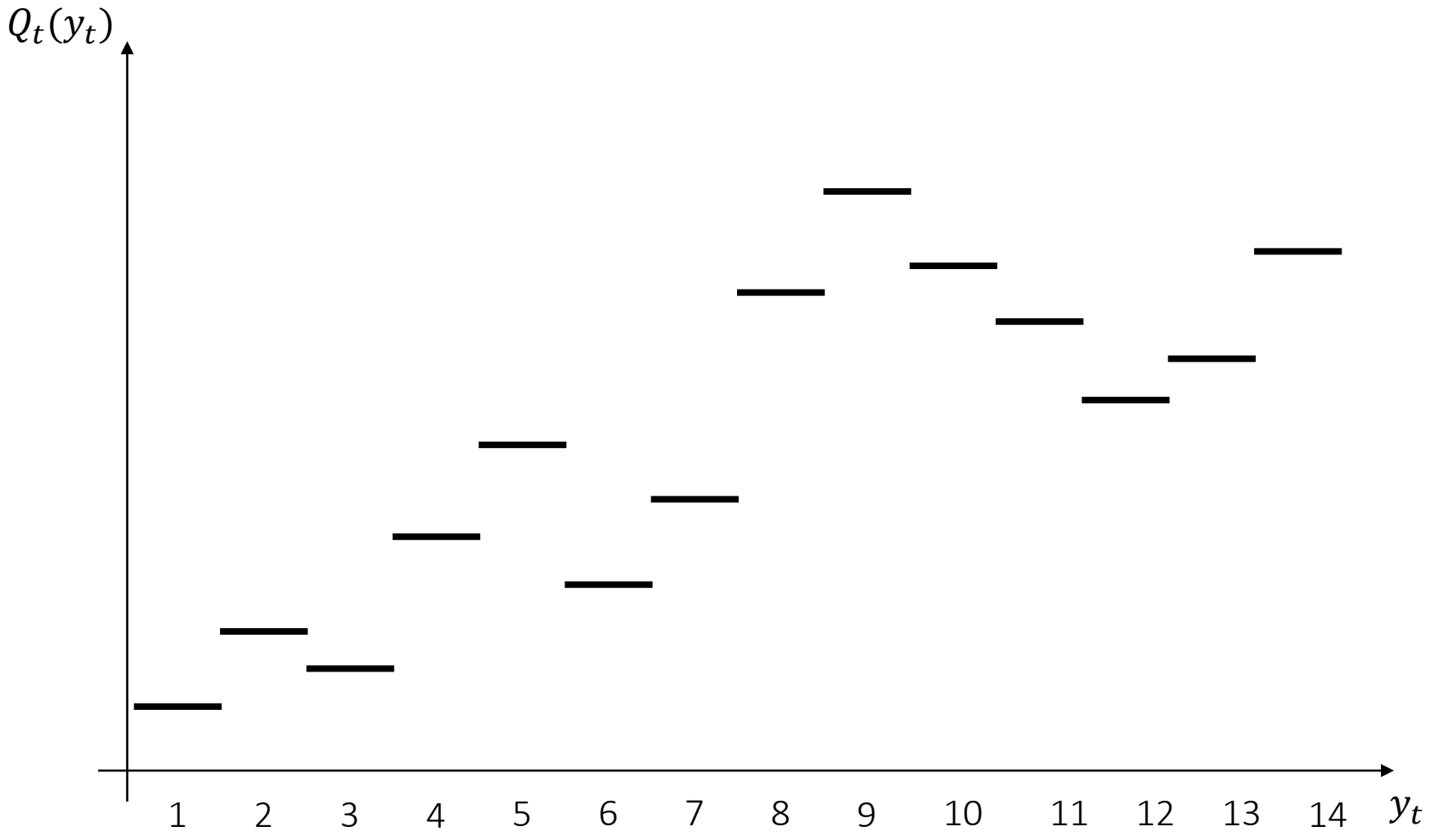


$(1,0,1)$



$(1,1,2)$

Agregación de estados:

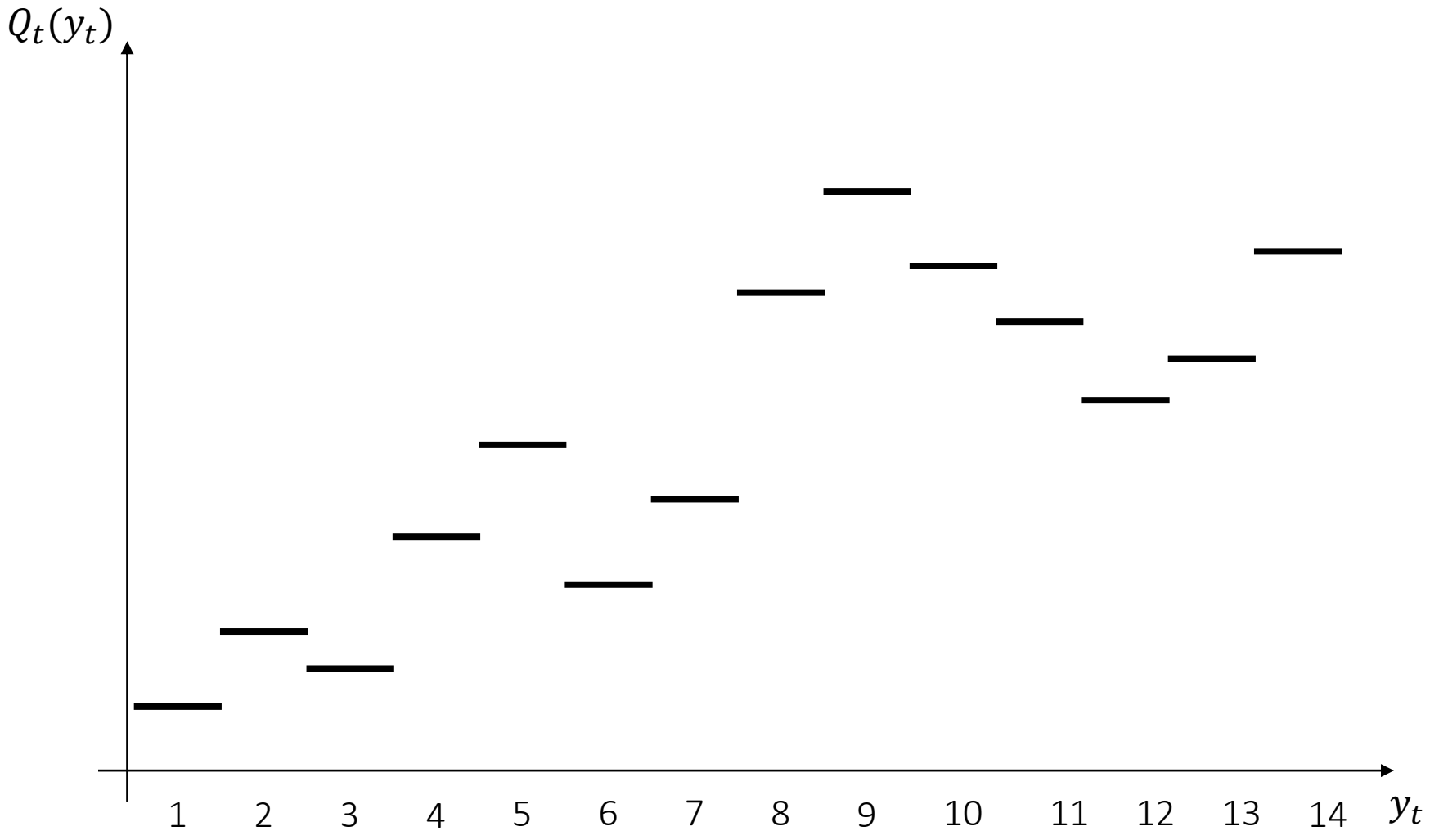


Agregación de estados:

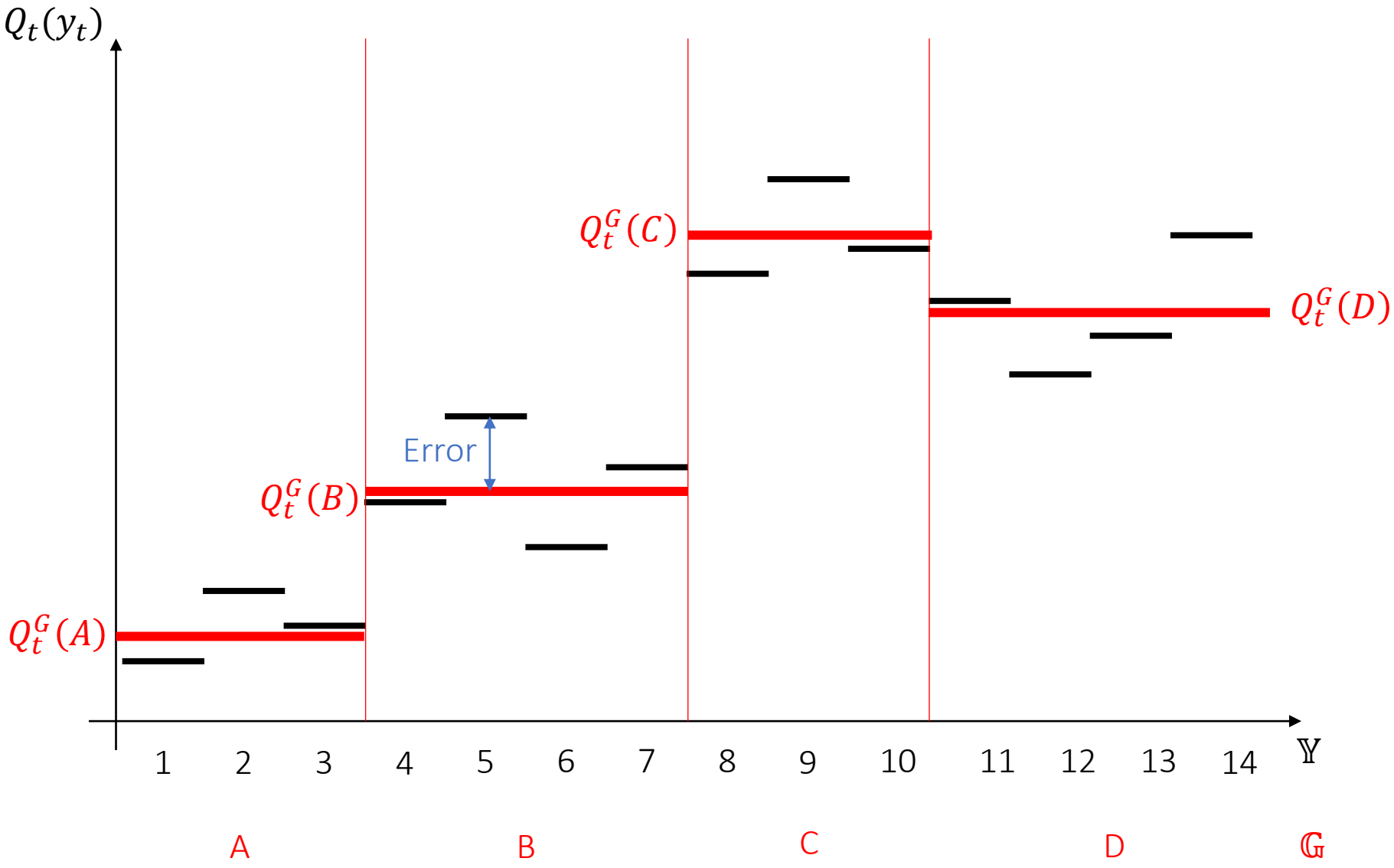
Agregación:

$$G(y) = \begin{cases} A & y \in \{1,2,3\} \\ B & y \in \{4,5,6,7\} \\ C & y \in \{8,9,10\} \\ D & y \in \{11,12,13,14\} \end{cases}$$

Agregación de estados:



Agregación de estados:



Algoritmo *AVI* + agregación de estados

1. Iniciar: $n \leftarrow 1$, $\bar{Q}_t^{G,0}$ para todo t y g_t
2. Mientras $n \leq N$:
   a. Iniciar estado inicial $s_1^n \leftarrow s_1$
   b. Para cada $t = 1, \dots, T$:
   - $x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{r_t(s_t^n, x) + \bar{Q}_t^{G,n-1}(G_t(y_t(s_t^n, x)))\}$,
   - $y_t^n \leftarrow y_t(s_t^n, x_t^n)$,
   - $g_t^n \leftarrow G_t(y_t^n)$,
   - $q_t^n \leftarrow r_t(s_t^n, x_t^n) + \bar{Q}_t^{G,n-1}(g_t^n)$.
   - $\bar{Q}_{t-1}^{G,n}(g_{t-1}^n) \leftarrow (1 - \alpha_n)\bar{Q}_{t-1}^{G,n-1}(g_{t-1}^n) + \alpha_n q_t^n$
   - $s_{t+1}^n \leftarrow f(y_t^n, \omega_t^n)$

   c. $n++$
3. Retornar $\bar{Q}_t^{G,N}$

Ejecución posterior:

- Una vez estimada la aproximación $\bar{Q}_t^{\mathrm{G}}(g_t)$ se puede ejecutar el MDP.

- **Política de decisión con agregación de estados** toma decisión en etapa t y estado s_t mediante:

$$d_t^{VFA}(s_t) \in \underset{x \in \mathbb{X}_t(s_t)}{\operatorname{argmax}} \left\{ r_t(s_t, x) + \bar{Q}_t^{\mathrm{G}}\left(G_t(y_t(s_t, x))\right) \right\}$$

, donde $\bar{Q}_t^{\mathrm{G}}\left(G_t(y_t)\right)$ aproxima a $Q_t(y_t)$.

- Nota: El paso de simulación no se ve afectado por la agregación de estados (solo la calibración y la optimización).

Agregación multicapa

Básicamente, trabajar con dos o más niveles de agregación a la vez.

G_t^1: nivel agregado

G_t^2: nivel más desagragdo

Luego estimar:

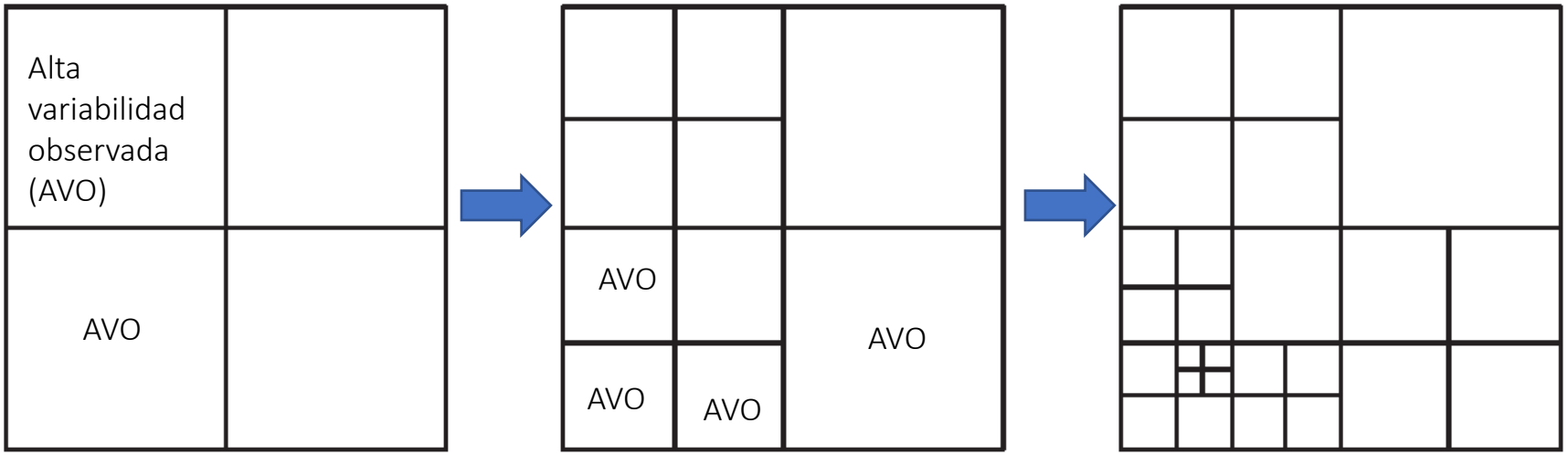
$$Q_t(y_t) \approx \beta_n \bar{Q}^{G^1}\left(G_t^1(y_t)\right) + (1-\beta_n)\bar{Q}^{G^2}(G_t^2(y_t))$$

- Ir reduciendo el peso del nivel más agregado a medida que se aprende más de Q.

- Backward-pass
- Agregación de estados
- Dynamic *look-up* tables

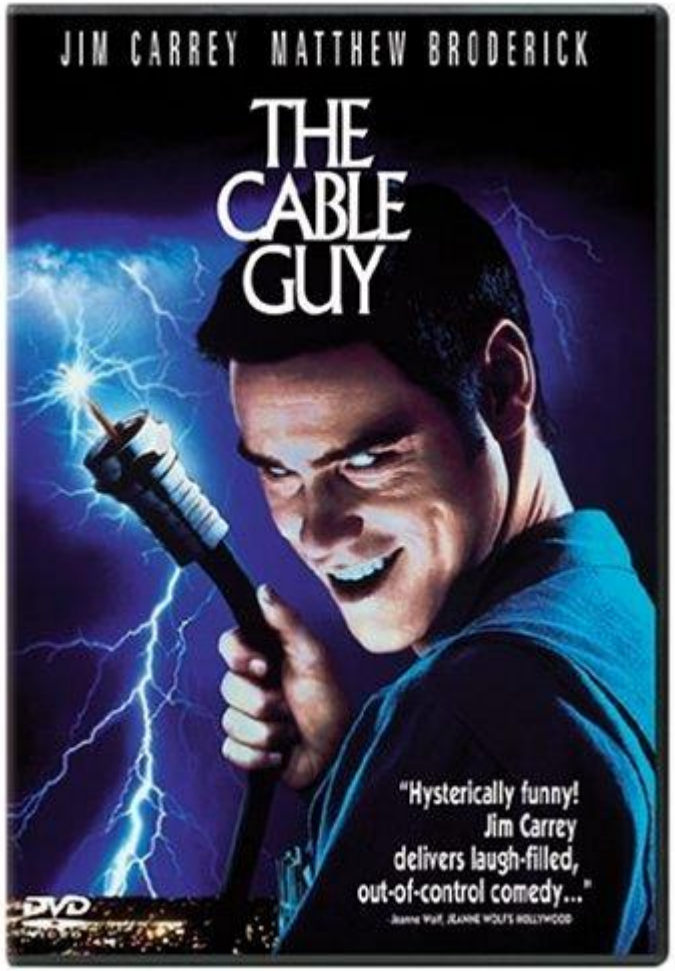
Dynamic Look-up table: (Ulmer *et al.*, 2017)

- Es una regla dinámica de agregación de estados.
- Permite trabajar con espacios de estado continuos o grillas discretas muy finas.
- Busca particionar la tabla desagregando estados a medida que los datos lo sugieren en función de la variabilidad observada de la estimación.



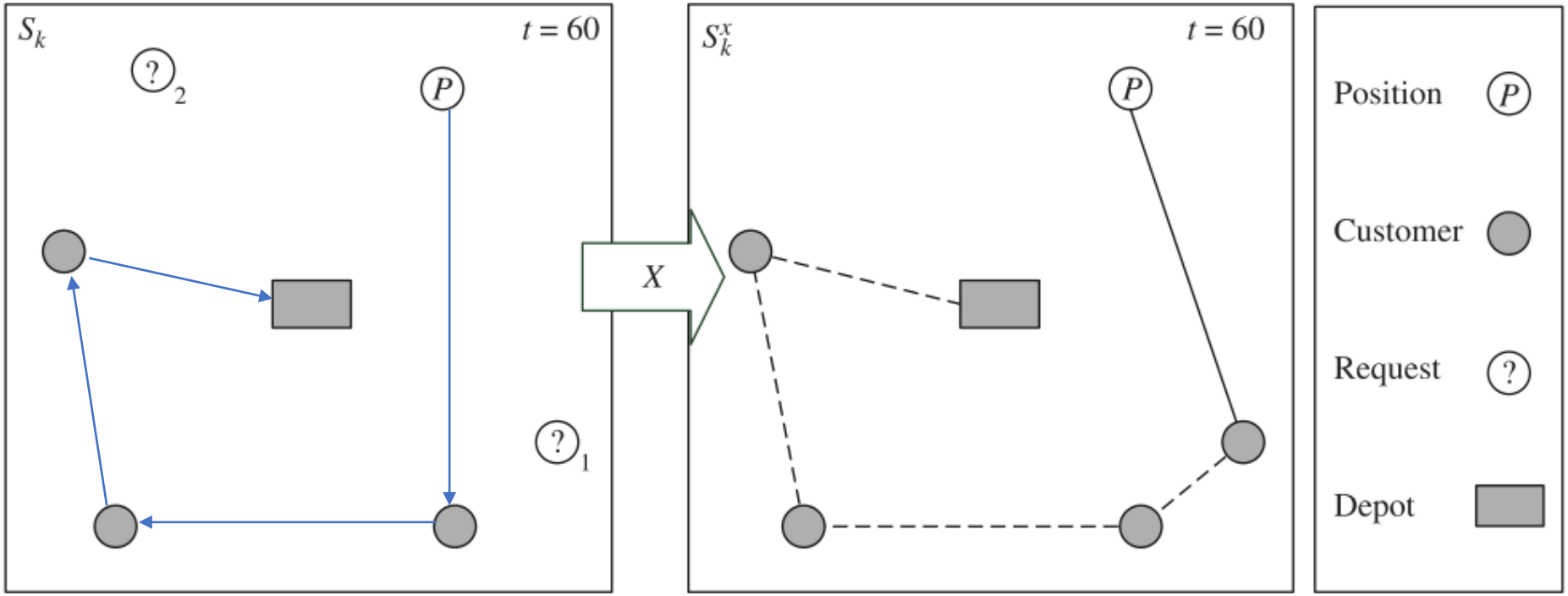
Agendamiento dinámico de visitas (Ulmer, 2018)

- Técnico visita clientes durante un día. Tiempos de viaje y servicio conocidos.
- Solicitud del servicio es dinámica.
- Efecto: Cada vez que técnico se despide de un cliente ya atendido, revisa el sistema y aparecen nuevas peticiones de servicio.
- Acción:
    1. Aceptar o rechazar cada petición de forma inmediata. Si acepta, la debe visitar, congestiona su agenda.
    2. Podría reconfigurar su futura ruta todavía no ejecutada.
- Objetivo: maximizar # de servicios.



Agendamiento dinámico de visitas

Dinámica del sistema, pre y post decisión



Agendamiento dinámico de visitas

Estado de post-decisión en tiempo $t \in [0, T]$:

- Posición actual $j \in N$
- $\rho = \{j, p_1, p_2, \dots, p_i, \dots, c\}$: ruta desde j que cubre pendientes hasta nodo terminal (c).

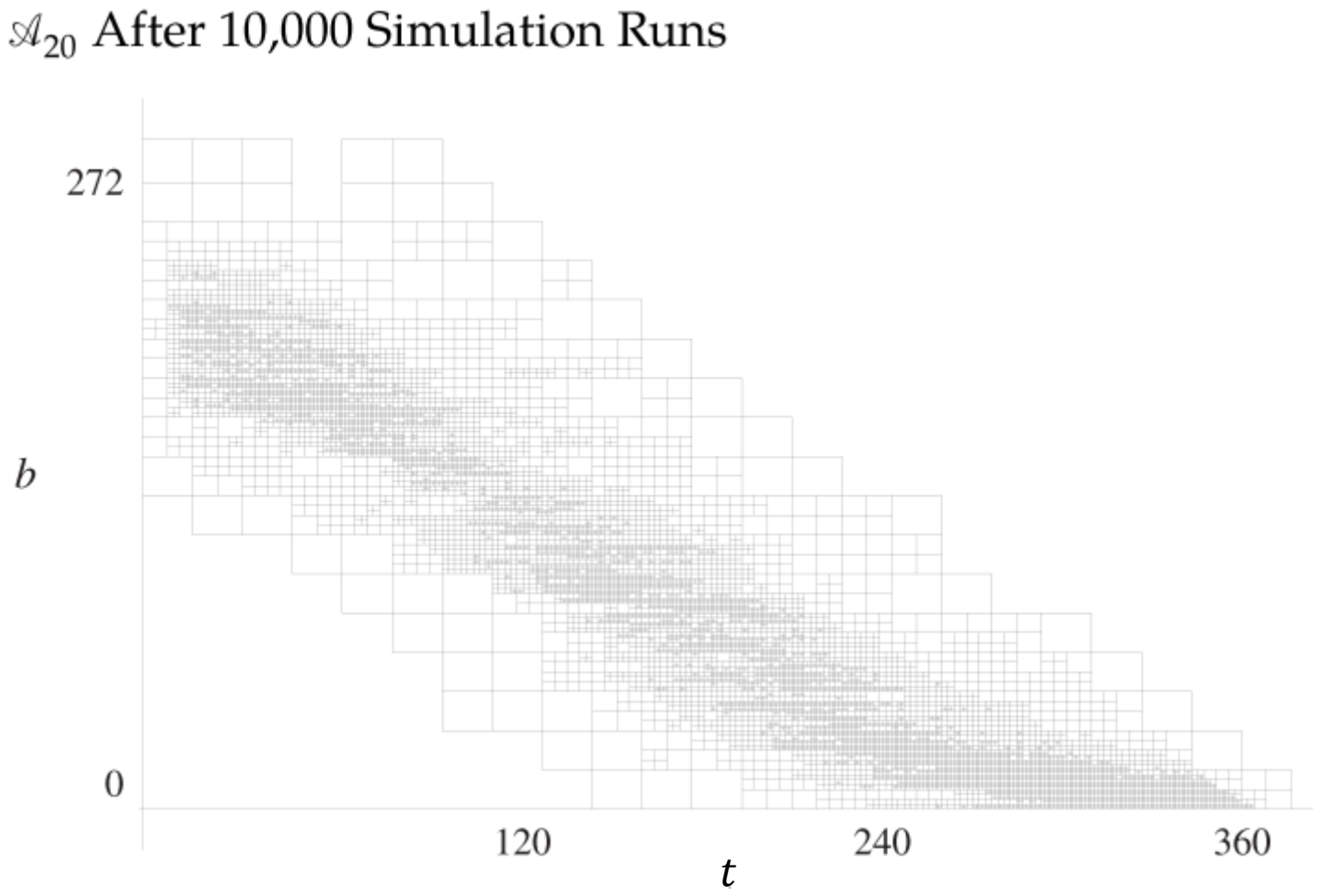
- Agregación trabajada:

$$Q_t(j, \rho) \approx Q_t^G(b)$$

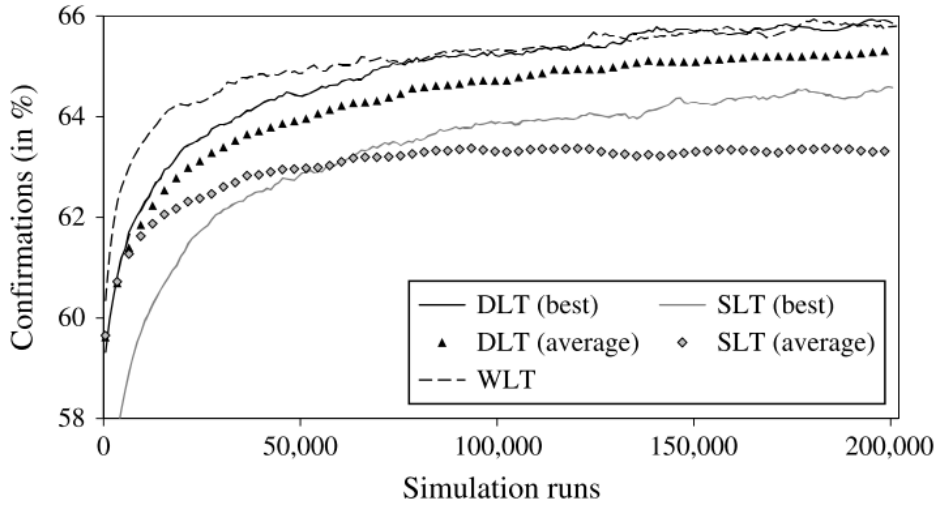
  - $b(\rho) = T - t - \bar{d}(\rho)$: Presupuesto de tiempo libre a futuro.

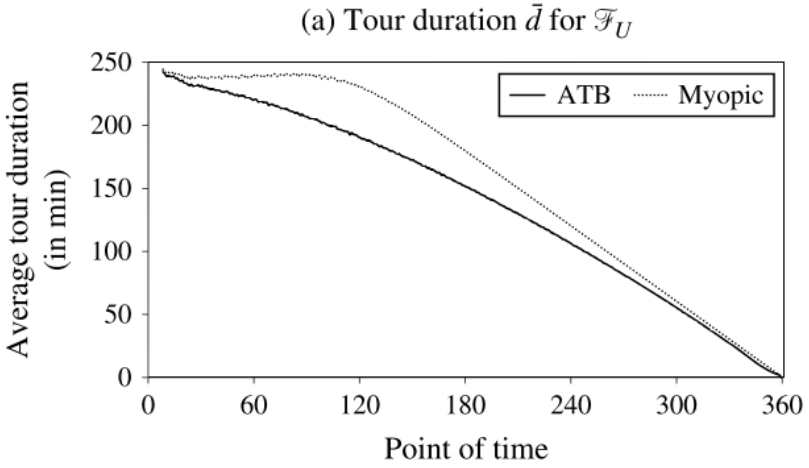
Agendamiento dinámico de visitas

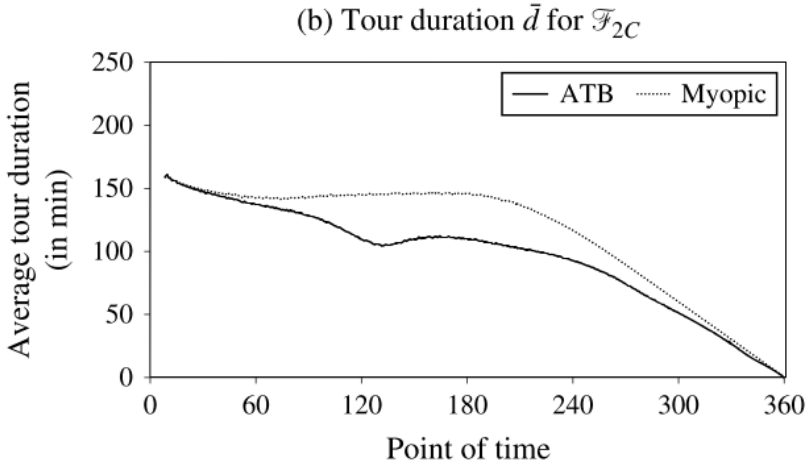
Calibración de Dynamic Lookup Table:



Agendamiento dinámico de visitas







Agendamiento dinámico de visitas

TRANSPORTATION SCIENCE

JOURNAL HOME ARTICLES IN ADVANCE CURRENT ISSUE ARCHIVES ABOUT SUBMIT

View PDF

Tools Share

Home > Transportation Science > Vol. 52, No. 1 >

Budgeting Time for Dynamic Vehicle Routing with Stochastic Customer Requests

Marlin W. Ulmer, Dirk C. Mattfeld, Felix Köster

Published Online: 24 Mar 2017 | https://doi.org/10.1287/trsc.2016.0719

VFA con *Lookup* table

Ventaja:

- Reduce esfuerzo *online.* Solo requiere resolver el problema de decisión.
- Proactivo, *feedforward* basado en entrenamiento.
- Explota viejos conocidos: simulación computacional, ajuste estadístico y optimización determinística.

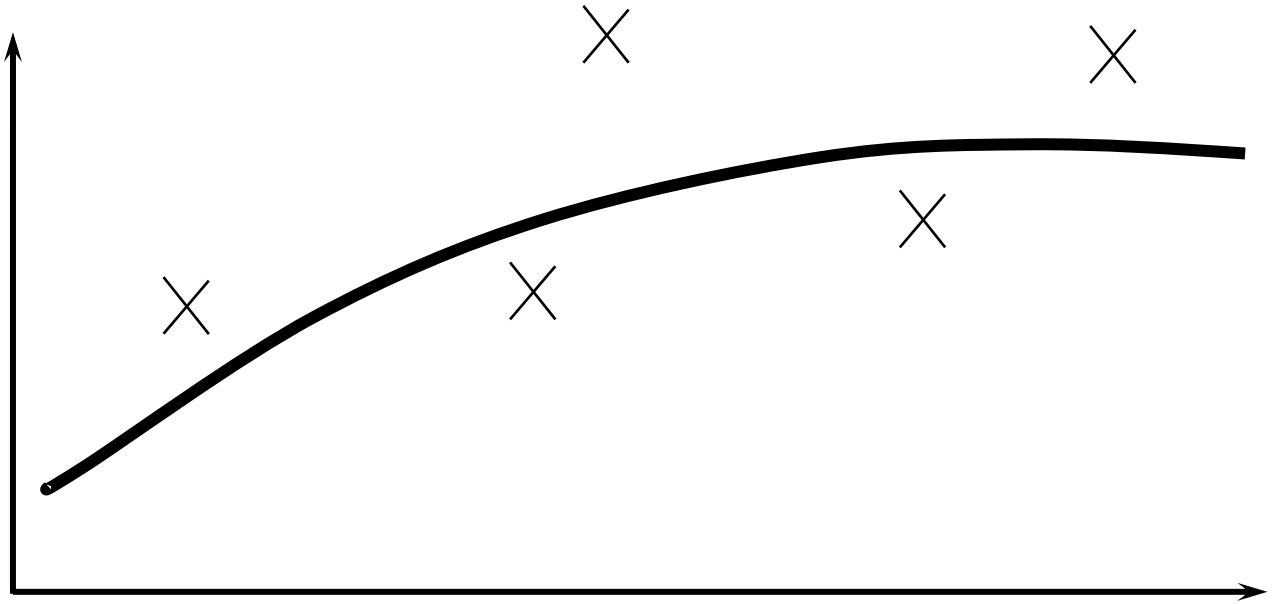
Desventaja:

- Depende del problema, de los datos y de la estimación de partida.
- Entrenamiento puede ser caro y largo.
- No más allá de las 4 o 5 dimensiones en el espacio de estados.
- No explota estructura del value-to-go.
- *Overfitting.*

Recomendación del Chef:

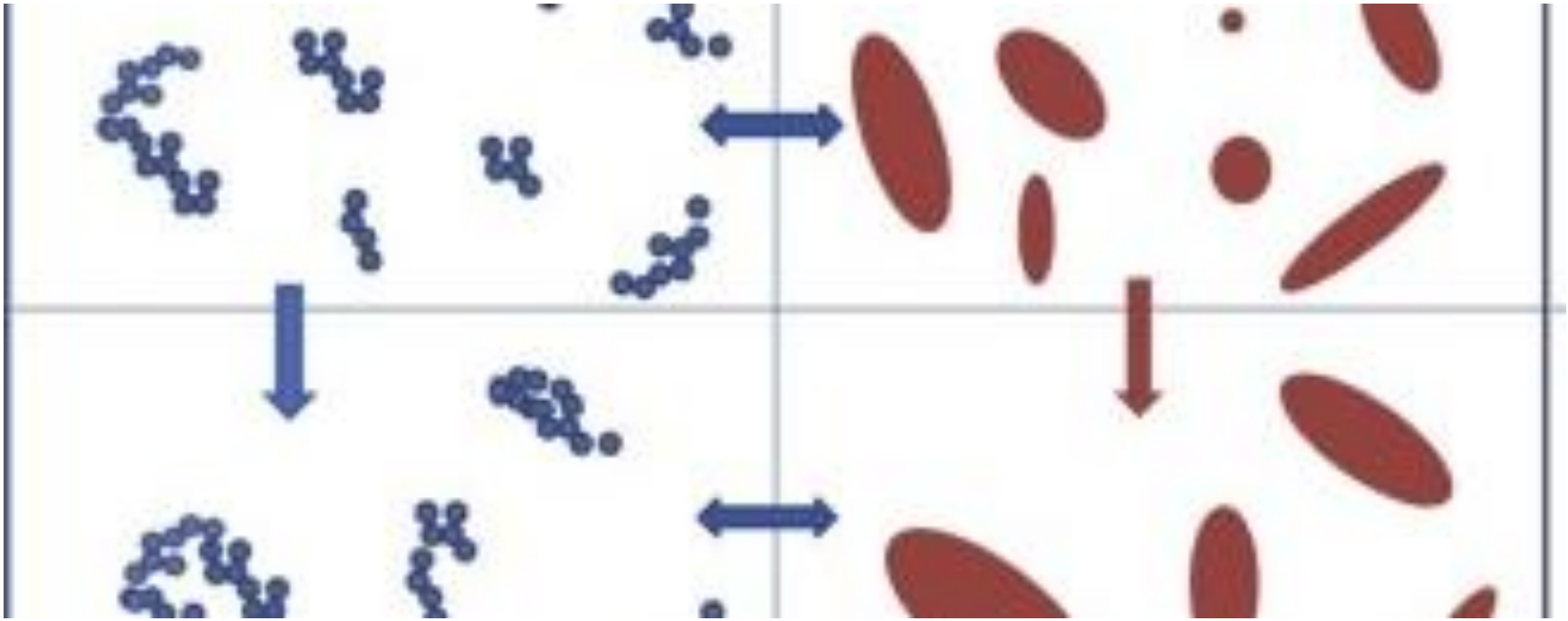
- Cuando no hay indicios de estructura predecible.
- Cuando hace sentido agregar el estado original.
- Para decisiones rápidas, cuando el cálculo *online* es caro.

Próxima clase: Aproximación paramétrica



Cuarta Parte:
Clase 5 – VFA – *Lookup Tables* 2

Optimización Dinámica - ICS



Mathias Klapp