Cuarta Parte:
Clase 2 – Decisiones Miopes

Optimización Dinámica - ICS

Mathias Klapp

Política de decisión miope

- Heurística **más simple** para tomar decisiones dinámicas.

- Una política miope π^M ejecuta decisiones en cada etapa t y estado visitado s_t resolviendo:

$$d_t^M(s_t) \in \underset{x \in \mathbb{X}_t(s_t)}{\operatorname{argmax}} \{r_t(s_t, x)\}$$

- Al planificar asume que $Q_t(y) = 0$, es decir, que **no hay valor futuro;**
  - En la prácitica puede haberlo.

- No es **proactiva** contra potenciales estados futuros, pero es **reactiva** al estado actual del sistema.

Evaluación simulada de una política miope:

- **Input:** Estado inicial s_1
- **Simular m ejecuciones de π** en una muestra Ω:

  Para cada corrida $\omega \in \Omega$:

  1. Inicializar: $s \leftarrow s_1, V_\omega \leftarrow 0$
  2. Para cada etapa $t = 1, \dots, T$:
     - Calcular: $d_t^M(s_t) \underset{x \in \mathbb{X}_t(s_t)}{\operatorname{argmax}} \{r_t(s_t, x)\}$
     - Actualizar valor: $V_\omega \leftarrow V_\omega + r_t\left(s, d_t^M(s_t)\right)$
     - Actualizar estado: $s \leftarrow f_t(s, x_t, \omega_t)$
  3. Guardar indicador: $V_\omega(s_1)$

- **Estimar el valor de la política:**
  - $\bar{V}_\Omega^\pi(s_1) := \frac{1}{m} \sum_{\omega \in \Omega} V_\omega(s_1)$
  - $S_\Omega^\pi = \sqrt{\frac{1}{|\Omega|-1} \sum_{\omega \in \Omega} (V_\omega(s_1) - \bar{V}_\Omega^\pi(s_1))^2}$
  - IdC para $V^\pi(s_1)$: $\left[\bar{V}_\Omega^\pi(s_1) - S_\Omega^\pi \frac{Z_{1-\frac{\alpha}{2}}}{\sqrt{m}}; \bar{V}_\Omega^\pi(s_1) + S_\Omega^\pi \frac{Z_{1-\frac{\alpha}{2}}}{\sqrt{m}}\right]$

Acerca de decisions Miopes

- También se conoce como **políticas reactivas** o de ***screenshot policy***, pues resuelve la ``foto visible'' del sistema.

- Solución simple y ampliamente utilizada porque:
  1. **No requiere información probabilística**.
  2. **Usa lo disponible**: Optimización determinística. Decisiones en dominios $\mathbb{X}$ complejos (LP, MILP, no-lineal).
  3. A veces, ser reactivo es suficiente.

- Puede resultar bien, pero puede ser un ``desastre'':
  - El ``pan para hoy`` puede generar ``hambre para mañana''.

- Cómputo *online* recae en optimizar $r(s, x)$ sobre $x \in \mathbb{X}(s)$

Ejemplo 1: Traveling Saleman Problem (TSP)

Estado:

- S: clientes por visitar
- i: ubicación actual

Ecuaciones del Bellman para todo $S \subset \{1, \dots, n\}$:

$$Q(i,S) = \begin{cases} \min\limits_{j \in S \setminus \{1\}} \{c_{ij} + Q(j, S \setminus \{j\})\} \\ c_{i1} \qquad\qquad si\ S = \{1\} \end{cases}$$

, es decir: $$d^*(i,S) \in \underset{j \in N_i^+ \cap S}{\operatorname{argmin}} \{c_{ij} + Q(j, S \setminus \{j\})\}$$

El problema está ``maldito", pues posee $\mathcal{O}(n2^n)$ estados.

Política heurística miope: vecino más cercano

Vecino más cercano asume $Q(i,S) = 0$. Luego:

$$d^{NN}(i,S) \leftarrow \underset{j \in N_i^+ \cap S}{\operatorname{argmin}}\{c_{ij}\}$$

- Ventaja: Simple

- Desventaja: No es proactivo con la distancia remanente del tour desde el vecino j en adelante.

Evaluación de la heurística:

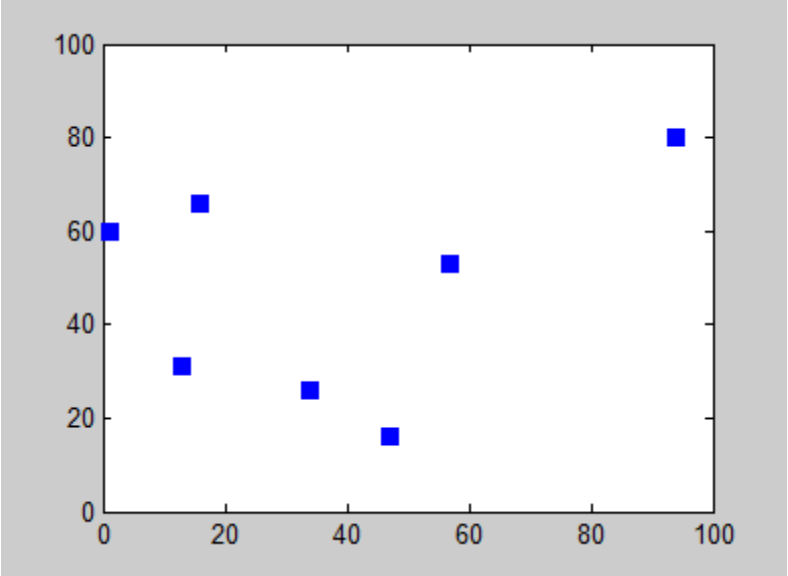
1. Inicializar: $i \leftarrow 1, S \leftarrow N \backslash \{1\}$

2. Para cada etapa $t = 1, \dots, n-1$:
   - Calcular: $j \in \underset{j \in N_i^+ \cap S}{\operatorname{argmin}} \{c_{ij}\}$
   - Actualizar valor: $Q \leftarrow Q + c_{ij}$
   - Actualizar estado: $(i, S) \leftarrow (j, S \backslash \{j\})$

3. Cierre del tour: $Q \leftarrow Q + c_{i1}$

Ejecuta en $\mathcal{O}(n^2)$ operaciones.

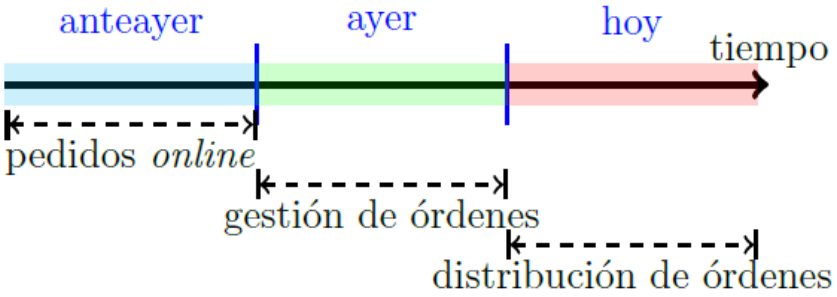
Ejemplo: DP del vendedor viajero



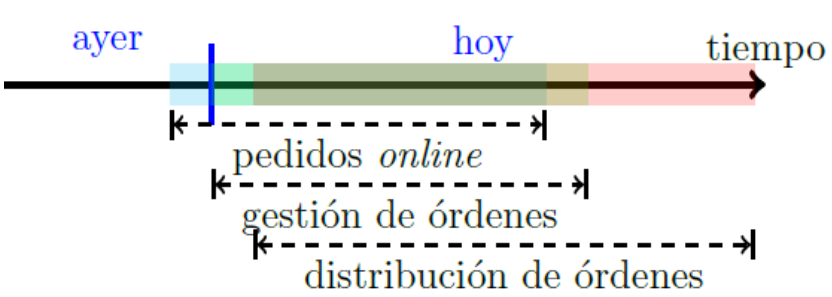
Ejemplo 2: Same-day delivery (SDD)

- *Two-day delivery* Compras online hoy, recibes pasado-mañana
- *Next-day delivery:* Compras online hoy, recibes mañana
- *Same-day delivery:* Pides online hoy, recibes hoy

Problema de SDD: Ruteo es dinámico.



(a) Entrega en dos días



(b) Same-day delivery

Dynamic Dispatch Waves Problem (DDWP)

2018 - Klapp, Erera & Toriello (European Journal of Operations Research)

Modela decisiones dinámicas de un centro despacho (CD) para same-day delivery a lo largo de un horizonte de T periodos (olas) de despacho.

- Un conjunto de N clientes arriba dinámicamente en el tiempo.
- En CD se preparan paquetes y se despachan en un grafo $([n] \cup \{0\}, A)$.
- Vehículos disponibles en el CD (uno para simplificar análisis).

- Cada cliente $i \in [n]$ posee tres características:
  1. Release date $r_i \in [T]$: Ola de despacho desde la cual el paquete es conocido y está para despacho.
  2. Tiempo de viaje c_{ij}: Tiempo que toma ir desde i a j.
  3. Penalidad p_i: Por no atender al cliente i.

- Objetivo: min $\mathbb{E}$(penalidades + costos de ruteo)

Dynamic Dispatch Waves Problem (DDWP)

- Sea R_t: conjunto de clientes pendientes para despacho desde CD en ola t.
- Decisiones cuando vehículo está disponible en el CD al inicio de la ola t:
  1. **Esperar** una ola o no.
  2. **Despachar** al vehículo y seleccionar un subconjunto $S \subseteq R_t$ de clientes a costo $\alpha \cdot T(S)$.

- Vehículo despachado en t queda inutilizado hasta retorno en $t + W(S)$, donde $W(S) = \lceil T(S)/\delta \rceil$.

Trade-offs:

1. Depachar (reducir cola) vs Esperar (acumular opciones).
2. Depachos cortos (caros y flexibles) vs largos (baratos e inflexibles).

Dynamic Dispatch Waves Problem (DDWP)

Ecuaciones de Bellman:

Para todo $t \in \{1, \dots, T-1\}$ y todo $R_t \subset N$:

$$V_t(R_t) = \min_{S \subseteq R_t \,:\, t+W(s) \leq T} \left\{ \alpha \cdot C(S) - \sum_{i \in S} p_i + \mathbb{E}\left(V_{t+W(S)}\left(R_t \backslash \mathrm{S} \cup N_{t,t+W(S)} \right) \right) \right\}$$

, donde $N_{t,t'}$ es el conjunto de órdenes reveladas entre t y t'.

Terminar: $V_T(R_t) = \sum_{i \in N_{1,T}} p_i$

¡Problema tiene **cuatro** maldiciones!

Trade-off 1:

¿Con qué frecuencia despachar a clientes que solicitan pedidos durante el día?

Despachar:

- reduce cola de pendientes, pero pierde la oportunidad de consolidar en ese vehículo potenciales pedidos futuros cercanos a los ya programados.

Esperar:

- Mejora consolidación, pero quema tiempo.

Trade-off 2:

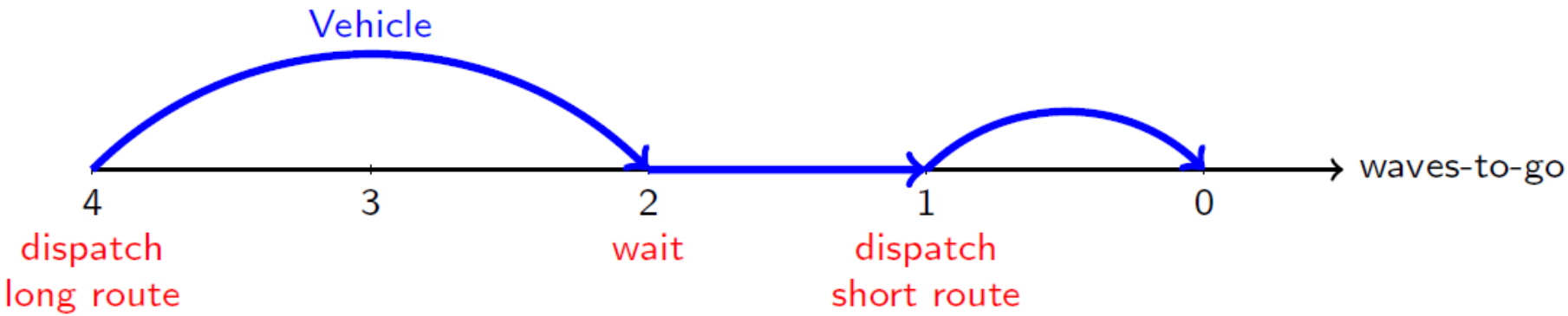
¿Rutas largas versus rutas cortas?

Ruta larga:

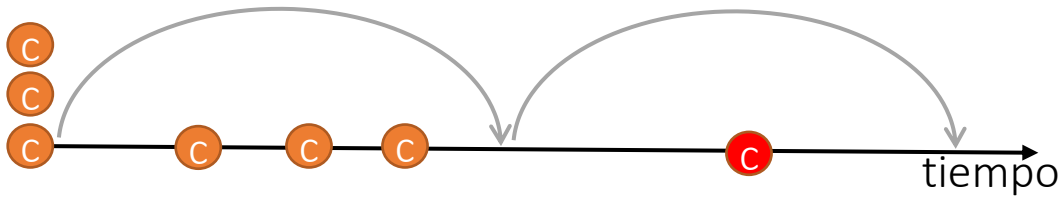
- Menor costo por orden, pero menor capacidad de reacción al postergar decisión futura.

Esperar:

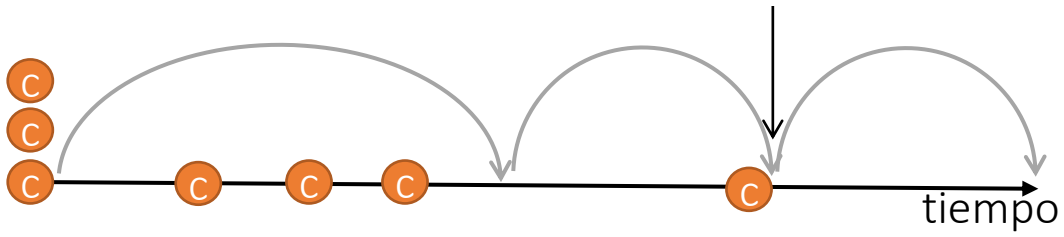
- Más cara por cliente, pero flexible (decisión futura más cerca del presente)



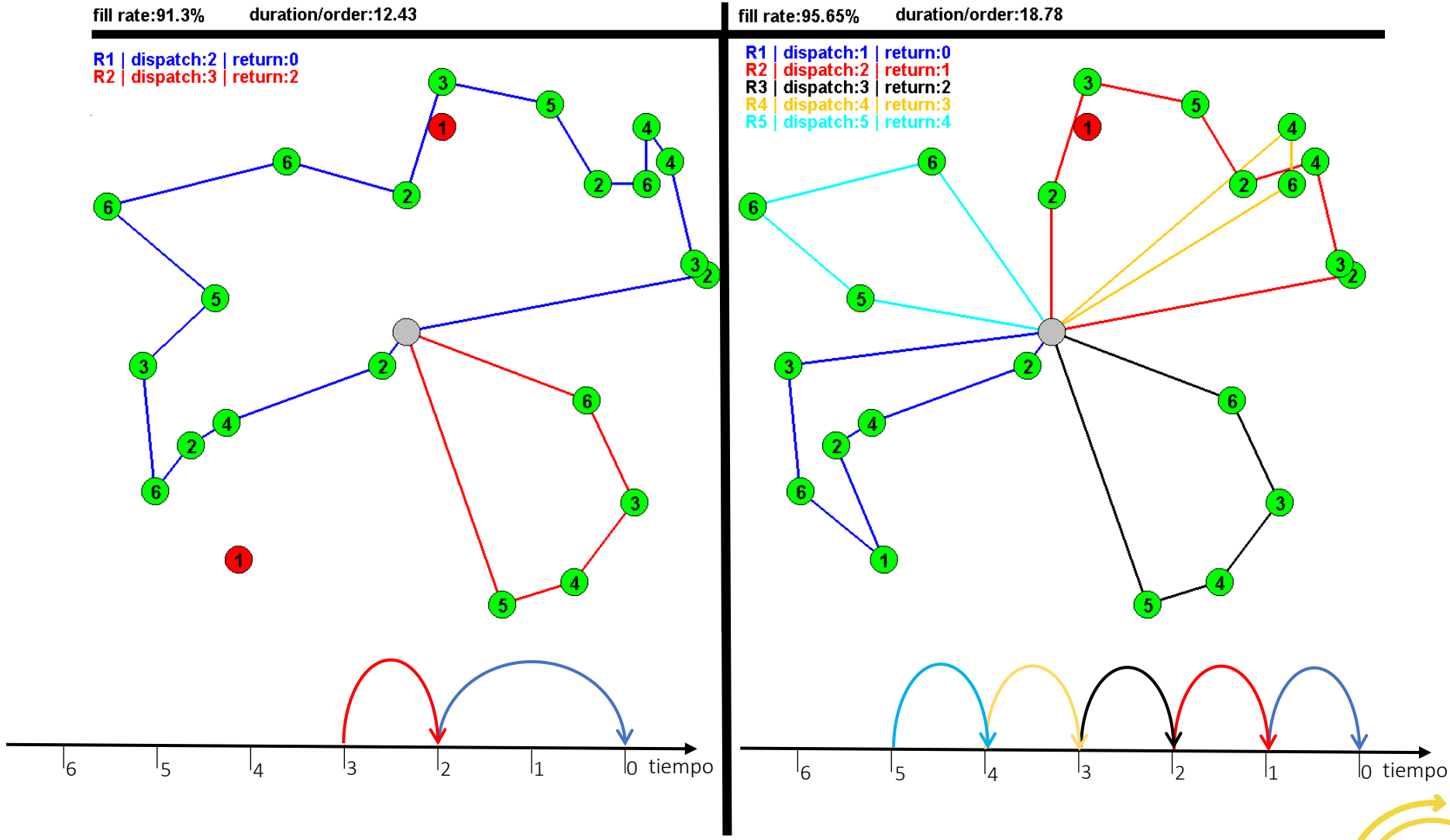
- *¿Cuántos pedidos cargar en el vehículo a despachar?*
- Rutas largas



- *¿Cuántos pedidos cargar en el vehículo a despachar?*
- Rutas largas



Trade-off 3: El peso de las penalidad:
Eficiencia operativa versus Cobertura



Dynamic Dispatch Waves Problem (DDWP)

Para todo $t \in \{1, \dots, T-1\}$ y todo $R_t \subset N$:

$$V_t(R_t) = \min_{S \subseteq R_t \,:\, t+W(s) \leq T} \left\{ \alpha \cdot C(S) - \sum_{i \in S} p_i + \mathbb{E}\left(V_{t+W(S)}\left(R_t \backslash \mathrm{S} \cup N_{t,t+W(S)} \right) \right) \right\}$$

, donde $N_{t,t'}$ es el conjunto de órdenes reveladas entre t y t'.

Enfoque miope sería omitir el costo esperado futuro y resolver:

$$d^H(R_t) = \operatorname*{argmin}_{S \subseteq R_t : W(s) \leq T-t} \left\{ \alpha \cdot C(S) - \sum_{i \in S} p_i \right\},$$

es decir, **evaluar decisión de despacho sólo considerando clientes pendientes**

- Descarta valor futuro de consolidación geográfica con potenciales clientes futuros.
- Requiere resolver un TSP con recolección de premios (prize colecting TSP).

Decisiones pseudo-miopes

- También conocidas como miopes calibradas.
- Decisiones miopes equipadas heurísticamente con parámetros en costos y espacio de decisiones que ``limitan la avaricia'' y permiten un buen comportamiento a futuro.
- Parámetros son típicamente calibrados mediante optimización-simulación.
- Muy utilizadas por su simpleza y practicidad.
- Ejemplos:
  - Decisiones de inventario con stock de seguridad predefinido.
  - Planificación de rutas aéreas con buffers de tiempo entre despegues.
  - Asignación dinámica de taxistas a pasajeros (Uber) con stock de seguridad de taxistas para el futuro.

DDWP con decisión miope calibrada

- Un enfoque miope es omitir el valor futuro y resolver:

$$d^H(R_t) = \underset{S \subseteq R_t : W(s) \leq (T-t)\cdot \gamma}{\operatorname{argmin}} \left\{ \alpha \cdot C(S) - \sum_{i \in S} \beta p_i \right\}$$

- Luego, se hace simulación computacional para evaluar $C^{MIOPE}(\beta, \gamma)$ en función de γ y β.
  - γ: evita consumir todo el tiempo de reacción a futuro
  - β: reduce o aumenta el valor relativo de cubrir a un cliente ahora ya.

- Se búsca el α y β que mejor funciona mediante optimización-simulación:

$$\min_{\beta, \gamma} \quad C^{MIOPE}(\beta, \gamma)$$

Cuarta Parte:
Clase 2 – Decisiones Miopes

Optimización Dinámica - ICS

Mathias Klapp