
# Diseño e implementación de metaheurísticas con Python

Sergio Pérez Peló

Jesús Sánchez-Oro

Universidad
Rey Juan Carlos

# Algoritmos voraces

- Muy utilizados para resolver problemas de optimización.
- Realizan la mejor elección en cada iteración.
- Esta elección no tiene por qué llevar a una solución óptima.

# Definiciones

- Conjunto de candidatos
- Función voraz
- Función de factibilidad
- Función objetivo

# Ventajas vs inconvenientes

## Ventajas

- Fáciles de implementar
- Soluciones eficientes

## Inconvenientes

- No todos los problemas admiten esta estrategia
- La búsqueda de óptimos locales no tiene por qué llevar a un óptimo global
- Lo que parece bueno ahora no tiene por qué ser bueno después

# GRASP

- **G**reedy **R**andomized **A**daptive **S**earch **P**rocedure
  - Fase de **construcción**
  - Fase de **mejora**

# GRASP

1. $CL \leftarrow \{v \in V\}$
2. $v_f \leftarrow \text{Random}(CL)$
3. $S \leftarrow \{v_f\}$
4. $CL \leftarrow CL \setminus \{v_f\}$
5. **while** $CL \neq \emptyset$ **do**
6. $\quad g_{min} \leftarrow \min_{v \in CL} g(v)$
7. $\quad g_{max} \leftarrow \max_{v \in CL} g(v)$
8. $\quad \mu \leftarrow g_{max} - \alpha \cdot (g_{max} - g_{min})$
9. $\quad RCL \leftarrow \{v \in CL : g(v) \leq \mu\}$
10. $\quad v_s \leftarrow \text{Random}(RCL)$
11. $\quad S \leftarrow S \cup \{v_s\}$
12. $\quad CL \leftarrow CL \setminus \{v_s\}$
13. **endwhile**
14. **return** $S$

# GRASP

1. $CL \leftarrow \{v \in V\}$
2. $v_f \leftarrow \text{Random}(CL)$
3. $S \leftarrow \{v_f\}$
4. $CL \leftarrow CL \setminus \{v_f\}$
5. **while** $CL \neq \emptyset$ **do**
6. $\quad g_{min} \leftarrow \min_{v \in CL} g(v)$
7. $\quad g_{max} \leftarrow \max_{v \in CL} g(v)$
8. $\quad \mu \leftarrow g_{max} - \alpha \cdot (g_{max} - g_{min})$
9. $\quad RCL \leftarrow \{v \in CL : g(v) \leq \mu\}$
10. $\quad v_s \leftarrow \text{Random}(RCL)$
11. $\quad S \leftarrow S \cup \{v_s\}$
12. $\quad CL \leftarrow CL \setminus \{v_s\}$
13. **endwhile**
14. **return** $S$

La **lista de candidatos** contiene a todos los elementos menos al primero, que se elige de manera aleatoria.

# GRASP

1. $CL \leftarrow \{v \in V\}$
2. $v_f \leftarrow \text{Random}(CL)$
3. $S \leftarrow \{v_f\}$
4. $CL \leftarrow CL \setminus \{v_f\}$
5. **while** $CL \neq \emptyset$ **do**
6. $\quad g_{min} \leftarrow \min_{v \in CL} g(v)$
7. $\quad g_{max} \leftarrow \max_{v \in CL} g(v)$
8. $\quad \mu \leftarrow g_{max} - \alpha \cdot (g_{max} - g_{min})$
9. $\quad RCL \leftarrow \{v \in CL : g(v) \leq \mu\}$
10. $\quad v_s \leftarrow \text{Random}(RCL)$
11. $\quad S \leftarrow S \cup \{v_s\}$
12. $\quad CL \leftarrow CL \setminus \{v_s\}$
13. **endwhile**
14. **return** $S$

La **lista de candidatos restringida** contiene a todos los elementos cuyo valor de una función voraz $g()$ es mejor que un cierto umbral.

# GRASP

1. $CL \leftarrow \{v \in V\}$
2. $v_f \leftarrow \text{Random}(CL)$
3. $S \leftarrow \{v_f\}$
4. $CL \leftarrow CL \setminus \{v_f\}$
5. **while** $CL \neq \emptyset$ **do**
6. $\quad g_{min} \leftarrow \min_{v \in CL} g(v)$
7. $\quad g_{max} \leftarrow \max_{v \in CL} g(v)$
8. $\quad \mu \leftarrow g_{max} - \alpha \cdot (g_{max} - g_{min})$
9. $\quad RCL \leftarrow \{v \in CL : g(v) \leq \mu\}$
10. $\quad v_s \leftarrow \text{Random}(RCL)$
11. $\quad S \leftarrow S \cup \{v_s\}$
12. $\quad CL \leftarrow CL \setminus \{v_s\}$
13. **endwhile**
14. **return** $S$

Se elige al azar un valor de la **RCL**, actualizando la lista de candidatos y la solución

# GRASP

## Fase de mejora

- Búsqueda local basada en el movimiento de **intercambio**.
- **First improvement**
  - Se aplica el primer movimiento que produzca una mejora.
- Exploración **aleatoria** de la vecindad.


# Diseño e implementación de metaheurísticas con Python

Sergio Pérez Peló

Jesús Sánchez-Oro

Universidad
Rey Juan Carlos