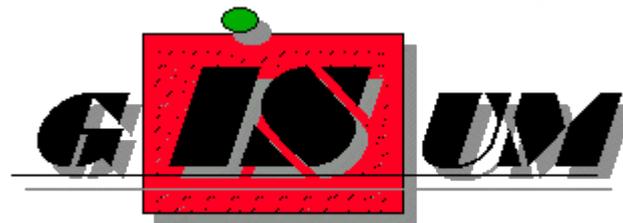




Testeo de Software con Dos Técnicas Metaheurísticas



LENGUAJES Y
CIENCIAS DE LA
COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA



Grupo de Ingeniería del Software de la Universidad de Málaga

Enrique Alba y Francisco Chicano

Introducción

Generador

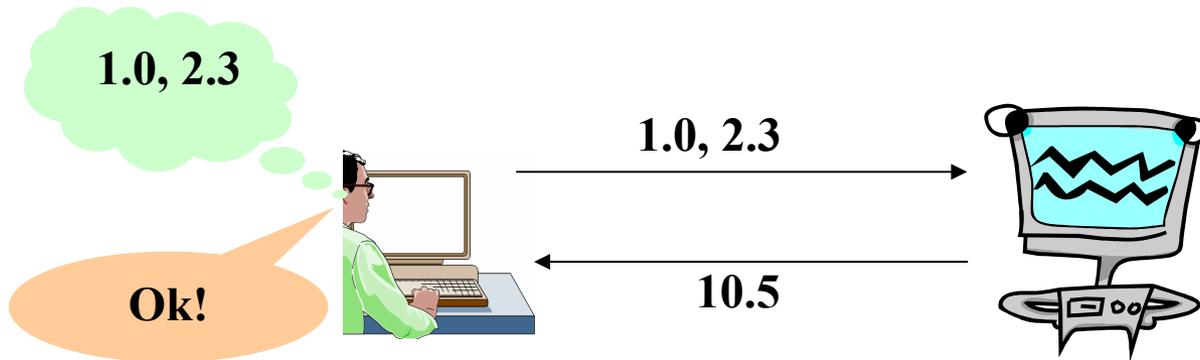
Algoritmos de
Optimización

Experimentos

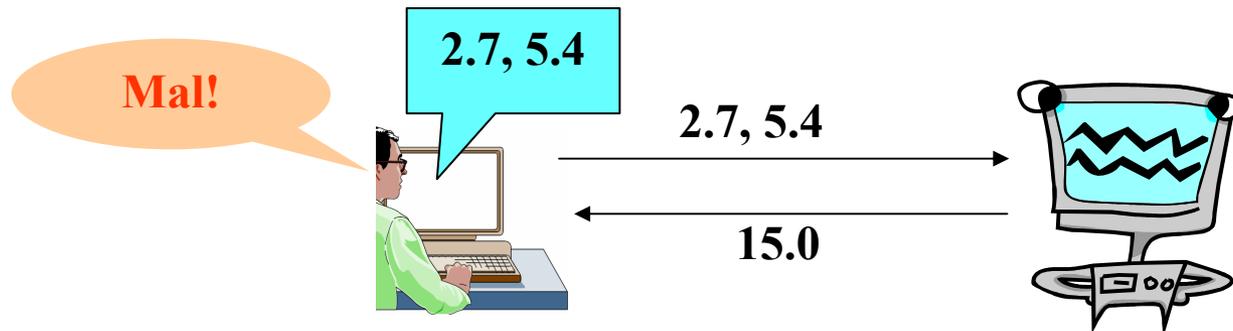
Conclusiones y
Trabajo Futuro

Introducción

- Tras la codificación, el software requiere una **fase de prueba**
- El objetivo es comprobar que el software cumple la **especificación**
- Las empresas software dedican aprox. el **50%** de recursos a dicha fase



- En este trabajo proponemos una herramienta automática basada en **Metaheurísticas para generar los casos de prueba**



Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro

Criterio de Adecuación

- Objetivo del generador de casos de prueba: proponer casos que encuentren el **máximo** número de errores en un software incorrecto



Introducción

Generador

Algoritmos de
Optimización

Experimentos

Conclusiones y
Trabajo Futuro

Criterio de Adecuación

- Objetivo del generador: **DIFÍCIL DE COMPROBAR** proponer casos que encuentren el ~~maximo~~ número de errores en un software incorrecto

Introducción

Generador

Algoritmos de Optimización

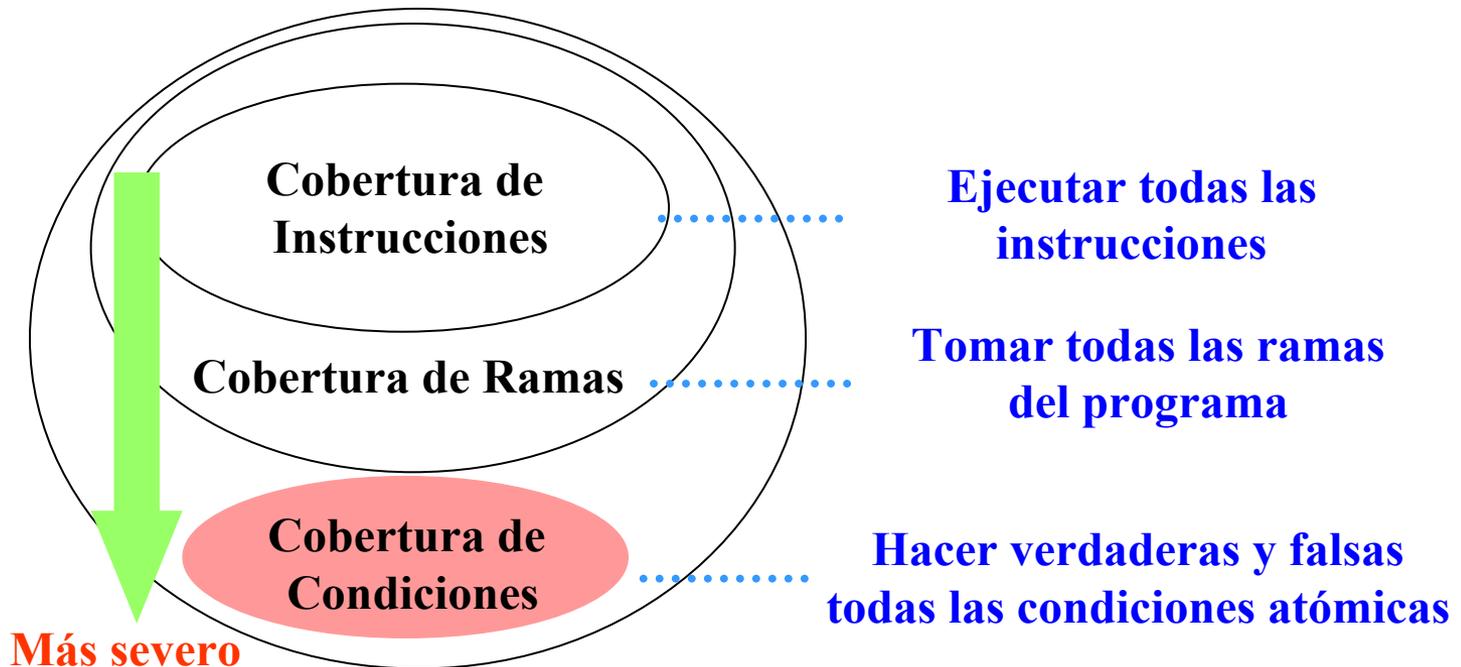
Experimentos

Conclusiones y Trabajo Futuro

Criterio de Adecuación

- Objetivo del generador de casos de prueba: **DIFÍCIL DE COMPROBAR** proponer casos que encuentren el **máximo** número de errores en un software incorrecto

- Ejemplos de **criterios de adecuación**



Paradigmas

- Tres paradigmas de generación de casos de prueba:

- Generación Aleatoria



1.2, 0.7

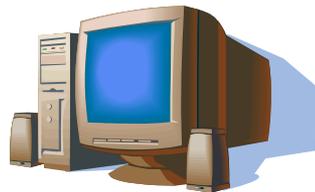
- Generación Simbólica

α, β

1.0, -2.0 ← $\alpha + \beta < 0$

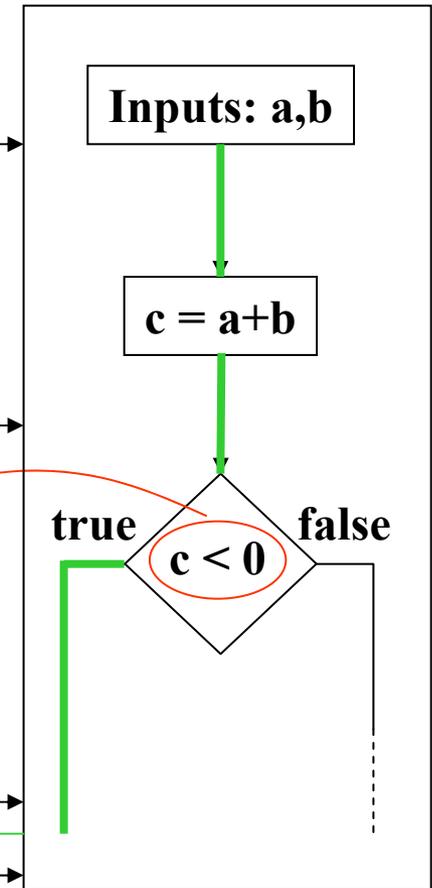
3.5, 1.2 ← $\alpha + \beta \geq 0$

- Generación Dinámica



-1.0, -0.5

1.0, -0.5



Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro

Paradigmas

- Tres paradigmas de generación de casos de prueba:

- Generación Aleatoria



1.2, 0.7

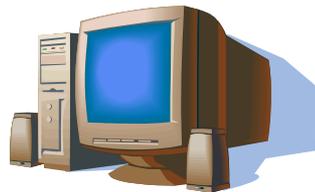
- Generación Simbólica

α, β

1.0, -2.0 ← $\alpha + \beta < 0$

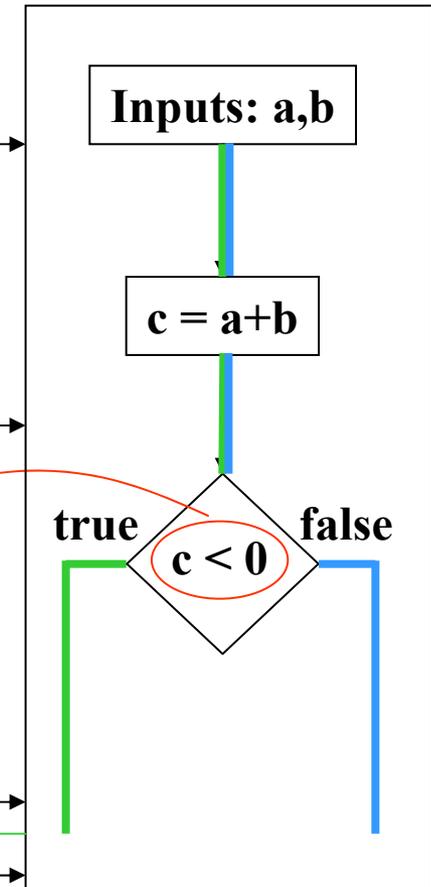
3.5, 1.2 ← $\alpha + \beta \geq 0$

- Generación Dinámica



-1.0, -0.5

1.0, -0.5



Introducción

Generador

Algoritmos de Optimización

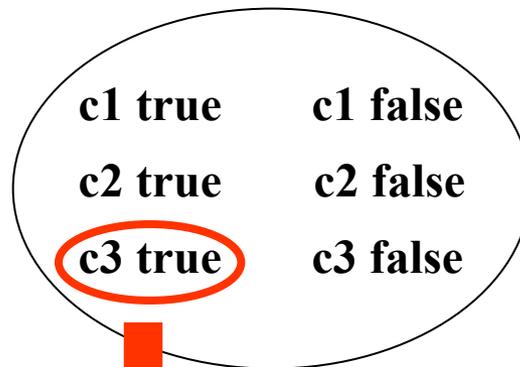
Experimentos

Conclusiones y Trabajo Futuro

Descomposición del Objetivo

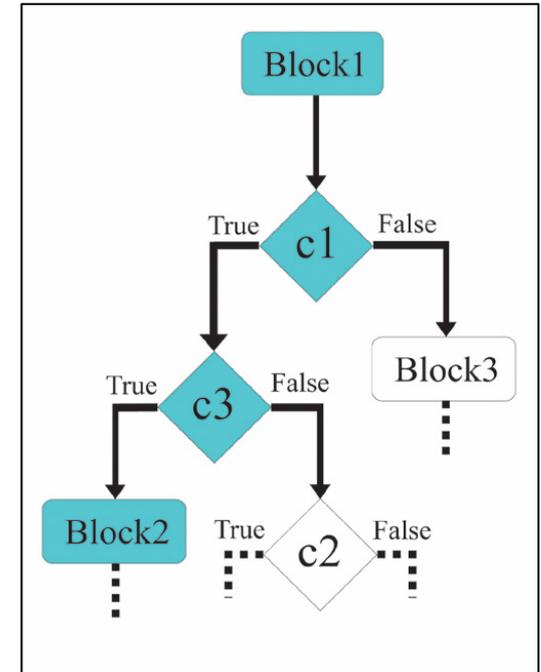
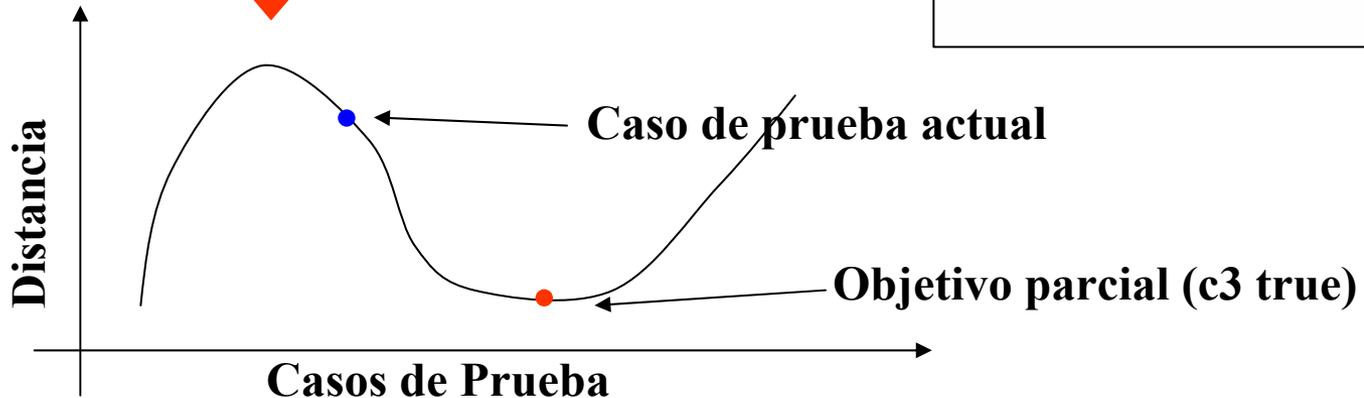
- El objetivo global se descompone en **pequeños objetivos parciales**

Seis objetivos parciales



Cobertura de
Condiciones

Problema de Minimización



Introducción

Generador

Algoritmos de
Optimización

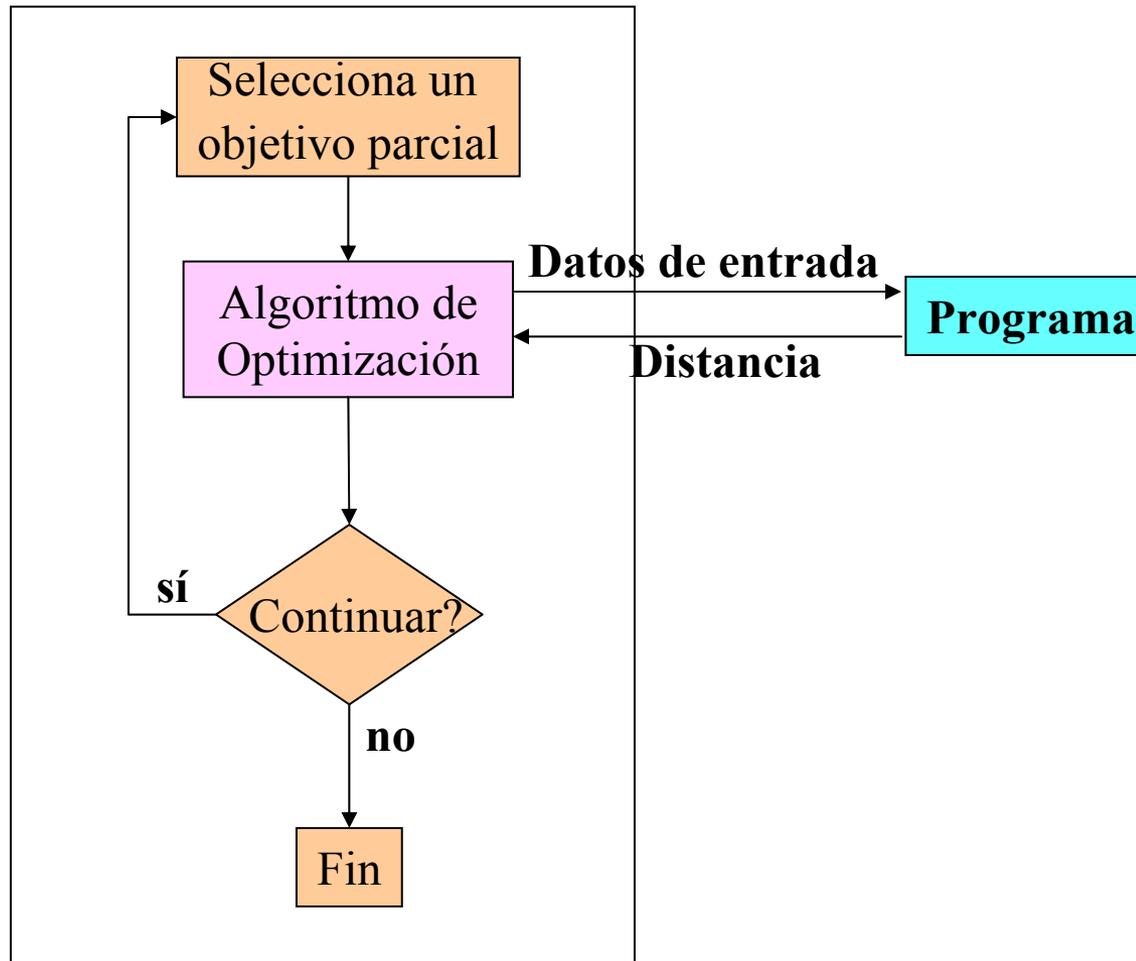
Experimentos

Conclusiones y
Trabajo Futuro



Generador de Casos de Prueba

Generador de Casos de Prueba



Introducción

Generador

Algoritmos de Optimización

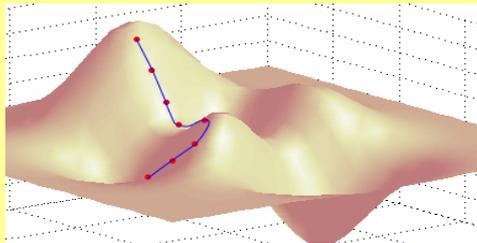
Experimentos

Conclusiones y Trabajo Futuro

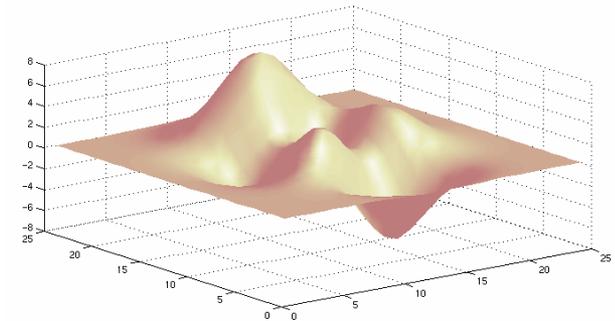
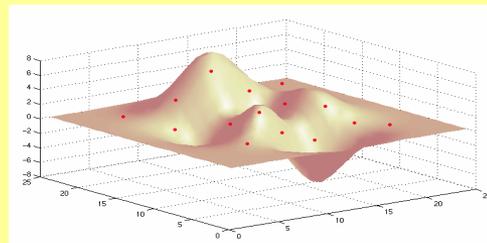
Algoritmos de Optimización

Metaheurísticas

Trayectoria



Población



Problema de Optimización

Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro

Algoritmos de Optimización

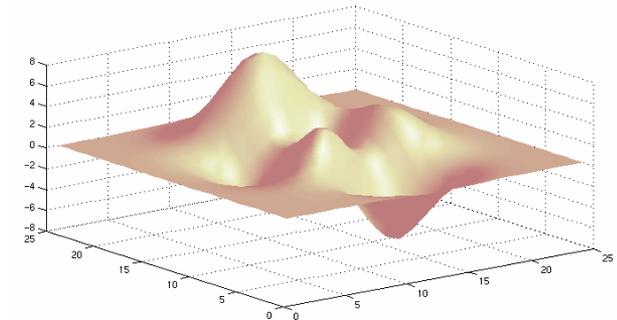
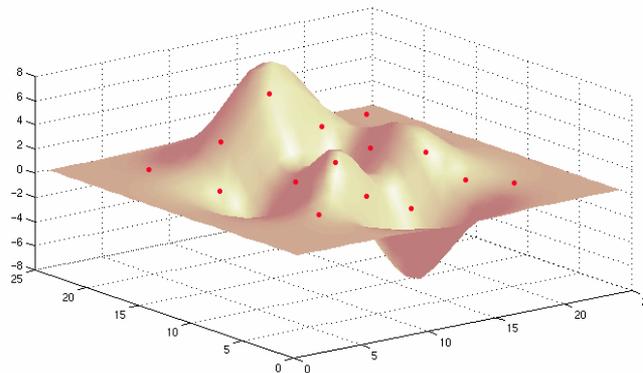
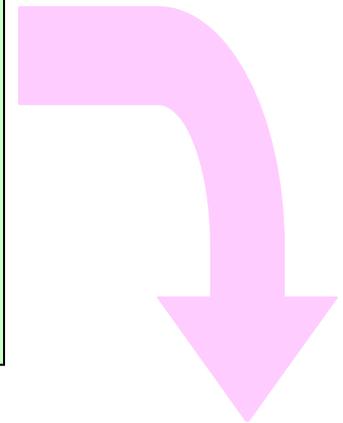
Metaheurísticas basadas en Población

Algoritmos
Evolutivos

Optimización basada en
Nubes de Partículas

Optimización basada en
Colonias de Hormigas

Búsqueda
Dispersa



Problema de Optimización

Introducción

Generador

Algoritmos de
Optimización

Experimentos

Conclusiones y
Trabajo Futuro

Nubes de Partículas (PSO)

• Optimización basada en Nubes de Partículas

➤ Partícula

(0.2, -1.4, 3.5) → Vector Solución (posición)

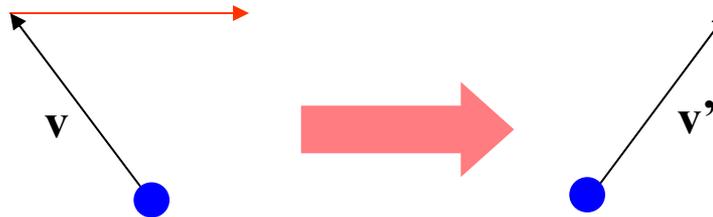
(1.0, 10.3, 7.2) → Velocidad

➤ Actualización

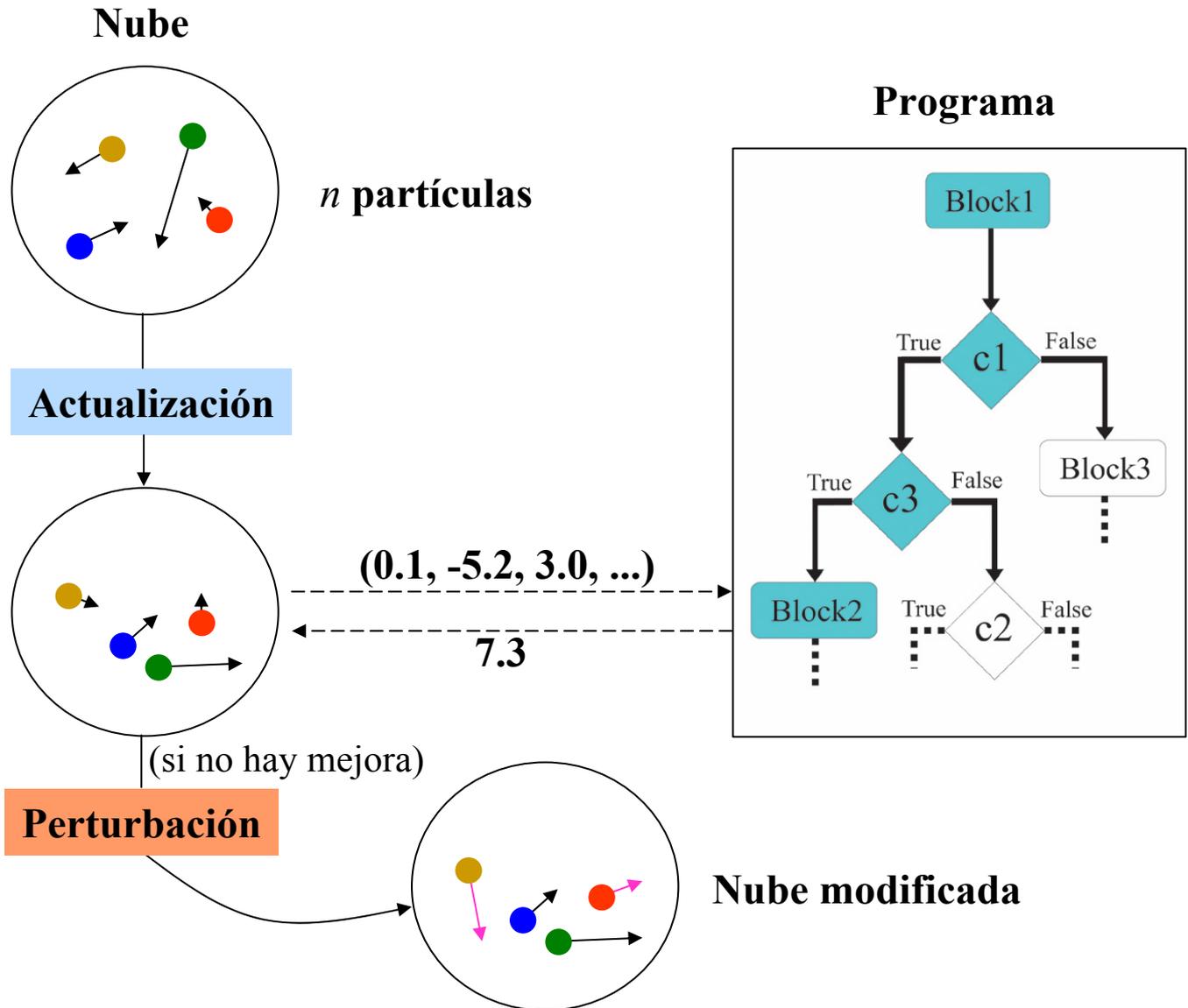
$$v_j^i(t+1) = \overset{\text{Inercia}}{w} \cdot v_j^i(t) + c_1 \cdot r_1 \cdot \overset{\text{Mejor personal}}{(p_j^i - x_j^i(t))} + c_2 \cdot r_2 \cdot \overset{\text{Mejor en vecindario}}{(n_j^i - x_j^i(t))}$$

$$x_j^i(t+1) = x_j^i(t) + v_j^i(t)$$

➤ Perturbación (cuando no mejoran los resultados)



Nubes de Partículas (PSO)



Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro



Estrategia Evolutiva (ES)

- Estrategia Evolutiva

- Individuo

$(0.2, -1.4, 3.5)$ → Vector Solución

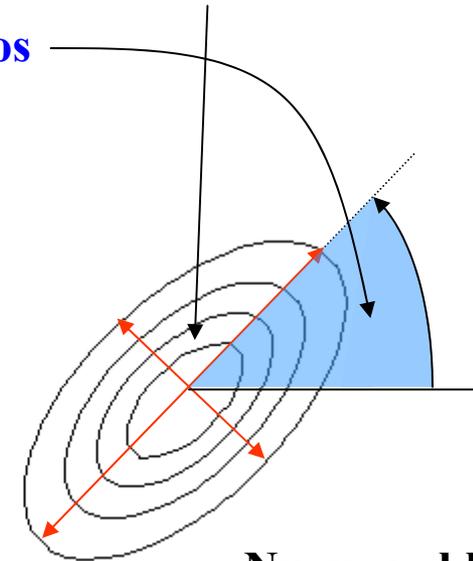
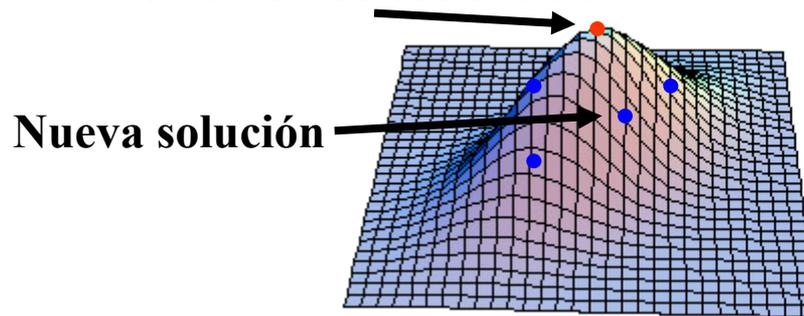
$(1.0, 10.3, 7.2)$ → Desviaciones estándar

$(1.7, 0.3, 2.1)$ → Ángulos

- Mutación Gaussiana

Solución Seleccionada

Nueva solución



Nueva población

- Reemplazo → $(\mu + \lambda)$ y (μ, λ)

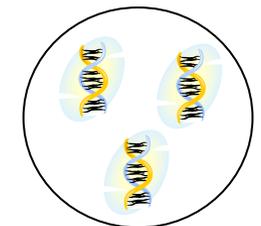
Población previa



Nuevos individuos



μ individuos



Estrategia Evolutiva (ES)

- Estrategia Evolutiva

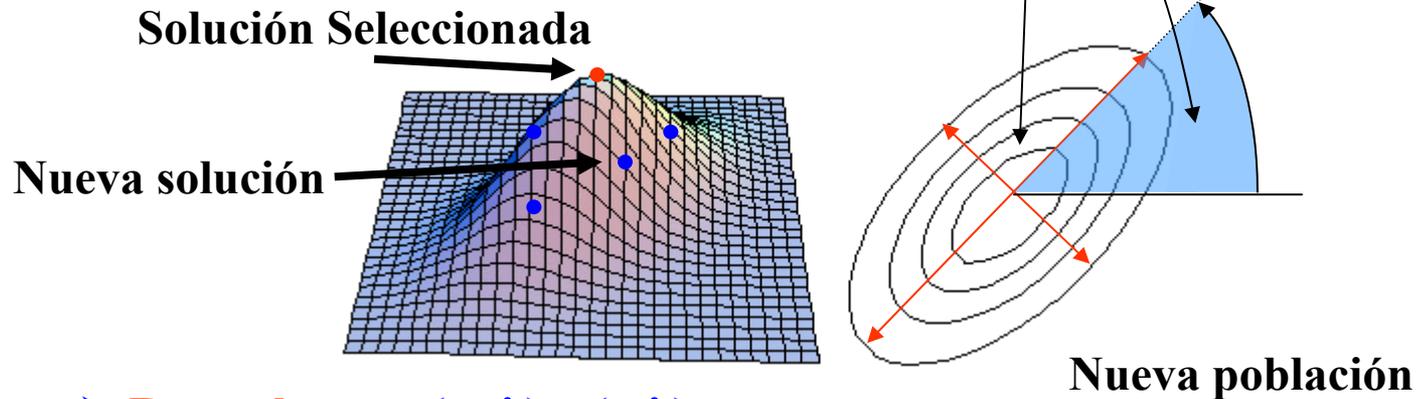
- Individuo

$(0.2, -1.4, 3.5)$ → Vector Solución

$(1.0, 10.3, 7.2)$ → Desviaciones estándar

$(1.7, 0.3, 2.1)$ → Ángulos

- Mutación Gaussiana

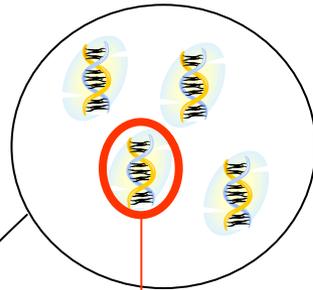


- Reemplazo → $(\mu + \lambda)$ y (μ, λ)



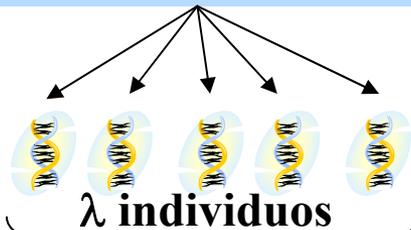
Estrategia Evolutiva (ES)

Población



μ individuos

Mutación Gaussiana

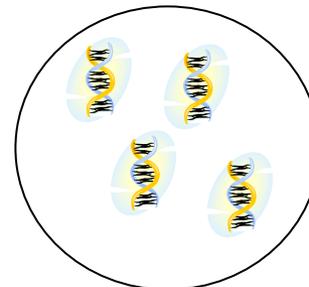


λ individuos

(0.1, -5.2, 3.0, ...)

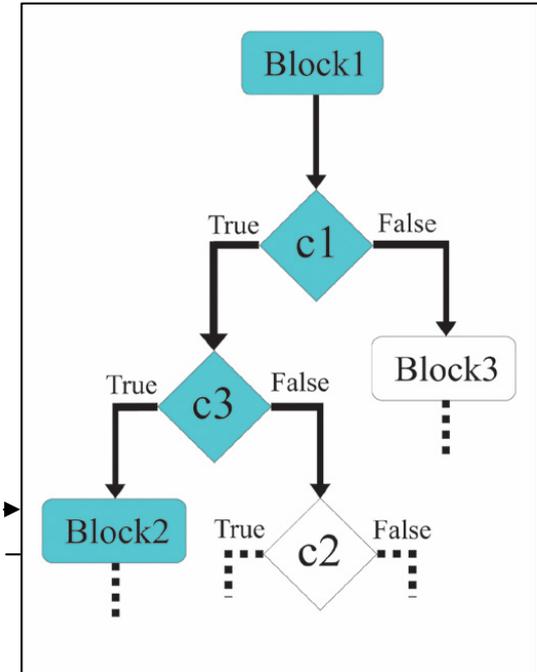
7.3

$(\mu + \lambda)$ Reemplazo



Nueva población

Programa



Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro

Medidas de Cobertura

- Cobertura:

$$c = \frac{\text{objetivos parciales cubiertos}}{\text{número total de objetivos parciales}}$$

- Inconvenientes: pérdidas inevitables de cobertura

```
while(1<2)
{
    /* algo */
}
```

**Pérdida de cobertura
dependiente del código**

```
p = malloc(4);
if (!p)
{
    error();
}
```

**Pérdida de cobertura
dependiente del entorno**

- Cobertura corregida:

$$cc = \frac{\text{objetivos parciales cubiertos}}{\text{número total de objetivos parciales alcanzables}}$$

Introducción

Generador

Algoritmos de
Optimización

Experimentos

Conclusiones y
Trabajo Futuro



Experimentos: Programas

- Benchmark: 6 programas en C

Programa	Conds.	LdC	Args.	Fuente
triangle	21	53	3	Michael et al., 2001
calday	11	72	3	C-Recipes
select	28	200	21	C-Recipes
bessel	21	245	2	C-Recipes
sa	30	332	23	C-Recipes
netflow	55	112	66	Wegener

Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro

Experimentos: Parámetros

- Parámetros de los algoritmos

PSO

Parámetro	Valor
Partículas	10
w	0.729
c1	1.494
c2	1.494
Prob. de perturb.	0.5
Parada	Obj. o 1000 evals.

ES

Parámetro	Valor
Población	25
Selección	Aleatoria
Mutación	Gaussiana
Hijos (λ)	5
Reemplazo	$(\mu+\lambda)$
Parada	Obj. o 1000 evals.

- Realizamos **30 ejecuciones independientes** para conseguir confianza estadística en los resultados

Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro



Experimentos: Resultados (I)

Programas	PSO		ES		GA	
	Cov.	Evals.	Cov.	Evals.	Cov.	Evals.
triangle	93.98	11295.77	99.84	2370.03	99.67	3209.47
calday	100.00	179.33	98.18	3166.47	90.91	75.03
select	88.89	380.13	83.33	13.27	83.33	83.20
bessel	97.56	116.90	97.56	350.63	97.56	533.03
sa	100.00	165.67	99.94	2337.30	96.72	176.63
netflow	97.77	4681.70	98.17	307.77	96.42	917.90

- PSO y ES tienen una eficacia **similar**
- La cobertura obtenida por GA es **igualada o superada** por PSO o ES en todos los casos



Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro

Experimentos: Resultados (y II)

Programas	PSO		ES		GA	
	Cov.	Evals.	Cov.	Evals.	Cov.	Evals.
triangle	93.98	11295.77	99.84	2370.03	99.67	3209.47
calday	100.00	179.33	98.18	3166.47	90.91	75.03
select	88.89	380.13	83.33	13.27	83.33	83.20
bessel	97.56	116.90	97.56	350.63	97.56	533.03
sa	100.00	165.67	99.94	2337.30	96.72	176.63
netflow	97.77	4681.70	98.17	307.77	96.42	917.90

Wegener: 40703 evals.

- Pérdida de cobertura dependiente del entorno en **select**
- Se encuentran casos de prueba que para los que **netflow no termina**

Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro



Conclusiones y Trabajo Futuro

Conclusiones

- Hemos aplicado **PSO** y **ES** al problema de testeo de software
- Hemos usado un conjunto de **seis programas** con diversas características para evaluar nuestra propuesta
- Las técnicas PSO y ES tienen una **eficacia similar**
- Ambas técnicas superan a los **algoritmos genéticos**

Trabajo Futuro

- **Hibridar** las técnicas para mejorar el balance entre explotación y exploración
- Extender la propuesta para programas con entradas **no numéricas**
- **Paralelizar** las técnicas siguiendo esquemas de población distribuida



Introducción

Generador

Algoritmos de Optimización

Experimentos

Conclusiones y Trabajo Futuro

FIN

Gracias por su Atención !!!



Introducción

Generador

Algoritmos de
Optimización

Experimentos

Conclusiones y
Trabajo Futuro