Generación Dinámica de Casos de Prueba utilizando Metaheurísticas

Tesina de Grado Juan La Battaglia

Dirección: Prof. Laura Lanzarini

Introducción

- Generación Dinámica de Casos de prueba
  - Testing de programas.
  - Cubrir todas líneas de código.

- Estudio y modificación de un método de optimización.

Índice

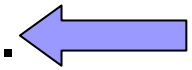
1. Problemas de Optimización en Ingeniería de Software.
2. Técnicas de Optimización.
3. Proceso de Generación de Casos de Prueba.
4. Método propuesto.
5. Experimentación.
6. Conclusiones y Trabajos Futuros.

Problemas de Optimización en Ingeniería de Software

Optimización

- ¿Que entendemos por optimización?
- Problemas de optimización.

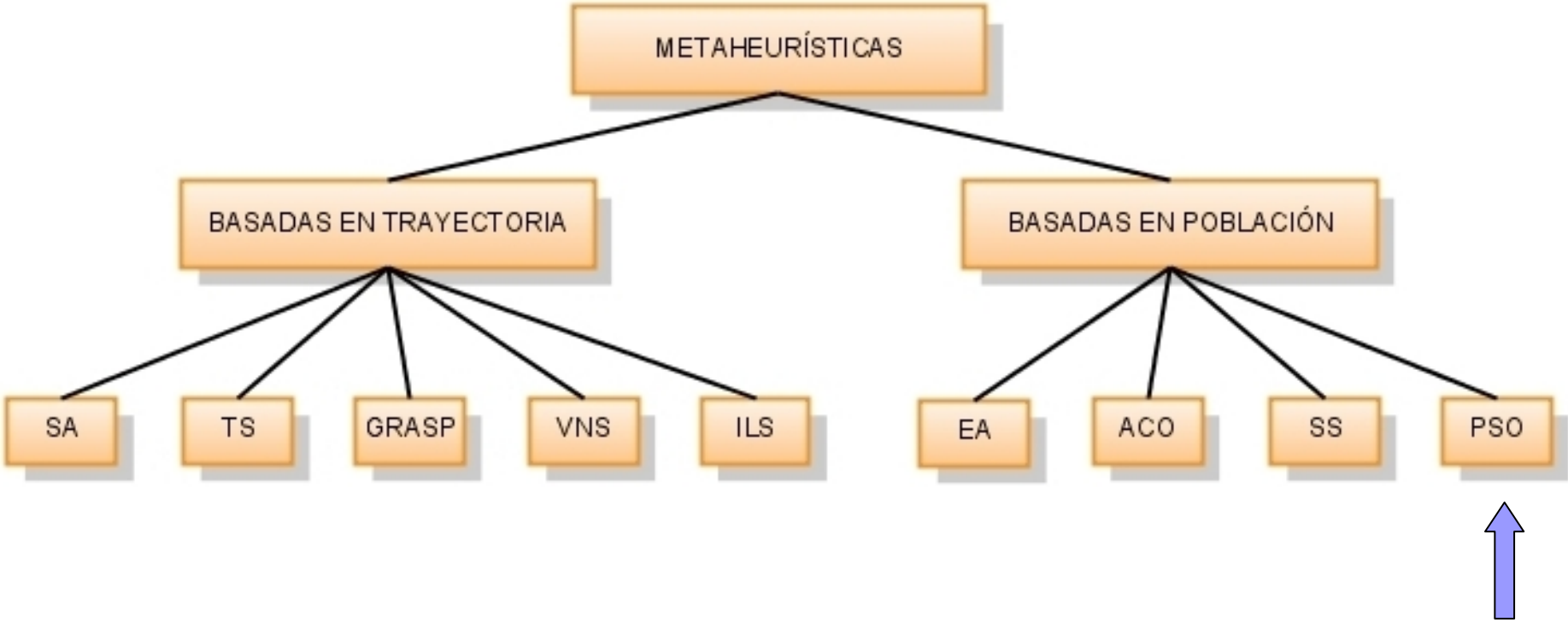
Problemas de Optimización en Ingeniería de Software

- **Análisis de Requerimientos**: Selección del orden de requisitos.
- **Diseño**: Orden de creación de componentes.
- **Implementación**: Optimización del código.
- **Testing**: Generación de casos de prueba. 
- **Implantación**: Planificación de tareas.
- **Mantenimiento**: Elegir modificaciones.
- **Gestión**: Estimación de costos.

Técnicas de Optimización

Técnicas de Optimización

Metaheurísticas

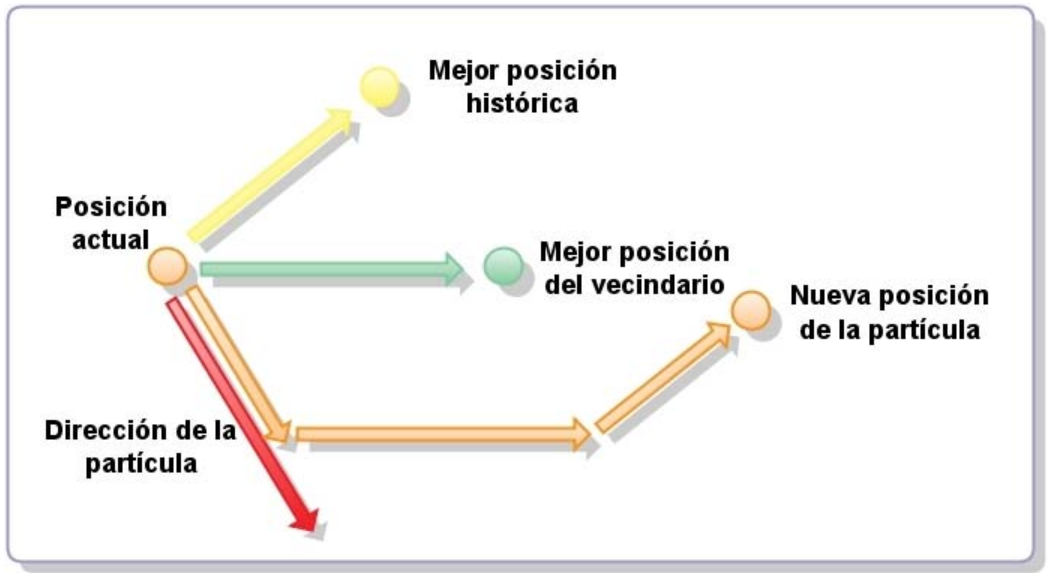


Optimización por Cúmulo de Partículas (PSO)

- Basado en “Metáfora Social”

Optimización por Cúmulo de Partículas (PSO)

- Información.
  - Conocimiento histórico propio.
  - Conocimiento social.
- Representación.
  - Vector de Posición.
  - Vector de Velocidad.
  - Fitness.



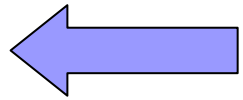
Proceso de generación de Casos de Prueba

Generación de Casos de Prueba

- Casos de prueba.
  - Par: Entrada – Salida Esperada.
- Problemas a sortear:
  - “Cubrir” la ejecución del programa.
  - Encontrar los casos en un tiempo razonable.
  - Conjunto de casos mínimo.

Automatización

Generación de Casos de Prueba

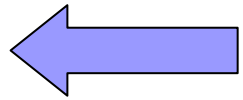
- Método existentes:
  - Generación random.
  - Prueba simbólica.
  - Técnicas de Optimización. 

Generación de Casos de Prueba

- Técnicas de Optimización.

1. Criterio de adecuación.
2. Conocimiento de la estructura del programa.
3. Función de Fitness.
4. Algoritmo de Optimización.

Generación de Casos de Prueba
Criterio de adecuación

- ¿Cuándo un programa está bien testeado?
  - Cobertura de Instrucciones.
  - Cobertura de ramas.
  - Cobertura de condiciones. 

Generación de Casos de Prueba
Conocimiento de la estructura del programa

- ¿Cuál fue el resultado de la comparación?
- ¿Qué valores fueron comparados?

Generación de Casos de Prueba

Conocimiento de la estructura del programa

- Instrumentador de código fuente.
- Asistente de ejecución.

```
1  a = 8
2  b = -1
3  if (a > 0) and (b > 0) #IF0 - false - AND - [[true], [false]] - [[8, 0, ">"], [-1, 0, ">"]]
4    puts("a y b son positivos")
5  end
```

Generación de Casos de Prueba
Función de Fitness

- ¿Cuán bueno es un individuo?
- ¿Cuál es la “distancia” que lo separa de invertir el resultado de la condición?

```
1  if a > 0
2  ...
3  end
```

a = 1000

a = 5

Generación de Casos de Prueba

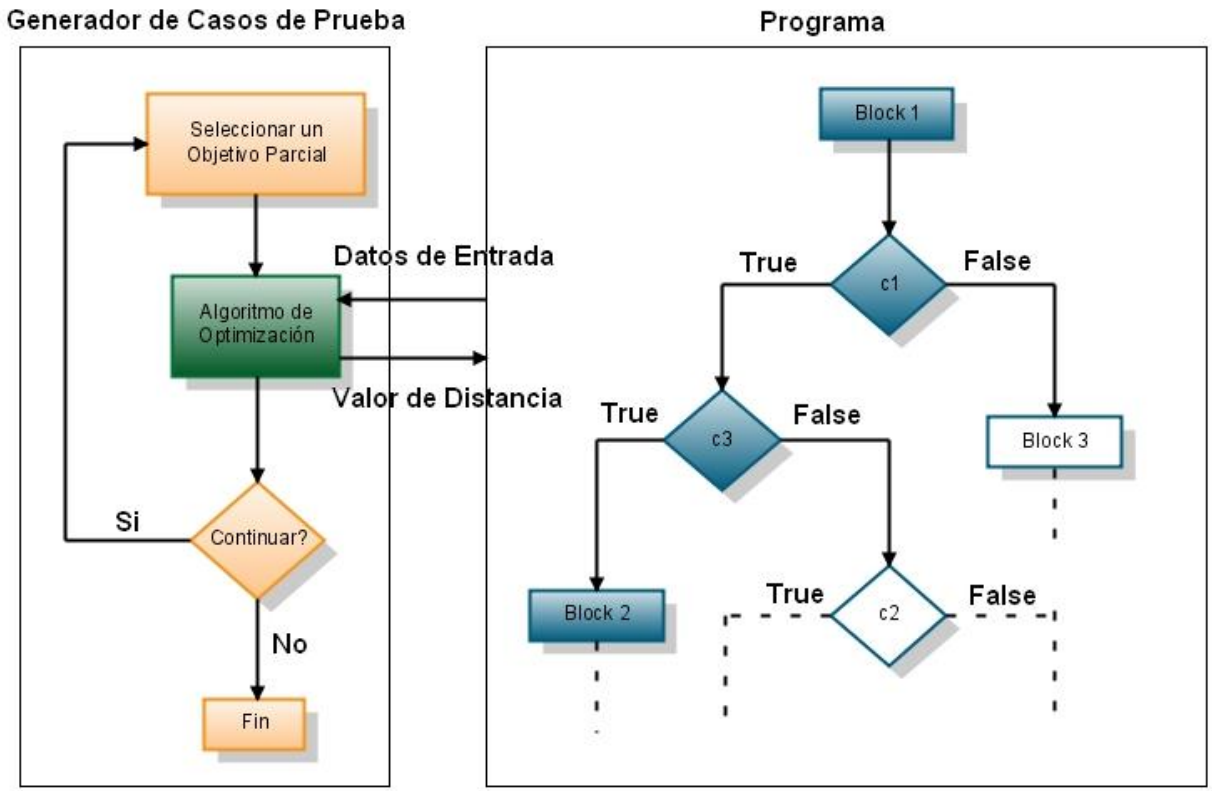
Función de Fitness

Condición	Función de fitness
$x=y$, $x\neq y$	abs(x-y)
$x<y$, $x\leq y$	y-x
$x>y$, $x\geq y$	x-y
$x \wedge y$	Min(cost(x),cost(y))

$x \vee y$	if x = TRUE and y = TRUE then
	Min(cost(x),cost(y))
	else $\sum_{c_i FALSE} \text{cost}(c_i)$
	end if

Generación de Casos de Prueba

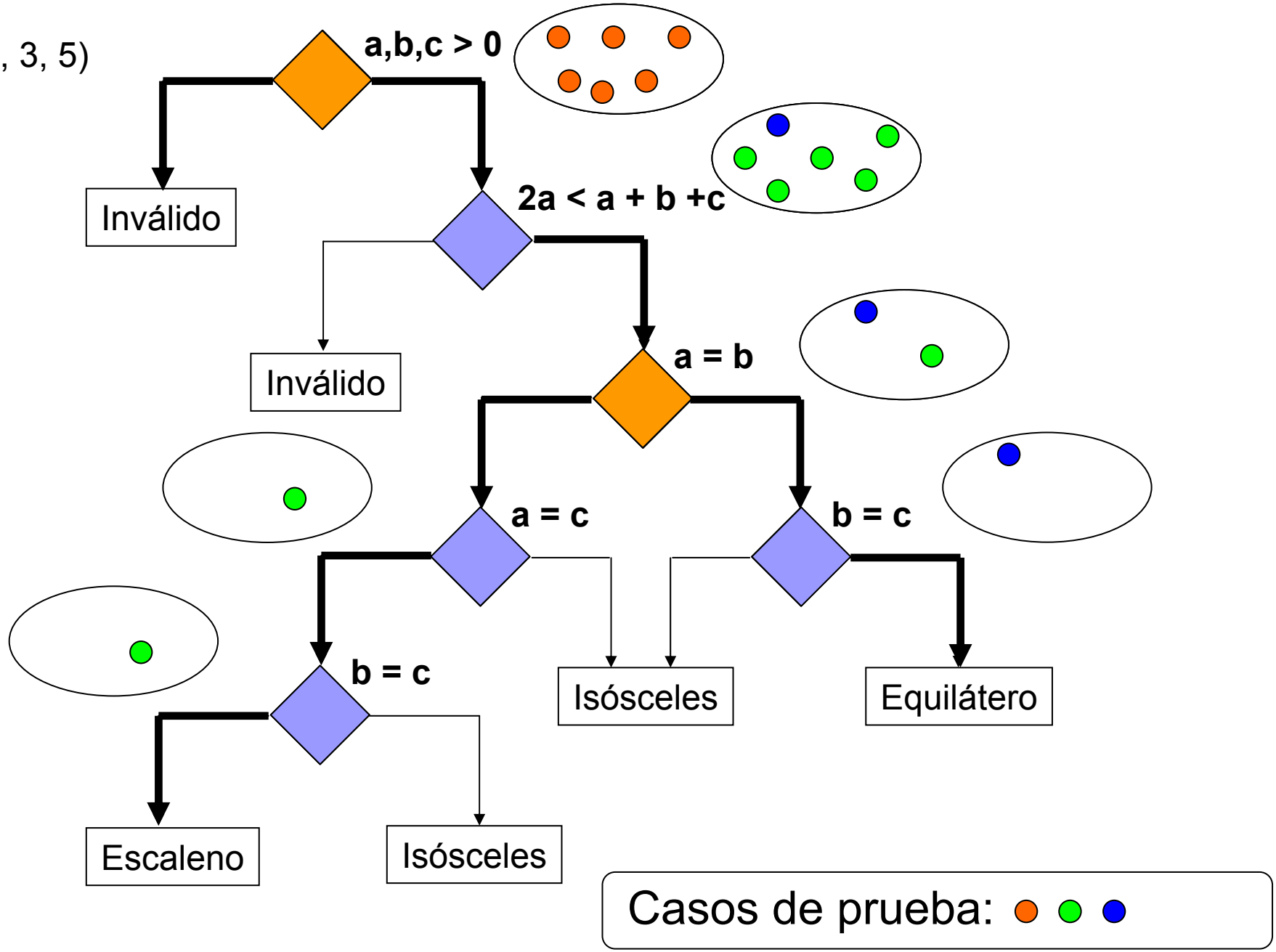
Algoritmo de Optimización



Ejemplo: Triángulos

```
1  if a > 0 and b > 0 and c > 0
2    if  2 * a < a + b + c and 2 * b < a + b + c and 2 * c < a + b + c
3
4      if a == b
5        if b == c
6          res = "equilatero"
7        else
8          res = "isosceles"
9        end
10     else
11       if a == c
12         res = "isosceles"
13       else
14         if b == c
15           res = "isosceles"
16         else
17           res = "escaleno"
18         end
19       end
20     end
21
22   else
23     res = "Entrada inválida"
24   end
25 else
26   res = "Entrada inválida"
27 end
```

(-2, 3, 5)


a,b,c > 0
Inválido
2a < a + b +c
Inválido
a = b
a = c
b = c
b = c
Isósceles
Equilátero
Escaleno
Isósceles
Casos de prueba:

Ejemplo: Triángulos

```
1  if a > 0 and b > 0 and c > 0
2    if  2 * a < a + b + c and 2 * b < a + b + c and 2 * c < a + b + c
3
4      if a == b
5        if b == c
6          res = "equilatero"
7        else
8          res = "isosceles"
9        end
10     else
11       if a == c
12         res = "isosceles"
13       else
14         if b == c
15           res = "isosceles"
16         else
17           res = "escaleno"
18         end
19       end
20     end
21
22   else
23     res = "Entrada inválida"
24   end
25 else
26   res = "Entrada inválida"
27 end
```

Ejemplo: Triángulos

- Primer IF, todos los valores tienen que ser mayores a cero.
- Ejecuta el instrumentador

```
1  if a > 0 and b > 0 and c > 0 #IF0
2  ...
3  end
```

Ejemplo: Triángulos

- Primer individuo al azar: (5, -8, 9)
- Comienza la optimización:
  - Crea población
  - Ejecuta el programa con los nuevos individuos
  - Mueve los individuos

```
1  if a > 0 and b > 0 and c > 0 #IF0 - false - AND - [[true],[false],[true]] - [[5,0,">"],[-8,0,">"],[9,0,">"]]
2  ...
3  end
```

-8 -8 -3 -6 -7

IF0 = (5, -8, 9), (2, -8, 9) , (5, -3, 9) , (5, -6, 5) , (8, -7, 9) …

IF0'= (3, -5, 2), (-1,-3, 8) , (5, 1, 8), (9, -1,-9) , (6, -6, 3) …

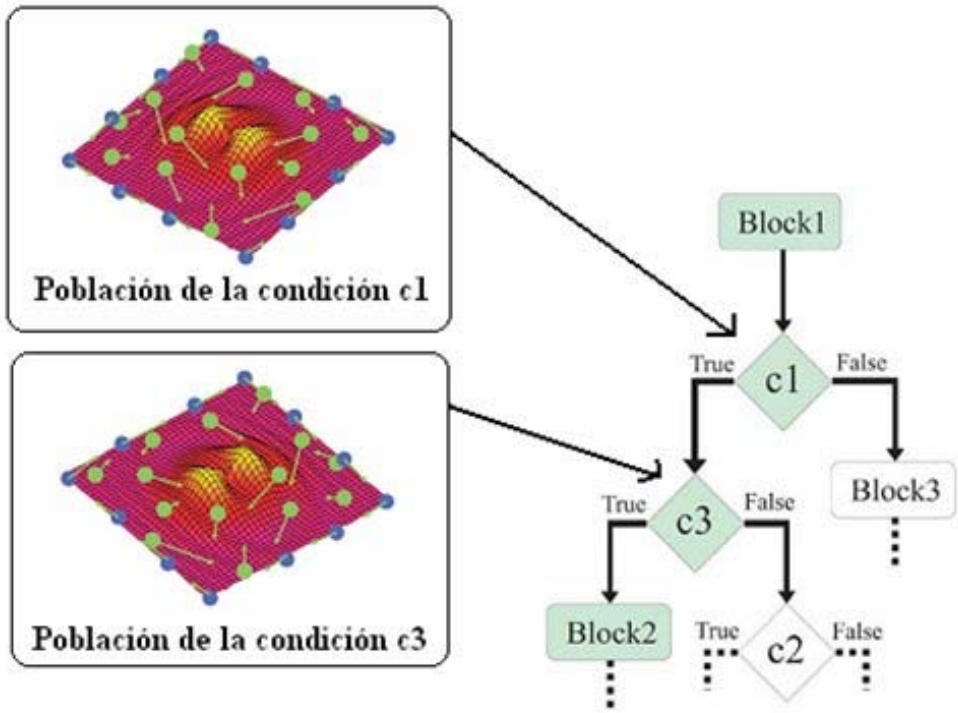
Método propuesto

Método propuesto

- Basado en PSO
- Adaptaciones particulares

Adaptaciones

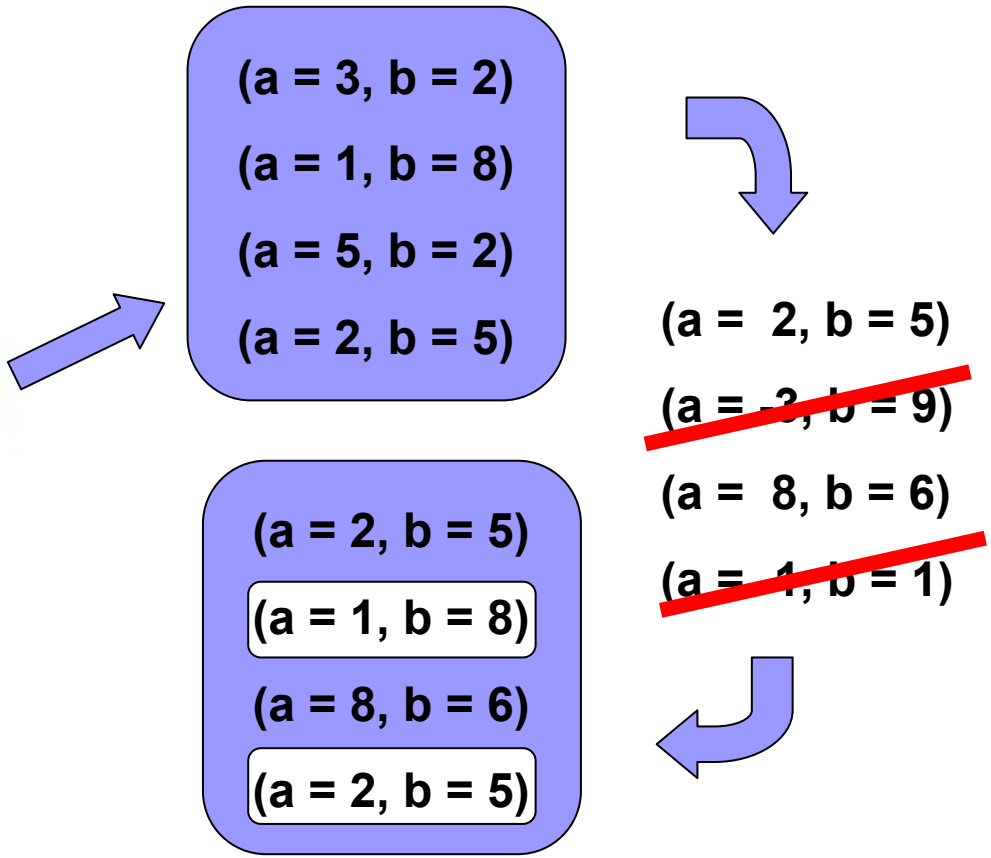
- Optimización multiobjetivo



Adaptaciones

- Función de Fitness no continua

```
1  if a > 0
2    if b < 10
3      ...
4    end
5  end
```



Adaptaciones

- Variaciones guiadas en para la creación de las poblaciones
  - (5, -8, 9): (2, -8, 9), (8, -8, 9)
    (5, -11, 9), (5, -5, 9)
    (5, -8, 6), (5, -8, 12)

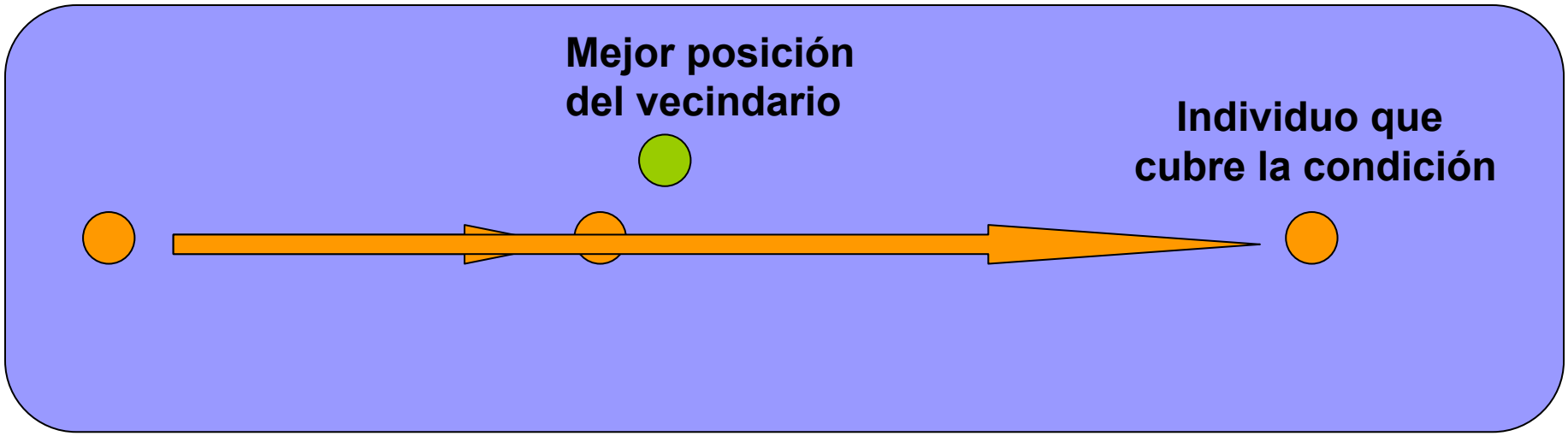
Adaptaciones

- Capacidad de exploración
  - Inserción de diversidad

(13, 2, 4), (5, -8, 9), (2, -8, 9) , (5, -3, 9) , (5, -6, 5) , (8, -7, 9) …

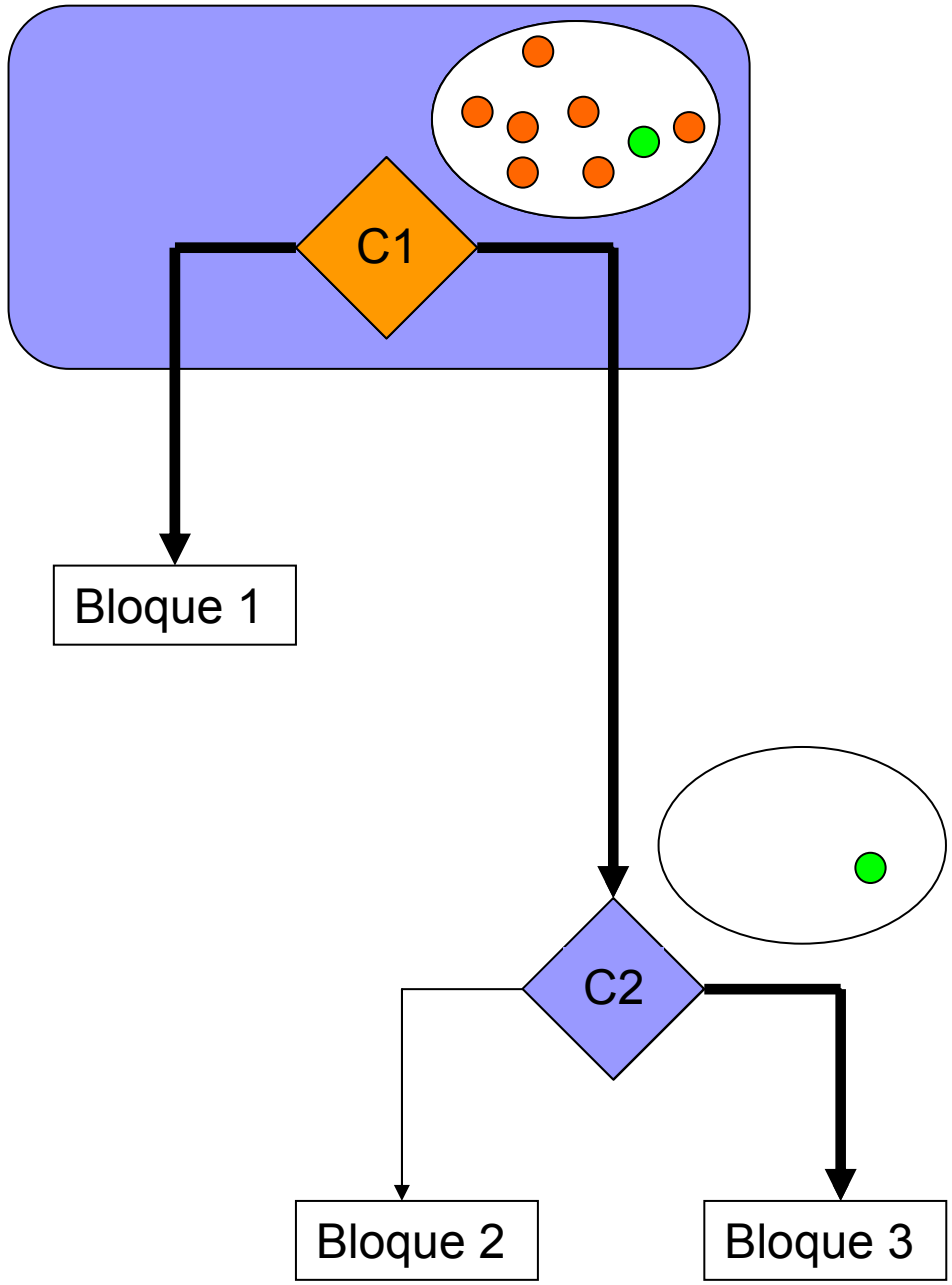
Adaptaciones

- Modificación del factor de inercia
  - No se busca el óptimo, sino su dirección.
  - Una inercia excesiva atenta contra el algoritmo.



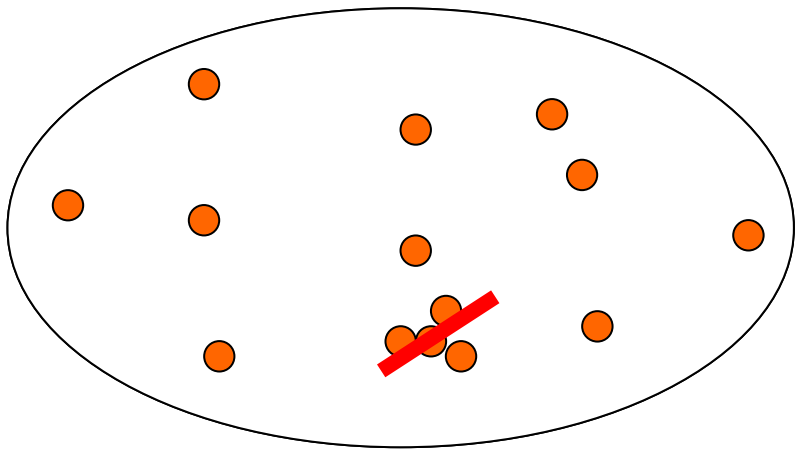
Adaptaciones

- Optimización de recursos.
  - Ejecución completa.
  - Lista tabú



Adaptaciones

- Cantidad de individuos.
  - Poblaciones variables.
  - Eliminación.



Experimentación

Experimentos

- Programas típicos en el área:
  - Triangle: clasificación de triángulos.
  - Calday: calcula día de la semana.
  - QuickSort: ordenamiento.
  - Select: k-ésimo elemento de una lista.
  - Bessel: funciones de Bessel.

Experimentos

- Comparación con otros métodos:
  - Random.
  - Tabu Search.

Experimentos

- Parámetros
  - Poblaciones iniciales: depende la cantidad de argumentos.
  - Rango de valores: específicos por variable.
  - Cantidad de intentos por condición: 150.
  - Total de pruebas por solucion: 100.

Experimentos

- Resultados
  - Cobertura

Programa	PSO con variación		Random		Tabu Search	
	Cov.	Evals.	Cov.	Evals.	Cov.	Evals.
triangle	**100**	**50,72**	95,75	34,76	73,75	55,58
calday	**99,4**	**512,74**	98,43	197,21	83,04	1491,26
select	**100**	72,56	**100**	**16,55**	98,83	139,13
bessel	**100**	**483,76**	99,03	320,08	96,63	2116,25
quicksort	**100**	2,21	**100**	**2,1**	**100**	7,99

Experimentos

- Resultados
  - Población

	Bessel
Cond	PSO var
1	4,05
2	4,32
3	4,23
4	3,68
5	4,97
6	4,29
7	1
8	5,15
9	6,92
10	4,34
11	1
12	1
13	11,64
14	2
15	30,13

	Calday
Cond	PSO var
1	5,67
2	3,27
3	4,79
4	4,51
5	4,76
6	4,5
7	4,48
8	4,47
9	4,88
10	4,51
11	15,47
12	6,7
13	5,16
14	6,16
15	5,31
16	6,83
17	5,89
18	4,54
19	6,59
20	8,05
21	5,67
22	6,69

	Select
Cond	PSO var
1	2,52
2	1,73
3	1
4	3,45
5	4,17
6	5,08
7	2,78
8	4,63
9	3,12
10	1,41
11	2,62
12	3,29

	Quicksort
Cond	PSO var
1	1
2	2,4
3	2,47
4	3,65
5	2,84
6	2,33

	Triangle
Cond	PSO var
1	4,51
2	3,41
3	4,38
4	3,21
5	5,97
6	5,83

Experimentos

- Resultados
  - Generación aleatoria

	Calday	Triangle	Bessel	Select	Quicksort
PSO	648,83	124,59	369,08	19,64	1,85
Random	129,27	9,82	35,27	75,98	1,15

Conclusiones y Trabajos Futuros

Conclusiones

- Comparación con otros métodos:
  - Aumento de la cobertura.
  - Disminución de proceso.

Trabajos Futuros

- Optimización Independiente.
  - Paralelización.
- Argumentos numéricos.
  - Extensión de la función de Fitness.