



ALGORITMO *PARTICLE SWARM OPTIMIZATION* PARA RESOLUÇÃO DO PROBLEMA DA PROGRAMAÇÃO DA PRODUÇÃO JOB-SHOP FLEXÍVEL

CCO-727 – Otimização Inteligente de Sistemas Produtivos

Diego Luiz Cavalca

TÓPICOS

- **Introdução**
- Definição do FJSP
- Objetivos
- Conceitos Gerais
- Trabalhos Relacionados
- Proposta
- Considerações Finais

PROGRAMAÇÃO DA programação

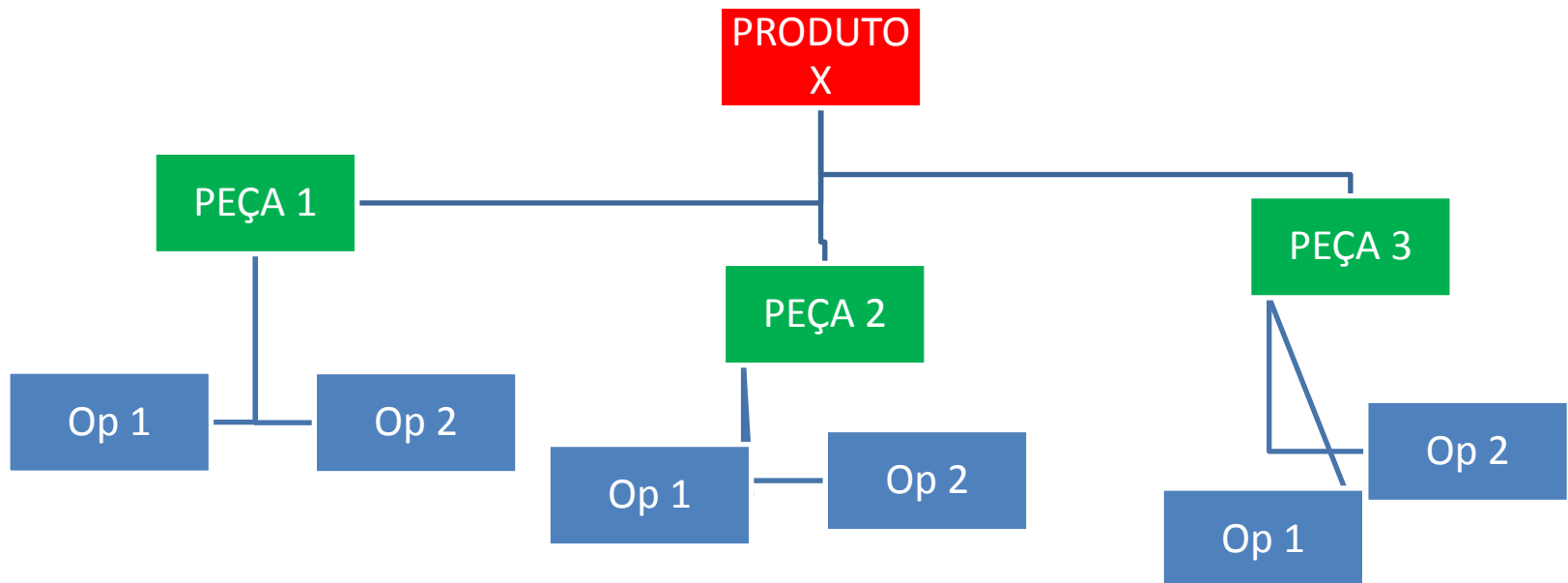
- Consiste em uma área da manufatura;
- O objetivo principal dá-se **tanto no planejamento** quanto no controle dos recursos do processo produtivo.

PROGRAMAÇÃO DA programação

- **Ideia:** processar n tarefas em m recursos produtivos (máquinas) em **tempo viável**, com o intuito de obter um produto com o menor tempo e custo de programação possível, respeitando as restrições.
- **Exemplos:** Montagem de automóveis, construção de aparelhos de eletrodomésticos, indústria têxtil, construção civil, etc.

PROGRAMAÇÃO DA programação

- **Desafio:** como otimizar a programação com os recursos disponíveis, diminuindo custos e maximizando o lucro produtivo?



MODELOS DE PROGRAMAÇÃO

DA programação

- Antes de resolver, é preciso modelar de maneira eficiente, de acordo com o contexto estudado;
- Existem alguns modelos de programação, os quais cabe destacar: **Flow-shop Scheduling (FS)**, **Job-shop Scheduling (JS)**, **Open-shop Scheduling (OS)** e **Flexible Job-shop Scheduling (FJSP)**.

MODELOS DE PROGRAMAÇÃO

DA programação

- **Flow-shop Scheduling (FS):** cenário em que existem m recursos e n *jobs*, sendo que cada *job* apresenta um **conjunto de operações (estágios de processamento) pré-ordenadas** e as sequências de execução nos recursos disponíveis são as mesmas para todos os *jobs* envolvidos;
- **Job-shop Scheduling (JS):** tal forma de produção difere do anterior pelo fato de que, nesse caso, **cada *job* apresenta uma sequência própria de execução de suas operações** nos recursos produtivos disponíveis, podendo o recurso ser ou não compartilhado por outro *job* do sistema considerado;
- **Open-shop Scheduling (OS):** este tipo de produção pode ser interpretado como um caso particular do JS, onde os *jobs* verificados não apresentam restrições internas de ordenamento de operações, ou seja, suas operações não possuem relações de precedência pré-estabelecidas.

TÓPICOS

- Introdução
- **Definição do FJSP**
- Objetivos
- Conceitos Gerais
- Trabalhos Relacionados
- Proposta
- Considerações Finais

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

- **Generalização** que respeita as premissas do modelo base (JS), porém possui complexidades complementares;
- Imposição de se processar n *jobs* utilizando m recursos produtivos;
 - **Cada recurso** está contido em um conjunto de recursos M que processa este determinado *job*;
- Quando submetido à programação, um *job* pode optar por um recurso produtivo **dentre todos os disponíveis** em seu conjunto de opções, respeitando as condições atuais do recurso;
 - Este cenário aumenta significativamente o **número de soluções factíveis**.

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

- **Característica:** pelo menos uma operação do problema ser processada por dois ou mais recursos (flexível);
- Neste caso, dada uma Operação O_{ij} , os recursos possuem as mesmas aptidões, sendo **factíveis** para esta tarefa, porém podem variar seus tempos de processamento (ou não).

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

- **Exemplo FJSP-3x3:** deve-se construir um produto, levando em conta as seguintes variáveis:
 - 3 Jobs devem ser executados;
 - Cada ***Job(i)*** é composto de 3 operações;
 - 3 recursos produtivos (máquinas) estão disponíveis para processar as operações.

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

job	$O_{i,j}$	M_1	M_2	M_3
1	$O_{1,1}$	2	0	3
	$O_{1,2}$	6	0	9
	$O_{1,3}$	8	7	7
2	$O_{2,1}$	3	2	6
	$O_{2,2}$	2	2	3
	$O_{2,3}$	5	2	4
3	$O_{3,1}$	2	0	1
	$O_{3,2}$	4	6	5
	$O_{3,3}$	0	1	3

Tabela 1: FJSP-3x3 (ARANHA, 2016)

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)



job	$O_{i,j}$	M_1		M_3
1	$O_{1,1}$	2	0	3
	$O_{1,2}$	6	0	9
	$O_{1,3}$	8	7	7
2	$O_{2,1}$	3	2	6
	$O_{2,2}$	2	2	3
	$O_{2,3}$	5	2	4
3	$O_{3,1}$	2	0	1
	$O_{3,2}$	4	6	5
	$O_{3,3}$	0	1	3

Tabela 1

Roteamento Job 1:

$O_{1,1}$	$O_{1,2}$	$O_{1,3}$
M1

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)




job	$O_{i,j}$	M_1	M_2	M_3
1	$O_{1,1}$	2	0	3
	$O_{1,2}$	6	0	9
	$O_{1,3}$	8	7	7
2	$O_{2,1}$	3	2	6
	$O_{2,2}$	2	2	3
	$O_{2,3}$	5	2	4
3	$O_{3,1}$	2	0	1
	$O_{3,2}$	4	6	5
	$O_{3,3}$	0	1	3



Tabela 1

Roteamento Job 1:

$O_{1,1}$	$O_{1,2}$	$O_{1,3}$
M1	M1	...



FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)





job	$O_{i,j}$	M_1	M_2	M_3
1	$O_{1,1}$	2	0	3
	$O_{1,2}$	6	0	9
	$O_{1,3}$	8	7	7
2	$O_{2,1}$	3	2	6
	$O_{2,2}$	2	2	3
	$O_{2,3}$	5	2	4
3	$O_{3,1}$	2	0	1
	$O_{3,2}$	4	6	5
	$O_{3,3}$	0	1	3

Tabela 1

Roteamento Job 1:

$O_{1,1}$	$O_{1,2}$	$O_{1,3}$
M1	M1	M3



FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

job	$O_{i,j}$	M_1	M_2	M_3
1	$O_{1,1}$	2	0	3
	$O_{1,2}$	6	0	9
	$O_{1,3}$	8	7	7
2	$O_{2,1}$	3	2	6
	$O_{2,2}$	2	2	3
	$O_{2,3}$	5	2	4
3	$O_{3,1}$	2	0	1
	$O_{3,2}$	4	6	5
	$O_{3,3}$	0	1	3

Tabela 1

Roteamento Job 2:

$O_{2,1}$	$O_{2,2}$	$O_{2,3}$
M2

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

job	$O_{i,j}$	M_1	M_2	M_3
1	$O_{1,1}$	2	0	3
	$O_{1,2}$	6	0	9
	$O_{1,3}$	8	7	7
2	$O_{2,1}$	3	2	6
	$O_{2,2}$	2	2	3
	$O_{2,3}$	5	2	4
3	$O_{3,1}$	2	0	1
	$O_{3,2}$	4	6	5
	$O_{3,3}$	0	1	3

Tabela 1

Roteamento Job 2:

$O_{2,1}$	$O_{2,2}$	$O_{2,3}$
M2	M1	...

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

job	$O_{i,j}$	M_1	M_2	M_3
1	$O_{1,1}$	2	0	3
	$O_{1,2}$	6	0	9
	$O_{1,3}$	8	7	7
2	$O_{2,1}$	3	2	6
	$O_{2,2}$	2	2	3
	$O_{2,3}$	5	2	4
3	$O_{3,1}$	2	0	1
	$O_{3,2}$	4	6	5
	$O_{3,3}$	0	1	3

Tabela 1

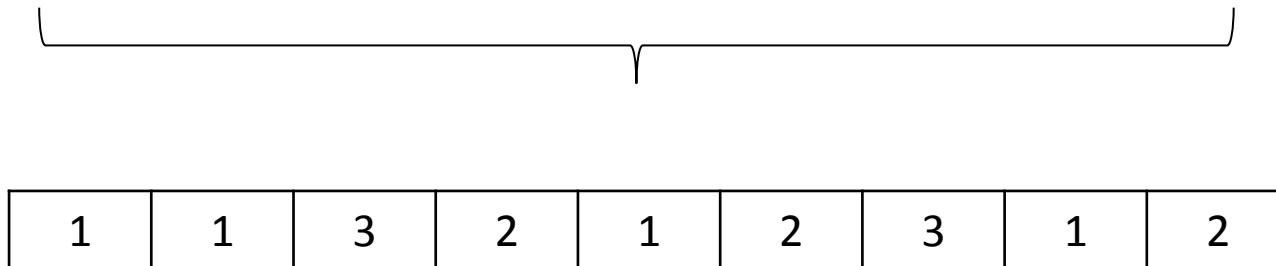
Roteamento Job 2:

$O_{2,1}$	$O_{2,2}$	$O_{2,3}$
M2	M1	M2

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

... E assim para o **Job 3**, resultando em:

JOB 1			JOB 2			JOB 3		
$O_{1,1}$	$O_{1,2}$	$O_{1,3}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$	$O_{3,3}$
1	1	3	2	1	2	3	1	2



Roteamento das operações

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

	JOB 1			JOB 2			JOB 3		
	$O_{1,1}$	$O_{1,2}$	$O_{1,3}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$	$O_{3,3}$
	1	1	3	2	1	2	3	1	2
Índice (O_{ij})	1	2	3	4	5	6	7	8	9
Valor (M)	1	1	3	2	1	2	3	1	2

Roteamento das operações

- Este **roteamento** determina os caminhos que as operações percorrerão através dos recursos:
 - Índices das colunas representam **cada operação** $O_{i,j}$;
 - Valores representam **o recurso** que irá processar a operação O_{ij} .

FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

JOB 1			JOB 2			JOB 3		
$O_{1,1}$	$O_{1,2}$	$O_{1,3}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$	$O_{3,3}$
1	1	3	2	1	2	3	1	2



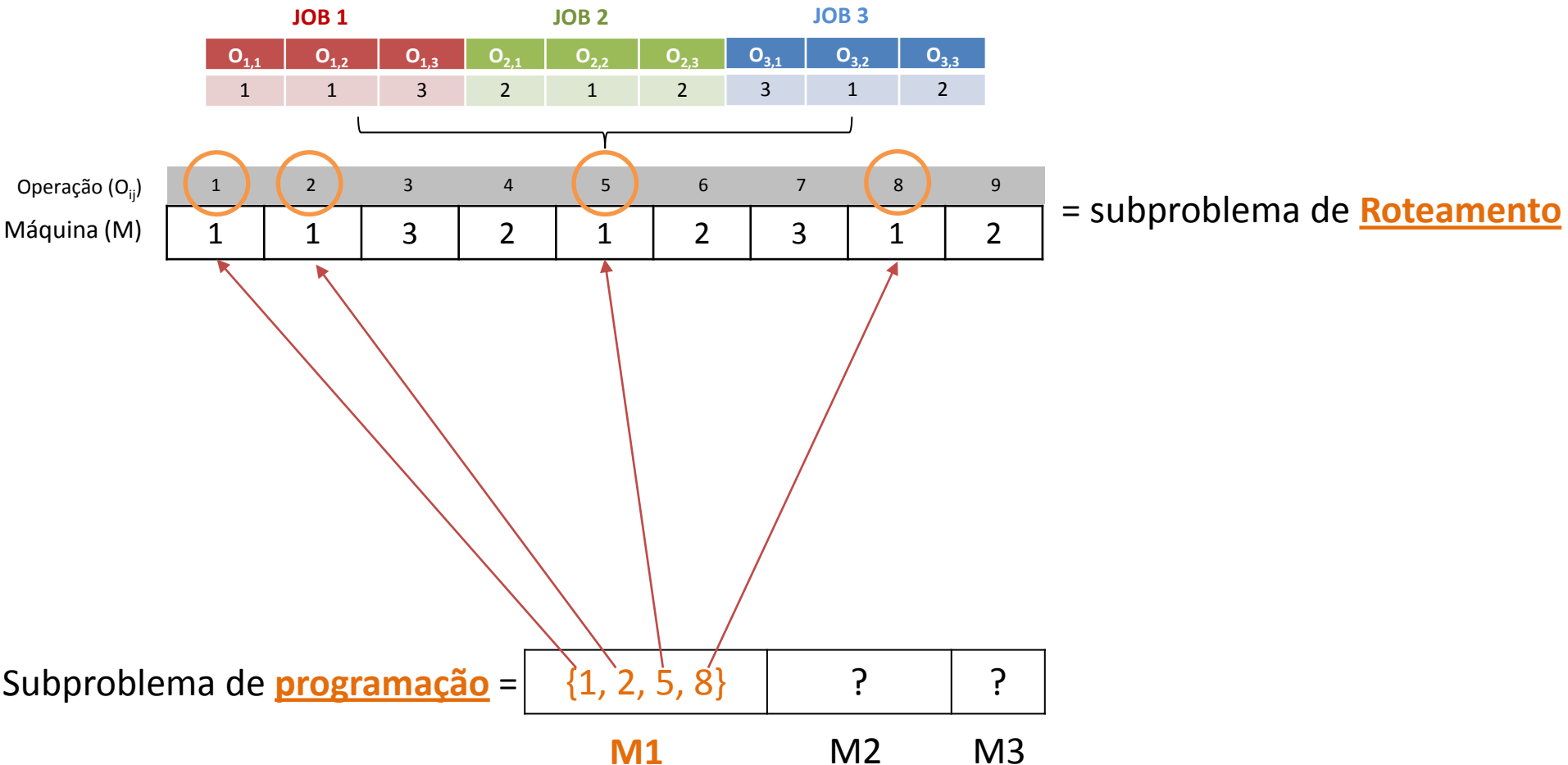
Operação (O_{ij})	1	2	3	4	5	6	7	8	9
Máquina (M)	1	1	3	2	1	2	3	1	2

= subproblema de Roteamento

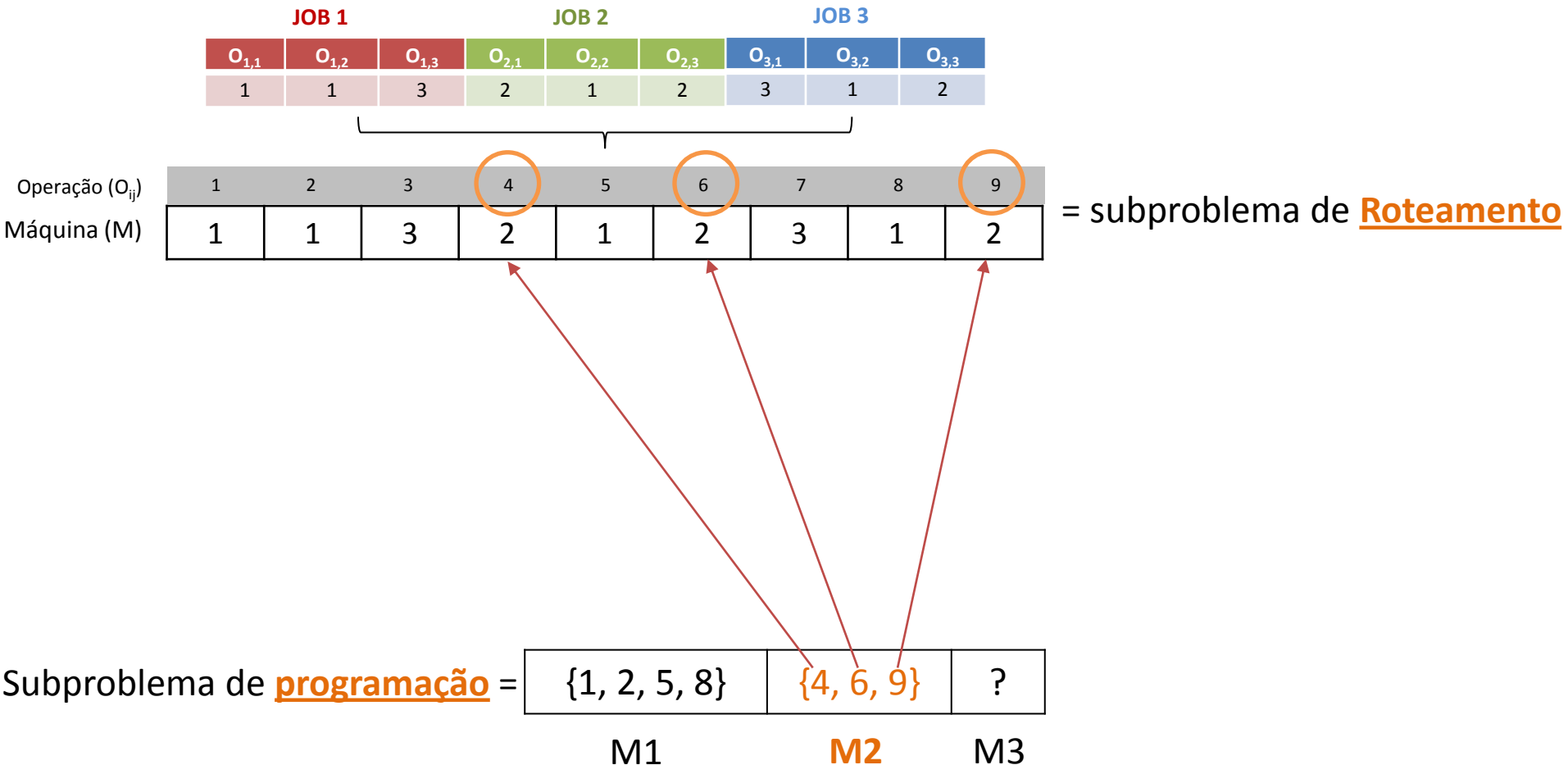
	M1	M2	M3
Subproblema de <u>programação</u> =	?	?	?

A partir do Roteamento, podemos extrair a programação!

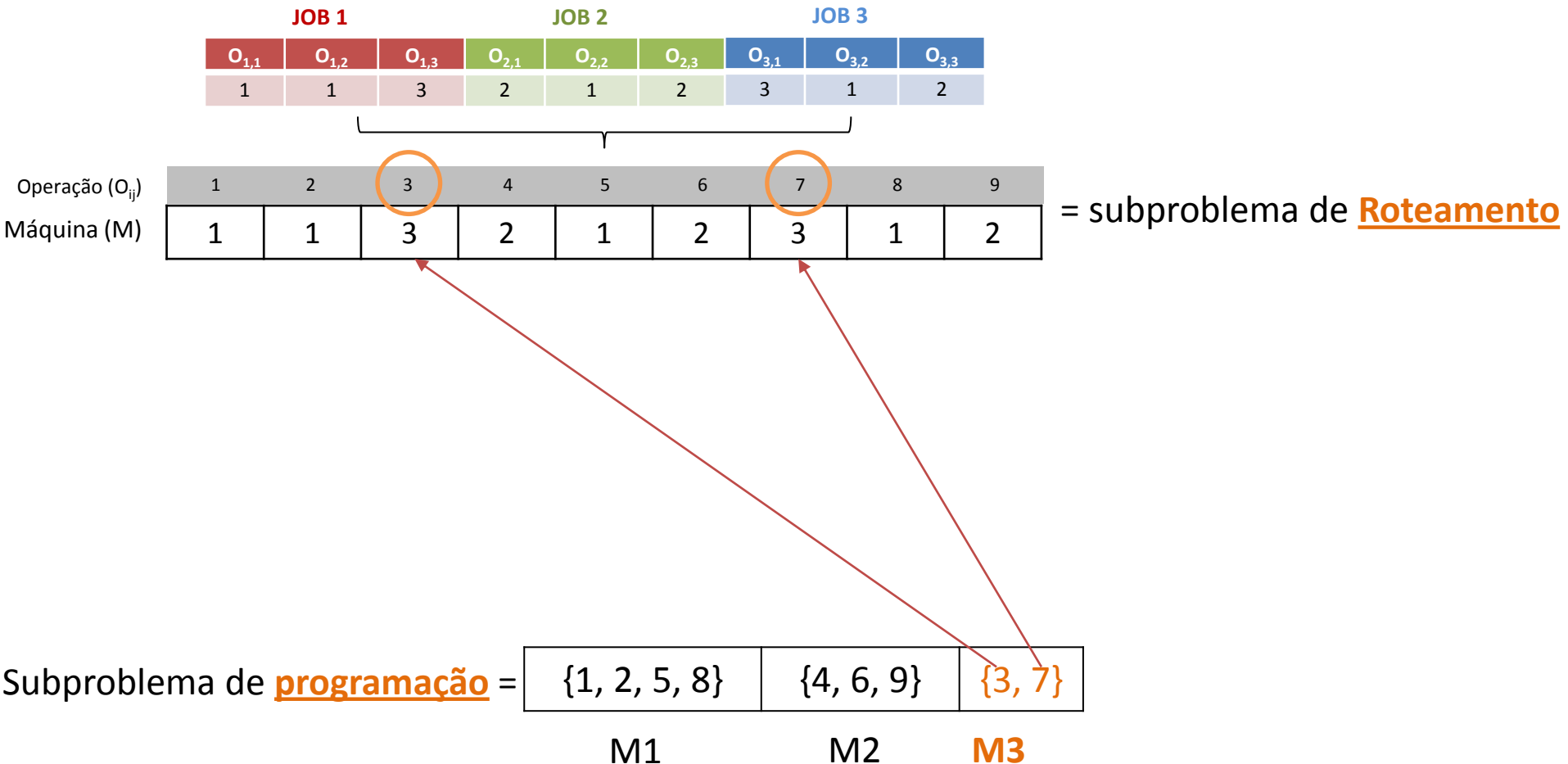
FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)



FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)



FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)



FLEXIBLE JOB-SHOP SCHEDULING PROBLEM (FJSP)

Subproblema de Roteamento

1	2	3	4	5	6	7	8	9
1	1	3	2	1	2	3	1	2

= É o MELHOR Roteamento?

Subproblema de programação

{1, 2, 5, 8}	{4, 6, 9}	{3, 7}
M1	M2	M3

= Dado o Roteamento, é a MELHOR ordem de programação?

Otimização através de representação hierárquica!

TÓPICOS

- Introdução
- Problema
- **Objetivo da Pesquisa**
- Conceitos Gerais
- Trabalhos Relacionados
- Proposta
- Considerações Finais

OBJETIVOS DA PESQUISA

- Resolução do *FJSP* através da aplicação do algoritmo ***Enxame de Partículas (PSO)*** combinado com a utilização do mecanismo de busca local ***Arrefecimento Simulado (do inglês Simulated Annealing - SA)***;
- Critério de minimização do problema: **tempo de completude do roteiro de programação (*maskepan*)**.

TÓPICOS

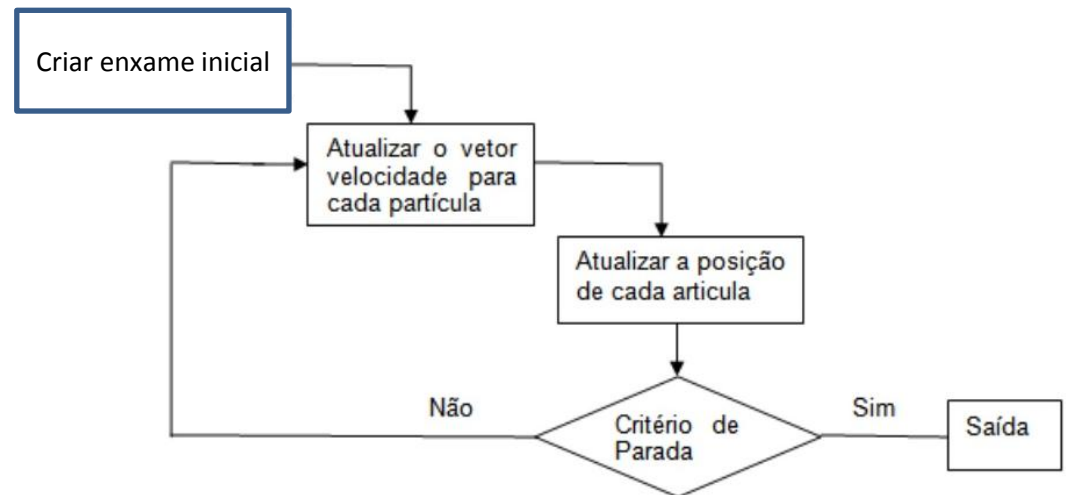
- Introdução
- Problema
- Objetivos
- **Conceitos Gerais**
- Trabalhos Relacionados
- Proposta
- Considerações Finais

ENXAME DE PARTÍCULAS

- Otimiza um problema **iterativamente** ao tentar melhorar a solução candidata com respeito a uma dada medida de qualidade;
- Utiliza o conceito de **inteligência de enxame** para resolver um problema de maneira heurística;
- Mimetiza o comportamento e mecanismos de sobrevivência de animais (aves, peixes, lobos, etc.)

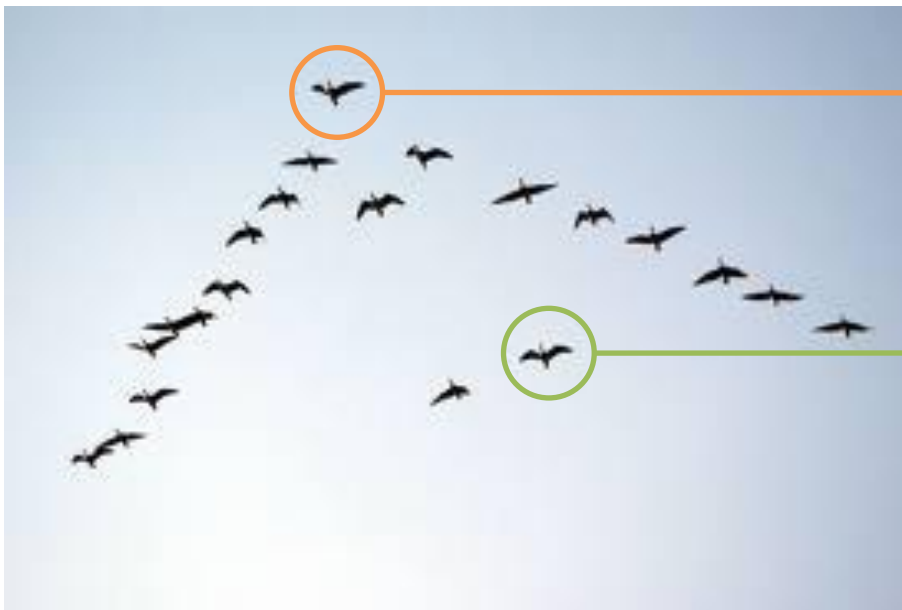
ENXAME DE PARTÍCULAS

- **Particle Swarm Optimization (PSO)**: algoritmo baseado em um modelo simplificado da teoria de enxames (Swarm Theory);
- Considera um **grupo de agentes** (enxame ou nuvem) em deslocamento pelo espaço de busca, considerando que nesse 'voo' cada partícula é independente (Goldbarg et. al, 2016).
- Regido por experiência pessoal ($Pbest$), experiência global ($Gbest$) e o movimento individual atual para decidir as posições seguintes no espaço de busca.



ENXAME DE PARTÍCULAS

- O movimento das partículas (nuvem) é regido por 2 funções:
 - (1a) Velocidade: $V_{t+1} = w * V_t + c1 * r1 * (pBest - P_t) + c2 * r2 * (gBest - P_t)$
 - (1b) Posição: $P_{t+1} = P_t + V_{t+1}$



Ótimo **global** = *gBest*

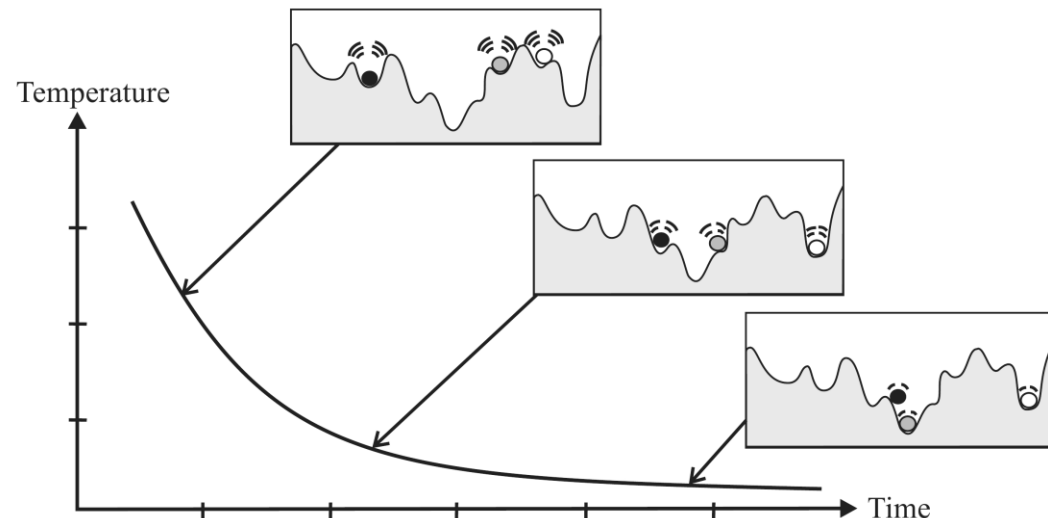
Ótimo **local** = *pBest*

SIMULATED ANNEALING (SA)

- Meta-heurística para **otimização** que consiste numa técnica de busca local;
- Inspirada no processo de **arrefecimento industrial** (aquecimento e controle de resfriamento) a fim de melhorar uma solução inicial;

checar a variação de cada iteração:
 $\Delta = f(S') - f(S)$

Em caso de piora:
Se $\Delta < 0$ então $S = S'$
Senão $\exp(-\Delta/T)$



TÓPICOS

- Introdução
- Problema
- Objetivos
- Conceitos Gerais
- **Trabalhos Relacionados**
- Proposta
- Considerações Finais

TRABALHOS RELACIONADOS

- A pesquisa foi apoiada nos trabalhos de:

XIA; WU; (2005)

Algoritmo PSO híbrido multiobjetivo que utiliza a abordagem hierárquica para a resolução do FJSP, apoiado no algoritmo de busca local Arrefecimento Simulado (SA) para a resolução do subproblema de programação.

ARANHA; G. (2016)

Enxame de Partículas híbrido de caráter multiobjetivo que também utiliza a abordagem hierárquica para a resolução do FJSP. A fim de resolver o subproblema de programação utiliza os algoritmos de busca local Reinício Aleatório de Subida de Colina (RRHC), Arrefecimento Simulado (SA) e Busca Tabu (TS).

TRABALHOS RELACIONADOS

- Diferenças entre trabalhos, considerando apenas SA:

ITEM	XIA, W.; WU, Z. (2005)	ARANHA, G. (2016)
Codificação estocástica	NÃO	SIM
Função de vizinhança (SA)	‘Swap’ de operações, por máquina	Caminho crítico
Controle antiestagnação	Não possui	Valida as soluções a cada iteração do PSO (amplia o espaço de busca)
Máquinas factíveis (problema flexível, 8x8)	Insere valores 9999 (‘infinito’) na matriz de tempos, inutilizando assim a máquina para operação não factível	Endereça a operação o maior nível possível na composição da solução

RESULTADOS

NxM (dataset)	ARANHA, G. (2016)			XIA, W.; WU, Z. (2005)
	RRHC	SA	TS	
8x8	14	14	14	15
10x10	7	7	7	7
15x10	11	11	11	12

TÓPICOS

- Introdução
- Problema
- Objetivos
- Conceitos Gerais
- Trabalhos Relacionados
- **Proposta**
- Considerações Finais

PROPOSTA

- Desenvolver um algoritmo híbrido, utilizando:
 - Modelagem apoiada na **abordagem hierárquica** do FJSP;
 - Algoritmo **PSO** para o roteamento;
 - **SA** para a programação;

PROPOSTA

- **Algoritmo PSO:** otimizar subproblema de **roteamento**;
 1. Inicializar **parâmetros** do PSO e SA;
 2. Gerar **nuvem de partículas** inicial estocásticamente;
 3. **Avaliar** cada partícula através do SA;
 - i. Definir *pBest* e *gBest*;
 4. Enquanto n^o máximo de iterações PSO não for atingido faça:
 - i. Guarda a nuvem no **registro de soluções anteriores**;
 - ii. Atualiza a nuvem de com base nas equações 1a e 1b;
 - iii. Valida as partículas geradas através do **controle de estagnação**;
 - iv. **Avaliar nuvem**:
 - Verifica a aptidão individual através do SA;
 - Define o novo *pBest_i* e *gBest* por comparação aos valores conhecidos;
 5. Retorna a programação otimizada (Gantt).

PROPOSTA

- Representação da partícula (roteamento):
 - Foi utilizado o conceito de de **partícula baseada em nível** (XIA; WU, 2005).

	JOB 1			JOB 2			JOB 3		
	$O_{1,1}$	$O_{1,2}$	$O_{1,3}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$	$O_{3,3}$
	1	1	3	2	1	2	3	1	2
Índice (O_{ij})	1	2	3	4	5	6	7	8	9
Valor (M)	1	1	3	2	1	2	3	1	2

Roteamento: partícula baseada em operações

PROPOSTA

- Representação da partícula (roteamento):
 - Para **cada operação**, organiza-se os recursos em níveis de acordo com seus tempos relativos;

		Machine				Priority order				
		M_1	M_2	M_3	M_4	1	2	3	⋮	4
$J1$	$O_{1,1}$	2	3	4	1	M_4	M_1	M_2		M_3
	$O_{1,2}$	3	1	8	2	M_2	M_4	M_1		M_3
$J2$	$O_{2,1}$	1	4	1	2	M_1	M_3	M_4		M_2
	$O_{2,2}$	5	3	2	9	M_3	M_2	M_1		M_4
	$O_{2,3}$	3	1	1	4	M_2	M_3	M_1		M_4
$J3$	$O_{3,1}$	7	6	3	5	M_3	M_4	M_2		M_1
	$O_{3,2}$	4	5	6	2	M_4	M_1	M_2		M_3



PROPOSTA

- Representação da partícula (roteamento):

		Machine				Priority order				
		M_1	M_2	M_3	M_4	1	2	3	⋮	4
$J1$	$O_{1,1}$	2	3	4	1	M_4	M_1	M_2		M_3
	$O_{1,2}$	3	1	8	2	M_2	M_4	M_1		M_3
$J2$	$O_{2,1}$	1	4	1	2	M_1	M_3	M_4		M_2
	$O_{2,2}$	5	3	2	9	M_3	M_2	M_1		M_4
	$O_{2,3}$	3	1	1	4	M_2	M_3	M_1		M_4
$J3$	$O_{3,1}$	7	6	3	5	M_3	M_4	M_2		M_1
	$O_{3,2}$	4	5	6	2	M_4	M_1	M_2		M_3

Job1		Job2			Job3	
$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$
2	1	3	2	2	4	4
↓	↓	↓	↓	↓	↓	↓
M_1	M_2	M_4	M_2	M_3	M_1	M_3

Partícula baseada em níveis

Partícula baseada em operações

PROPOSTA

- Geração estocástica de partícula:
 - Com a representação do **roteamento em níveis**, é possível gerar partículas de modo estocástico.

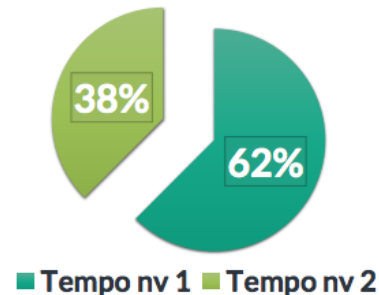
		Machine				Priority order				
		M_1	M_2	M_3	M_4	1	2	3	⋮	4
$J1$	$O_{1,1}$	2	3	4	1	M_4	M_1	M_2		M_3
	$O_{1,2}$	3	1	8	2	M_2	M_4	M_1		M_3
$J2$	$O_{2,1}$	1	4	1	2	M_1	M_3	M_4		
	$O_{2,2}$	5	3	2	9	M_3	M_2	M_1		
	$O_{2,3}$	3	1	1	4	M_2	M_3	M_1		
	$O_{2,4}$									
$J3$	$O_{3,1}$	7	6	3	5	M_3	M_4	M_2		M_1
	$O_{3,2}$	4	5	6	2	M_4	M_1	M_2		M_3

PROPOSTA

- Geração estocástica de partícula:
 - Otimiza a nuvem em direção a solução ótima;

Tempo nível 1 = 3s $1/3 = 0,333...$

Tempo nível 2 = 5s $1/5 = 0,2$



Exemplo de roleta para inicialização da população do algoritmo PSO e troca de recurso

PROPOSTA

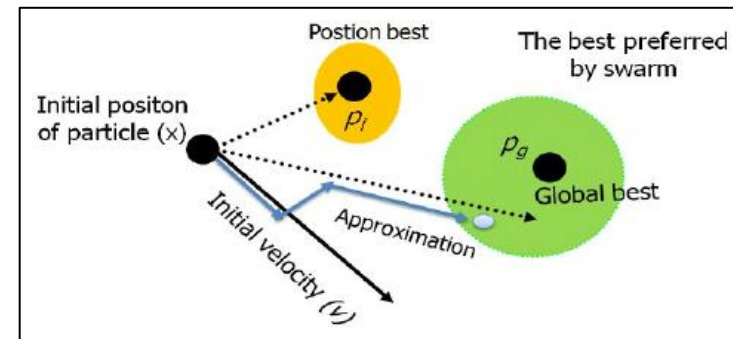
- Fator de inércia W:

- Dada a equação 1ª (V_{t+1}), desempenha papel **importante na otimização** do PSO (XIA; WU, 2005);
 - Valores **altos**: permite explorar o espaço de busca amplamente;
 - Valores **baixos**: permite explorar espaço de maneira refinada, guiando-se pelos melhores locais ($pBest$) e global ($gBest$);

Equação W:

$$W = W_{max} - \frac{W_{max} - W_{min}}{Iter_{max}} * Iter,$$

- W_{max} : Valor máximo do fator de inércia;
- W_{min} : Valor mínimo;
- $Iter_{max}$: Número máximo de iterações do PSO;
- $Iter$: Iteração corrente do algoritmo PSO.



Intuição sobre o papel dos termos da equação 1a

PROPOSTA

- **Controle antiestagnação:**
- Através da equação 1a, a exploração de busca do PSO é fortemente influenciado pelos melhores valor local (pBest) e global (gBest);
 - Pode ocorrer de ambos possuírem valores similares, o movimento das partículas pode ser fortemente induzida pelo melhor local corrente.
 - Algoritmo fica viesado no mínimo local, gerando, na maioria das vezes, a mesma solução.
- Conforme ARANHA (2016), é possível implementar um controle antiestagnação;

Controle antiestagnação simples:

- Criar um **registro de soluções** R ;
- A cada iteração do PSO, após **atualizar a nuvem de partículas**, faça
 - Se S_i **está contida em** R :
 - Enquanto S_i **ainda pertencer ao registro** R faça
 - Escolhe uma operação O_i pertencente a S_i **aleatoriamente**
 - Realiza a **troca do nível** em O_i
 - Retorna **nova solução** S_i ;

PROPOSTA

- Algoritmo SA: otimizar subproblema de **programação**;

```
procedimento  $SA(f(.), N(.), \alpha, SAmax, T_0, s)$ 
1   $s^* \leftarrow s$ ;   {Melhor solução obtida até então}
2   $IterT \leftarrow 0$ ; {Número de iterações na temperatura T}
3   $T \leftarrow T_0$ ;  {Temperatura corrente}
4  enquanto ( $T > 0$ ) faça
5      enquanto ( $IterT < SAmax$ ) faça
6           $IterT \leftarrow IterT + 1$ ;
7          Gere um vizinho qualquer  $s' \in N(s)$ ;
8           $\Delta = f(s') - f(s)$ ;
9          se ( $\Delta < 0$ )
10             então
11                  $s \leftarrow s'$ ;
12                 se ( $f(s') < f(s^*)$ ) então  $s^* \leftarrow s'$ ;
13             senão
14                 Tome  $x \in [0, 1]$ ;
15                 se ( $x < e^{-\Delta/T}$ ) então  $s \leftarrow s'$ ;
16         fim-se;
17     fim-enquanto;
18      $T \leftarrow \alpha \times T$ ;
19      $IterT \leftarrow 0$ ;
20 fim-enquanto;
21  $s \leftarrow s^*$ ;
22 Retorne  $s$ ;
fim SA;
```

Subproblema de **programação**

{1, 2, 5, 8}	{4, 6, 9}	{3, 7}
M1	M2	M3

PROPOSTA

- Função de vizinhança:

- Papel fundamental na exploração eficiente do espaço de busca;

Troca de 1-par por máquina aleatória:

- Recebe o **vetor de programação** M
- Escolhe uma **máquina aleatória** M_i
- Enquanto **$operações(M_i) < 2$**
 - Escolhe uma **máquina aleatória** M_i
- Em M_i , escolhe uma **operação aleatória** O_i
- Caso O_i seja a **última operação** de M_i
 - O_i **troca de valor** com O_1
- Senão
 - O_i **troca de valor** com O_{i+1}
- Retorna o **vetor de programação** M^*

Subproblema de **programação**

{1, 2, 5, 8}	{4, 6, 9}	{3, 7}
M1	M2	M3

PROPOSTA

- Função de vizinhança:

- Papel fundamental na exploração eficiente do espaço de busca;

Troca de 1-par por máquina aleatória:

- Recebe o **vetor de programação** M
- Escolhe uma **máquina aleatória** M_i
- Enquanto **$operações(M_i) < 2$**
 - Escolhe uma **máquina aleatória** M_i
- Em M_i , escolhe uma **operação aleatória** O_i
- Caso O_i seja a **última operação** de M_i
 - O_i **troca de valor** com O_1
- Senão
 - O_i **troca de valor** com O_{i+1}
- Retorna o **vetor de programação** M^*

Subproblema de programação

{1, 2, 5, 8}	{4, 6, 9}	{3, 7}
--------------	-----------	--------

M1

M2

M3



PROPOSTA

- Função de vizinhança:

- Papel fundamental na exploração eficiente do espaço de busca;

Troca de 1-par por máquina aleatória:

- Recebe o **vetor de programação** M
- Escolhe uma **máquina aleatória** M_i
- Enquanto **$operações(M_i) < 2$**
 - Escolhe uma **máquina aleatória** M_i
- Em M_i , escolhe uma **operação aleatória** O_i
- Caso O_i seja a **última operação** de M_i
 - O_i **troca de valor** com O_1
- Senão
 - O_i **troca de valor** com O_{i+1}
- Retorna o **vetor de programação** M^*

Subproblema de **programação**

{1, 2, 5, 8}	{4, 6, 9}	{3, 7}
--------------	-----------	--------

M1

M2

M3



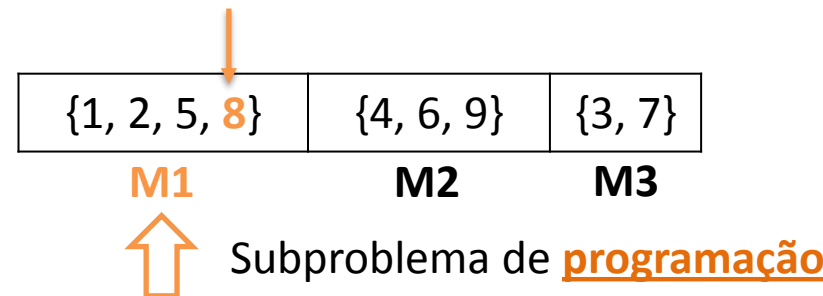
PROPOSTA

- Função de vizinhança:

- Papel fundamental na exploração eficiente do espaço de busca;

Troca de 1-par por máquina aleatória:

- Recebe o **vetor de programação** M
- Escolhe uma **máquina aleatória** M_i
- Enquanto **$operações(M_i) < 2$**
 - Escolhe uma **máquina aleatória** M_i
- Em M_i , escolhe uma **operação aleatória** O_i
- Caso O_i seja a **última operação** de M_i
 - O_i **troca de valor** com O_1
- Senão
 - O_i **troca de valor** com O_{i+1}
- Retorna o **vetor de programação** M^*



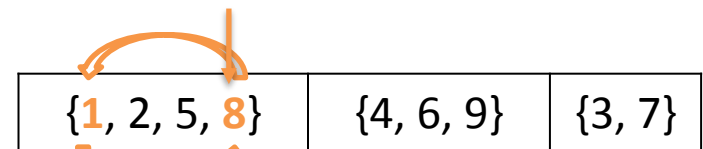
PROPOSTA

- Função de vizinhança:

- Papel fundamental na exploração eficiente do espaço de busca;

Troca de 1-par por máquina aleatória:

- Recebe o **vetor de programação** M
- Escolhe uma **máquina aleatória** M_i
- Enquanto **$operações(M_i) < 2$**
 - Escolhe uma **máquina aleatória** M_i
- Em M_i , escolhe uma **operação aleatória** O_i
- **Caso O_i seja a última operação de M_i**
 - O_i **troca de valor** com O_1
- Senão
 - O_i **troca de valor** com O_{i+1}
- Retorna o **vetor de programação** M^*



M1

M2

M3



Subproblema de programação

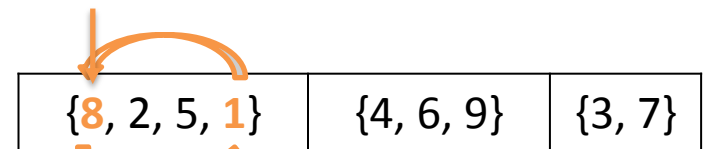
PROPOSTA

- Função de vizinhança:

- Papel fundamental na exploração eficiente do espaço de busca;

Troca de 1-par por máquina aleatória:

- Recebe o **vetor de programação** M
- Escolhe uma **máquina aleatória** M_i
- Enquanto **$operações(M_i) < 2$**
 - Escolhe uma **máquina aleatória** M_i
- Em M_i , escolhe uma **operação aleatória** O_i
- Caso O_i seja a **última operação** de M_i
 - O_i **troca de valor** com O_1
- Senão
 - O_i **troca de valor** com O_{i+1}
- Retorna o **vetor de programação** M^*



M1

M2

M3



Subproblema de programação

PROPOSTA

- Função de vizinhança:

- Papel fundamental na exploração eficiente do espaço de busca;

Troca de 1-par por máquina aleatória:

- Recebe o **vetor de programação** M
- Escolhe uma **máquina aleatória** M_i
- Enquanto ***operações*** $(M_i) < 2$
 - Escolhe uma **máquina aleatória** M_i
- Em M_i , escolhe uma **operação aleatória** O_i
- Caso O_i seja a **última operação** de M_i
 - O_i **troca de valor** com O_1
- Senão
 - O_i **troca de valor** com O_{i+1}
- **Retorna o vetor de programação** M^*

Subproblema de **programação**

{8, 2, 5, 1}	{4, 6, 9}	{3, 7}
M1	M2	M3

PROPOSTA

- Resultados:
 - Testes executados*: 10
 - Base: KACEM

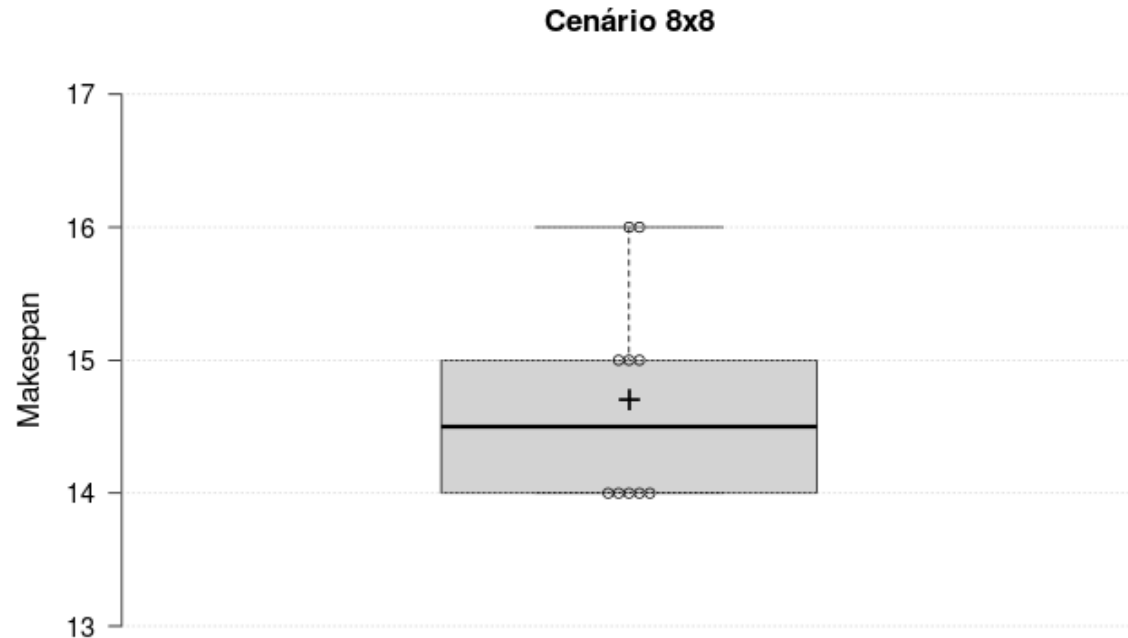
NxM (dataset)	ARANHA, G. (2016)			XIA, W.; WU, Z. (2005)	Diego Cavalca
	RRHC	SA	TS		
8x8	14	14	14	15	14
10x10	7	7	7	7	8
15x10	11	11	11	12	14

* A base 15x10 não permitiu executar 10 testes a tempo da apresentação.

PROPOSTA

- **Resultado 8x8:**
 - Melhor makespan: **14**
 - Atingiu o benchmark: **SIM**
 - Margem de aceite*: **SIM**

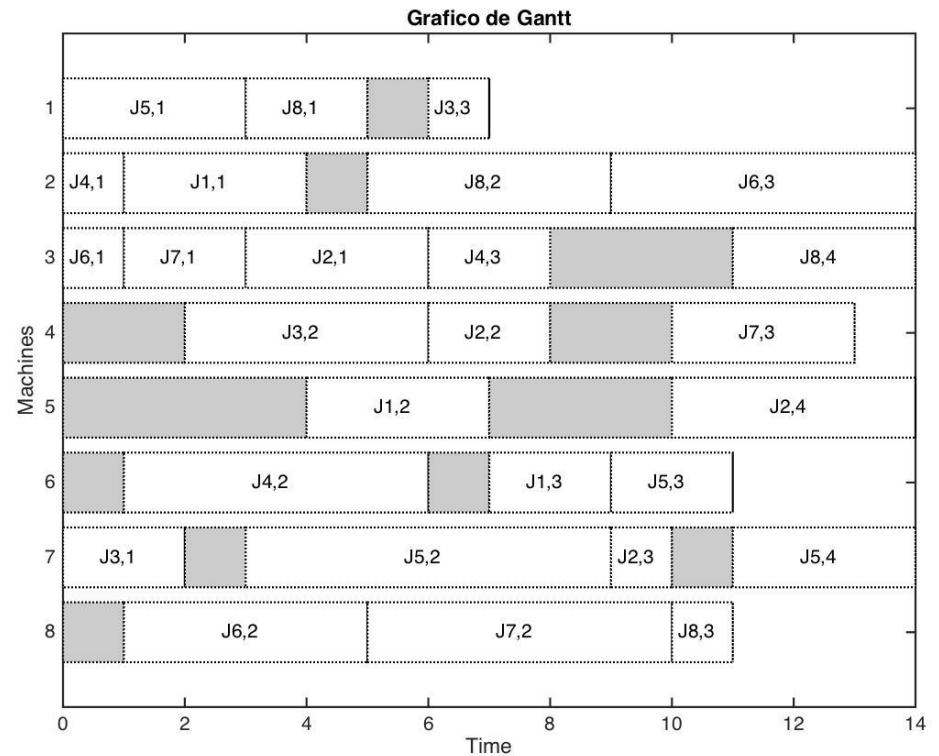
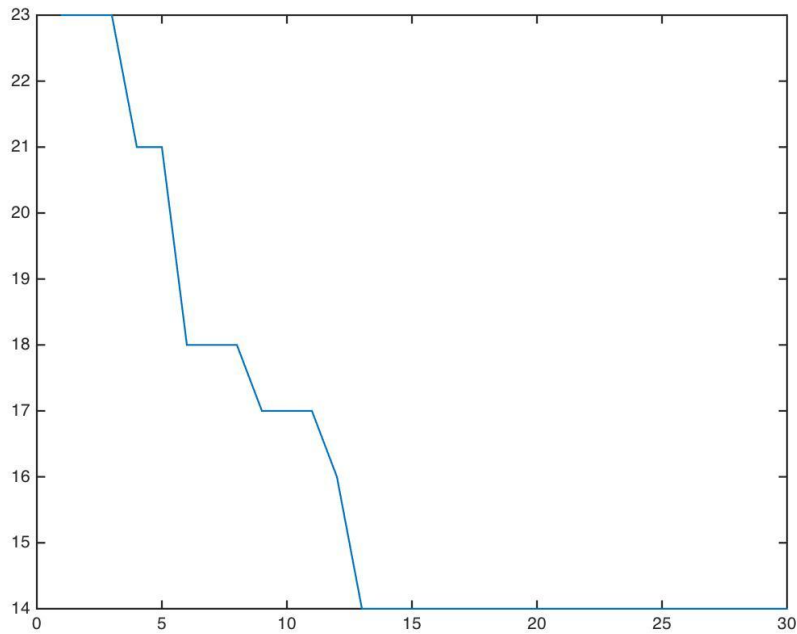
PARÂMETRO	VALOR
(PSO) Iterações PSO	30
(PSO) Tam. da nuvem	50
(PSO) Fator inércia (w)	0.4~1.2
(PSO) C1 = C2	2
Recursos descartados	4
(SA) Temp. Inicial	3
(SA) Temp. Final	0.01
(SA) Fator de resfr.	0.9
(SA) Iter. por temp.	20



* Valor obtido no máximo 10% acima do benchmark conhecido

PROPOSTA

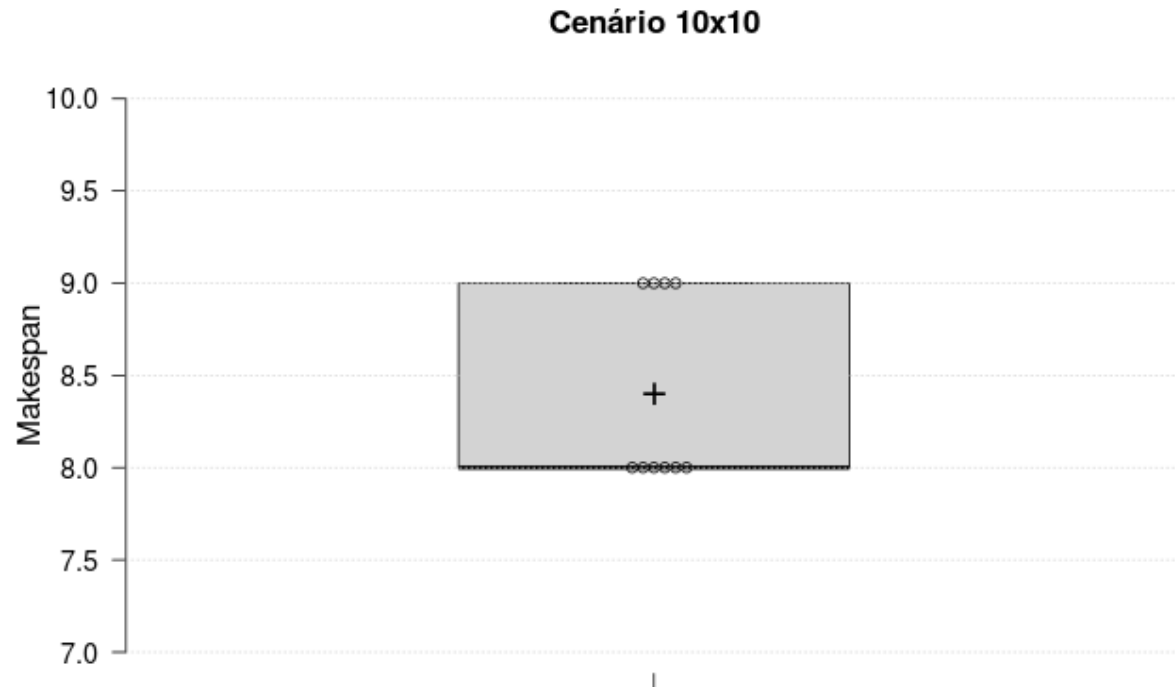
- Resultado 8x8:



PROPOSTA

- **Resultado 10x10:**
 - Melhor makespan: **8**
 - Atingiu o benchmark: **NÃO**
 - Margem de aceite*: **SIM**

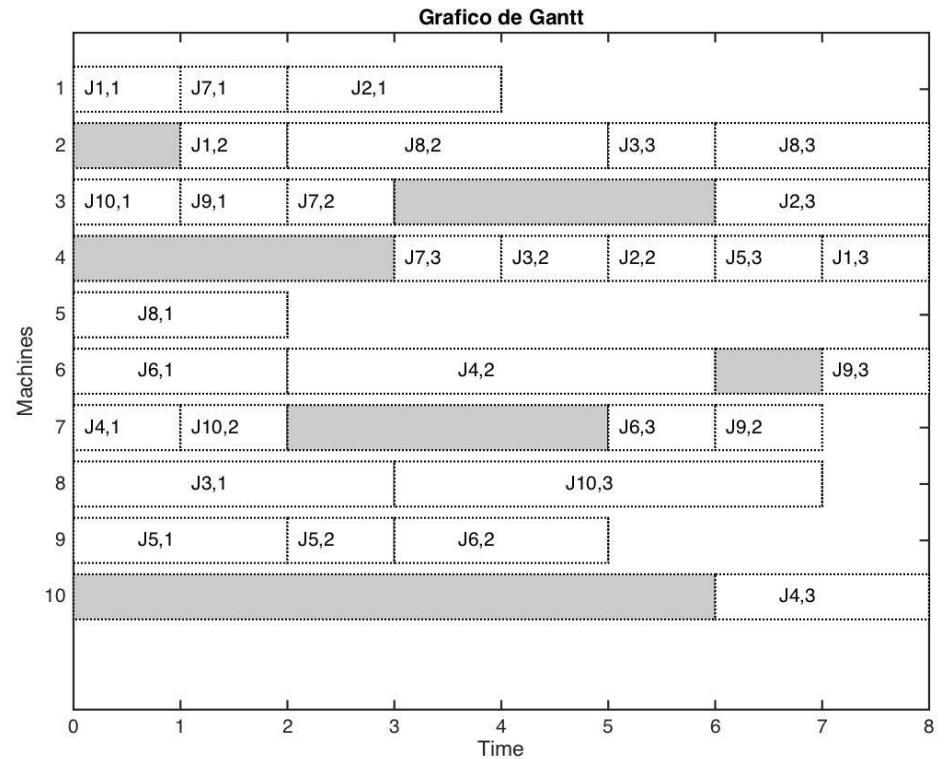
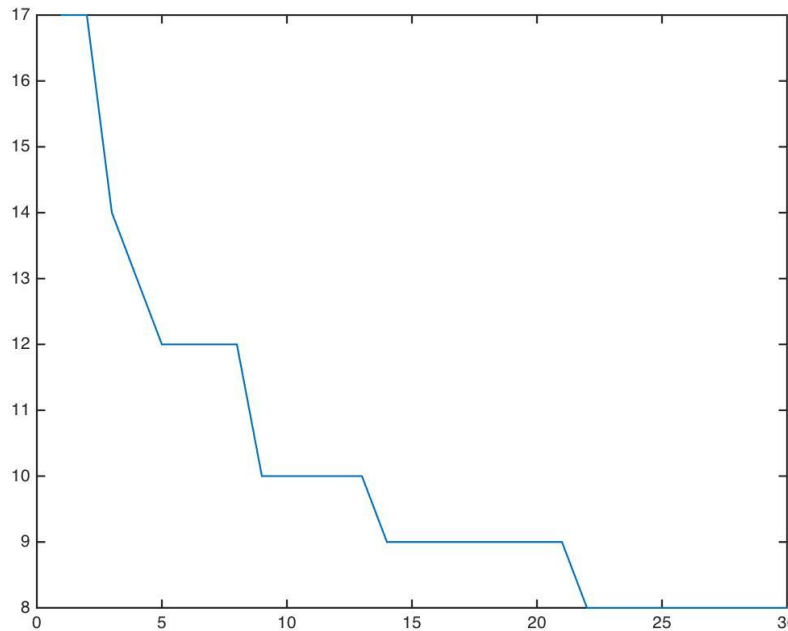
PARÂMETRO	VALOR
(PSO) Iterações	30
(PSO) Tam. da nuvem	50
(PSO) Fator inércia (w)	0.4~1.2
(PSO) C1 = C2	2
Recursos descartados	4
(SA) Temp. Inicial	5
(SA) Temp. Final	0.01
(SA) Fator de resfr.	0.9
(SA) Iter. por temp.	20



* Valor obtido no máximo 10% acima do benchmark conhecido


PROPOSTA

- Resultado 10x10:



PROPOSTA

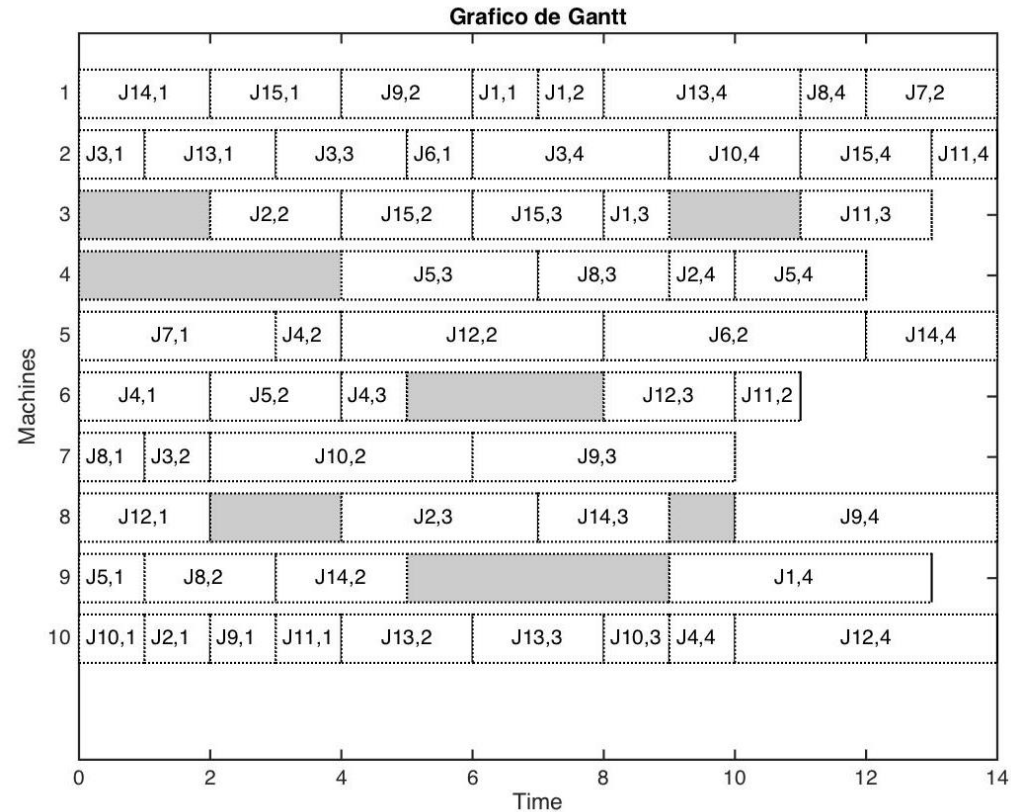
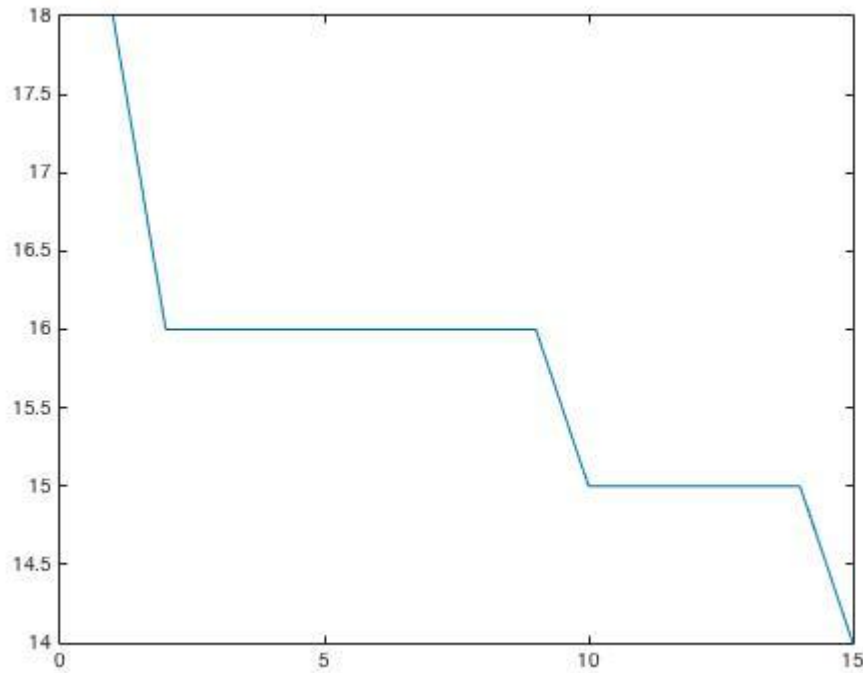
- **Resultado 15x10:**
 - Melhor makespan: **14**
 - Atingiu o benchmark: **NÃO**
 - Margem de aceite*: **NÃO**
 - ~16% Xia; Wu (2005)
 - ~27% Aranha (2016)

PARÂMETRO	VALOR
(PSO) Iterações	15 
(PSO) Tam. da nuvem	100
(PSO) Fator inércia (w)	0.4~1.2
(PSO) C1 = C2	2
Recursos descartados	4
(SA) Temp. Inicial	10
(SA) Temp. Final	0.01
(SA) Fator de resfr.	0.95
(SA) Iter. por temp.	20

* Valor obtido no máximo 10% acima do benchmark conhecido

PROPOSTA

- Resultado 15x10:**



TÓPICOS

- Introdução
- Problema
- Objetivos
- Conceitos Gerais
- Trabalhos Relacionados
- Proposta
- **Considerações Finais**

CONSIDERAÇÕES FINAIS

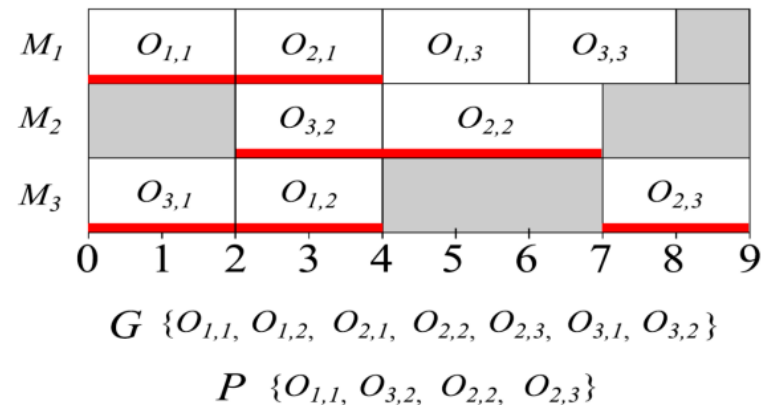
- **Algoritmo PSO proposto:**
 - Teve desempenho aceitável no que tange a função objetivo (*makespan*) para as 3 bases consideradas;
 - Atingiu benchmark na base 8x8;
 - Ficou dentro da margem de aceite estabelecida para o *dataset* 10x10 e próximo desta métrica no 15x10;
 - Computacionalmente custoso;
 - Código não otimizado (paralelismo);
 - Busca local e **função de vizinhança** tem um papel fundamental neste quesito;

CONSIDERAÇÕES FINAIS

- Trabalhos futuros:

- Função de **Vizinhança**;

- Swap de **1-par de operações** por máquina aleatória se mostra computacionalmente pouco eficiente;
 - Exige em média **800~1.000 iterações** do SA para um resultado aceitável (melhoria da solução original);
- Alternativa:
 - Vizinhança baseada em **caminho crítico** (Mastrolilli e Gambardella, 2000);

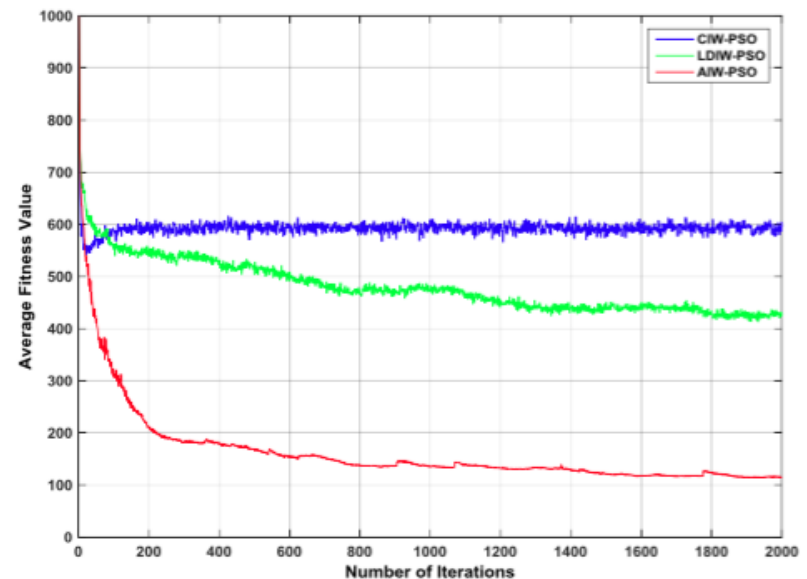
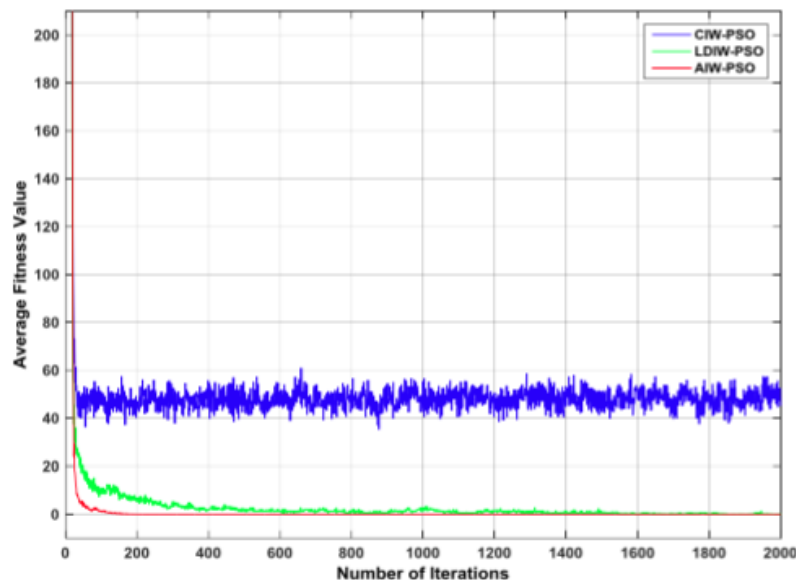


CONSIDERAÇÕES FINAIS

- Trabalhos futuros:

- Fator de Inércia Adaptativo Individual (AIW-PSO);

- Peso de Inércia Baseado nos **Melhores, Piores e Aptidão Individual** da Nuvem de Partículas.
 - Otimiza a **convergência** do PSO;
 - Proposto para **problemas contínuos** (SPAVIERI et. al, 2015);

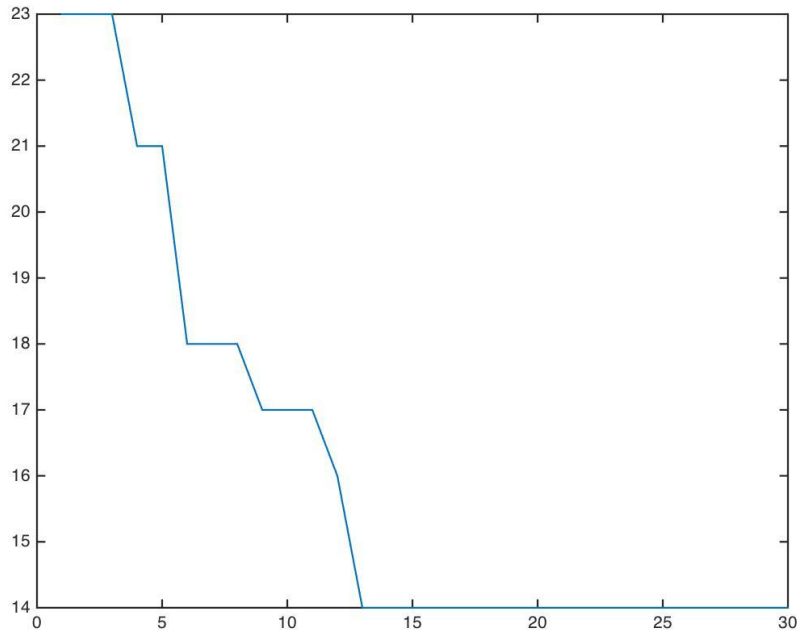


Fator de Inércia Adaptativo Individual (AIW-PSO);

CENÁRIO FJSP-8x8

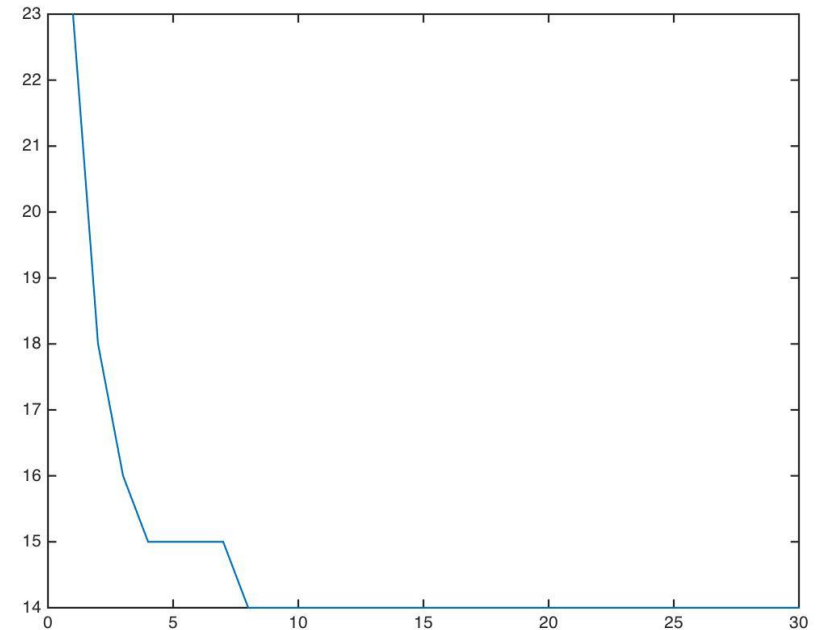
XIA; WU (2005)

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{\text{iter}_{\max}} \text{iter}_i$$



SPAVIERI et. al (2015)*

$$\omega_i^{k+1} = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \frac{(f_i^k - f_{\text{worst}}^k)}{(f_{\text{best}}^k - f_{\text{worst}}^k)}$$



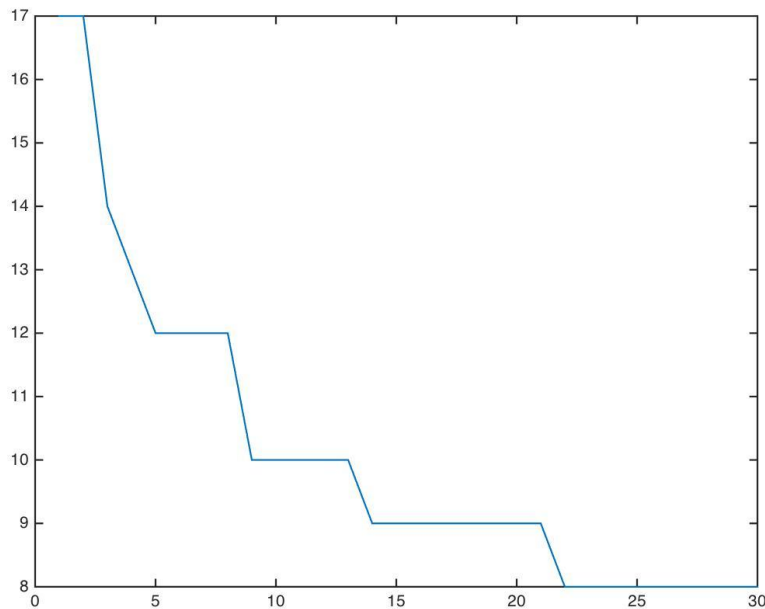
* Adaptado ao contexto discreto do FJSP

Fator de Inércia Adaptativo Individual (AIW-PSO);

CENÁRIO FJSP-10x10

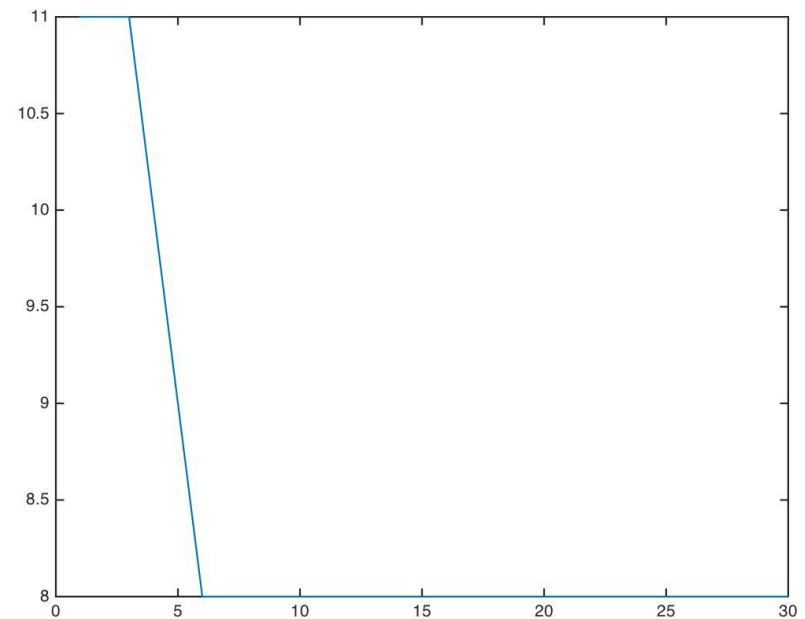
XIA; WU (2005)

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{\text{iter}_{\max}} \text{iter}_i$$



SPAVIERI et. al (2015)*

$$\omega_i^{k+1} = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \frac{(f_i^k - f_{\text{worst}}^k)}{(f_{\text{best}}^k - f_{\text{worst}}^k)}$$



* Adaptado ao contexto discreto do FJSP

CONSIDERAÇÕES FINAIS

- **Trabalhos futuros:**
 - **Otimização** do algoritmo;
 - Controle antiestagnação;
 - Validação de operações iterativas;
 - Paralelismo;
 - Ajuste de parâmetros de acordo com a sensibilidade dos testes;
 - Etc.

Obrigado.