Universidade do Minho
Escola de Engenharia

Ana Sofia Alves Ferreira

Aplicação do *Particle Swarm Optimization* a um problema de escalonamento de máquinas paralelas não relacionadas com tempos de *setup* dependentes da sequência

novembro de 2020

Universidade do Minho
Escola de Engenharia

Ana Sofia Alves Ferreira

Aplicação do *Particle Swarm Optimization* a um problema de escalonamento de máquinas paralelas não relacionadas com tempos de *setup* dependentes da sequência

Dissertação de Mestrado
Mestrado em Engenharia de Sistemas

Trabalho efetuado sob a orientação de:
Professora Doutora Maria Leonilde Varela
Professora Doutora Ana Cristina Braga

novembro de 2020

AGRADECIMENTOS

Este documento encerra um capítulo importante da minha vida e quero utilizar este espaço para agradecer a todos os que me ajudaram nesta fase.

Quero deixar os meus sinceros agradecimentos à minha orientadora, a professora Maria Leonilde Varela, por todo o apoio, motivação, ensinamentos, colaboração e disponibilidade que foram essenciais para a realização deste trabalho.

À minha coorientadora, a professora Ana Cristina Braga, agradeço a disponibilidade e o apoio na realização deste trabalho.

Ao professor Lino Costa, agradeço a ajuda com a elaboração e implementação do modelo.

Deixo também um agradecimento muito especial à minha família por todo o amor, compreensão e carinho.

Por fim, agradeço a todos os meus amigos pelo apoio e por viverem esta fase comigo.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Esta dissertação aborda um problema de escalonamento de máquinas paralelas não relacionadas com tempos de *setup* dependentes da sequência e o objetivo é minimizar o *makespan* de um conjunto de trabalhos. Para tal, é implementado o algoritmo *Particle Swarm Optimization*, que é usado para resolver um problema da literatura, dividido em pequenos e grandes problemas, consoante o número de trabalhos que são utilizados.

O desempenho deste algoritmo foi avaliado através de uma análise comparativa das suas soluções com as soluções obtidas usando o *Ant Colony Optimization*, o *Simulated Annealing* e o *Genetic Algorithm*. Na implementação do algoritmo em estudo foi utilizado a *toolbox particleswarm* do software MATLAB, que tenta otimizar utilizando o algoritmo *Particle Swarm Optimization*.

Os resultados da implementação mostram que para pequenos problemas o *Particle Swarm* consegue superar o *Genetic Algorithm* em algumas instâncias, sendo que os outros três algoritmos apresentam valores de *makespan* inferiores. Para grandes problemas, é clara a superioridade do *Particle Swarm* em relação ao *Genetic Algorithm*, no entanto, relativamente aos restantes algoritmos o mesmo não acontece. Existe também a tendência crescente da variação percentual entre os algoritmos à medida que o número de máquinas aumenta para o mesmo número de trabalhos.

PALAVRAS-CHAVE

Escalonamento da produção; *Makespan*; Máquinas paralelas; *Particle Swarm Optimization*

ABSTRACT

This dissertation addresses the unrelated parallel machine scheduling problem with sequence-dependent setup times and the objective is to minimize the makespan of a set of jobs. It is implemented the Particle Swarm Optimization, used to solve a problem from the literature, divided into small and large problems, depending on the number of jobs that are used.

Particle Swarm performance is evaluated through a comparative analysis between its solutions and the solutions obtained using Ant Colony Optimization, Simulated Annealing and Genetic Algorithm. For implementing the algorithm under study, the particle swarm toolbox from the MATLAB software was used, which tries to optimize using the Particle Swarm Optimization.

The results of the implementation show that for small problems the Particle Swarm can overcome the Genetic Algorithm in some instances, with the other three algorithms having lower makespan values. For large problems, the Particle Swarm superiority over Genetic Algorithm is clear, however, in relation to the other algorithms the same does not happen. There is also as increasing trend in the percentage variation between the algorithms as the number of machines increases for the same number of jobs.

KEYWORDS

Scheduling Production; Makespan; Parallel Machines; Particle Swarm Optimization

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÓNIMOS

ACO – *Ant Colony Optimization*

CMFOA – *Collaborative Multiobjective Fruit Fly Optimization Algorithm*

DPSO – *Discrete Particle Swarm Optimization*

EP – Escalonamento da Produção

EPMP – Escalonamento da produção de máquinas paralelas

EPMU – Escalonamento da produção de máquinas únicas

GA – *Genetic Algorithm*

MOPSO – *Multi-objective Particle Swarm Optimization*

NTS – *Neighbourhood Search Techniques*

RBS – *Recovering Beam Search*

PSO – *Particle Swarm Optimization*

PIM – Programação inteira mista

PLIM – Programação linear inteira mista

SA – *Simulated Annealing*

TS – *Tabu Search*

UPMSPST – *Unrelated Parallel Machines Scheduling Problem with Sequence-Dependent Setup Times*

1. INTRODUÇÃO

1.1 Enquadramento

A gestão da produção em ambiente industrial enfrenta diversos desafios devido às mudanças dinâmicas na produção e à volatilidade dos mercados. As empresas procuram cada vez mais ter curtos períodos de entrega de forma económica devido ao aumento da competitividade nos mercados emergentes (Schuh, Reuter, Prote, Brambring, & Ays, 2017). Os problemas industriais como o escalonamento, nomeadamente o sequenciamento de trabalhos, entre outros são afetados pelas exigências dos clientes por variedade de pedidos e rapidez na entrega (Liao, Gen, Tiwari, & Chang, 2013).

O escalonamento da produção (EP) é um tema bastante estudado, definindo-se como um processo de tomada de decisão muito importante para os sistemas de fabricação e produção. Tem como objetivo otimizar uma ou mais medidas de desempenho através da alocação de um conjunto de recursos a um conjunto de tarefas/trabalhos ao longo de um intervalo de tempo (Pinedo, 2016). Os problemas de EP são comuns no planeamento e controlo da produção, uma vez que, sempre que uma ordem de produção é lançada é necessário definir a sequência de execução dos trabalhos e determinar os recursos que serão utilizados.

Dependendo da necessidade ou não de alocação dos recursos, o escalonamento da produção pode ser em máquinas únicas ou máquinas paralelas. Segundo Carmo-Silva (2015) no caso do EP em processador único apenas existem problemas de sequenciamento. No entanto, no caso de máquinas paralelas, surge o problema de afetação dos trabalhos às máquinas aliado aos problemas de sequenciamento. De modo geral, o problema de sequenciamento consiste em encontrar a ordem pela qual os trabalhos devem ser processados no sistema de produção. Por outro lado, o problema de afetação pretende descobrir o processador, de entre os disponíveis, que será responsável por executar o fabrico do trabalho.

Os problemas de EP de máquinas paralelas têm sido objeto de estudo nas últimas décadas. No entanto, o caso em que as máquinas não estão relacionadas foi muito menos estudado, especialmente quando são utilizados os tempos de *setup* (Vallada & Ruiz, 2011).

Segundo Rabadi, Moraga e Al-Salem (2006) as máquinas são consideradas não relacionadas quando os tempos de processamento dos trabalhos dependem das máquinas às quais eles

são atribuídos e quando não há relação entre as velocidades das máquinas. Os tempos de *setup* são dependentes uma vez que os seus valores dependem tanto da sequência do trabalho como da matriz de tempos de *setup* de cada máquina.

São muitos os autores que consideraram o uso de algoritmos, heurísticas e meta-heurísticas para a resolver problemas de escalonamento de máquinas paralelas não relacionadas. Especialmente para problemas complicados ou grandes instâncias de problemas, as meta-heurísticas são frequentemente capazes de oferecer uma melhor troca entre a qualidade da solução e o tempo de computação.

1.2 Motivação

Para que uma indústria tenha sucesso são necessários a otimização dos processos de produção e o uso de métodos inovadores que a diferenciem no mercado. Assim, o escalonamento da produção é crucial para que uma empresa seja competitiva e seja superior em relação aos seus concorrentes. Este traduz-se na alocação de tarefas a recursos escassos, num determinado período, com a pretensão de otimização de uma ou mais medidas de desempenho.

Um escalonamento adequado evita custos de produção desnecessários ou cancelamento de encomendas relativas a atrasos, daí a sua importância para processo de produção e para a indústria a que pertence.

1.3 Objetivos

Com esta dissertação pretende-se avaliar o comportamento do algoritmo *Particle Swarm Optimization (PSO)* aplicado ao problema de escalonamento de máquinas paralelas não relacionadas, com o objetivo de minimizar o *makespan* de um conjunto de trabalhos. A avaliação será feita através da comparação dos resultados com as soluções obtidas pela aplicação de outros algoritmos presentes na literatura.

Espera-se que no final do projeto se consiga provar qual o algoritmo mais eficaz e com potencial para resolver problemas de escalonamento da produção de máquinas paralelas não relacionadas em contexto industrial.

Assim, delinearam-se como objetivos:

1. Identificação de problemas de escalonamento da produção, com ênfase em problemas de máquinas paralelas.
2. Pesquisa de conhecimento relativo aos métodos de escalonamento da produção adequados para resolver o problema em estudo.
3. Contextualização do atual estado de arte do tema a desenvolver.
4. Aplicação do método de escalonamento da produção ao caso de estudo.
5. Avaliação do método de escalonamento aplicado através de análise comparativa em termos das medidas de desempenho.
6. Análise comparativa com o trabalho presente na literatura acerca do problema considerado.

1.4 Metodologia

A metodologia utilizada nesta dissertação é o caso de estudo (*case study*) que permite explorar o processo em questão e responder às perguntas "Porquê?", "O quê?" ou "Como?". Este trabalho pode ser dividido em duas fases principais, a primeira corresponde à pesquisa para responder às perguntas "O quê?" e "Como?" e a segunda de aplicação e análise do caso de estudo com o objetivo de responder à pergunta "Porquê?".

Na primeira fase foi feita uma pesquisa relativa ao processo de escalonamento da produção, nomeadamente os tipos de problemas que daí advêm. Deu-se especial enfase ao ambiente de produção de máquinas paralelas, com a apresentação das suas características, as subcategorias, os métodos de escalonamento e as medidas de desempenho. É ainda feita a revisão de literatura fazendo referência a alguns trabalhos efetuados relativos ao escalonamento da produção, com foco em ambientes de máquinas paralelas e com o objetivo de minimizar o *makespan*. Nesta fase são utilizados todos os tipos de referências bibliográficas. Foram utilizadas as fontes primárias com a consulta de teses e dissertações, as secundárias através de livros e artigos e, por fim, as terciárias com o objetivo de identificar os outros dois tipos de fontes mencionados.

A segunda fase do trabalho foi iniciada com a análise de um artigo retirado da literatura sobre máquinas paralelas não relacionadas, aplicando o algoritmo e comparando os resultados obtidos com os encontrados na literatura.

1.5 Estrutura

O documento está organizado da seguinte forma. No primeiro capítulo é feita uma breve introdução do tema abordado na dissertação, apresentando também os objetivos, a motivação para a realização do trabalho e a forma como está estruturado.

O segundo capítulo corresponde à contextualização na área de estudo, o escalonamento da produção, apresentando ambientes de produção, métodos de escalonamento e medidas de desempenho. No terceiro capítulo é feita a revisão de literatura do tema em estudo, com a apresentação do estado de arte que serve de base a esta dissertação.

No quarto capítulo é introduzido o caso de estudo, são descritos os dados do problema e é feita a explicação da implementação do algoritmo em estudo. O quinto capítulo corresponde à análise de resultados e no sexto capítulo é feita a conclusão do trabalho desenvolvido.

2. ESCALONAMENTO DA PRODUÇÃO

2.1 Introdução

O escalonamento, como um problema a que o decisor tem de responder, tem um papel muito importante no processo de tomada de decisão em diferentes ambientes, desde a produção, ao processamento de informações, ao transporte e distribuição ou outros tipos de serviços (Pinedo, 2016).

O escalonamento da produção (EP) implica a definição do tempo inicial e final de processamento de cada trabalho de determinado conjunto, assim como a sua alocação aos recursos, embora existam determinadas restrições que podem envolver as tarefas e/ou os recursos (Artiba & Elmaghraby, 1996).

Brucker (1995) destaca que o escalonamento de um sistema de produção está dependente de vários fatores, de realçar, as ordens de fabrico, os recursos disponíveis, as máquinas disponíveis e as operações a realizar em cada uma, sendo que poderão existir temos de processamento distintos em cada máquina. O objetivo é sequenciar e alocar eficientemente os recursos e as máquinas disponíveis às operações a executar.

O escalonamento da produção referido, frequentemente, como programação da produção, tem como objetivo a programação da execução de um conjunto de trabalhos num conjunto de máquinas ou recursos, durante um determinado intervalo de tempo (Varela, 2007).

2.2 Problemas de Escalonamento

Os problemas de escalonamento começaram a surgir na literatura no início dos anos cinquenta (Pinedo, 2016) e, desde então, têm sido o objeto de estudo de inúmeras publicações. Estes problemas são de grande importância, uma vez que podem ser encontrados em aplicações reais.

Para resolver um problema de escalonamento é necessário responder a duas questões:

- Quais os recursos que devem ser alocados para executar cada tarefa?
- Quando cada tarefa deve ser executada?

Assim, o escalonamento requer decisões tanto de sequenciamento dos trabalhos como de alocação dos recursos. O sequenciamento corresponde a uma permutação dos trabalhos ou a ordem com que estes deverão ser processados na máquina. Por outro lado, a alocação dos recursos refere-se à escolha de qual máquina irá processar os trabalhos (Baker & Trietsch, 2009).

Nos problemas de escalonamento tem-se um conjunto de *n* tarefas $\mathrm{T} = \{\mathrm{T}_1; \ldots; \mathrm{T}_\mathrm{n}\}$ e um conjunto de *m* máquinas $\mathrm{M} = \{\mathrm{M}_1; \ldots; \mathrm{M}_\mathrm{m}\}$, no qual T e M são conjuntos finitos. Estes problemas têm como objetivo a afetação das tarefas às máquinas e a definição dos períodos que cada tarefa é processada na máquina. O planeamento desta etapa será ótimo se minimizar ou maximizar o critério de otimização ou a função objetivo que se pretende atingir (Brucker, 1995). O escalonamento pode ser representado através de gráficos de Gantt, como representados na Figura 1. Os gráficos de Gantt tanto podem ser orientados para as máquinas (Figura 1 a)) ou para os trabalhos (Figura 1 b)).

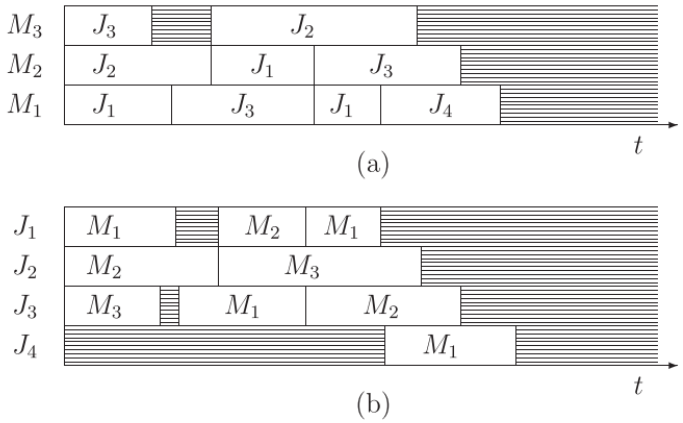

Figura 1 – Gráficos de Gantt orientados para máquinas e trabalhos (Brucker, 1995).

2.2.1 Sistemas ou ambientes de produção

Para a conclusão da produção de um artigo podem ser necessárias realizar uma ou mais operações, em uma ou mais máquinas ou processadores. Existem duas classes genéricas de ambientes de produção: uni-fase e multi-fase.

Em ambientes de produção uni-fase, os trabalhos são processados apenas numa fase ou estágio. O objetivo é fazer o sequenciamento dos trabalhos, ou seja, a ordem pela qual devem ser processados, e caso seja necessário, a alocação dos recursos, ou seja, decidir em que máquinas, entre as disponíveis, irão ser processados os trabalhos dessa fase. Dentro dos

ambientes uni-fase pode-se ter escalonamento com máquinas únicas ou paralelas, dependendo da necessidade ou não de alocação de recursos.

Existem ambientes de produção mais complexos, denominados de multi-fase. Neste tipo de ambientes, os trabalhos necessitam de mais do que uma operação para o seu processamento, o que significa que têm mais do que uma fase na sua produção. Estão incluídos neste tipo os ambientes de produção em linha (*flow-shop*), oficina clássica (*job-shop*) ou sistema aberto (*open-shop*) (Varela, 2007; Artiba & Elmaghraby, 1996).

2.2.2 Escalonamento da produção de máquinas únicas

O escalonamento da produção em máquinas únicas (EPMU) é, fundamentalmente, um problema de sequenciamento de trabalhos, tendo apenas como objetivo a definição da ordem com que os trabalhos ou lotes devem ser processados em determinada máquina, ou seja, os trabalhos necessitam apenas de uma operação (Morton & Pentico, 1993).

Existe apenas uma fila de espera e, consequentemente, uma única decisão a tomar, que consiste na determinação da ordem pela qual os trabalhos devem ser executados no processador, Figura 2.

Figura 2 – Sistema de máquina única.

Os problemas mais complexos podem ser decompostos em problemas de máquina única, de forma a que a sua resolução seja mais rápida e simples (Madureira & Pereira, 2010; Madureira, Pereira, & Sousa, 2011; Madureira, Ramos, & Do Carmo Silva, 2002).

2.2.3 Escalonamento da produção de máquinas paralelas

No escalonamento da produção de máquinas paralelas (EPMP) surge o problema de afetação dos trabalhos às máquinas aliado ao problema de sequenciamento. De modo geral, o problema de sequenciamento consiste em encontrar a ordem pela qual os trabalhos devem ser processados no sistema de produção. Por outro lado, o problema de afetação pretende

descobrir a máquina, de entre as disponíveis que será responsável por executar o trabalho, Figura 3.

Este possibilita a realização de um conjunto de trabalhos/tarefas em simultâneo, em que cada trabalho é constituído apenas por uma operação. Basicamente, funcionam como um conjunto de máquinas únicas, que no sistema global, realizam a mesma função todos em paralelo (Blazewicz, Domscke, & Pesch, 1996).

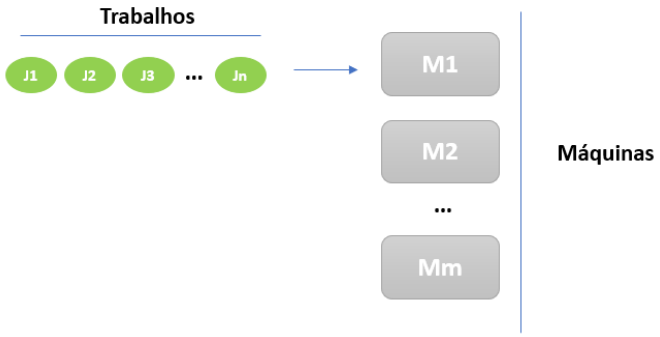

Figura 3 – Sistema de máquinas paralelas.

Dentro da categoria de máquinas paralelas existem três subcategorias, como apresentado na Figura 4.

Figura 4 – Subcategorias das máquinas paralelas.

Sistema de máquinas paralelas idênticas

Neste sistema, todas as máquinas operam em paralelo a uma velocidade de execução igual para todos os trabalhos. Isto significa que para determinado trabalho o tempo de processamento da máquina um ou dois será o mesmo (Brucker, 1995).

Sistema de máquinas paralelas uniformes

No caso das máquinas uniformes, a velocidade de execução não é a mesma para todas as máquinas. Como o que varia é a velocidade a que as máquinas operam, então cada trabalho terá um tempo de processamento específico (Brucker, 1995).

Sistema de máquinas paralelas não relacionadas

Neste caso, o tempo de processamento depende da relação produto-máquina, ou seja, cada máquina tem uma velocidade própria de processamento para cada um dos trabalhos a considerar. Assim, a velocidade, comparando com as máquinas idênticas, será diferente em cada máquina e, comparando com as máquinas uniformes, será diferente para cada trabalho (Blazewicz et al., 1996).

2.2.4 Características adicionais de um problema de escalonamento

Para além do número de recursos e da sua disposição, em qualquer ambiente de produção existem outros fatores a variar que tornam o problema mais complexo.

Interrupção de tarefas

A preempção significa a interrupção do processamento de um trabalho em qualquer altura e a colocação de outro trabalho na máquina. Mais tarde, o trabalho interrompido pode voltar para a mesma máquina ou continuar o seu processamento numa máquina ou recurso de produção diferentes (Rahmani, Mahdavi, Moradi, Khorshidian, & Solimanpur, 2011). Pelo contrário, nos problemas sem preempção, a interrupção de tarefas não é permitida, ou seja, uma vez começado o processamento do trabalho, não pode deixar a máquina até estar concluído.

No caso de problemas de máquinas paralelas, é possível o processamento do mesmo trabalho em duas máquinas diferentes (Su, Cheng, & Chou, 2013), como se pode ver na Figura 5, com um exemplo do gráfico de Gantt em ambiente de máquinas paralelas com interrupção de tarefas (Pinedo, 2016).

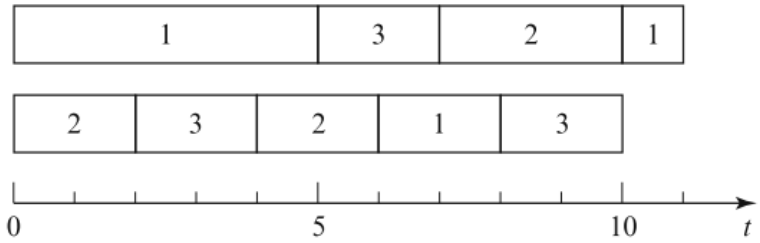

Figura 5 – Gráfico de Gantt com interrupção de tarefas (Pinedo, 2016).

Tempo de preparação das máquinas (*setup*)

Nos problemas de escalonamento em que são consideradas múltiplas máquinas e diferentes tipos de trabalhos, os tempos de processamento são, frequentemente, diferentes e podem depender da sequência de trabalhos. Assim, existe um tempo de preparação dependente da sequência (s_{jk}) sempre que depois de processar o trabalho *j*, é necessário um tempo de preparação s_{jk} antes de processar o trabalho *k*. Sempre que estes tempos são também dependentes das máquinas, a variável *i* é adicionada (S. W. Lin, Lu, & Ying, 2011; Zeidi & MohammadHosseini, 2015).

Tal como já explicado na secção 2.2.3 acerca das máquinas paralelas, no caso das uniformes ou não relacionadas, a existência de tempos de preparação das máquinas pode variar consoante o recurso (Jungwattanakit, Reodecha, Chaovalitwongse, & Werner, 2009).

Restrições de precedência

Quando existem restrições de precedência, é necessário que um ou mais trabalhos estejam completos antes que seja possível começar o processamento de outro trabalho, ou seja, o processamento de um trabalho depende da execução de outro (Gacias, Artigues, & Lopez, 2010; Hassan, Kacem, Martin, & Osman, 2016).

Segundo Carmo-Silva (2015), quando o processamento de um trabalho implica a realização prévia de outros trabalhos, estes consideram-se dependentes. Esta relação de dependência deve-se à existência de trabalhos, referidos como tarefas ou atividades, que fazem parte de um trabalho mais complexo ou à dependência da ordem pela qual os trabalhos são processados.

Disponibilidade das máquinas

Nos problemas de escalonamento é usual que se assuma que as máquinas estejam sempre disponíveis, no entanto, na prática, esta assunção pode não se verificar e é necessário que a indisponibilidade das máquinas seja considerada aquando da programação da produção. Assim, uma máquina pode estar indisponível devido à alocação de outros trabalhos, a atividades de manutenção ou a avarias das máquinas (Yazdani, Khalili, Babagolzadeh, & Jolai, 2017; Pinedo, 2016).

A manutenção periódica é uma forma de diminuir a indisponibilidade das máquinas causada pelas avarias, apesar de criar momentos de indisponibilidade (Benmansour, Allaoui, Artiba, & Hanafi, 2014; Tan, Chen, & Zhang, 2011).

Prioridades ou penalizações

A penalização é um valor perdido que acontece sempre que a execução de um trabalho não corresponde ao prazo de entrega, podendo ainda ter um valor associado, por exemplo o custo de armazenamento, devido à antecipação da execução de um trabalho (Wan & Yen, 2009).

Relativamente à prioridade, a chegada ao sistema de um trabalho de elevada prioridade pode levar à interrupção de alguma máquina, o que leva à reorganização do escalonamento feito (Lee & Pinedo, 1997).

2.2.5 Nomenclaturas de problemas de escalonamento

Os problemas de escalonamento têm um conjunto de características relacionadas com os trabalhos e os processadores. Estas características devem ser esclarecidas de forma a ser possível resolver o problema e é sempre benéfico recorrer a nomenclaturas para classificar o problema e, assim, facilitar a sua representação e especificação (Varela, 2007).

São várias as nomenclaturas usados por diversos autores. Para Conway (2003) os problemas são classificados segundo os parâmetros A/B/C/D, em que A descreve o processo de chegada dos trabalhos ao sistema, B corresponde ao número de processadores, C descreve o sistema de produção e D é o critério, ou critérios, de otimização.

French (1982) classifica os problemas de escalonamento de acordo com quatro parâmetros *n*/*m*/A/B, no qual *n* representa o número de trabalhos e *m* o número de máquinas no sistema.

O parâmetro A descreve o sistema de produção, no entanto, se *m* = 1, A é deixado em branco. Por fim, B descreve o critério de otimização que irá avaliar o escalonamento.

O sistema de classificação de Brucker (1995) utiliza a nomenclatura α | β | γ, inicialmente introduzida por (Graham, Lawler, Lenstra, & Kan, 1979). Esta nomenclatura pode ser decomposta em $\alpha_1 \mid \alpha_2 \mid \beta_1 \mid \beta_2 \mid \beta_3 \mid \beta_4 \mid \beta_5 \mid \beta_6 \mid \gamma$, permitindo uma classificação mais detalhada do problema a resolver, Tabela 1. Desde a sua introdução que esta notação tem sido usada e reformulada por vários autores e muitas classificações têm sido adicionadas à medida que vários problemas aparecem.

Tabela 1 – Descrição do nomenclatura α | β | γ

<table>
<tr><td rowspan="2">α</td><td rowspan="2">Características do ambiente de produção</td><td>α_1</td><td>Ambiente de produção</td></tr>
<tr><td>α_2</td><td>Número de processadores</td></tr>
<tr><td rowspan="6">β</td><td rowspan="6">Características dos trabalhos e dos processadores</td><td>β_1</td><td>Existência ou não de interrupção dos trabalhos</td></tr>
<tr><td>β_2</td><td>Restrições de precedência</td></tr>
<tr><td>β_3</td><td>Chegada de trabalhos</td></tr>
<tr><td>β_4</td><td>Tempos de processamento (arbitrário ou unitário)</td></tr>
<tr><td>β_5</td><td>Existência de prazos de entrega</td></tr>
<tr><td>β_6</td><td>Existência de lotes de trabalho</td></tr>
<tr><td>γ</td><td colspan="3">Critério de otimização (medidas de desempenho)</td></tr>
</table>

O campo α é composto por dois elementos, $\alpha_1\alpha_2$. O primeiro representa o ambiente e o segundo o número de processadores.

$\alpha_1 = P$, máquinas idênticas;

$\alpha_1 = Q$, máquinas uniformes;

$\alpha_1 = R$, máquinas paralelas;

$\alpha_2 = 1$, máquina única. Quando α_2 é omitido o número de máquinas é variável.

Por exemplo, sempre que $\alpha = P2$, está-se perante um problema de máquinas idênticas com duas máquinas. Existem outros problemas que são contemplados pela notação, como *Job shop* ou *Flow shop*, mas que não são abordados neste trabalho.

Relativamente ao parâmetro β, este refere-se às características das tarefas e pode não conter nenhum elemento. As restrições de precedência ($\beta = prec$), se a interrupção de uma tarefa é permitida ($\beta = pmtn$), se uma tarefa tem data de disponibilidade ($\beta = r_j$) ou se o modelo considera tempos de preparação ($\beta = s_j$) são exemplos de características deste parâmetro.
Como exemplo, quando um modelo permite interrupção e as tarefas têm datas de disponibilidade (*release date*) é representado da seguinte forma:

$\beta = pmtn; r_j$, no qual $pmtn$ significa a possibilidade de interrupção de tarefas e r_j a existência de data de disponibilidade.

No terceiro campo γ refere-se à função objetivo, que traduz o desempenho do sistema, representado pelas medidas de desempenho. Por exemplo, num modelo que apresenta como objetivo a minimização do *makespan*, então $\gamma = C_{max}$.

2.3 Critérios de otimização

No escalonamento da produção, o desempenho ou eficiência do sistema é, usualmente, avaliado através dos critérios de otimização. Estes são critérios que têm por base medidas de desempenho para avaliar a qualidade do funcionamento e das decisões do sistema de produção (Varela, 2007). Estes critérios têm como base medidas que permitem avaliar o desempenho do sistema, para que se possa averiguar se as decisões tomadas são as melhores para o funcionamento do sistema.
Os critérios de otimização mais comuns são o tempo de percurso máximo (F_{max}), o tempo de conclusão ou *makespan* (C_{max}), o tempo médio de fluxo ($\bar{F}$) ou o atraso máximo (L_{max}).

2.3.1 *Makespan*

Em escalonamento da produção uma variável importante é o tempo de conclusão dos trabalhos (C_j). O tempo de conclusão refere-se ao instante em que um trabalho termina o seu processamento e sai do sistema. A medida de desempenho que representa o instante de tempo em que a última tarefa é concluída corresponde ao *makespan* e é representado simbolicamente por C_{max}. O *makespan* pode ser definido como a diferença entre a primeira tarefa e aquela que determina o final da sequência de tarefas, ou o tempo a que a última

tarefa da sequência está completa (Blazewicz, Machowiak, Weglarz, Kovalyov, & Trystram, 2004).

Um dos objetivos muito utilizado na literatura é a minimização do *makespan*. Minimizar o *makespan* garante que um conjunto de trabalhos é processado o mais rápido possível e, normalmente, significa uma boa utilização das máquinas.

2.3.2 Atraso (*Lateness*)

Outro critério usado para avaliar o desempenho do sistema está relacionado com os atrasos dos trabalhos. O atraso (L_j) representa até que ponto o tempo de conclusão de um trabalho está dentro da data de vencimento. Um atraso positivo significa que o trabalho está concluído depois da data prevista, pelo contrário, um atraso negativo significa que o trabalho foi concluído antes da data de vencimento (Baker & Trietsch, 2009).

Os trabalhos atrasados (*tardy jobs*) normalmente significam entregas atrasadas o que pode prejudicar a confiança existente entre cliente e empresa. No entanto, em alguns casos, trabalhos adiantados (*early jobs*) também são prejudiciais, uma vez que significam um aumento no nível de inventário.

Assim, os critérios mais comuns são para minimizar o atraso total ($\sum L_j$) ou o atraso máximo ($\sum L_{max}$) de todos os trabalhos.

2.4 Métodos ou Algoritmos de Escalonamento da Produção

A complexidade dos problemas de escalonamento da produção dificulta a procura por uma solução ótima. Além disso, os problemas reais geralmente incluem um grande número de variáveis e trabalhos, tornando a procura ainda mais difícil. Os métodos para a resolução destes problemas podem ser exatos ou de aproximação. Os métodos exatos encontram sempre soluções ótimas para os problemas (Morton & Pentico, 1993). No entanto, no caso de problemas difíceis ou de grande escala, estes métodos têm um tempo de resposta muito elevado, sendo até, por vezes, inaplicável (Martí & Reinelt, 2011).

Cada solução encontrada irá dar lugar a uma tomada de decisão e qualquer indústria competitiva não pode esperar grandes períodos por uma solução. Para superar este problema são utilizados frequentemente os métodos heurísticos, caracterizados por serem abordagens

mais práticas. As heurísticas são rápidas, fáceis de implementar e capazes de fornecer boas soluções em menos tempo, apesar de não garantirem a solução ótima (Pinedo, 2016). Os algoritmos heurísticos podem ser divididos em duas categorias, construtivos (*constructive*) ou de melhoria (*improvement*).

Os algoritmos construtivos constroem as soluções a partir do zero. Começam sem um cronograma e gradualmente adicionam um trabalho, fazendo sempre a escolha que parece ser a melhor no momento, sendo a maneira mais rápida de obter soluções viáveis. As heurísticas construtivas são usadas essencialmente se uma solução razoavelmente boa é aceitável, se a solução precisa ser encontrada rapidamente ou para fornecer soluções iniciais para a heurística de melhoria (Pinedo, 2016).

Os algoritmos de melhoria são diferentes dos construtivos na medida em que começam com um cronograma completo, que pode ser selecionado arbitrariamente, tentando obter uma melhor solução através da manipulação do atual (Jungwattanakit et al, 2009; Pinedo, 2016).

Técnicas de pesquisa de vizinhança

As técnicas de pesquisa de vizinhança, também referidas como NTS (*Neighbourhood Search Techniques*) são baseadas numa pesquisa local, tentando encontrar uma solução melhor através da pesquisa na vizinhança do programa atual. Para construir um método de pesquisa local são necessários uma solução inicial para o problema (semente original), um mecanismo de geração de vizinhança, um critério de seleção da próxima semente e um critério de paragem da pesquisa, normalmente quando encontram o ótimo local (Artiba & Elmaghraby, 1996; Pinedo, 2016).

A pesquisa de vizinhança está presente em vários métodos heurísticos ou meta-heurísticos, dos quais se destacam *Simulated Annealing* (SA) e a *Tabu Search* (TS), que são usados para problemas bastante complicados em que a solução é muito complexa, permitindo uma solução mais refinada (Morton & Pentico, 1993; Pinedo, 2016; Xhafa & Abraham, 2008).

Tabu Search

Tabu Search é um método de pesquisa da vizinhança que utiliza um processo determinístico para o critério de aceitação, construindo uma lista de troca de pares tabu, que não podem ser utilizados na próxima geração da vizinhança. Este método evita a criação de ciclos e pesquisas por soluções anteriormente visitadas. Tem como principal vantagem a criação de um histórico

de forma a evitar pesquisas desnecessárias. Quando uma solução ótima local é encontrada, a TS aceita uma nova semente, mesmo que a solução seja pior do que a atual semente (Baker & Trietsch, 2009; Pinedo, 2016).

Genetic Algorithm

O *Genetic Algorithm* baseia-se no processo de seleção natural, no qual apenas as melhoras soluções prevalecem. Um programa (*schedule*) é visto como um membro da população. Cada iteração tem uma população composta por membros da iteração anterior mais os novos programas. Estes novos programas são criados através de técnicas de reprodução ou de mutação de membros que compunham iterações anteriores. Através do mecanismo de *crossover*, Figura 6, pode também ser gerado um novo programa mediante a combinação de diferentes partes de outros programas da população (Artiba & Elmaghraby, 1996; Pinedo, 2016).

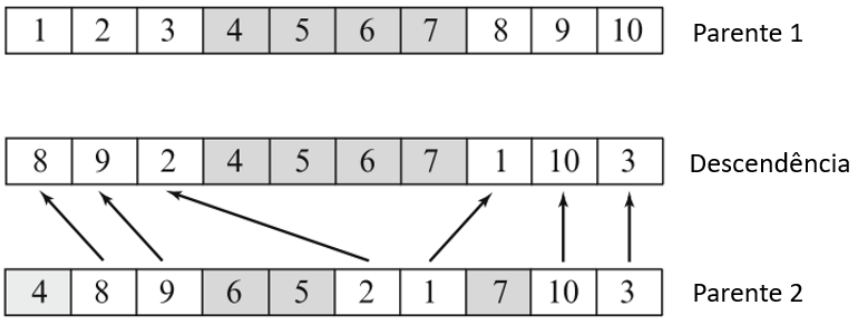

Figura 6 – Exemplo de geração de descendência (adaptado de Pinedo (2016))

Simulated Annealing

O *SA*, inicialmente proposto por Kirkpatrick et al., em 1983 e por Cerny, em 1988, tem como base uma analogia ao processo de fundição (*annealing*) na metalúrgica (Artiba & Elmaghraby, 1996).

A cada iteração, uma solução aleatória é selecionada da vizinhança da solução atual. Esta solução é aceite como nova solução com uma probabilidade calculada com base na função objetivo e na temperatura. As soluções não tendem para um ótimo local, mas traçam um trajeto aleatório que tende a ser orientado numa direção "favorável" (Varela, 2007).

No início há flexibilidade para escolher soluções piores, mas com o avançar da pesquisa apenas se escolhem os que levam a melhores soluções (Baker & Trietsch, 2009).

Ant Colony Optimization

O algoritmo *Ant Colony Optimization* (ACO) combina técnicas de pesquisa local, regras de sequenciamento e outras técnicas (Pinedo, 2016). Este método é inspirado no caminho que segue o comportamento das formigas e no uso da substância química (feromonas) que estas libertam durante o seu caminho para atrair os outros para o melhor caminho, neste caso, o melhor programa de produção. Para aplicar o ACO, o problema de otimização transforma-se no problema para encontrar o melhor caminho num gráfico ponderado.

O algoritmo ACO assume que a colónia de formigas (artificiais) constroem, iterativamente, soluções para o problema, usando os feromonas associados tanto às soluções anteriores como às informações heurísticas (Pinedo, 2016; Arnaout, Musa, & Rabadi, 2014; Arnaout, Rabadi, & Musa, 2010)

Particle Swarm Optimization

No inicio dos anos 1990 foram desenvolvidos vários estudos sobre o comportamento social de grupos de animais. Estes estudos mostraram que certos animais são capazes de partilhar informação entre o seu grupo, o que lhes confere uma vantagem de sobrevivência (Kennedy & Eberhart, 1995). Inspirados por estes estudos, em 1995, Kennedy & Eberhart (1995) propuseram o *Particle Swarm Optimization*, que é uma técnica de computação evolutiva, inspirada pelo comportamento social das aves ou peixes. A versão clássica foi proposta em 1995, mas desde então outras versões têm sido propostas.

Um bando de pássaros a voar sobre um lugar deve encontrar um local para pousar, mas a definição deste é um problema complexo, pois depende de vários fatores como maximizar a disponibilidade de comida ou minimizar o risco de existência de predadores. O problema de encontrar o melhor local apresenta um problema de otimização. O bando deve identificar o melhor ponto a fim de maximizar as condições de sobrevivência dos seus membros. Para tal, cada pássaro voa avaliando os critérios de sobrevivência. Cada um deles tem a vantagem de saber a melhor localização até que seja do conhecimento de todo o bando.

Semelhante aos Algoritmos Genéticos, o PSO é uma ferramenta de otimização baseada na população. É distintamente diferente de outros métodos do tipo evolutivo uma vez que não usa o *crossover* e a mutação e os membros de toda a população são mantidos através de procedimentos de busca, para que as informações sejam partilhadas socialmente entre os

indivíduos a fim de direcionar a pesquisa para a melhor posição no espaço de pesquisa (Lian, Gu, & Jiao, 2008).

No PSO cada indivíduo é chamado de “partícula” e cada partícula voa pelo espaço de pesquisa com uma certa velocidade. Em cada iteração, a partícula move-se do local anterior para um novo local à velocidade atualizada recentemente, que é calculada com base na própria experiência da partícula e na experiência de todo o grupo (Y. K. Lin, 2013).

3. REVISÃO DE LITERATURA

A programação de máquinas paralelas tem sido uma área de pesquisa em crescimento desde os primeiros trabalhos (McNaughton, 1959). Este tipo de problema tem, desde então, recebido um contínuo interesse pelos investigadores devido à sua relevância para os sistemas de produção.

Esta dissertação aborda o problema de escalonamento de máquinas paralelas não relacionadas, um tema que foi menos estudado do que os problemas de escalonamento mais comuns. Quando o problema considera os temos de preparação dependentes da sequência (*sequence-dependent setup times*) a pesquisa é ainda menor, uma vez que a complexidade aumenta consideravelmente. No entanto, este problema tem atraído atenção mais recentemente devido à sua complexidade e importância.

Allahverdi (2015) apresenta uma revisão sobre problemas de escalonamento. Este classifica-os com base no ambiente de produção como máquina única, máquinas paralelas, *flow shop*, *job shop* ou *open shop*. Classifica também os problemas como familiar/não familiar e tempos/custos de *setup* dependentes/independentes da sequência.

O problema de escalonamento de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência tem bastante importância, uma vez que pode ser encontrado em várias áreas como a indústria eletrónica, siderúrgica ou têxtil (Kim, Kim, Jang, & Frank Chen, 2002; Gendreau, Laporte, & Guimarães, 2001; Tang & Wang, 2009)

Kim, Kim, Jang, & Frank Chen (2002) utilizaram o algoritmo *Simulated Annealing* para resolver um problema de escalonamento na indústria eletrónica. Foi possível concluir que o método SA proposto supera significativamente o método de pesquisa de vizinhança em termos de atraso total.

Tang & Wang (2009) formularam um problema de escalonamento na indústria siderúrgica como um programa não linear inteiro misto e propuseram a heurística *Tabu Search* para obter soluções satisfatórias. Os resultados mostram que o modelo e a heurística apresentados são mais eficazes e eficientes do que o planeamento manual.

Na indústria têxtil Gendreau, Laporte, & Guimarães (2001) aplicaram uma heurística para o problema de escalonamento do multiprocessador com tempos de preparação dependentes da sequência. Os resultados mostram que a heurística proposta é mais rápida do que a TS enquanto fornece soluções de qualidade semelhante.

Apesar dos problemas de máquinas paralelas não relacionadas terem grandes implicações industriais, estão entre os problemas de otimização combinatória mais difíceis, uma vez que são NP-difíceis. A sua complexidade advém do conjunto vasto de possíveis combinações de soluções a explorar no espaço de soluções possíveis e também à adição de outros tipos de dados, como condições ou restrições impostas nos problemas (Pfund, Fowler, & Gupta, 2004).

A complexidade do problema de escalonamento de máquinas paralelas não relacionadas tem levado ao aumento do interesse por procedimentos heurísticos para encontrar soluções num tempo razoável. D. W. Kim, Na, & Chen (2003) propuseram quatro heurísticas de pesquisa para o problema acima mencionado. Estas heurísticas podem ser facilmente aplicadas para programação prática da produção. Ghirardi & Potts (2005) também estudaram o problema de máquinas paralelas não relacionadas com o objetivo de minimizar o *makespan*, no qual a heurística usado foi uma aplicação do *Recovering Beam Search* (RBS). Os resultados computacionais mostram que este algoritmo é capaz de gerar soluções aproximadas para grandes instâncias (até 50 máquinas e 1000 trabalhos) em apenas alguns minutos.

A meta heurística Meta-RaPS foi introduzida por Rabadi, Moraga e Al-Salem (2006) para minimizar o *makespan* de um problema de máquinas paralelas não relacionadas com tempos de *setup* dependentes da sequência. O desempenho da nova meta heurística foi avaliado comparando as suas soluções com as obtidas pelas heurísticas existentes para o mesmo problema. Os resultados mostram que a Meta-RaPS encontrou soluções ótimas para problemas pequenos e teve melhor desempenho do que as heurísticas existentes para grandes problemas.

Para o mesmo problema e objetivo Rabadi et al. (2010) introduziram o *Ant Colony Optimization*. Para avaliar o desempenho, as soluções do algoritmo ACO foram comparadas com as soluções de outras heurísticas como a *Tabu Search*, a *Partitioning Heuristic* e a Meta-RaPS (Rabadi et al., 2006). Os resultados permitiram concluir que o ACO teve melhor desempenho do que os restantes algoritmos.

Rabadi et al. (2014) propuseram o algoritmo *ACO* melhorado e obtiveram melhor desempenho do que a versão anterior, tendo a capacidade de resolver problemas difíceis de otimização combinatória, particionando-os em subproblemas.

A minimização do *makespan* é um dos critérios mais estudados na literatura de escalonamento de produção, seja de máquinas paralelas seja de únicas. Por exemplo, Woo, Jung, & Kim (2017) desenvolveram um modelo de programação linear inteira mista (PLIM)

para encontrar a solução ótima para o problema de escalonamento de máquinas paralelas não relacionadas com o objetivo de minimizar o *makespan*. Eles propuseram uma nova regra baseada no *GA* com um cromossoma que representa a sequência de atribuição dos trabalhos a uma máquina e o escalonamento dos trabalhos em cada máquina são determinados por uma heurística baseada no tempo de conclusão durante o processo de decodificação do cromossoma.

Considerando os tempos de *setup* dependentes da sequência dos trabalhos (Vallada & Ruiz, 2011) apresentaram o GA para o problema de máquinas paralelas não relacionadas com o objetivo de minimizar o *makespan*. Este algoritmo foi comparado com os melhores encontrados na literatura e teve um desempenho claramente melhor.

Recentemente, foi proposto por Zheng & Wang (2018) um novo problema relativo ao escalonamento de máquinas paralelas não relacionadas de manufatura verde com restrições de recursos. Este problema tem como objetivos a minimização do *makespan* e do total de emissões de carbono e para o resolver é proposto o *Collaborative Multiobjective Fruit Fly Optimization Algorithm* (CMFOA). Os resultados mostram que o algoritmo multiobjectivo conseguiu obter mais e melhores soluções não dominadas do que os outros algoritmos.

O PSO foi aplicado com sucesso a vários problemas de otimização e, nos anos mais recentes, tem sido aplicado a estudos para resolver problemas de escalonamento, nomeadamente de máquinas paralelas. Vários são os autores que usam o PSO para resolver problemas semelhantes ao problema abordado nesta dissertação. Kashan & Karimi (2009) aplicam o algoritmo PSO para resolver um problema de máquinas paralelas idênticas com o objetivo de minimizar o *makespan* (C_{max}). Já Damodaran, Diyadawagamage, Ghrayeb, & Vélez-Gallego (2012) utilizaram o algoritmo PSO para escalonar máquinas paralelas não idênticas de processamento em lote também com o objetivo de minimizar o *makespan*. A eficácia do algoritmo é avaliada comparando os seus resultados com um algoritmo genético e um *solver* comercial usado para resolver problemas de PLIM.

Salehi Mir e Rezaeian (2016) estudaram uma abordagem híbrida do PSO e do GA aplicada a um problema de escalonamento de máquinas paralelas não relacionadas e avaliaram o método comparando os seus resultados com o GA e o PSO. Para problemas pequenos os algoritmos mencionados são eficientes a encontrar soluções ótimas, mas quando o tamanho do problema aumenta, o método híbrido obtém melhores resultados.

Torabi, Sahebjamnia, Mansouri e Bajestani (2013) consideraram um problema de escalonamento de máquinas paralelas com vários objetivos, com tempos de *setup* dependentes da sequência e das máquinas e restrições de recursos secundários para os trabalhos. Apresentaram também um algoritmo *Multi-objetive Particle Swarm Optimization* (MOPSO) para encontrar uma boa aproximação à fronteira de Pareto. O MOPSP foi comparado com um PSO convencional e supera este último em termos de métricas de qualidade, diversidade e espaçamento.

Behnamian (2014) sugere um algoritmo *Discrete Particle Swarm Optimization* (DPSO) para problemas de escalonamento de máquinas paralelas difusas, que compreende duas componentes, o PSO e o GA.

4. CASO DE ESTUDO

4.1 Introdução

Este trabalho foca-se no problema de escalonamento de trabalhos com tempos de *setup* dependentes da máquina e da sequência em máquinas paralelas não relacionadas, com o objetivo de minimizar o tempo máximo de conclusão – *makespan*. O problema de escalonamento de máquinas paralelas não relacionadas tem um conjunto $N = \{1, \dots, n\}$ de *n* trabalhos que devem ser processados numa máquina do conjunto $M = \{1, \dots, m\}$ de *m* máquinas paralelas não relacionadas (R_m). Quando o tempo de processamento do trabalho depende da máquina atribuída e não há relação entre essas máquinas, as máquinas são consideradas não relacionadas. Os tempos de *setup* são dependentes das máquinas e da sequência (S_{ijk}). Cada máquina tem a sua matriz de tempo de *setup* e cada matriz é diferente das outras.

Na teoria do escalonamento, o *makespan* (C_{max}) é definido como o tempo de conclusão do trabalho final (quando sai do sistema). O objetivo deste estudo é minimizar o *makespan*.

O problema estudado nesta dissertação é classificado na literatura como $R_m \left| S_{ijk} \right| C_{max}$. A minimização do *makespan* de um problema de escalonamento com *m* máquinas paralelas idênticas e tempos de *setup* dependentes da sequência é categorizado como NP-difícil. Deste modo, o problema mais complexo de máquinas paralelas não relacionadas também é considerado NP-difícil.

O problema é descrito usando as seguintes suposições e notações:

- Cada máquina apenas processa um trabalho da cada vez, sem interrupções;
- No instante inicial, tempo zero, todos os trabalhos estão disponíveis e não há restrições de precedência entre os mesmos;
- Em cada máquina *k*, cada trabalho *i* tem um tempo de processamento p_{ik};
- Em cada máquina *k*, para processar o trabalho *j* depois do trabalho *i* existe um tempo de setup s_{ijk}. O tempo de *setup* é diferente para cada máquina.

4.2 Descrição dos dados

Nesta secção são apresentados os dados usados para a aplicação dos algoritmos. Os dados usados estão disponíveis na página *Scheduling Research Virtual Centre*. Resumidamente, os tempos de processamento e de *setup* do *dataset* do problema estão aleatoriamente distribuídos de U (50, 100) e U (125, 175). Os limites destas duas distribuições uniformes determinam o nível de dominância dos tempos de processamento e de *setup*. Quando os tempos de processamento e *setup* são equilibrados (*balanced*), ambos são gerados a partir de U (50, 100).

Os tempos de processamento e *setup* são gerados a partir de U (125, 175) e U (50, 100), respetivamente, quando os tempos de processamento são dominantes (P). Quando os tempos de *setup* são dominantes (S), eles são gerados a partir de U (50, 100) e U (125, 175), respetivamente.

Os tempos de processamento e de *setup* das máquinas estão organizados por matrizes, como representado nas tabelas Tabela 2 e Tabela 3. Na Tabela 2 estão presentes os tempos de processamento do trabalho *j* na máquina *k* ($p_{j,k}$), no caso de seis trabalhos e duas máquinas.

Tabela 2 – Exemplo de matriz dos tempos de processamento

	Machine1	*Machine2*
Job1	59	68
Job2	76	60
Job3	68	73
Job4	52	73
Job5	84	77
Job6	85	59

Na Tabela 3 está presente um exemplo da matriz dos tempos de *setup* do trabalho *j* depois de processado o trabalho *i* na máquina *k* (s_{ijk}).

Tabela 3 – Exemplo de matriz dos tempos de *setup*

	Job1	*Job2*	*Job3*	*Job4*	*Job5*	*Job6*
Job1	57	91	98	70	95	56
Job2	68	66	52	93	86	95
Job3	84	89	62	98	59	60
Job4	65	89	89	57	89	52
Job5	59	86	90	56	73	52
Job6	73	94	68	99	97	67

Os dados estão divididos em pequenos e grandes problemas, pelo que o número de trabalhos e de máquinas variam consoante o tamanho do problema. Assim, para pequenos problemas existem seis, sete, oito, nove, dez e onze trabalhos e duas, quatro, seis e oito máquinas, enquanto que para problemas grandes existem 20, 40, 60, 80, 100 e 120 trabalhos e duas, quatro, seis, oito, dez e doze máquinas. Para além disso, existem 15 instâncias para cada um dos problemas mencionados.

4.3 Definição do problema

A programação inteira mista (PIM) é utilizada para encontrar soluções ótimas para o problema de escalonamento da produção de máquinas paralelas não relacionadas com tempos de *setup* dependentes da sequência. Uma formulação semelhante foi usada por (Guinet, 1991).

Minimizar C_{max}

Sujeito a

$$\sum_{\substack{i=0 \\ i\neq j}}^{n}\sum_{k=1}^{m} x_{i,j,k} = 1 \qquad \forall j = 1,\dots,n \tag{2}$$

$$\sum_{\substack{i=0 \\ i\neq h}}^{n} x_{i,h,k} - \sum_{\substack{j=0 \\ j\neq h}}^{n} x_{h,j,k} = 0 \qquad \forall h = 1,\dots,n, \quad \forall k = 1,\dots,m \tag{3}$$

$$C_j \geq C_i + \sum_{k=1}^{m} x_{i,j,k}\left(S_{i,j,k} + p_{j,k}\right) + M\left(\sum_{k=1}^{m} x_{i,j,k} - 1\right)$$

$$\forall i = 0,\dots,n, \quad \forall j = 1,\dots,n \tag{4}$$

$$\sum_{j=0}^{n} x_{0,j,k} = 1 \qquad \forall k = 1, \ldots, m \tag{5}$$

$$x_{i,j,k} \in \{0,1\} \qquad \forall i = 0, \ldots, n, \quad \forall j = 0, \ldots, n, \quad \forall k = 1, \ldots, m \tag{6}$$

$$C_0 = 0 \tag{7}$$

$$C_j \geq 0 \qquad \forall j = 1, \ldots, n \tag{8}$$

No qual:

C_j: tempo de conclusão do trabalho *j*

$p_{j,k}$: tempo de processamento do trabalho *j* na máquina *k*

$S_{i,j,k}$: tempo de setup dependente da sequência para processar o trabalho *j* depois do *i* na máquina *k*

$S_{0,j,k}$: tempo de setup para processar primeiro o trabalho *j* na máquina *k*

$x_{i,j,k}$: 1 se o trabalho *j* é diretamente depois do trabalho *i* na máquina *k* e 0 se não for

$x_{0,j,k}$: 1 se o trabalho *j* é o primeiro a ser processado na máquina *k* e 0 se não for

$x_{i,0,k}$: 1 se o trabalho *j* é o último a ser processado na máquina *k* e 0 se não for

M: um número positivo grande

O objetivo é minimizar o *makespan*. A restrição (2) assegura que cada trabalho é apenas escalonado uma vez e processado por uma única máquina. A restrição (3) assegura que cada trabalho não deve ser precedido nem sucedido por mais do que um trabalho. A restrição (4) calcula o tempo de conclusão e assegura que nenhum trabalho precede nem sucede o mesmo trabalho. A restrição (5) assegura que apenas um trabalho pode ser escalonado primeiro em cada máquina. Não há necessidade de mais restrições para garantir que apenas um trabalho é escalonado em último em cada máquina porque as restrições (5) e (3) garantem isso. A restrição (6) assegura que a variável de decisão x é binária em todos os domínios. A restrição (7) estabelece que o tempo de conclusão para o trabalho *dummy* 0 é zero e a restrição (8) assegura que o tempo de conclusão é não negativo. Resolvendo o problema de PIM descrito anteriormente podem ser obtidas soluções ótimas.

4.4 Explicação do PSO

O PSO tem como base a metáfora da interação social e comunicação entre os bandos de pássaros e os cardumes de peixes. Neste caso, o algoritmo é explicado utilizando um bando de pássaros. Nestes grupos, existe um líder, que apresenta melhor desempenho, que é o guia do movimento de todo o bando. No PSO, cada indivíduo é chamado de partícula e cada partícula voa no espaço de procura a uma certa velocidade. Em cada iteração, a partícula move-se da posição anterior para um novo local à velocidade recém atualizada, que é calculada com base na própria experiência da partícula e na experiência de todo o bando.

O objetivo de um problema de otimização é determinar a variável representada pelo vetor $X = [x_1 x_2 x_3 \dots x_n]$ que minimiza ou maximiza dependendo da formulação da função $f(X)$. A variável vetor X é conhecido como o vetor posição. A função $f(X)$ é chamada de função *fitness* ou objetivo, que é uma função que pode avaliar quão boa ou má é a posição X. Considerando um bando com P partículas, existe um vetor posição $X_i^t = (x_{i1} x_{i2} x_{i3} \dots x_{in})^T$ e um vetor velocidade $V_i^t = (v_{i1} v_{i2} v_{i3} \dots v_{in})^T$ na iteração t para cada uma das partículas i que o compõem. Estes vetores são atualizados através da dimensão j de acordo com as seguintes equações:

$$V_{ij}^{t+1} = wV_{ij}^t + c_1 r_1^t \left(pbest_{ij} - X_{ij}^t\right) + c_2 r_2^t (gbest_j - X_{ij}^t) \qquad \text{Equação 1}$$

$$X_{ij}^{t+1} = X_{ij}^t + V_{ij}^{t+1} \qquad \text{Equação 2}$$

$$i = 1,2, \dots, P \text{ e } j = 1,2, \dots, n$$

O parâmetro w é a constante do peso da inércia e para a versão clássica do PSO é um valor constante positivo. O primeiro termo da equação de atualização de velocidade é um produto entre o parâmetro w e a velocidade da partícula anterior. O segundo termo da equação 1 é calculado através da diferença entre a melhor posição da partícula ($pbest_{ij}$) e a posição atual (X_{ij}^t). O parâmetro c_1 é uma constante positiva que pesa a importância das próprias experiências anteriores da partícula. O outro parâmetro é o r_1 que é um valor aleatório do intervalo $[0,1]$ e desempenha o papel de evitar convergências prematuras, aumentando os ótimos globais mais prováveis (Kennedy & Eberhart, 1995).

O terceiro termo é o de aprendizagem social. É através dele que todos as partículas no bando conseguem partilhar informação sobre o melhor ponto encontrado independentemente da partícula que o encontrou. O formato é igual ao do segundo termo sobre a aprendizagem individual. A diferença $(gbest_j - X_{ij}^t)$ atua como uma atração para as partículas para o melhor ponto até então encontrado em alguma t iteração. Similarmente, o c_2 é o parâmetro de aprendizagem social e pesa a importância da aprendizagem global do bando, enquanto o r_2 tem o mesmo papel do r_1.

De forma resumida, na Figura 7, está presente o fluxograma do PSO, que permite perceber como é que o algoritmo funciona.

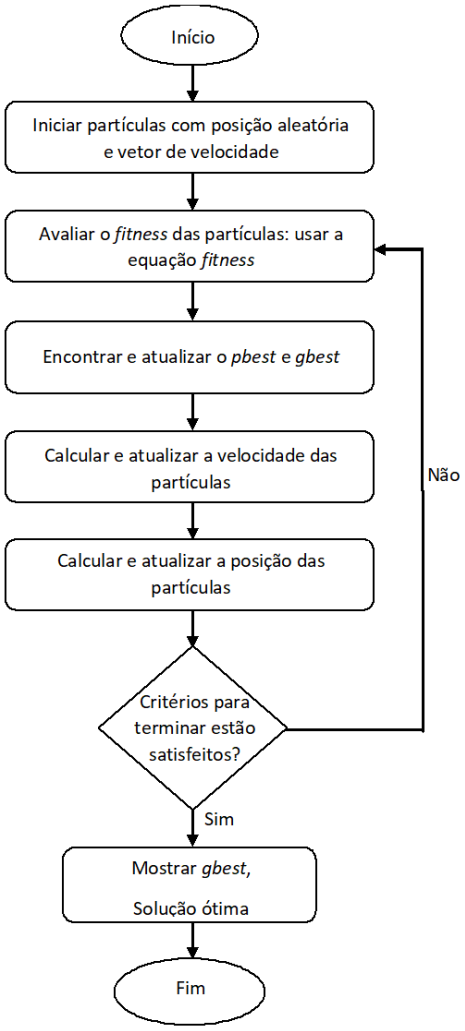

Figura 7 – Fluxograma do *Particle Swarm Optimization* (adaptado de Singh, Saxena, & Soni (2015)).

4.5 Implementação do PSO

O algoritmo PSO foi implementado em linguagem de programação MATLAB e corrido num computador com processador Intel Core i7 8th Gen.

4.5.1 MATLAB

O MATLAB é um software interativo de alta performance que integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos. O uso do MATLAB permite analisar dados, desenvolver algoritmos e criar modelos e aplicações. A linguagem, as aplicações e as funções matemáticas integradas permitem explorar com rapidez várias abordagens para chegar a uma solução. As caixas de ferramentas (*toolbox*) correspondem a coleções de funções criadas no ambiente de computação técnica do MATLAB. A *Global Optimization Toolbox* fornece funções que pesquisam soluções globais para problemas com múltiplos máximos ou mínimos. Os *solvers* da caixa de ferramentas incluem o GA, o PSO, o SA, a pesquisa múltipla e pesquisa global.

4.5.2 Implementação

A Figura 8 apresenta um exemplo de um modelo utilizado na aplicação do algoritmo, neste caso para duas máquinas, seis trabalhos e instância 1. A matriz p corresponde ao tempo de processamento das máquinas, sendo que a primeira coluna diz respeito à máquina um e a segunda à dois. As matrizes s correspondem ao tempo de *setup* dos trabalhos nas máquinas um e dois, respetivamente. Neste caso, é possível verificar que, por exemplo, o tempo de *setup* do trabalho três depois de processado o trabalho um na máquina um é de 84.

```
function model = CreateModel_2_6_1()

    p=[ 59 68
        76 60
        68 73
        52 73
        84 77
        85 59 ];

    %number of jobs
    I=size(p,1);
    %number of machines
    J=size(p,2);

%sequence dependent setup time
    s(:,:,1)=[57 91 98 70 95 56
              68 66 52 93 86 95
              84 89 62 98 59 60
              65 89 89 57 89 52
              59 86 90 56 73 52
              73 94 68 99 97 67];

    s(:,:,2)=[95 75 98 53 62 97
              76 75 55 85 88 51
              88 94 76 72 50 77
              79 92 86 63 93 81
              76 82 95 98 50 68
              98 97 99 99 96 72 ];

 % define model parameters
    model.I=I;
    model.J=J;
    model.p=p;
    model.s=s;
    model.nVar=I+J-1;
end
```

Figura 8 - *Script* do modelo.

Como já foi referido anteriormente, na implementação do algoritmo em estudo foi utilizado a caixa de ferramentas *particleswarm*, que tenta otimizar utilizando o algoritmo PSO. A forma como o algoritmo funciona já está implementada no *solver*, no entanto foi necessário criar algumas funções para que o algoritmo funcionasse por completo e para que fosse calculado o *makespan* e os resultados fossem apresentados na forma pretendida pelo utilizador. Assim, no Apêndice 1 – Modelo MATLAB estão presentes os *scripts* utilizados para implementar o PSO. A função *MyCost*, que tem como *output* o valor do *makespan* (C_{max}) do modelo que está a analisar. A função *ParseSolution* tem como objetivo encontrar a solução do modelo, ou seja, criar a lista de trabalhos, encontrar a posição do trabalho na máquina e o índice do primeiro trabalho na lista.

A função *PlotSolution* tem como objetivo formatar os gráficos que são criados pelo *solver* da forma que o utilizador pretende, como por exemplo os tipos de cores.

Por fim, tem-se o *script* que permite correr os modelos todos e que cria o ficheiro Excel com os resultados, assim como guarda os gráficos da ordem de trabalhos e do número de iterações necessárias. São também definidos o número de vezes que o modelo deve correr, neste caso, 20, e outros parâmetros do PSO, como o *SwarmSize* e o *MaxIterations*.

Por exemplo, o número de variáveis depende do modelo que está a ser analisado, uma vez que o exemplo com seis máquinas tem menos variáveis do que o modelo com dez. O número de vezes que os modelos devem ser corridos foi estipulado em 20, de forma a encontrar soluções mais fiáveis. O *SwarmSize* foi estipulado em 200 e como número máximo de iterações, o valor escolhido foi de 100 iterações, ou seja, após este valor o programa termina de correr e avança para a instância seguinte. Este valor foi limitado de forma a que os modelos conseguissem correr em tempo útil.

5. ANÁLISE DE RESULTADOS

Na análise de resultados feita neste capítulo os resultados são comparados com os da literatura, nomeadamente com o trabalho desenvolvido por Rabadi et al. (2014, 2010), que aplicaram o *Ant Colony Optimization* (ACOI e ACOII) e por Amaral, Costa, Rocha, Varela, & Madureira (2020) que implementou dois algoritmos, o SA e o GA. Esta comparação é feita com o objetivo de perceber qual o método que melhor se aplica a este problema específico.
De referir que para todos os casos de estudo e cada instância foram obtidos dois gráficos distintos, no entanto, apenas serão apresentados os que se consideram relevantes, uma vez que devido ao elevado número de casos de estudo, não se considera viável a apresentação de todos.

5.1 Número máximo de Iterações

Uma das opções da caixa de ferramentas *particleswarm* permite a escolha do número máximo de iterações. Este critério faz com que quando é atingido o número estipulado, o programa termine de correr e avance para a instância seguinte. Inicialmente, o número máximo de iterações escolhido foi de 50 e conseguiu-se obter resultados para todos os modelos. No entanto, pela análise das iterações necessárias para encontrar o valor ótimo, foi percetível que existiam modelos em que eram necessárias mais iterações. Assim, decidiu-se aumentar este valor para 100 iterações.
Os resultados desta alteração permitiram perceber que, na sua maioria, houve uma diminuição do *makespan* tanto para pequenos como para grandes problemas, uma vez que existiam mais iterações para que fosse possível encontrar o valor ótimo.
Nas tabelas Tabela 4 e Tabela 5 são apresentadas as percentagens de variação entre os resultados com 100 iterações e com 50, tanto para pequenos problemas, Tabela 4, como para grandes problemas, Tabela 5.
Como se pode observar pela Tabela 4, existem alguns modelos em que as 50 iterações eram suficientes para encontrar os melhores valores, uma vez que estes são iguais para os dois casos. Na maior parte dos casos os valores encontrados com 100 iterações são melhores, no

entanto, esta variação é bastante pequena, estando quase todos os casos abaixo de 1%. De realçar também que à medida que o número de máquinas e de trabalhos aumentam, a variação percentual é também maior o que significa que quanto maior o número de máquinas e trabalhos mais iterações são necessárias para encontrar o *makespan* ótimo. Exemplificando, no caso de seis máquinas e onze trabalhos, o *makespan* para os problemas equilibrados, quando se utilizam 50 iterações é de 269. No entanto, com o aumento para 100 iterações, este valor diminui para 265,47, em média. Isto significa que houve uma descida de 1,31% no valor do *makespan* para este exemplo, como pode ser observado na Tabela 4.

Tabela 4 – Percentagem de variação entre 100 e 50 iterações, pequenos problemas.

Máquinas	**Trabalhos**	**Equilibrados**	**Proc. Dom**	***Setup* Dom**
2	**6**	0,00	0,00	0,00
	7	-0,01	0,00	0,00
	8	-0,04	0,01	0,05
	9	0,00	0,00	0,00
	10	0,01	-0,01	-0,03
	11	-0,23	-0,02	0,00
4	**6**	0,00	0,00	0,00
	7	0,00	0,05	0,00
	8	-0,08	-0,14	-0,03
	9	0,00	-0,08	0,00
	10	-0,13	-0,28	-0,42
	11	-0,19	-0,16	-0,15
6	**8**	-0,06	-0,12	0,00
	9	0,74	-0,12	-0,76
	10	-0,57	-0,10	-0,07
	11	-1,31	-0,45	-0,34
8	**10**	-1,22	-1,02	-0,70
	11	-0,41	-0,61	-0,17

Na Tabela 5 estão os valores para grandes problemas. Ao contrário do que aconteceu com os pequenos problemas, neste caso houve sempre melhorias, mesmo que algumas com

percentagens muito pequenas, em relação ao *makespan*. Isto significa que 50 iterações não foram suficientes para encontrar o melhor valor e aumentando para 100, os valores do *makespan* diminuíram. Exemplificando, para tempos de processamento dominantes com dez máquinas e 40 trabalhos, o *makespan* com 50 iterações era de 937,6. Depois de aumentar as iterações para 100 este valor fixou-se nos 891,6, que equivale a uma descida de 4,91%, observado na Tabela 5. No entanto, existe uma conclusão que é transversal a grandes e pequenos problemas e resume-se a que, normalmente, quando o número de trabalhos e de máquinas aumenta, as variações percentuais também aumentam. Isto significa que, para esses casos, são necessárias mais iterações para encontrar o melhor valor.

Tabela 5 – Percentagem de variação entre 100 e 50 iterações, grandes problemas.

		Trabalhos					
	Máquinas	**20**	**40**	**60**	**80**	**100**	**120**
Equilibrados	**2**	0,24	-1,07	-1,10	-1,60	-2,05	-2,08
	4	0,05	-1,60	-1,54	-1,70	-1,81	-2,07
	6	-1,59	-1,82	-2,17	-1,59	-2,13	-2,06
	8	-1,36	-1,49	-3,21	-3,08	-1,44	-2,77
	10	-2,92	-5,33	-3,16	-2,90	-2,99	-3,19
	12	-2,32	-3,51	-3,15	-4,37	-4,53	-4,22
Proc. Dom	**2**	0,08	-0,69	-0,92	-1,10	-1,09	-1,28
	4	-0,59	-0,77	-0,72	-1,23	-1,15	-1,11
	6	-0,72	-0,53	-1,05	-1,43	-1,22	-1,63
	8	-0,97	-1,91	-1,59	-2,40	-1,79	-1,56
	10	-1,80	-4,91	-4,08	-2,90	-3,24	-2,53
	12	-1,51	-1,65	-4,85	-2,96	-3,26	-3,37
***Setup* Dom**	**2**	-0,23	-0,84	-1,18	-1,17	-1,01	-1,56
	4	-0,50	-0,91	-1,06	-1,08	-1,38	-1,49
	6	-1,25	-0,48	-1,20	-1,06	-1,36	-1,35
	8	-0,97	-2,21	-1,57	-2,40	-2,02	-1,66
	10	-2,81	-6,04	-4,30	-3,31	-3,82	-2,43
	12	-1,64	-2,07	-4,64	-3,89	-3,09	-3,73

Concluindo a análise desta secção, o aumento do número máximo de iterações de 50 para 100 fez com que os resultados do PSO melhorassem, principalmente quando o número de trabalhos é maior. Este valor não continuou a ser aumentado, uma vez que era necessário restringir o intervalo que se poderia utilizar a fim de conseguir resultados em tempo útil.

5.2 Pequenos problemas

Nesta secção irão ser analisados os resultados obtidos para o caso de estudo de pequenos problemas, com duas, quatro, seis e oito máquinas e seis, sete, oito, nove, dez e onze trabalhos. A análise é feita consoante a classificação dos tempos de processamento e de *setup*, estando dividida entre tempos de processamento e *setup* equilibrados, tempos de processamento dominantes e tempos de *setup* dominantes, dependendo do intervalo da distribuição uniforme utilizado para gerar os dados.

5.2.1 Tempos de processamento e *setup* equilibrados (*balanced*)

O primeiro cenário a ser analisado diz respeito aos tempos de processamento e *setup* equilibrados (*balanced*), os quais foram gerados a partir da distribuição U (50, 100).

O tempo de execução do modelo, C_{max} e a ordem do trabalho são produtos do algoritmo. Para cada caso de estudo e para cada execução associada a cada instância, dois tipos de gráficos são gerados, como representado nas Figura 9 e Figura 10. A Figura 9 mostra a sequência de trabalho associada ao caso de estudo com duas, quatro, seis e oito máquinas. Na Figura 9 é possível visualizar quais e quantos trabalhos foram alocados a cada máquina e qual a máquina que finalizou a execução das tarefas planeadas. Por exemplo, para o gráfico com duas máquinas, os trabalhos posicionados na parte inferior referem-se à máquina um e os posicionados na parte superior, à máquina dois. É ainda possível verificar o tempo de execução total dos trabalhos, C_{max}, no canto superior direito de cada gráfico, que neste caso é de 390. Para quatro máquinas, este valor é de 258, para seis é de 239 e para oito é de 227. Os espaços vazios entre o início e o fim de cada trabalho referem-se ao tempo de *setup* alocado entre cada trabalho.

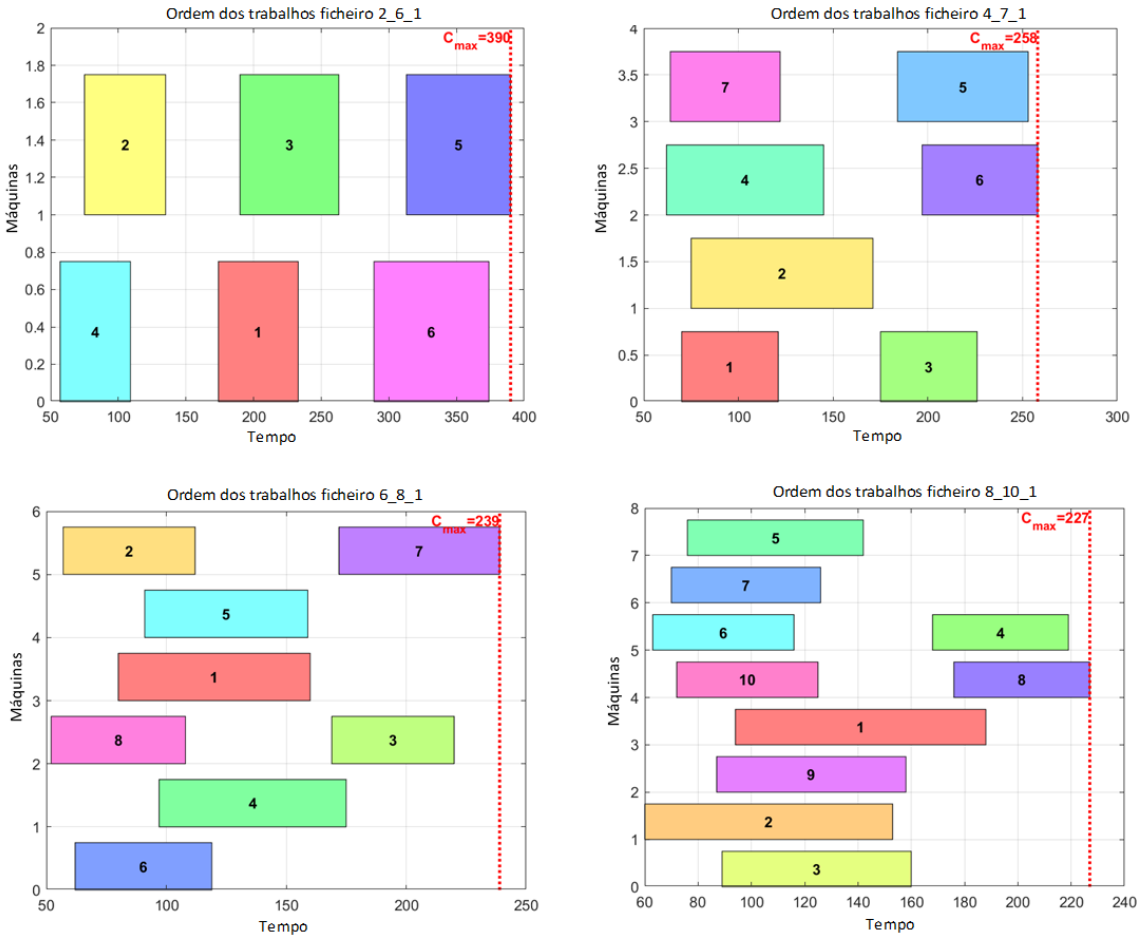

Figura 9 - Ordem de trabalhos, tempos de processamento e *setup* equilibrados

A Figura 10 mostra como o valor de C_{max}, representado pela variável *Best Cost*, varia ao longo das iterações. É possível constatar que esta variação é diferente dependendo do caso em estudo. Assim, o ficheiro 2_6_1 nas primeiras 2 iterações tem uma diminuição do valor do *makespan*. Para as restantes iterações o C_{max} mantem-se constante, com um valor de 390. Já o ficheiro 4_7_1 apresenta o melhor valor encontrado apenas na décima primeira iteração. Os restantes dois ficheiros por terem mais máquinas e trabalhos necessitam de mais iterações até encontrar o melhor valor de C_{max}, no entanto, conseguem encontrar os resultados antes de atingir as 100 iterações.

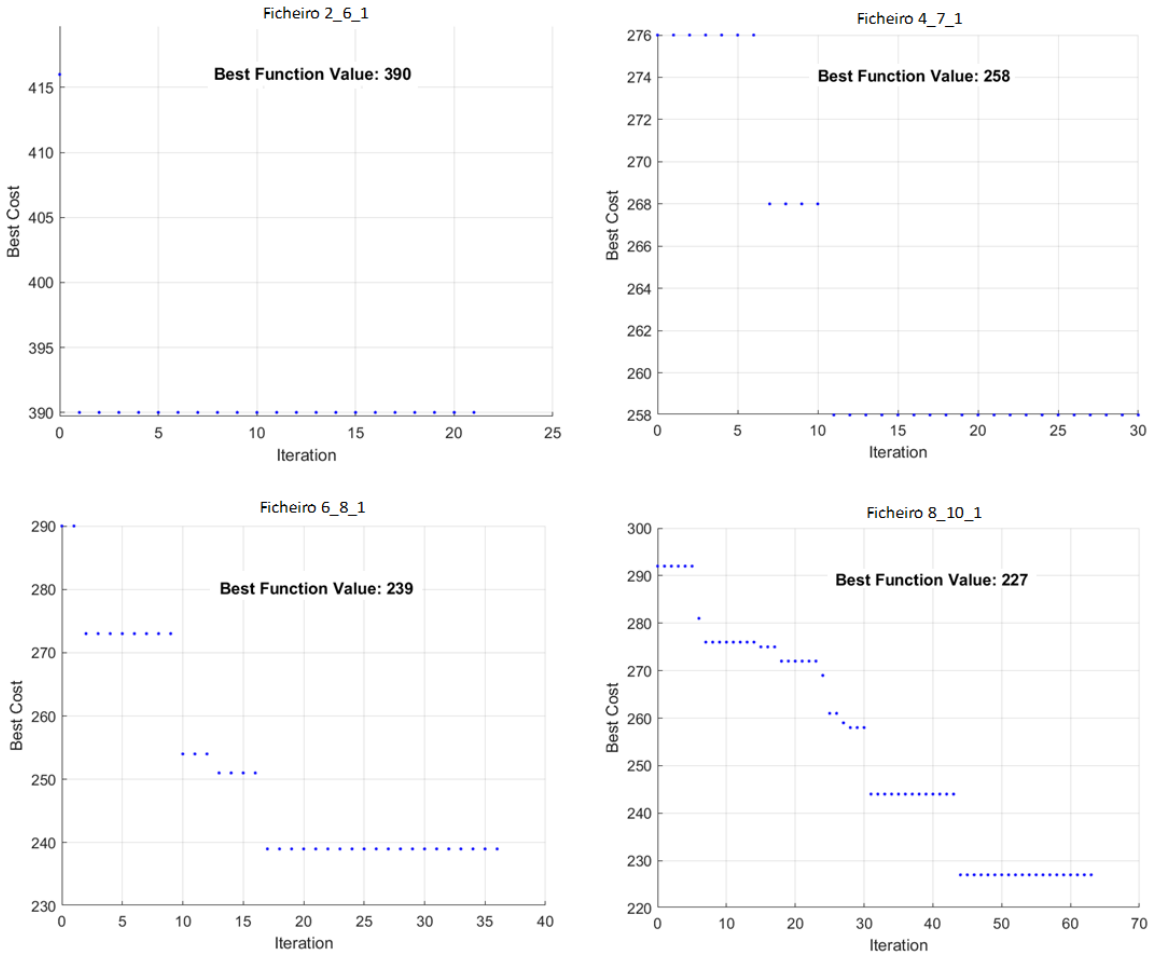

Figura 10 – Escala das iterações, tempo de processamento e *setup* equilibrados.

Na Tabela 6 estão representados os valores médios do *makespan* para pequenos problemas com tempos de processamento e de *setup* equilibrados. Estes valores são uma média das 15 instâncias para os diferentes números de máquinas e de trabalhos.

Tabela 6 - Média do *makespan* para pequenos problemas, Pij, Sij equilibrados

Máquinas	Trabalhos	ACOI	ACOII	SA	GA	PSO
2	6	394,73	394,73	394,73	394,73	394,73
	7	491,00	491,00	491,00	491,00	491,00
	8	517,40	517,40	517,40	517,40	518,40
	9	598,47	598,47	598,47	598,47	598,47
	10	638,93	638,93	638,93	645,00	639,40
	11	710,73	710,73	710,73	715,60	711,40
4	6	245,00	245,00	245,00	245,00	245,00
	7	252,27	252,27	252,27	252,27	252,27
	8	264,73	264,73	264,73	264,80	265,20
	9	346,07	346,07	346,07	346,07	346,07
	10	359,53	359,47	359,47	362,73	359,80
	11	366,47	366,33	366,67	378,87	375,80
6	8	234,47	234,47	234,47	234,87	234,73
	9	238,53	238,53	238,53	243,40	244,60
	10	246,47	246,00	246,87	254,73	255,07
	11	251,60	251,27	252,87	265,73	265,47
8	10	226,13	226,13	226,13	231,13	232,80
	11	232,47	232,47	232,53	243,20	244,40

Na Tabela 7 são apresentados os valores da variação do *makespan* comparando o PSO com os restantes quatro algoritmos. Estes valores foram calculados através da seguinte equação:

$$Variação = \frac{C_{max}\ (algoritmo) - C_{max}(PSO)}{C_{max}(PSO)} \qquad \text{Equação 3}$$

É possível verificar que há casos de estudo em que os valores são melhores, quando a percentagem de variação é negativa. De realçar que isto apenas acontece quando se compara o PSO com o GA.

Tabela 7 – Variação do PSO, Pij, Sij equilibrados

Máquinas	Trabalhos	ACOI	ACOII	SA	GA
2	6	0,00	0,00	0,00	0,00
	7	0,00	0,00	0,00	0,00
	8	0,19	0,19	0,19	0,19
	9	0,00	0,00	0,00	0,00
	10	0,07	0,07	0,07	-0,87
	11	0,09	0,09	0,09	-0,59
4	6	0,00	0,00	0,00	0,00
	7	0,00	0,00	0,00	0,00
	8	0,18	0,18	0,18	0,15
	9	0,00	0,00	0,00	0,00
	10	0,07	0,09	0,09	-0,81
	11	2,55	2,58	2,49	-0,81
6	8	0,11	0,11	0,11	-0,06
	9	2,54	2,54	2,54	0,49
	10	3,49	3,69	3,32	0,13
	11	5,51	5,65	4,98	-0,10
8	10	2,95	2,95	2,95	0,72
	11	5,13	5,13	5,10	0,49

A evolução e a comparação dos valores obtidos para as 15 instâncias do problema podem ser visualizadas na Figura 11. Pela análise da figura, é possível verificar que os dados obtidos pelo PSO apresentam uma variação nas instâncias semelhante aos outros algoritmos. Os valores do *makespan* obtidos pelo PSO são iguais aos valores obtidos por Rabadi et al. (2010), Rabadi et al. (2014) e Amaral et al. (2020), para os algoritmos ACOI, ACOII e SA, respetivamente. Analisando o gráfico, pode-se observar que o PSO apresenta, em algumas instâncias, melhores valores de C_{max} do que o GA. Exemplos disso são as instâncias três, cinco e onze, que apresentam valores de C_{max} inferiores em mais de 2%.

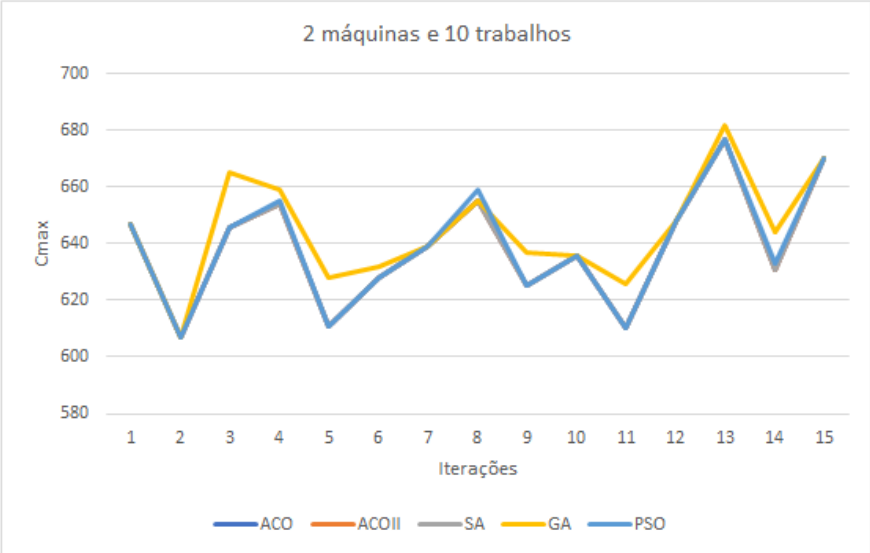

Figura 11 - Comparação de algoritmos para duas máquinas e dez trabalhos.

Na Figura 12 são apresentados quatro gráficos que representam os quatro casos de estudo com maior número de máquinas, seis e oito, e de trabalhos, dez e onze. É possível verificar que os valores apresentam alguma discrepância entre os algoritmos. O ACOI e ACOII têm valores muito parecidos em todos os casos, sendo que em nenhum caso é possível obter valores melhores do que estes, apenas iguais. Para os quatro gráficos, os valores do algoritmo SA são inferiores ao PSO, salvo exceções em que se consegue obter valores iguais como, por exemplo, nas duas primeiras instâncias do modelo com oito máquinas e dez trabalhos. Relativamente à comparação com o GA, os valores oscilam ao longo de todos os modelos, independentemente do número de máquinas ou de trabalhos, o que significa que para algumas instâncias o GA apresenta melhores resultados, mas para outras o melhor é o PSO.

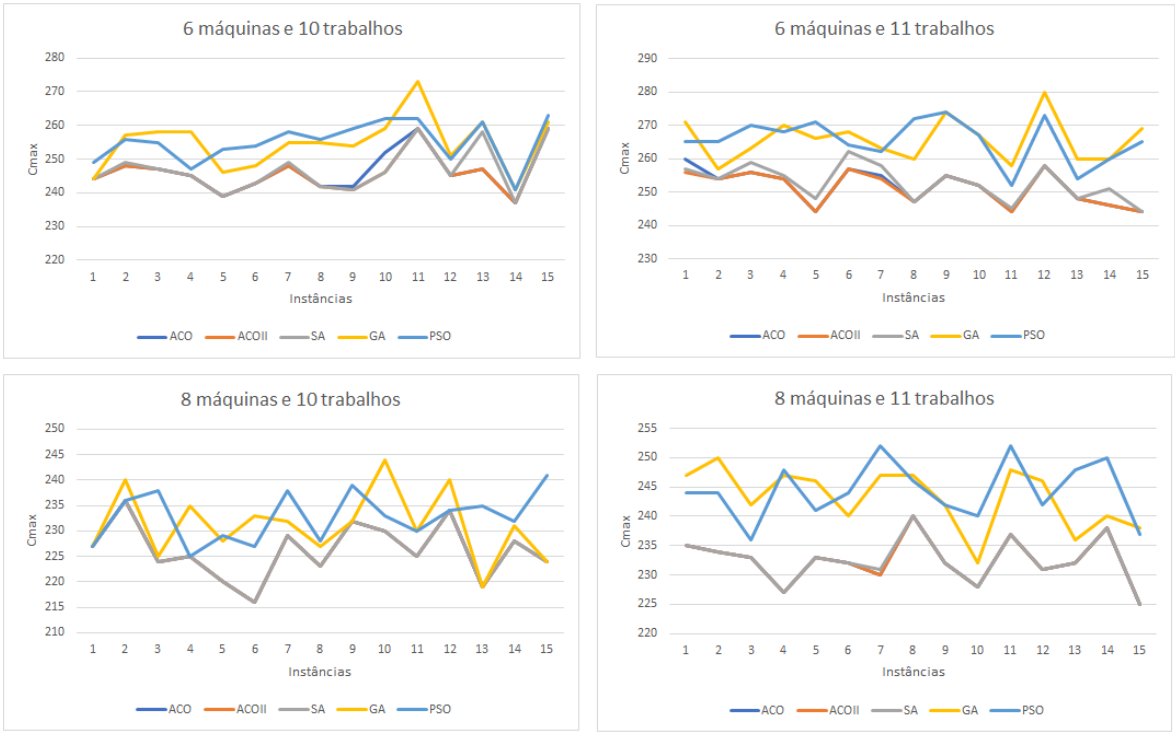

Figura 12 - Comparação de algoritmos com seis e oito máquinas.

Fazendo uma análise geral dos resultados para o caso dos tempos de processamento e *setup* equilibrados, conclui-se que com o aumento do número de trabalhos o algoritmo aplicado consegue piores resultados do que os restantes, principalmente para os modelos com seis e oito máquinas, comparando com os algoritmos ACO, ACOII e SA.

No caso do GA, as conclusões são diferentes, uma vez que com o aumento do número de trabalhos, existem instâncias em que o PSO conseguiu, em média, melhores resultados, com exceção dos modelos com oito máquinas.

No entanto, há também muitas instâncias em que se conseguiu atingir os valores obtidos pelos restantes algoritmos, por exemplo, nos casos em que existem seis e sete trabalhos para duas e quatro máquinas.

5.2.2 Tempo de processamento dominante

Nesta secção serão analisados os resultados quando o tempo de processamento é dominante. Foi escolhido apresentar apenas um exemplo da ordem de trabalhos para cada caso de estudo de duas, quatro, seis e oito máquinas. Na Figura 13 é possível observar a ordem dos trabalhos e o respetivo C_{max} para esses exemplos, com duas máquinas e seis trabalhos o *makespan* é de 633. Para o caso de existirem quatro máquinas e oito trabalhos, o C_{max} é de 409. Com o

aumento do número de máquinas, para seis, o *makespan* diminui para 388, uma vez que os trabalhos têm mais opções para serem alocados. Por fim, para oito máquinas e dez trabalhos, o C_{max} é de 374. Este valor é mais baixo do que o anterior, uma vez que apesar de serem mais trabalhos, o número de máquinas também é superior.

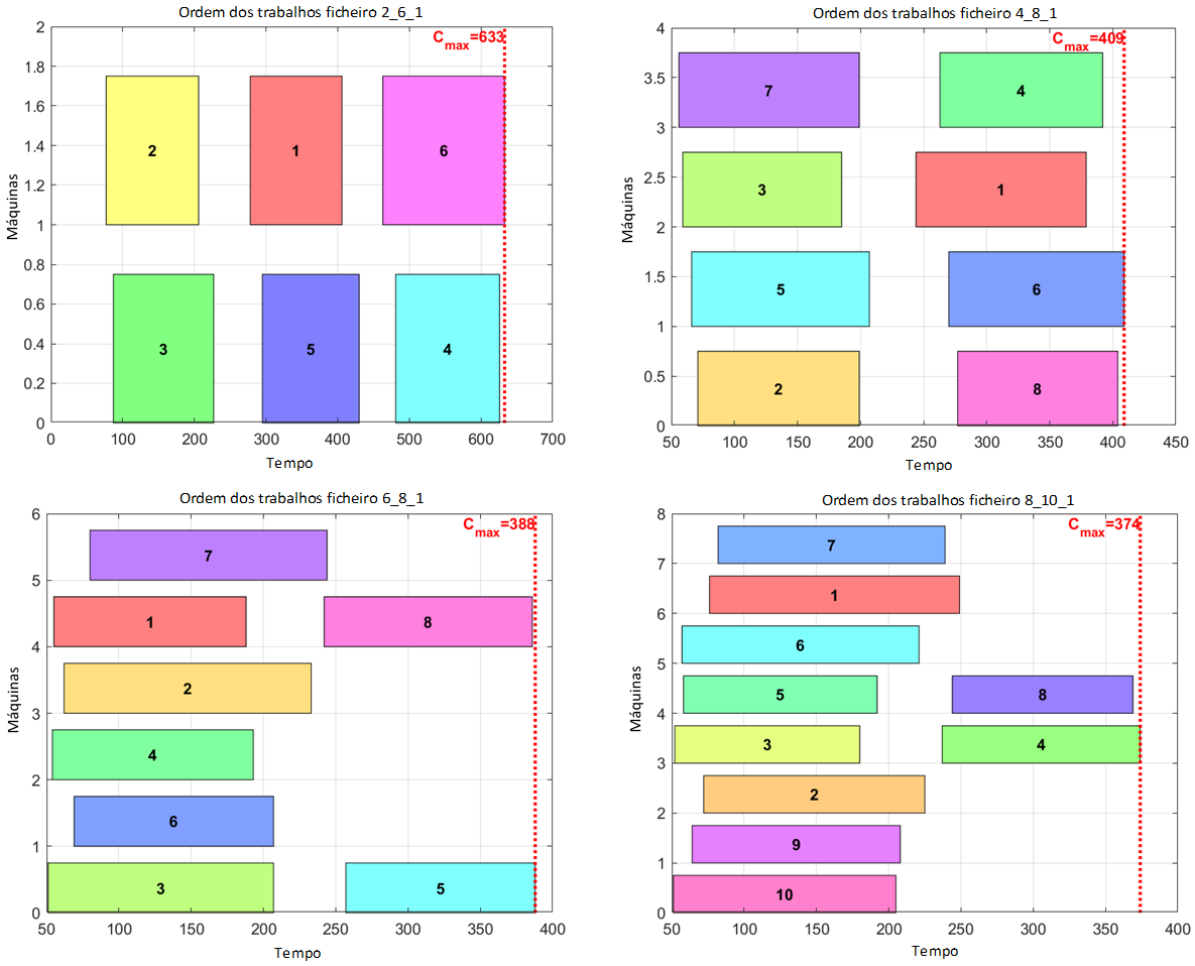

Figura 13 - Ordem de trabalhos, tempo de processamento dominante.

A variação do valor do C_{max} (*Best Cost*) ao longo das iterações para quatro casos é apresentada na Figura 14. Para o ficheiro 2_6_1 encontrar a solução ótima foram precisas apenas três iterações, atingindo depois o valor de 633. O exemplo do ficheiro 6_8_1 foi o que precisou de mais iterações até encontrar o melhor valor, apenas conseguido na vigésima primeira iteração, mantendo-se estável com um valor de 388. Para o modelo com quatro máquinas, o valor do C_{max} desceu nas primeiras dez iterações e para oito máquinas desceu nas primeiras catorze iterações. A partir daí manteve-se constante e com o valor ótimo de 409 e 374, respetivamente.

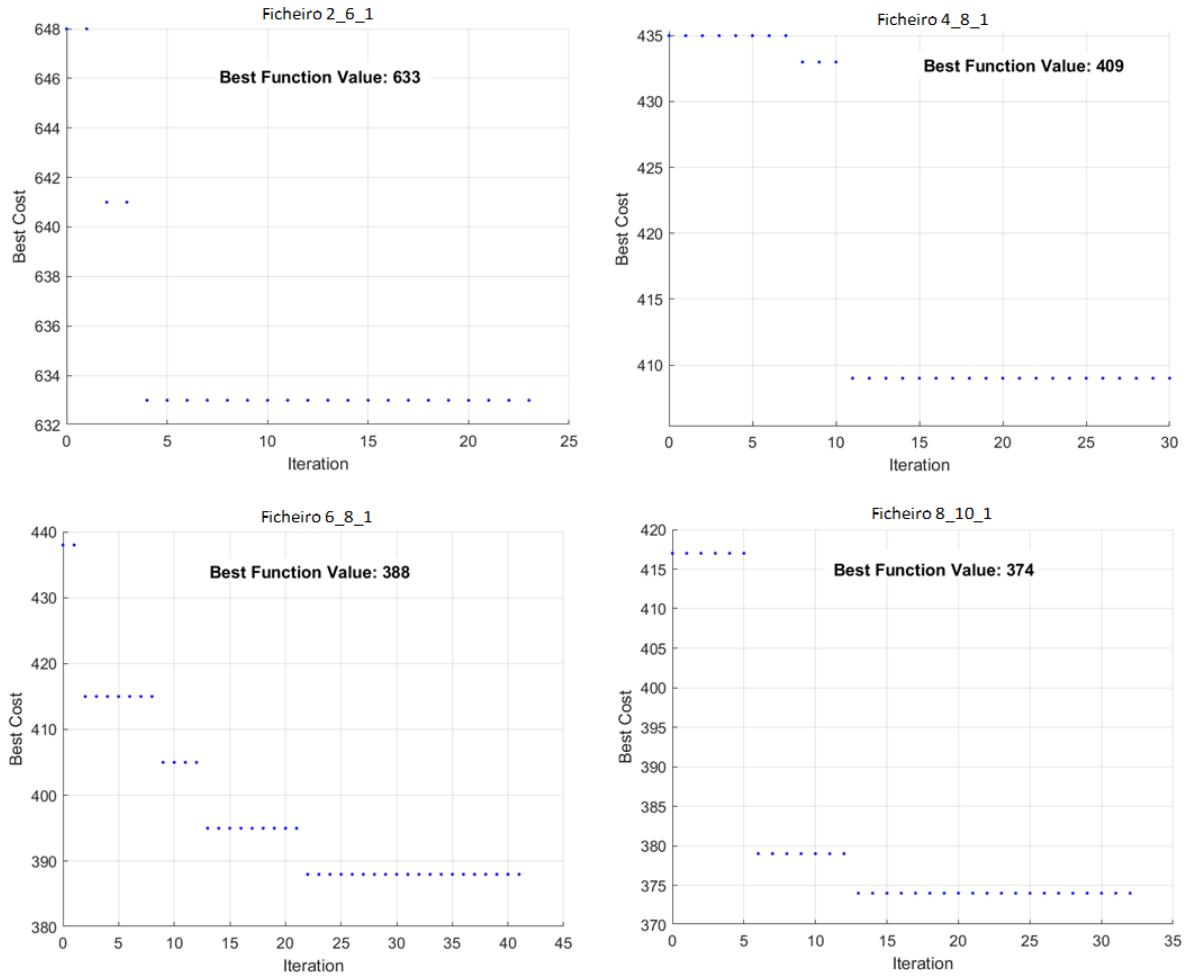

Figura 14 – Escala das iterações, tempo de processamento dominante.

Na Tabela 8 estão representados os valores médios do *makespan* para pequenos problemas com tempos de processamento dominantes. Estes valores são uma média das quinze instâncias para os diferentes números de máquinas e de trabalhos. De realçar que existem três valores dos algoritmos SA e GA que não estavam disponíveis para comparação.

Quando os tempos de processamento são dominantes, analisando apenas os modelos com duas e quatro máquinas, na maior parte das instâncias os valores de *makespan* obtidos são iguais aos algoritmos já aplicados, existindo alguns casos em que os valores são inferiores. No Apêndice 5 – Gráficos para 20 trabalhos de tempo de *setup* dominante estão presentes alguns gráficos que permitem ver a evolução dos valores ao longo das iterações.

Tabela 8 - Média do *makespan* para pequenos problemas, Pij dominante

Máquinas	Trabalhos	ACOI	ACOII	SA	GA	PSO
2	6	630,00	630,00	630,00		630,00
	7	787,13	787,13	787,13	787,13	787,13
	8	823,53	823,53	823,53	823,60	823,60
	9	988,60	988,60		989,07	988,60
	10	1012,87	1012,87	1013,80	1016,20	1013,67
	11	1164,40	1164,40	1164,67	1166,20	1164,40
4	6	397,40	397,40	397,27		397,27
	7	404,53	404,53	404,53	405,47	404,73
	8	414,40	414,40	414,40	415,47	416,60
	9	569,07	569,07	567,47	567,53	567,47
	10	583,60	583,60	587,00	587,00	585,80
	11	592,60	591,47	592,00	600,80	599,00
6	8	385,07	385,07	385,07	385,80	385,87
	9	391,20	391,20	391,20	394,67	394,87
	10	394,33	394,20	394,53	401,60	402,93
	11	399,93	399,87	400,33	413,60	414,60
8	10	376,20	376,13	376,13	380,27	381,20
	11	379,73	379,73	380,80	389,73	392,67

Na Tabela 9 estão presentes as percentagens de variação dos valores médios do PSO em comparação com os restantes quatro algoritmos. Tal como referido anteriormente, para duas e quatro máquinas muitos dos valores têm variação nula, uma vez que os resultados são iguais. No entanto, existem sete casos em que foi possível encontrar uma solução melhor do que a solução do GA, com duas máquinas e nove, dez e onze trabalhos e com quatro máquinas e sete, nove, dez e onze trabalhos. De realçar, na comparação com os algoritmos ACOI e ACOII,

as percentagens -0,03 e -0,28 correspondentes a quatro máquinas e seis e nove trabalhos, respetivamente. Isto significa que, em algumas instâncias, foi possível encontrar melhores resultados do que os obtidos por Rabadi et al. (2010, 2014).

Tabela 9 – Variação do PSO, Pij dominante

Máquinas	Trabalhos	ACOI	ACOII	SA	GA
2	6	0,00	0,00	0,00	
	7	0,00	0,00	0,00	0,00
	8	0,01	0,01	0,01	0,00
	9	0,00	0,00		-0,05
	10	0,08	0,08	-0,01	-0,25
	11	0,00	0,00	-0,02	-0,15
4	6	-0,03	-0,03	0,00	
	7	0,05	0,05	0,05	-0,18
	8	0,53	0,53	0,53	0,27
	9	-0,28	-0,28	0,00	-0,01
	10	0,38	0,38	-0,20	-0,20
	11	1,08	1,27	1,18	-0,30
6	8	0,21	0,21	0,21	0,02
	9	0,94	0,94	0,94	0,05
	10	2,18	2,22	2,13	0,33
	11	3,67	3,68	3,56	0,24
8	10	1,33	1,35	1,35	0,25
	11	3,41	3,41	3,12	0,75

Os quatro gráficos apresentados na Figura 15 são exemplos de modelos em que se conseguiu obter melhores resultados com a implementação do PSO comparando com os restantes algoritmos. Analisando os gráficos presentes na parte superior da figura correspondentes a duas máquinas, estes mostram que em várias instâncias os valores do *makespan* do PSO são menores em relação aos valores do *Genetic Algorithm*.

Para dez trabalhos são destacadas as descidas de 1,56% e 0,98%, nas instâncias 5 e 15, respetivamente. No caso de onze trabalhos, a maior descida é na instância 11, com uma variação de -1,03%. Comparando com o SA, houve uma descida de 0,98% na instância 5. Este foi o único caso em que se conseguiu melhor resultado do que o SA.

Para quatro máquinas e sete trabalhos, os resultados obtidos pelo PSO são iguais aos resultados dos algoritmos ACO, ACOII e SA. No entanto, comparando com o GA há uma descida de 0,5%, 1,22% e 1,67% em três instâncias.

De destacar o caso de quatro máquinas e nove trabalhos em que é possível observar que, na instância 12, o PSO obteve uma melhoria de 4,6% em relação ao ACO e ACOII.

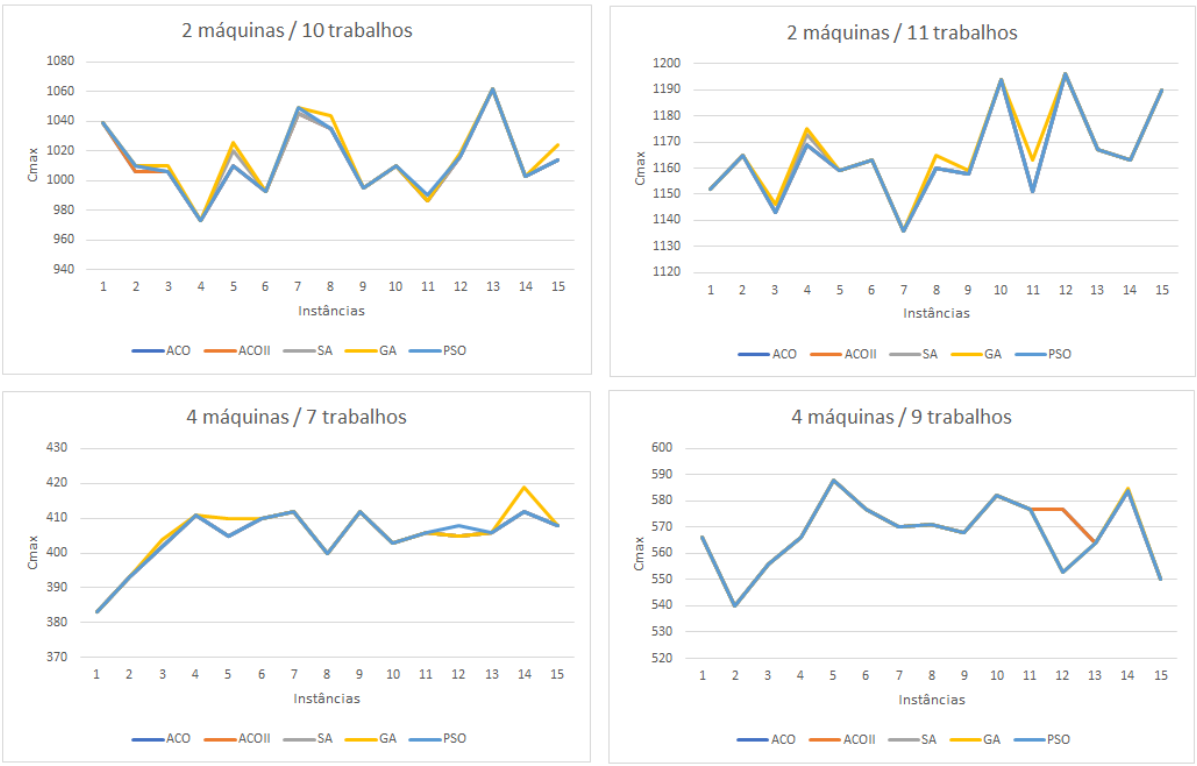

Figura 15 - Comparação de algoritmos com duas e quatro máquinas

De forma a perceber a influência do número de trabalhos no resultado decidiu-se comparar os valores obtidos quando o número de máquinas é seis, Figura 16. Assim, é percetível que à medida que o número de trabalhos aumenta, as disparidades entre algoritmos são maiores. No primeiro gráfico, com oito trabalhos, as diferenças entre os resultados dos quatro algoritmos são pequenas, embora existam algumas instâncias em que os resultados do PSO são piores do que os restantes. No entanto, na instância três, a variação é de -2,76% em

relação ao GA. Para nove trabalhos, já é notório que há mais disparidades entre resultados, no entanto, ainda existem algumas instâncias em que os valores são iguais.

Da análise dos gráficos com dez e onze trabalhos conclui-se que os valores, tanto do PSO como do GA, estão acima dos valores apresentados pelo ACO, ACOII e SA. Ao longo das quinze instâncias, comparando apenas com o GA, os valores oscilam, ou seja, tanto são melhores como piores, no entanto, a diferença entre estes dois algoritmos é a mais pequena em relação aos restantes, ou seja, apresentam valores muito próximos.

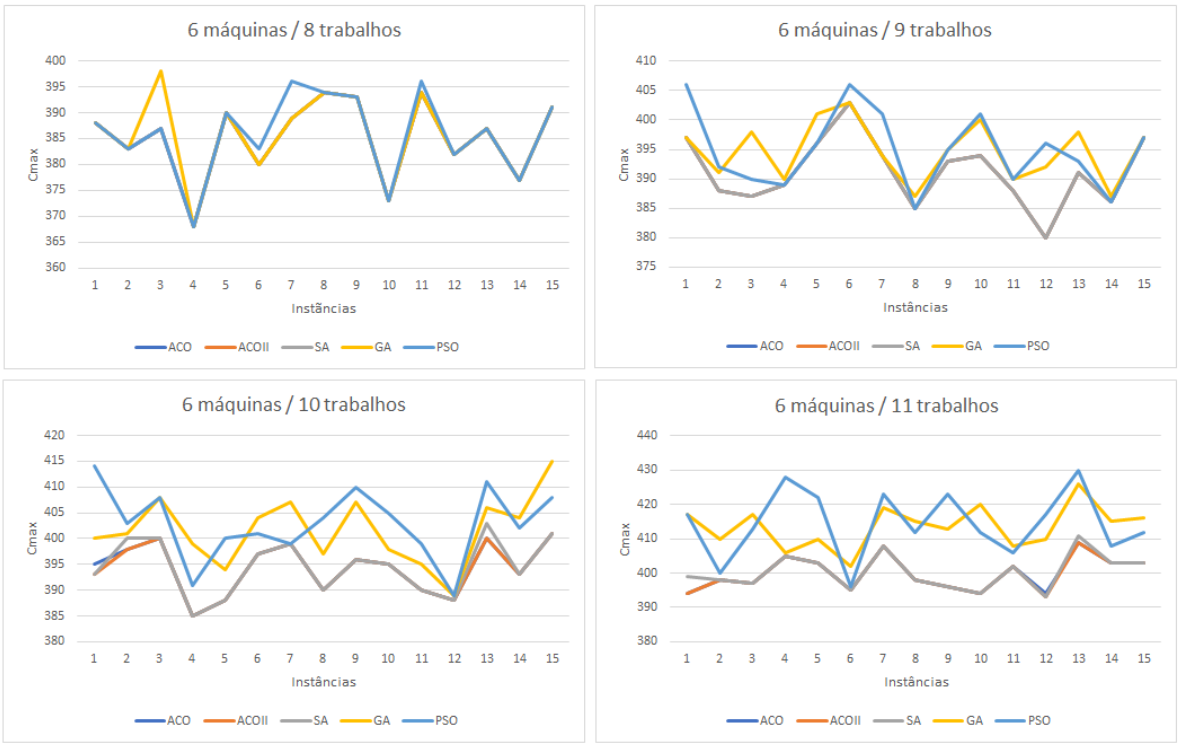

Figura 16 - Comparação entre algoritmos para seis máquinas

Na Figura 17 são apresentados os gráficos com os resultados para oito máquinas. Mais uma vez os resultados dos algoritmos PSO e GA são os que mais se aproximam, sendo que, em média, os valores do PSO são superiores em 0,25 e 0,75% para dez e onze trabalhos, respetivamente. Com o aumento do número de trabalhos, as disparidades entre o PSO e os algoritmos ACO e ACOII aumentam, apresentando uma subida de, aproximadamente, 1,3% para dez trabalhos e 3,4% para onze trabalhos, Tabela 9.

Figura 17 - Comparação de algoritmos com oito máquinas

Concluindo a análise dos resultados quando o tempo de processamento é dominante, é possível afirmar que com o aumento do número de trabalhos, os valores do PSO tendem a afastarem-se dos resultados dos algoritmos ACO, ACOII e SA, sendo que os valores com maior percentagem de diferença correspondem aos modelos com dez e onze trabalhos. Quando tanto o número de máquinas como de trabalhos são menores, os resultados são, na maior parte das vezes, iguais ou melhores do que os restantes algoritmos. Isto acontece uma vez que é mais fácil encontrar os melhores valores quando o número de trabalhos e máquinas é mais baixo, visto que existem menos soluções possíveis.

O GA é o algoritmo que apresenta valores mais próximos do PSO, seja a diferença positiva ou negativa, sendo a diferença mais alta apenas de 0,8%.

5.2.3 Tempo de *setup* dominante

Nesta secção são analisados os resultados quando o tempo de *setup* é dominante, sendo escolhido apresentar um exemplo para cada caso de estudo de duas, quatro, seis e oito máquinas. A ordem dos trabalhos e o respetivo *makespan* para esses exemplos estão presentes na Figura 18. Quando existem duas máquinas e seis trabalhos o *makespan* é de 626, sendo alocados os trabalhos um, três e seis à máquina um e os restantes à máquina dois. Com o aumento das máquinas, e apesar do aumento do número de trabalhos para sete, o *makespan* diminui para 396, devido à existência de mais opções para alocar os mesmos. Para o caso de seis máquinas e oito trabalhos, o C_{max} atinge os 381 e, por fim, para oito máquinas e dez trabalhos, o C_{max} é de 365. Este valor é mais baixo do que o anterior, uma vez que apesar de serem mais trabalhos, o número de máquinas também é superior. De realçar que,

nos casos em que existem mais máquinas, apesar de todas as máquinas serem ocupadas, a maior parte apenas processa um trabalho.

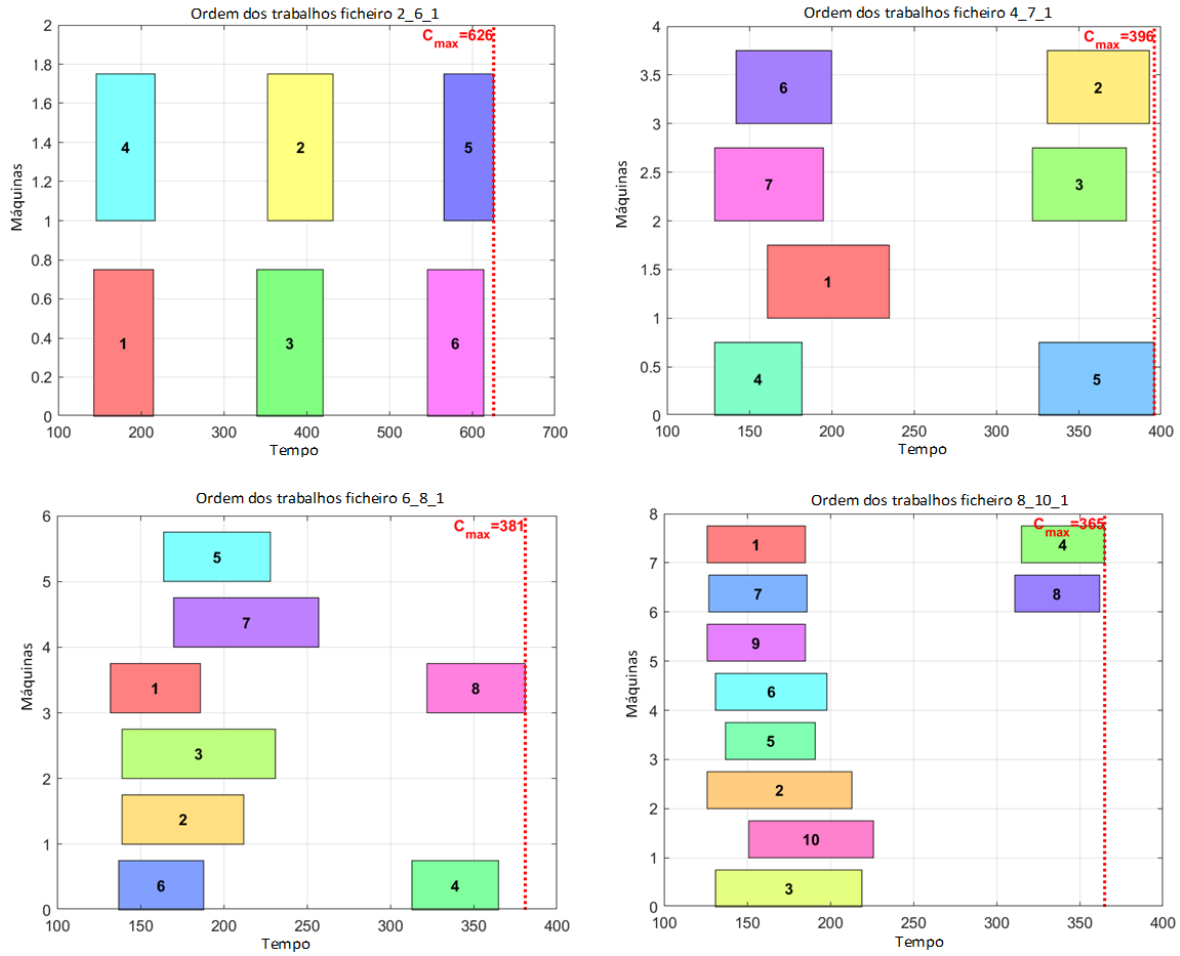

Figura 18 - Ordem de trabalhos, tempo de *setup* dominante.

Na Figura 19 estão presentes exemplos da variação do C_{max} ao longo de várias iterações até ser encontrado o melhor valor. No ficheiro 2_6_1 foram apenas precisas três iterações para encontrar esse valor, enquanto que para quatro máquinas foram necessárias onze iterações para atingir um *makespan* de 396.

Os dois últimos exemplos, para seis e oito máquinas, precisaram de mais iterações para encontrar o melhor valor. Para encontrar este valor foram necessárias 44 e 57 iterações, respetivamente.

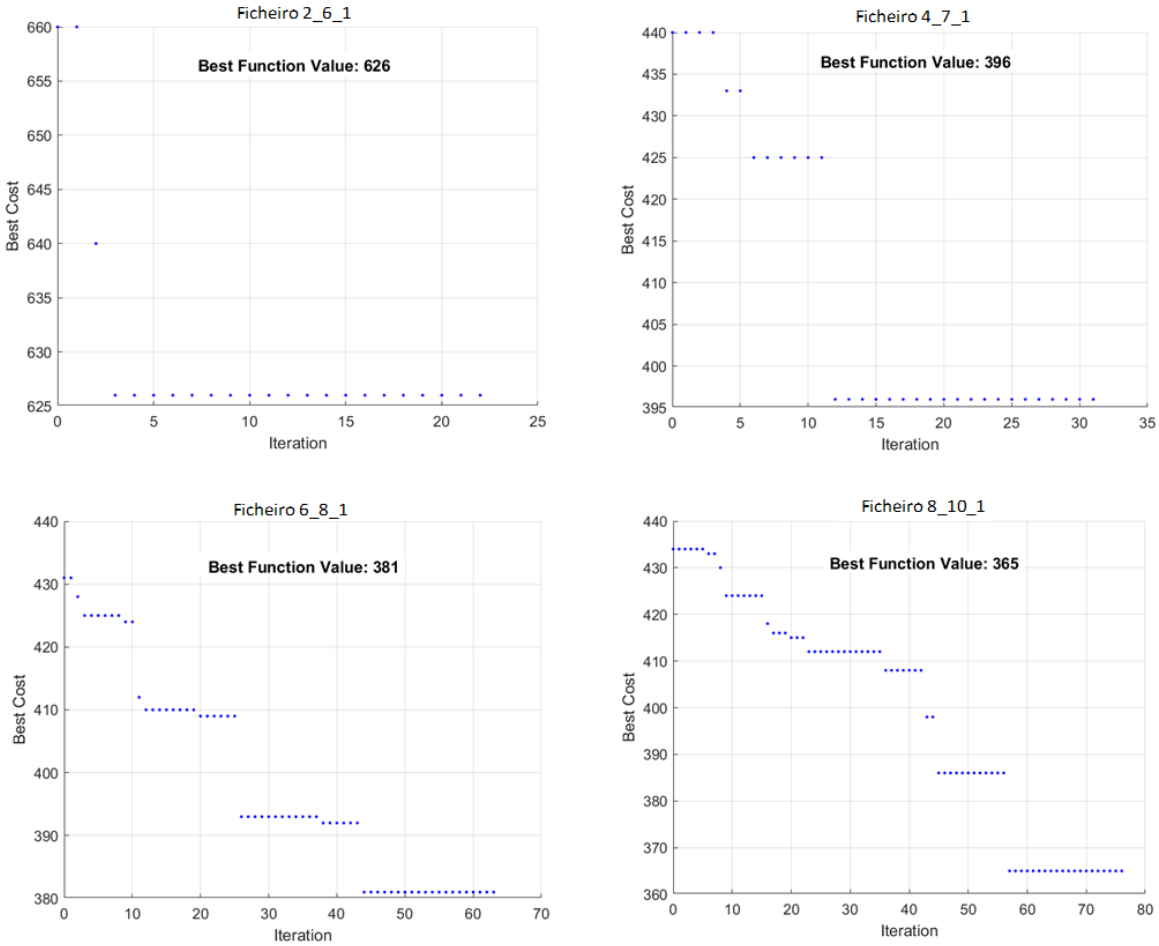

Figura 19 – Escala de iterações, tempo de *setup* dominante.

Os valores médios do *makespan* para vários números de máquinas e trabalhos quando o tempo de *setup* é dominante são apresentados na Tabela 10. Os resultados para os casos de estudo com duas e quatro máquinas são, na sua maioria, iguais aos resultados já presentes na literatura, salvo poucas exceções em que tanto foi possível melhorar os valores como também houve resultados piores do que os utilizados na comparação. De realçar que a partir do caso com quatro máquinas e onze trabalhos há uma tendência que o valores do C_{max} do PSO sejam maiores do que os dos restantes algoritmos, com exceção do *Genetic Algorithm*.

Tabela 10 - Média do *makespan* para pequenos problemas, Sij dominante

Máquinas	Trabalhos	ACOI	ACOII	SA	GA	PSO
2	6	628,47	628,47	628,47	628,47	628,47
	7	784,40	784,40	784,40	784,40	784,40
	8	820,27	820,27	820,27	820,67	820,87
	9	977,13	977,13	977,13	977,13	977,13
	10	1020,67	1020,33	1020,60	1022,67	1022,33
	11	1162,40	1162,40	1162,40	1164,33	1162,40
4	6	396,67	396,67	396,67	396,67	396,67
	7	399,87	399,87	400,07	400,20	400,07
	8	415,47	415,47	415,47	416,20	416,87
	9	568,20	568,20	568,13	568,13	568,13
	10	579,40	579,40	579,40	582,33	580,87
	11	593,87	593,87	594,40	606,67	601,53
6	8	385,07	385,07	385,07	385,40	386,33
	9	388,40	388,40	388,40	393,60	391,93
	10	392,27	392,27	394,13	403,80	402,47
	11	399,20	399,20	400,87	412,33	414,33
8	10	374,93	374,93	374,93	381,40	380,40
	11	379,87	379,87	381,13	391,33	392,40

No seguimento desta análise apresenta-se a Tabela 11, na qual é apresentada a percentagem de variação do *makespan* entre os quatro algoritmos escolhidos para o presente estudo. Comparando com os algoritmos ACOI, ACOII e SA, para os casos de duas e quatro máquinas, não foi possível encontrar melhores resultados do que os apresentados por estes, o máximo que se atingiu foi o mesmo valor. No entanto, comparando com o GA, em cinco dos casos estudados, foi possível encontrar melhores soluções, em média, apesar das variações serem muito pequenas.

Tabela 11 - Variação do PSO, Sij dominante

Máquinas	Trabalhos	ACOI	ACOII	SA	GA
2	6	0,00	0,00	0,00	0,00
	7	0,00	0,00	0,00	0,00
	8	0,07	0,07	0,07	0,02
	9	0,00	0,00	0,00	0,00
	10	0,16	0,20	0,17	-0,03
	11	0,00	0,00	0,00	-0,17
4	6	0,00	0,00	0,00	0,00
	7	0,05	0,05	0,00	-0,03
	8	0,34	0,34	0,34	0,16
	9	-0,01	-0,01	0,00	0,00
	10	0,25	0,25	0,25	-0,25
	11	1,29	1,29	1,20	-0,85
6	8	0,33	0,33	0,33	0,24
	9	0,91	0,91	0,91	-0,42
	10	2,60	2,60	2,11	-0,33
	11	3,79	3,79	3,36	0,49
8	10	1,46	1,46	1,46	-0,26
	11	3,30	3,30	2,96	0,27

Depois da análise da Tabela 11 percebe-se que os modelos com dez e onze trabalhos apresentam maior variação quando o número de máquinas é dois. Assim, na Figura 20 estão presentes os gráficos que permitem a comparação do *makespan* entre os cinco algoritmos. Com dez trabalhos a descida em relação ao GA é menor, apenas 0,03%, comparando com onze trabalhos que é de 0,17%. No entanto, neste último caso os resultados do PSO são iguais aos restantes três algoritmos. Pelo contrário, com dez trabalhos, existe um aumento do *makespan* de 0,16% comparando com o ACO, de 0,2% com o ACOII e de 0,17% com o SA.

Figura 20 – Comparação de algoritmos para duas máquinas

Os resultados para quatro máquinas são apresentados na Figura 21. Dos quatro exemplos escolhidos, o único que apresenta melhores resultados do que o ACO e ACOII é o caso com nove trabalhos, que na instância 12 tem um decréscimo de 0,18% no *makespan*. Os restantes três exemplos apresentam todos, em média, valores do *makespan* maiores do que os algoritmos ACO, ACOII e SA. No entanto, em comparação com o GA, existem instâncias com diferenças tanto positivas como negativas, mas para dez e onze trabalhos há um decréscimo, em média, de 0,25 e 0,85%, respetivamente.

Para onze trabalhos a maior parte das instâncias dos algoritmos ACO, ACOII e SA apresentam melhores resultados do que o PSO. Pelo contrário, quando comparado com o GA, os valores do *makespan* do PSO são mais baixos em todas as instâncias, exceto dois casos, instâncias 13 e 14, destacando a maior diferença de 2,13%.

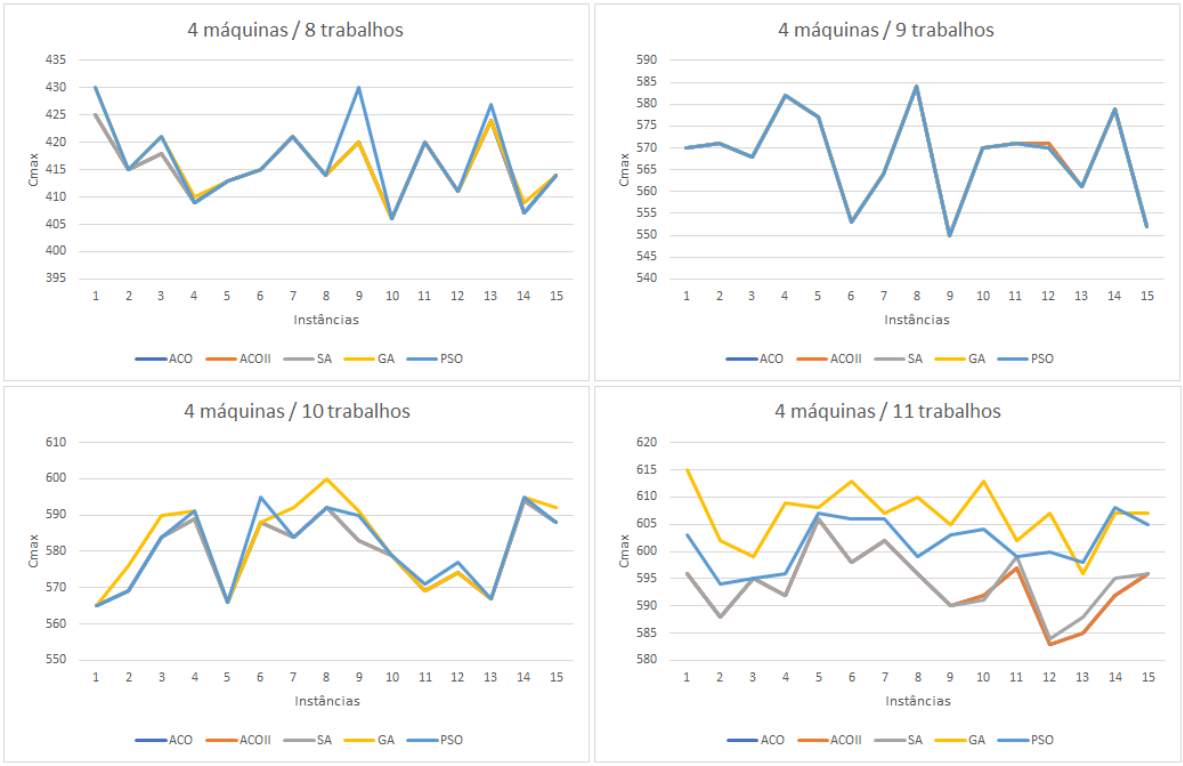

Figura 21 - Comparação entre algoritmos para quatro máquinas

Os resultados do *makespan* para problemas com seis máquinas são apresentados na Figura 22. Pela análise desta figura é percetível que quanto mais trabalhos são utilizados, maior é a disparidade entre os resultados dos algoritmos. Com oito trabalhos a maior parte das instâncias apresentam os mesmos valores que os restantes algoritmos, no entanto, existem dois casos em que o *makespan* aumentou quando aplicado o PSO. Pelo contrário, existem também duas instâncias em que o *makespan* diminui, no entanto, isto apenas acontece quando comparado com o GA.

Para nove trabalhos existem mais resultados em que foi possível atingir um valor de *makespan* menor do que o GA, destacando a descida de 2,75% na instância 6. Pelo contrário, nos restantes algoritmos há um aumento do número de instâncias em que o *makespan* do PSO é superior aos restantes, existindo apenas cinco instâncias em que os resultados são iguais.

Para os restantes dois modelos, com dez e onze trabalhos, os valores têm um comportamento semelhante entre si, ou seja, quando comparado com o ACO, ACOII e SA os valores do *makespan* do PSO são, em média, superiores aos encontrados pelos três algoritmos. Quando comparado com o GA, mais uma vez, os valores variam, destacando que, para dez trabalhos,

em média, os valores do PSO são melhores 0,33%, enquanto que para onze trabalhos há um aumento de 0,49%, Tabela 11.

Em geral, analisando a Tabela 11 e a Figura 22 percebe-se que à medida que o número de trabalhos aumenta as diferenças entre os resultados entre o PSO e os restantes algoritmos tendem a aumentar.

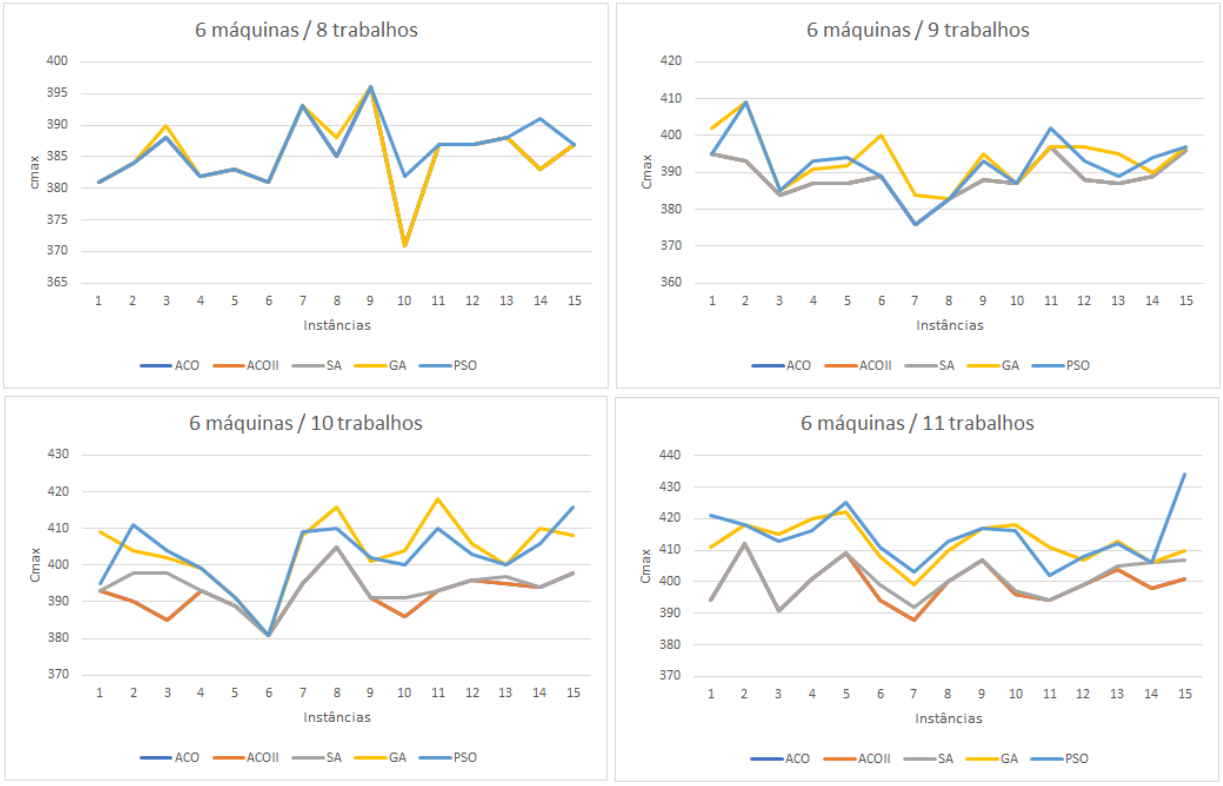

Figura 22 - Comparação entre algoritmos para seis máquinas com tempos de *setup* dominantes

Na Figura 23 estão presentes os resultados do modelo com oito máquinas. Comparando apenas com os algoritmos *Ant Colony* em todas as instâncias o PSO obteve valores mais altos, em média, 1,46 e 3,3%, para dez e onze trabalhos, respetivamente. Destacar com onze trabalhos, a instância 7 em que o PSO desceu 1,29% em relação ao SA. Relativamente ao GA, a análise é igual para o modelo com seis máquinas, ou seja, com dez trabalhos foi possível encontrar melhores resultados, descendo, em média, 0,26%, mas para onze trabalhos o *makespan* aumentou 0,27%, em média.

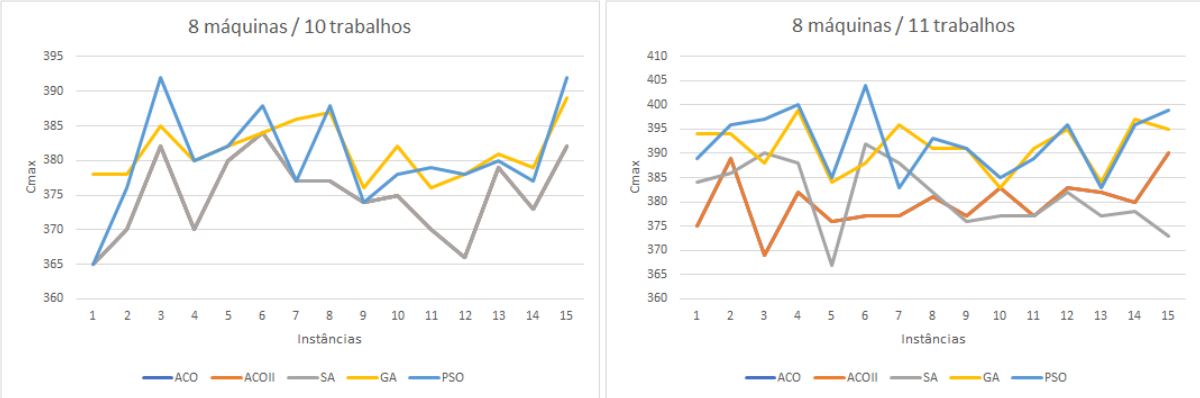

Figura 23 - Comparação de algoritmos para oito máquinas

Concluindo a análise dos resultados quando o tempo de *setup* é dominante, é possível afirmar que com o aumento do número de trabalhos, os valores do PSO tendem a afastarem-se dos resultados dos algoritmos ACO, ACOII e SA, sendo que os valores com maior percentagem de diferença correspondem aos modelos com dez e onze trabalhos. Quando o número de máquinas e de trabalhos são menores, os resultados são, na sua maioria, iguais ou melhores do que os restantes algoritmos. O GA é o algoritmo que apresenta valores mais próximos do PSO, seja a diferença positiva ou negativa.

5.3 Grandes problemas

Nesta secção irão ser analisados os resultados obtidos para o caso de estudo de grandes problemas com duas, quatro, seis, oito, dez e doze máquinas e 20, 40, 60, 80, 100 e 120 trabalhos. A análise é feita consoante a classificação dos tempos de processamento e de *setup*, estando dividida entre tempos de processamento e *setup* equilibrados, tempos de processamento dominantes e tempos de *setup* dominante.

5.3.1 Tempos de processamento e *setup* equilibrados (*balanced*)

Tal como já foi referido anteriormente, existem três casos diferentes relativamente à geração dos tempos de processamento e de *setup* dos trabalhos nas máquinas. Assim, começa-se por analisar os resultados relativos aos tempos de processamento e de *setup* equilibrados, ou seja, quando ambos são gerados a partir da distribuição U (50, 100).

De forma a ilustrar a ordem dos trabalhos obtida com a aplicação do PSO, foram escolhidas duas sequências de trabalhos bastantes diferentes, apresentadas na Figura 24. No gráfico do

lado esquerdo da figura, existem apenas duas máquinas nas quais são alocados 20 trabalhos, correspondendo um C_{max} de 1335. Pelo contrário, no gráfico do lado direito, estão alocados 80 trabalhos a oito máquinas, com um C_{max} de 1429. No Apêndice 9 – Ordem dos trabalhos para tempos de processamento e *setup* equilibrados estão presentes exemplos de cada um dos modelos.

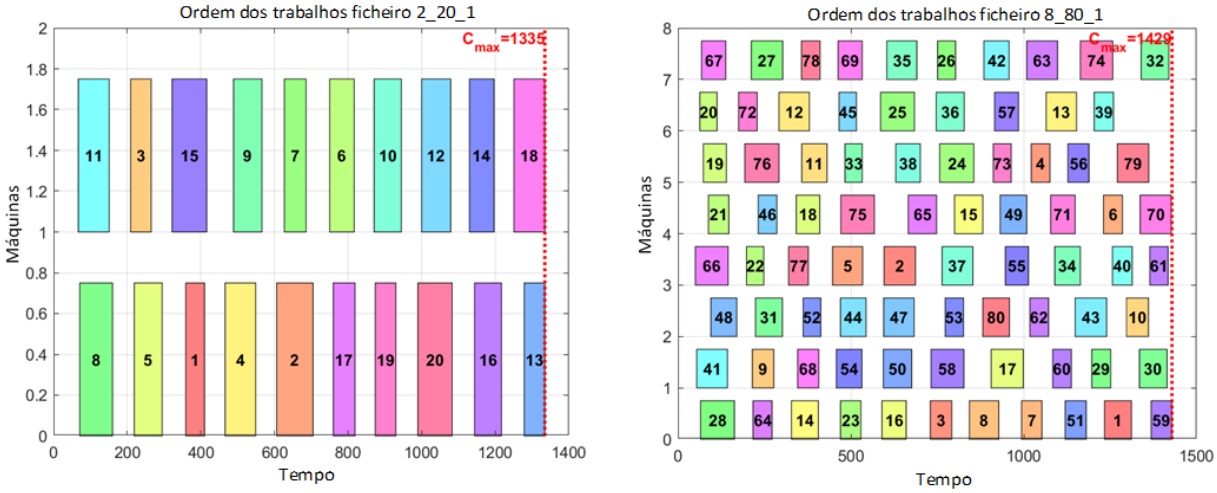

Figura 24 - Ordem de trabalhos com duas e oito máquinas

Na Tabela 12 estão presentes as percentagens de variação entre o *Particle Swarm* e os algoritmos utilizados na comparação. Há um claro destaque para o *Genetic Algorithm* por ser o único que apresenta valores negativos, o que significa que o PSO apenas conseguiu superar este algoritmo para o caso de grandes problemas e com tempos de processamento e *setup* equilibrados. Os restantes três algoritmos apresentam valores de *makespan* mais baixos do que o PSO, sendo que ordenando por ordem crescente as diferenças, encontram-se o SA, o ACOII e, por fim, o ACO.

Tabela 12 - Variação do PSO para grandes problemas, Pij Sij equilibrados

Máquinas	Trabalhos	ACOI	ACOII	SA	GA
2	20	3,22	3,43	1,96	-1,83
	40	8,00	8,13	3,38	-3,27
	60	10,78	11,07	4,22	-3,35
	80	12,64	12,82	4,46	-3,48
	100	13,54	13,85	4,16	-3,56
	120	14,68	14,84	4,76	-3,51
4	20	7,54	8,89	5,32	-1,09
	40	13,26	14,52	6,16	-2,50
	60	16,63	17,46	5,90	-3,39
	80	18,71	19,21	5,59	-3,52
	100	19,56	20,07	4,82	-4,29
	120	21,43	21,98	5,29	-3,75
6	20	7,50	9,15	6,76	-0,61
	40	14,97	16,93	6,44	-3,29
	60	19,36	20,99	6,49	-4,72
	80	20,50	21,68	5,96	-4,72
	100	22,61	23,59	5,03	-4,75
	120	24,23	25,33	4,73	-4,64
8	20	11,49	13,89	8,23	0,02
	40	19,47	23,28	8,55	-4,75
	60	19,83	21,65	6,05	-5,38
	80	24,99	26,23	6,32	-5,59
	100	24,58	26,08	6,55	-5,41
	120	26,90	28,09	4,98	-5,90
10	20	19,46	22,86	13,55	-0,22
	40	22,23	24,64	10,60	-6,68
	60	24,80	28,35	8,85	-6,42
	80	26,95	30,15	7,33	-7,11
	100	28,66	30,38	6,58	-7,35
	120	29,94	31,43	6,08	-7,73
12	20	15,05	18,82	8,87	-0,26
	40	19,34	22,40	10,39	-4,33
	60	28,86	34,22	10,80	-7,87
	80	27,31	30,10	7,14	-8,44
	100	27,46	30,09	7,68	-9,87
	120	31,68	34,67	6,70	-9,80

Pela análise tanto da Tabela 12 como do Apêndice 2 – Resultados dos algoritmos para grandes problemas e tempos de processamento e setup equilibrados, percebe-se que à medida que o número de trabalhos aumenta, para cada máquina, a diferença entre os resultados do PSO e dos restantes algoritmos também aumenta. Por exemplo, comparando com o ACO, para dez máquinas, o *makespan* do PSO aumentou 19,46% com 20 trabalhos, 22,23% com 40 trabalhos, e assim sucessivamente, até atingir um aumento de 29,94% com 120 trabalhos.

Esta conclusão em relação aos resultados é transversal ao caso dos tempos de processamento e *setup* equilibrados pelo que foi decidido apresentar apenas alguns gráficos ilustrativos do comportamento dos resultados de forma a simplificar a análise.

Nesse sentido, na Figura 25 estão presentes quatro exemplos de modelos com 20 trabalhos, e com seis, oito, dez e doze máquinas. Há um claro destaque do PSO e do GA, pois são os dois algoritmos que apresentam maior *makespan*. Os resultados dos dois oscilam ao longo das quinze instâncias, não existindo uma tendência clara do seu comportamento.

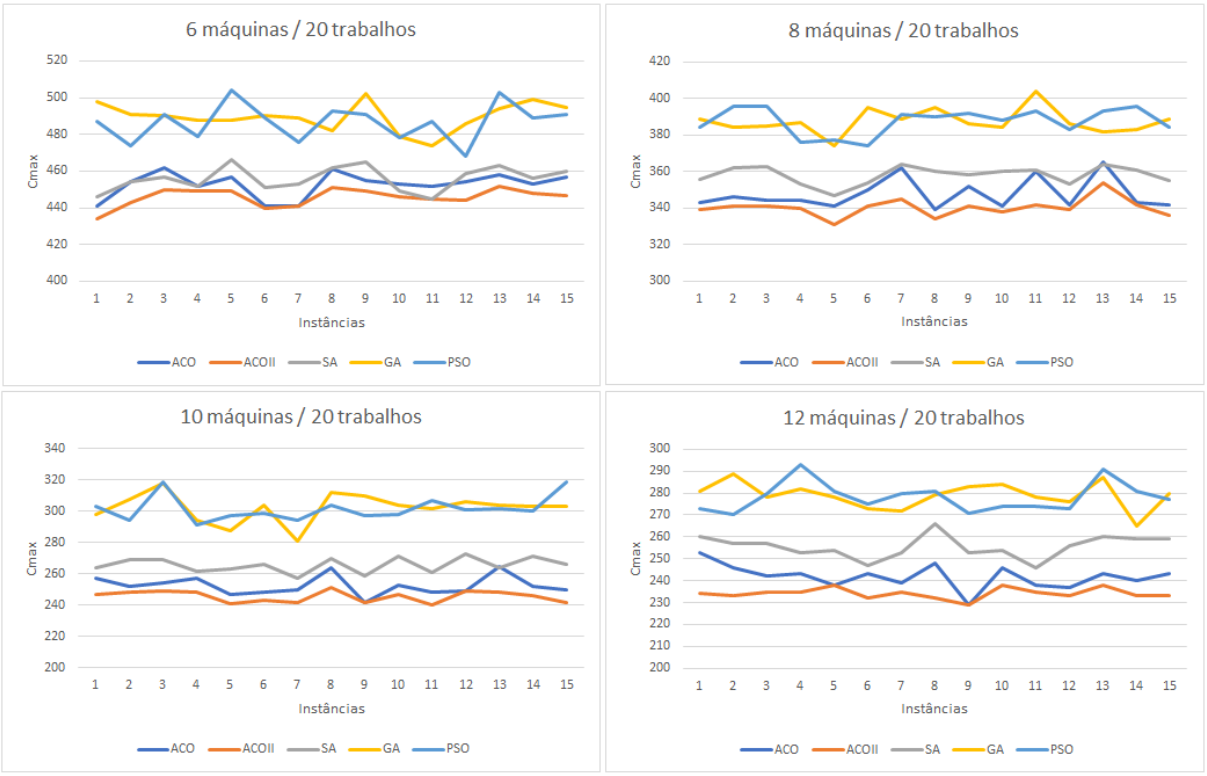

Figura 25 - Comparação de algoritmos para 20 trabalhos

Na Figura 26 estão presentes quatro gráficos ilustrativos do comportamento dos algoritmos ao longo das quinze instâncias. Os gráficos escolhidos são meros exemplos, uma vez que todos os modelos com 40, 60, 80, 100 e 120 trabalhos tinham a mesma tendência, embora

apresentem valores de *makespan* diferentes. O ACO e o ACOII apresentam valores iguais em muitas instâncias, sendo considerados os algoritmos mais eficientes, no entanto existem algumas instâncias em que o segundo revela melhores resultados do que o primeiro. O terceiro algoritmo com melhores resultados é o SA, seguido do PSO implementado nesta dissertação. Por fim, encontra-se o GA, que de todos é o que apresenta resultados mais elevados de *makespan*.

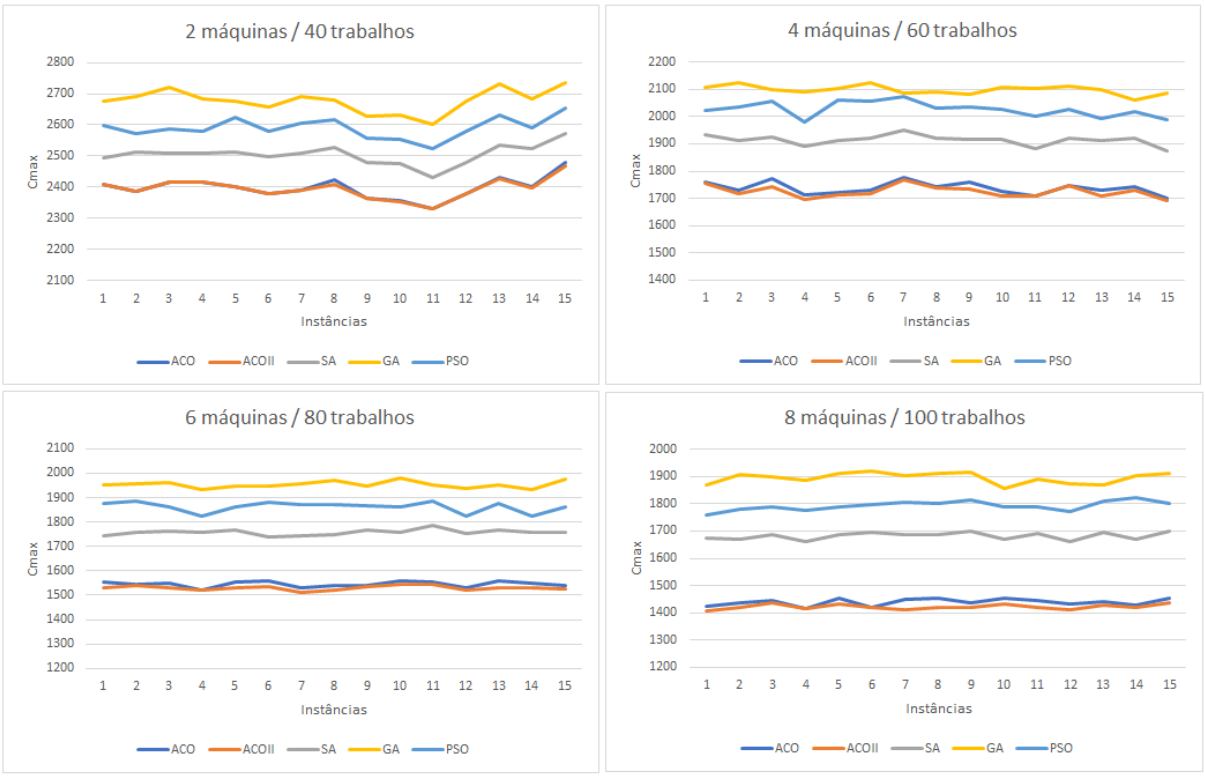

Figura 26 - Comparação de algoritmos, vários exemplos

A variação percentual dos trabalhos pode ser observada na Figura 27. Através desta é percetível a tendência crescente da variação percentual à medida que o número de máquinas aumenta, tanto para 60 trabalhos como para 80, independentemente dos algoritmos em análise. Isto significa que quanto mais máquinas os resultados do PSO tendem a afastar-se dos resultados dos restantes algoritmos, seja para melhor ou pior. Este comportamento é visível em todos os modelos, para 20, 40, 100 e 120 trabalhos, pelo que no Apêndice 6 – Gráficos da variação percentual dos tempos de processamento e *setup* equilibrados estão presentes os gráficos correspondentes a esses casos.

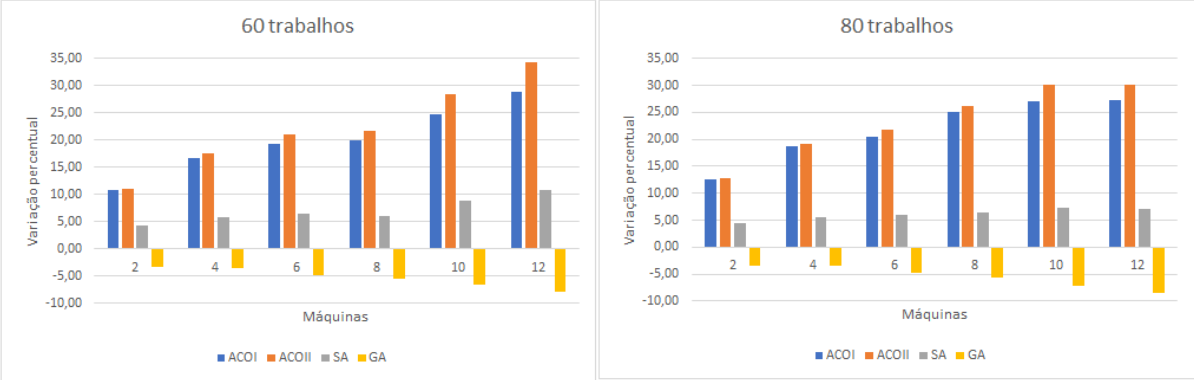

Figura 27 - Variação percentual para 60 e 80 trabalhos

Concluindo a análise aos modelos desta secção, conseguiu-se tanto melhores resultados como piores. Comparando com os algoritmos ACO, ACOII e SA, o PSO obteve valores de *makespan* mais altos, logo teve um desempenho mais baixo, no entanto, comparando com o GA, o desempenho do PSO foi superior, uma vez que conseguiu reduzir, em média, o *makespan* de todos os modelos, com exceção do caso com oito máquinas e 20 trabalhos.

5.3.2 Tempos de processamento dominantes

Nesta secção são analisados os resultados dos modelos em que o tempo de processamento é dominante, ou seja, este tempo foi gerado a partir da distribuição U (125, 175) e o tempo de *setup* a partir da distribuição U (50, 100).

Na Figura 28 estão presentes os dois exemplos da ordem de trabalhos escolhidos para esta secção. No primeiro gráfico estão 40 trabalhos alocados a quatro máquinas, com um C_{max} de 2064. No segundo gráfico o número de máquinas é substancialmente maior, sendo alocados 80 trabalhos em dez máquinas, atingindo um C_{max} de 1810. No Apêndice 10 – Ordem dos trabalhos para tempos de processamento dominantes encontram-se mais quatro exemplos da ordem de trabalhos para os restantes modelos.

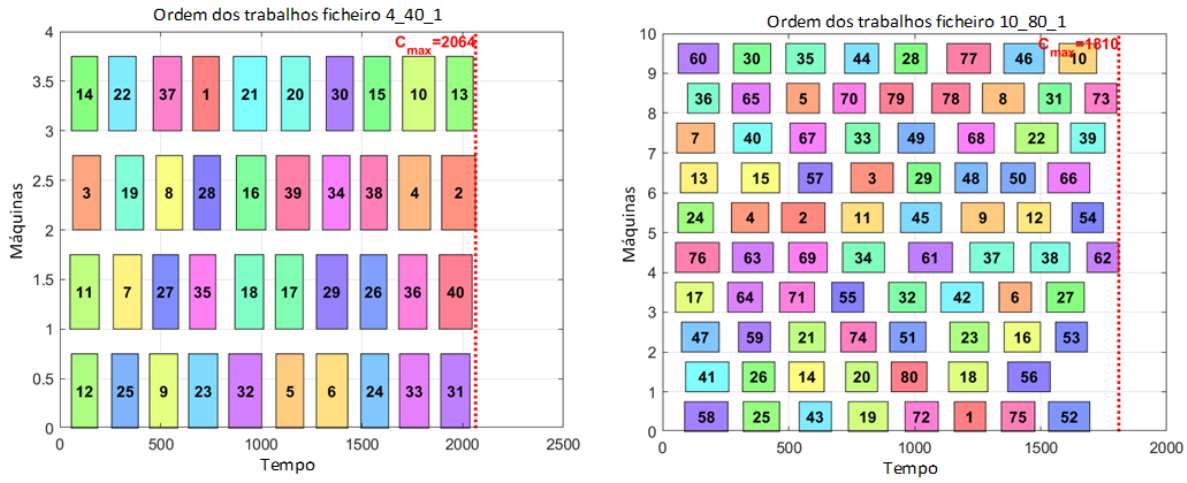

Figura 28 - Ordem de trabalhos com quatro e dez máquinas.

A variação percentual entre o PSO e os restantes algoritmos está presente na Tabela 13. O único algoritmo que o PSO conseguiu superar foi o *Genetic Algorithm*, uma vez que é o único que apresenta variação negativa. O segundo algoritmo que apresenta valores mais próximos do PSO é o SA, seguido do ACO e, por fim, do ACOII.

À medida que é aumentado tanto o número de máquinas como de trabalhos, as variações percentuais entre o PSO e os ACO e ACOII tendem a aumentar, por exemplo para duas máquinas e 20 trabalhos a variação é de aproximadamente 2% e para 120 trabalhos está em 8%, aproximadamente.

Tabela 13 - Variação do PSO para grandes problemas, Pij dominante

Máquinas	Trabalhos	ACOI	ACOII	SA	GA
2	20	1,79	1,87	0,94	-1,40
	40	5,02	5,20	2,40	-1,78
	60	6,86	6,99	2,82	-1,84
	80	7,70	7,81	2,72	-2,08
	100	6,84	7,96	3,25	-2,16
	120	8,19	8,36	3,06	-2,16
4	20	5,31	5,76	3,77	-0,13
	40	8,87	9,41	4,33	-1,60
	60	10,70	10,98	4,13	-1,76
	80	11,73	11,91	3,81	-2,29
	100	12,30	12,59	3,62	-2,40
	120	12,83	13,10	3,46	-2,74
6	20	4,42	4,98	3,98	-0,27
	40	9,70	10,84	4,87	-1,98
	60	11,93	12,48	4,57	-3,23
	80	11,06	12,70	4,15	-3,15
	100	13,54	13,90	3,69	-3,45
	120	14,72	15,09		-3,67
8	20	7,07	7,55	4,93	-0,24
	40	12,22	14,15	5,92	-3,37
	60	11,61	13,16	4,81	-3,24
	80	14,83	16,22	4,71	-5,38
	100	13,66	14,26	4,03	-4,14
	120	17,30	17,67	4,10	-4,88
10	20	10,96	12,46	8,02	-0,18
	40	13,94	14,64	6,46	-7,21
	60	15,41	17,85	6,10	-8,30
	80	18,12	20,37	6,76	-7,28
	100	18,01	18,58	4,73	-6,72
	120	18,85	19,48	4,80	-7,10
12	20	10,26	12,17	6,62	1,06
	40	11,18	12,30	6,13	-2,75
	60	21,81	23,95	11,02	-6,89
	80	16,00	17,70	4,84	-10,14
	100	15,36	17,16	5,14	-9,28
	120	21,38	23,11	6,85	-7,88

Os gráficos com a comparação do *makespan* dos cinco algoritmos para 20 trabalhos e duas e quatro máquinas estão presentes na Figura 29. Para duas máquinas os valores do *makespan* estão mais próximos entre todos os algoritmos, comparando com quatro máquinas. Em todas as instâncias o PSO tem um *makespan* mais baixo do que o GA, existindo até algumas instâncias em que consegue superar o SA (instâncias 1 e 7). No caso de quatro máquinas, a análise não é tão clara, uma vez que, comparando o PSO com o GA, os valores variam ao longo das instâncias entre qual deles é o melhor. No entanto, os valores de ambos são sempre superiores aos restantes três algoritmos.

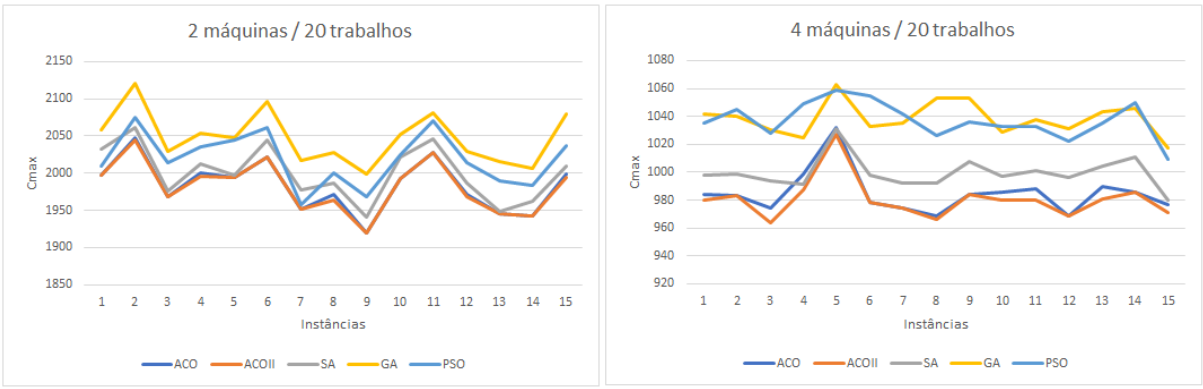

Figura 29 - Comparação de algoritmos para 20 trabalhos e duas e quatro máquinas

Na Figura 30 estão presentes gráficos ilustrativos do comportamento dos algoritmos, para seis e oito máquinas e 60 e 120 trabalhos. Os gráficos escolhidos são apenas exemplos, uma vez que existem modelos com 40, 60, 80, 100 e 120 trabalhos com a mesma tendência, embora apresentem valores de *makespan* diferentes.

Apesar do número diferente de máquinas e trabalhos, os resultados comportam-se de forma semelhante, uma vez que em ambos os algoritmos podem ser ordenados por ordem crescente de *makespan* da seguinte forma ACO, ACOII, SA, PSO e GA.

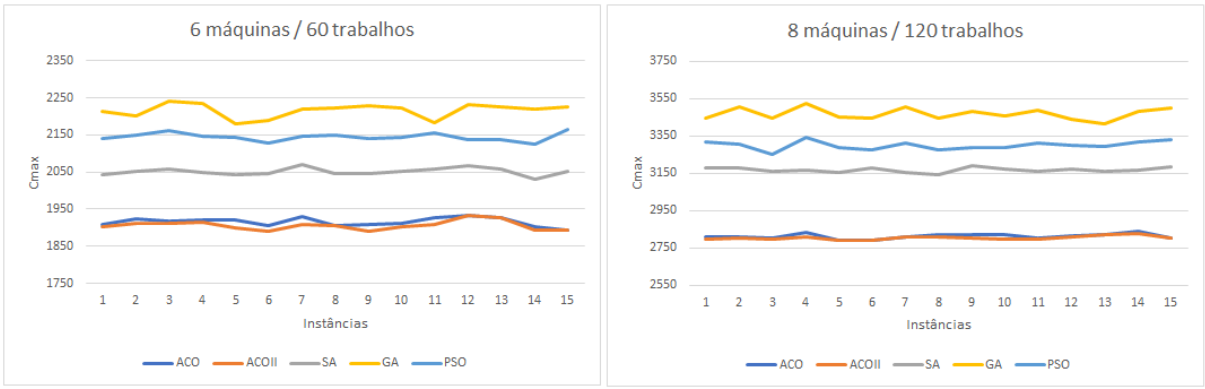

Figura 30 - Comparação de algoritmos para seis e oito máquinas e 60 e 120 trabalhos

A Figura 31 apresenta dois gráficos com a variação percentual entre os resultados do PSO e dos restantes algoritmos, para 100 e 120 trabalhos. Em ambos é percetível a tendência crescente da variação, tanto positiva como negativa, em relação a todos os algoritmos. Isto significa que com o aumento do número de máquinas, para o mesmo número de trabalhos as diferenças entre o PSO e os algoritmos ACO, ACOII, SA e GA tendem a aumentar também. Os gráficos escolhidos são meros exemplos do comportamento da variação para todos os modelos, como é possível verificar no Apêndice 7 – Gráficos da variação percentual dos tempos de processamento dominantes.

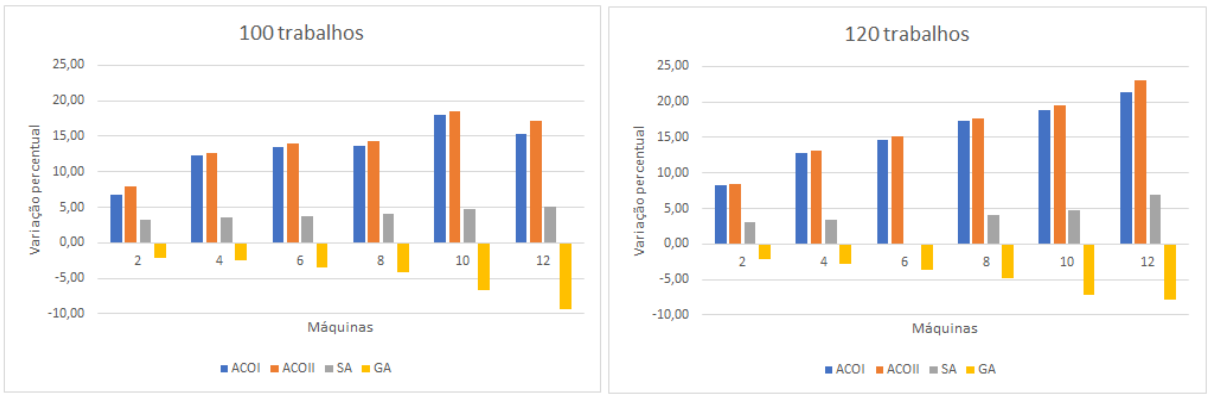

Figura 31 - Variação percentual para 100 e 120 trabalhos.

Concluindo a análise dos resultados do tempos de processamento dominante, pode-se afirmar que o PSO foi mais eficiente se comparado com o GA e menos eficiente se comparado com os restantes três algoritmos. Existe uma tendência para a variação percentual aumentar à medida que se aumentam tanto o número de máquinas como os trabalhos.

5.3.3 Tempos de *setup* dominantes

Os últimos modelos em análise correspondem aos casos em que o tempo de *setup* é dominante, ou seja, quando estes foram gerados a partir da distribuição U(125, 175) e os tempos de processamento a partir de U(50, 100).

Na Figura 32 estão presentes dois exemplos da ordem de trabalhos criada com a aplicação do *Particle Swarm*. O primeiro gráfico mostra a alocação de 60 trabalhos a seis máquinas, totalizando um *makespan* de 2149. O segundo refere-se à alocação de 80 trabalhos a oito máquinas e com um *makespan* de 2179. No Apêndice 11 – Ordem dos trabalhos para tempos de *setup* dominantes estão presentes exemplos dos restantes modelos com diferentes máquinas e trabalhos.

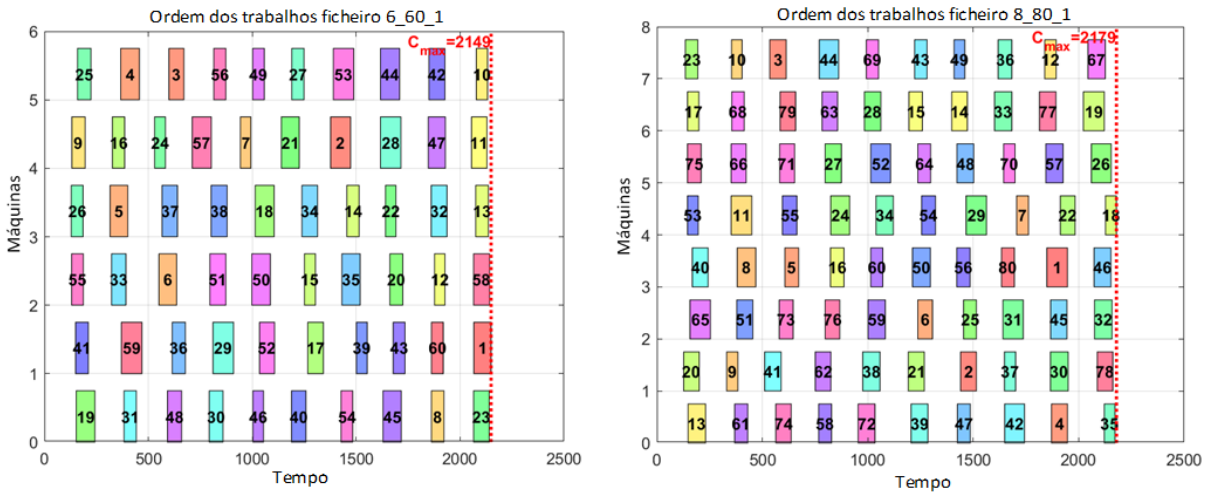

Figura 32 - Ordem de trabalhos para seis e oito máquinas.

Os valores da variação entre os resultados do PSO e os restantes algoritmos estão presentes na Tabela 14. O PSO teve pior desempenho do que três dos quatro algoritmos escolhidos para a comparação, sendo eles o ACO, ACOII e SA, em todos os casos, chegando mesmo a atingir valores superiores em 20%, aproximadamente. No entanto, teve melhor desempenho do que o GA em mais de 90% dos casos, aumentando apenas naqueles em que existem seis, dez e doze máquinas e 20 trabalhos. Porém, este aumento é inferior a 1%.

Tabela 14 - Variação do PSO para grandes problemas, Sij dominante

Máquinas	Trabalhos	ACOI	ACOII	SA	GA
2	20	1,90	2,07	1,07	-1,50
	40	4,92	5,17	2,03	-2,11
	60	6,52	6,69	2,67	-2,14
	80	7,93	8,04	2,94	-2,06
	100	7,74	7,90	2,94	-2,11
	120	8,37	8,49	3,07	-2,23
4	20	5,08	5,84	3,71	-0,39
	40	8,36	8,78	4,08	-1,76
	60	7,92	8,28	4,19	-2,06
	80	11,53	11,76	3,87	-2,64
	100	12,32	12,61	3,57	-2,40
	120	12,97	13,10	3,35	-2,84
6	20	4,55	5,11	4,27	0,06
	40	9,82	10,51	5,00	-1,55
	60	12,19	12,71	4,49	-2,68
	80	11,58	12,72	4,34	-2,90
	100	13,45	13,98	3,51	-3,35
	120	14,53	15,40	3,45	-3,86
8	20	6,69	7,68	5,40	-0,60
	40	11,57	13,17	5,44	-4,06
	60	11,50	13,03	4,95	-2,83
	80	15,23	15,82	4,84	-5,75
	100	13,92	14,54	3,90	-4,43
	120	16,73	17,07	3,78	-5,01
10	20	10,96	12,34	7,78	0,09
	40	13,54	14,29	6,21	-8,70
	60	15,87	18,39	6,41	-8,19
	80	16,95	19,25	5,68	-7,51
	100	18,07	18,93	5,05	-7,16
	120	19,09	19,74	4,94	-6,53
12	20	10,78	12,84	6,40	0,71
	40	11,67	13,49	6,59	-2,36
	60	21,97	24,97	11,57	-5,94
	80	16,52	18,32	5,12	-9,35
	100	15,69	17,59	5,50	-6,81
	120	21,54	23,68	6,66	-8,30

Na Figura 33 são apresentados os gráficos que comparam o *makespan* de todos os algoritmos quando o número de trabalhos é 20 e existem seis e oito máquinas. Verifica-se que em ambos os gráficos se destacam os algoritmos PSO e GA por terem valores acima dos restantes em todas as instâncias. Assim, no primeiro gráfico, os valores destes dois algoritmos balanceiam ao longo das instâncias, sendo numas instâncias melhor o PSO e noutras o GA. No segundo gráfico isto também acontece, mas não de forma tão evidente, uma vez que a maior parte dos resultados são inferiores ao GA. A Figura 33 é apenas um exemplo do comportamento dos algoritmos quando o número de trabalhos é 20, uma vez que para quatro, dez e doze máquinas os valores do PSO e do GA são os mais elevados de entre os algoritmos e são oscilantes ao longo das quinze instâncias. Assim, no Apêndice 5 – Gráficos para 20 trabalhos de tempo de *setup* dominante encontram-se os gráficos que provam o que foi dito anteriormente.

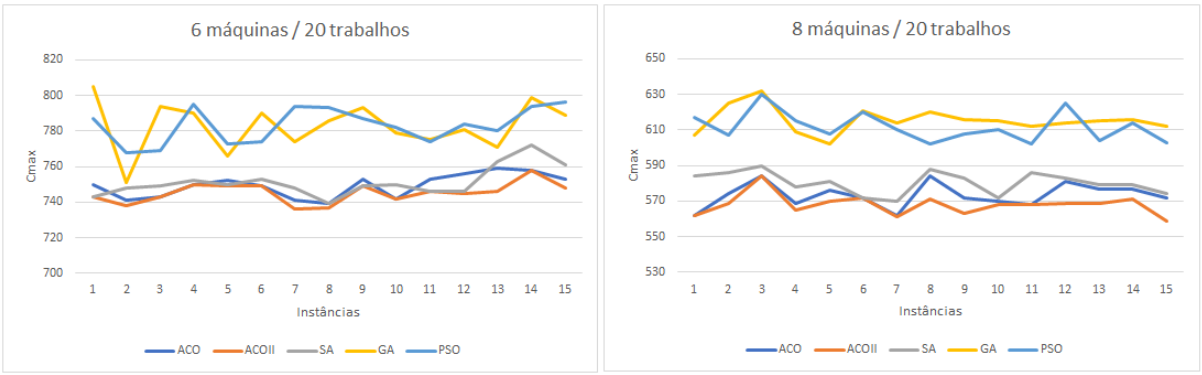

Figura 33 - Comparação de algoritmos para 20 trabalhos e seis e oito máquinas

O gráfico com os resultados do modelo com quatro máquinas e 60 trabalhos é apresentado na Figura 34. É percetível que o *makespan* tem um comportamento estável até à instância 13, na qual há uma grande descida do *makespan* tanto para o PSO como para o GA e o SA, ficando 25% abaixo dos valores do ACO e do ACOII, aproximadamente.

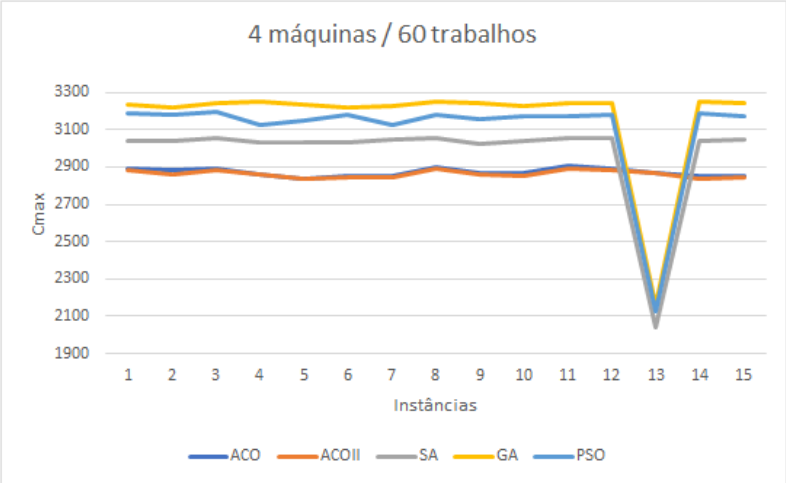

Figura 34 - Comparação de algoritmos para quatro máquinas e 60 trabalhos

Continuando a análise desta secção foram escolhidos dois gráficos em que os resultados se comportam de forma muito semelhante. Na Figura 35 são apresentados os resultados para 80 trabalhos, alocados em duas e em dez máquinas. Apesar do número de máquinas ser bastante díspar, os resultados comportam-se de forma semelhante, uma vez que em ambos os algoritmos podem ser ordenados por ordem crescente de *makespan* da seguinte forma ACO, ACOII, SA, PSO e GA.

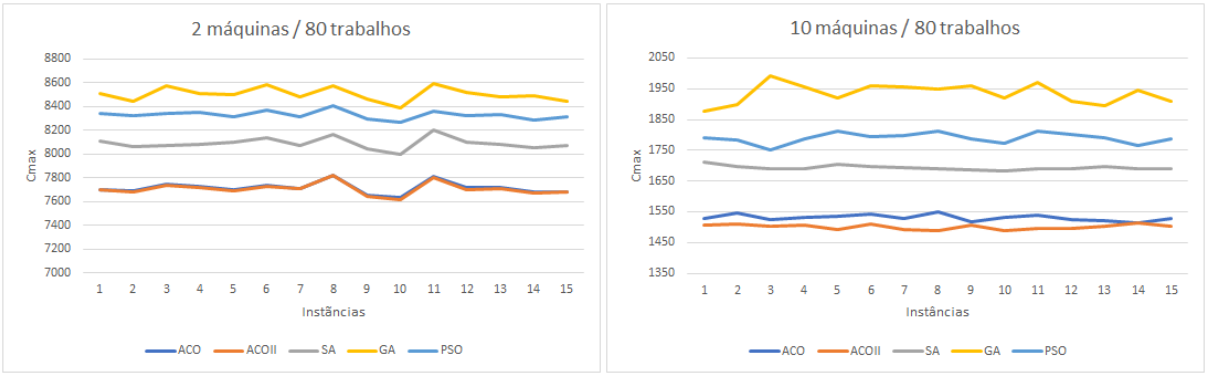

Figura 35 - Comparação de algoritmos para 80 trabalhos com duas e dez máquinas

A Figura 36 apresenta a evolução da variação percentual dos modelos com 20 e 40 trabalhos. Tal como já foi evidenciado em secções anteriores, o comportamento destes exemplos é semelhante a outros exemplos já analisados. Mais uma vez há uma tendência crescente da percentagem de variação com o aumento do número de máquinas em uso. O único caso em que isto não se verifica é na comparação com o GA para 20 trabalhos, pois não tem uma tendência clara, havendo várias oscilações. No Apêndice 8 – Gráficos da variação percentual

dos tempos de *setup* dominantes estão presentes os gráficos para todos os restantes trabalhos, uma vez que estes seguem uma tendência semelhante aos apresentados.

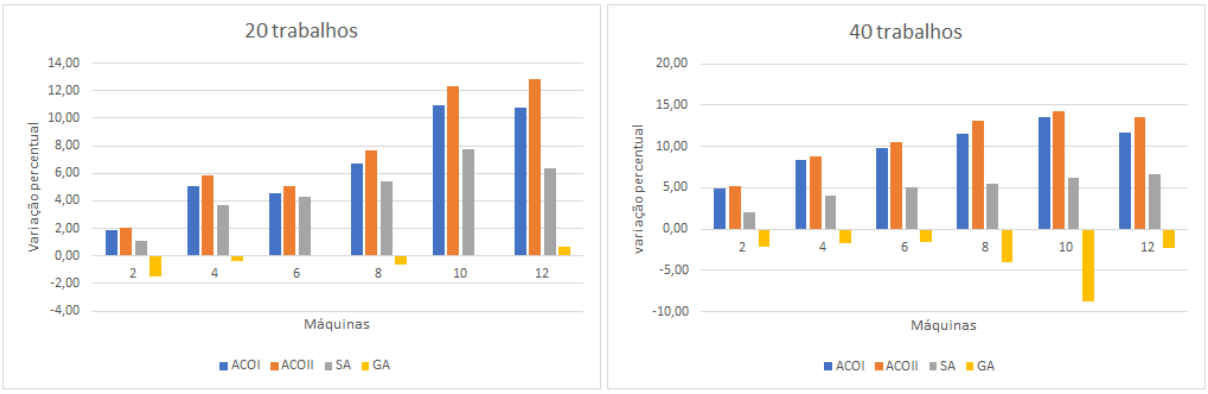

Figura 36 - Evolução da variação percentual para 20 e 40 trabalhos

Em suma, os resultados do PSO para tempos de *setup* dominantes são semelhantes aos outros dois casos. Existe uma clara melhoria do PSO em relação ao GA na maior parte dos modelos, existindo apenas três casos em 36 em que os resultados foram piores. Os restantes três algoritmos apresentam valores mais baixos de *makespan*, podendo ser organizados por ordem crescente da diferença entre os resultados em SA, ACO e ACOII.

5.4 Variação dos algoritmos em relação ao PSO

Nesta secção são comparados os resultados do PSO através da variação percentual entre os vários algoritmos. As figuras Figura *37* e Figura *38* representam a variação do PSO em relação aos diferentes algoritmos para pequenos e grandes problemas, respetivamente.

Para pequenos problemas a variação em todos os algoritmos é bastante inferior em comparação com os grandes problemas. Os maiores desvios são dos algoritmos ACO, ACOII e SA para Pij, Sij equilibrados, ou seja, o PSO apresenta valores superiores em 1,2%, aproximadamente. O GA apresenta valores negativos para dois casos, Pij, Sij equilibrados e Sij dominante, ou seja, o PSO tem resultados inferiores a este algoritmo, em média.

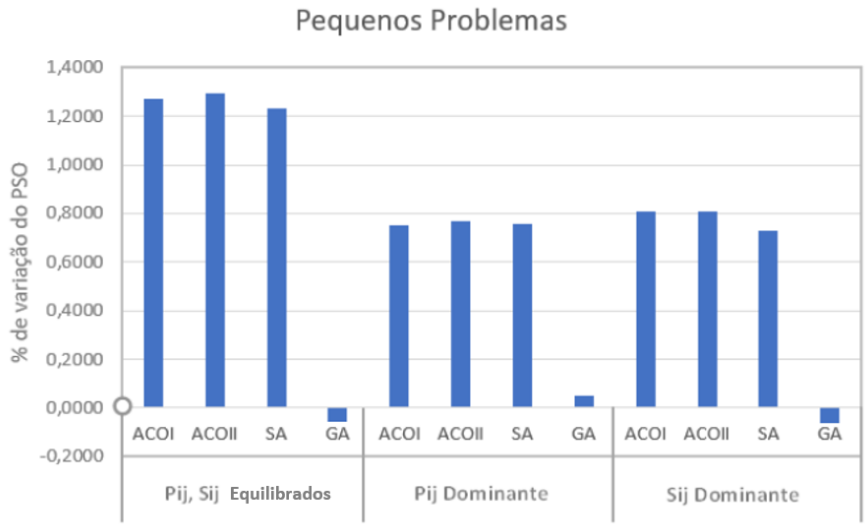

Figura 37 – Variação do PSO para pequenos problemas

Relativamente à variação dos grandes problemas presente na Figura 38, estes apresentam valores mais dispares do que os pequenos problemas. Destacam-se dois algoritmos que apresentam valores muito inferiores ao PSO, que são os ACO e ACOII, sendo que para Pij, Sij equilibrados a variação é a maior. Segue-se o algoritmo SA, que é o terceiro algoritmo com valores melhores do que o PSO. O único algoritmo que tem uma variação negativa é o GA, ou seja, em comparação com este, o PSO apresenta valores de *makespan* inferiores, em média, cerca de 4%.

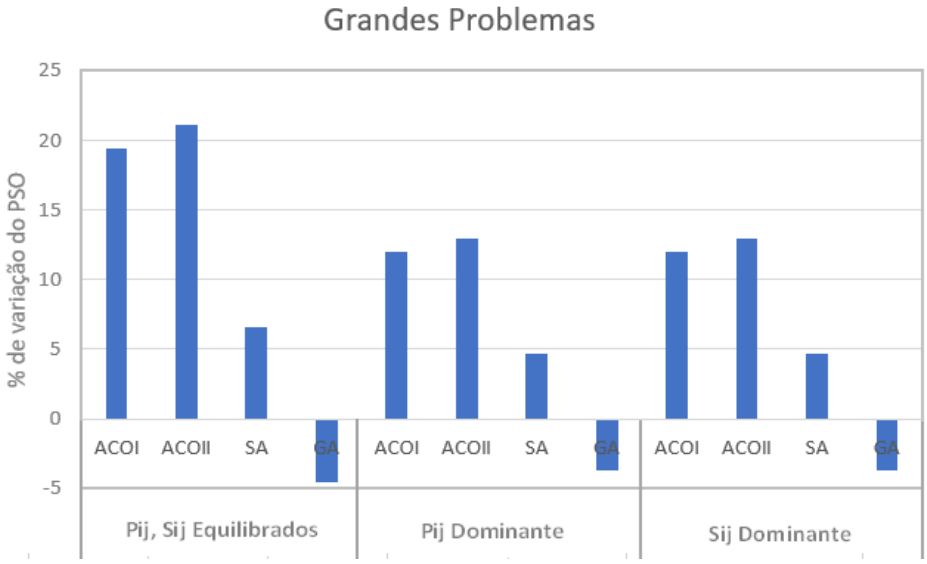

Figura 38 – Variação do PSO para grandes problemas

Conclui-se que tanto para pequenos como grandes problemas, o PSO apenas consegue superar o GA. Há uma grande diferença na percentagem de variação entre os dois problemas, uma vez que para pequenos problemas a percentagem máxima é de 1,2% e para grandes problemas atinge os 21%.

6. CONCLUSÃO

O principal objetivo desta dissertação consistia em estudar o problema de escalonamento de máquinas paralelas não relacionadas com o objetivo de minimiar o *makespan* do conjunto de trabalhos. Mais especificamente, os objetivos do estudo prendiam-se com a implementação do algoritmo escolhido, a análise dos resultados obtidos e a comparação com os resultados de quatro algoritmos presentes na literatura para o mesmo problema, de forma a perceber qual o melhor método para o problema apresentado.

Foi apresentada a revisão da literatura que abordou as principais áreas de intervenção deste trabalho. Começou com a descrição do escalonamento da produção e dos problemas que este apresenta, de forma a perceber a diferença entre o escalonamento de máquinas paralelas e únicas. Dentro das máquinas paralelas foi necessário perceber a diferença entre máquinas idênticas, uniformes e não relacionadas, uma vez que o estudo era precisamente sobre as últimas. Foram também estudados outros problemas como a interrupção de tarefas e o tempo de preparação das máquinas.

O algoritmo escolhido foi o *Particle Swarm Optimization*, depois de uma pesquisa sobre métodos e algoritmos de escalonamento da produção, sendo implementado no software MATLAB. O PSO foi escolhido por ser um algoritmo mais recente, que já teve algumas implementações a problemas semelhantes.

O algoritmo foi implementado utilizando a *toolbox particleswarm* do MATLAB. Um dos parâmetros que foi definido no *solver* consistia no número máximo de iterações que o algoritmo permitia. Inicialmente, este valor foi estipulado em 50 iterações, no entanto, com a análise dos resultados, foi percetível que muitas instâncias necessitavam de mais iterações para encontrar o valor ótimo de *makespan*. Como tal, o valor deste parâmetro foi aumentado para 100 iterações e verificou-se, então, uma descida nos valores do *makespan* dos modelos em análise. Assim, confirma-se que com o aumento do número máximo de iterações o algoritmo foi mais eficiente, no entanto, é de realçar que esta mudança fez com que o tempo de correr os modelos também aumentasse.

A análise dos resultados foi feita separando-os entre pequenos e grandes problemas. Dentro dos pequenos problemas, existem três casos distintos, os modelos com tempos de processamento e *setup* equilibrados, com tempos de processamento dominante e com tempos de *setup* dominante. As conclusões são iguais para estes três modelos, e prendem-se

com o facto de que com o aumento do número de trabalhos, os valores do PSO tendem a afastarem-se dos resultados dos algoritmos ACO, ACOII e SA, sendo que os valores com maior percentagem de diferença correspondem aos modelos com dez e onze trabalhos. Quando tanto o número de máquinas como de trabalhos são menores, os resultados são, na sua maioria, iguais ou melhores do que os restantes algoritmos. O GA é o algoritmo que apresenta valores mais próximos do PSO, seja a diferença positiva ou negativa, sendo que, em média, para os tempos de processamento e *setup* equilibrados e os tempos de *setup* dominantes o PSO apresenta valores de *makespan* mais baixos.

Para os grandes problemas, os modelos também estão divididos segundos os tempos de processamento e de *setup*. É percetível uma melhoria do PSO em relação ao *Genetic Algorithm* na maior parte dos modelos, existindo apenas três casos em 36 em que os resultados foram piores. Os restantes três algoritmos apresentam valores mais baixos de *makespan*, podendo ser organizados por ordem crescente da diferença entre os resultados em SA, ACO e ACOII. Existe também uma tendência para a variação percentual aumentar à medida que se aumentam tanto o número de máquinas como os trabalhos.

Como trabalho futuro sugere-se uma análise mais extensa do desempenho deste algoritmo neste tipo de problemas, podendo envolver outras medidas de desempenho, como os atrasos ou o tempo de percurso médio. Outra sugestão que poderá enriquecer o trabalho desenvolvido é testar a alteração de um parâmetro importante da *toolbox particleswarm*, com o aumento do número máximo de iterações que cada modelo pode correr. Esta alteração melhorou os resultados apresentados, pelo que este aumento poderá melhorar ainda mais os resultados. No entanto, deve-se estar atento ao acréscimo no tempo que demora a correr os modelos, uma vez que é de esperar que este aumente também.

REFERÊNCIAS BIBLIOGRÁFICAS

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, *246*(2), 345–378. https://doi.org/10.1016/j.ejor.2015.04.004

Amaral, G., Costa, L., Rocha, A. M. A. C., Varela, L., & Madureira, A. (2020). Application of the Simulated Annealing Algorithm to Minimize the makespan on the Unrelated Parallel Machine Scheduling Problem with Setup Times. *Advances in Intelligent Systems and Computing*, *923*, 398–407. https://doi.org/10.1007/978-3-030-14347-3_39

Artiba, A., & Elmaghraby, S. E. (1996). The Planning and Scheduling of Production Systems:
Methodologies and Applications: Springer. https://doi.org/10.1007/978-1-4613-1195-9

Baker, K. R., & Trietsch, D. (2009). Principles of Sequencing and Scheduling. In *Principles of Sequencing and Scheduling*. https://doi.org/10.1002/9780470451793

Behnamian, J. (2014). Particle swarm optimization-based algorithm for fuzzy parallel machine scheduling. *International Journal of Advanced Manufacturing Technology*, *75*(5–8), 883–895. https://doi.org/10.1007/s00170-014-6181-0

Benmansour, R., Allaoui, H., Artiba, A., & Hanafi, S. (2014). Minimizing the weighted sum of maximum earliness and maximum tardiness costs on a single machine with periodic preventive maintenance. *Computers and Operations Research*, *47*, 106–113. https://doi.org/10.1016/j.cor.2014.02.004

Blazewicz, J., Domscke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research 93 (1996) 1-33.

Blazewicz, J., Machowiak, M., Weglarz, J., Kovalyov, M. Y., & Trystram, D. (2004). Scheduling malleable tasks on parallel processors to minimize the makespan. *Annals of Operations Research*, 129(1-4), 65.

Brucker, P. (1995). Due-Date Scheduling. In *Scheduling Algorithms* (pp. 225-247). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-04550-3_7

Carmo-Silva, S. (2015). *GESTÃO DA PRODUÇÃO - Rev 04-2015*. Universidade do Minho, publicação interna.

Conway, R. W., Miller, L. W. & Maxwell, W. L. (2003). *Theory of scheduling*. Dover.

Damodaran, P., Diyadawagamage, D. A., Ghrayeb, O., & Vélez-Gallego, M. C. (2012). A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines. *International Journal of Advanced Manufacturing Technology*, *58*(9–12), 1131–1140. https://doi.org/10.1007/s00170-011-3442-z

French, S. (1982). *Sequencing and Scheduling An Introduction to the Mathematics of the Job-Shop*. John Wiley and Sons, Inc.

Gacias, B., Artigues, C., & Lopez, P. (2010). Parallel machine scheduling with precedence constraints and setup times. *Computers and Operations Research*, *37*(12), 2141–2151. https://doi.org/10.1016/j.cor.2010.03.003

Gendreau, M., Laporte, G., & Guimarães, E. M. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, *133*(1), 183–189. https://doi.org/10.1016/S0377-2217(00)00197-1

Ghirardi, M., & Potts, C. N. (2005). Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. *European Journal of Operational*

Research, *165*(2), 457–467. https://doi.org/10.1016/j.ejor.2004.04.015

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* (Vol. 5, pp. 287-326). Elsevier.

Guinet, A. (1991). Textile production systems: a succession of non-identical parallel processor shops. *Journal of the Operational Research Society*, 42(8), 655-671.

Hassan, M. A., Kacem, I., Martin, S., & Osman, I. M. (2016). Unrelated parallel machine scheduling problem with precedence constraints: Polyhedral analysis and branch-and-cut. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9849 LNCS*(May), 308–319. https://doi.org/10.1007/978-3-319-45587-7_27

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers and Operations Research*, *36*(2), 358–378. https://doi.org/10.1016/j.cor.2007.10.004

Kashan, A. H., & Karimi, B. (2009). A discrete particle swarm optimization algorithm for scheduling parallel machines. *Computers and Industrial Engineering*, *56*(1), 216–223. https://doi.org/10.1016/j.cie.2008.05.007

Kennedy, J., & Eberhart, R. (1995a). New optimizer using particle swarm theory. *Proceedings of the International Symposium on Micro Machine and Human Science*, 39–43. https://doi.org/10.1109/mhs.1995.494215

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks* (Vol. 4, pp. 1942-1948). IEEE.

Kim, D., Kim, K., Jang, W., & Frank Chen, F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, *18*(3–4), 223–231. https://doi.org/10.1016/S0736-5845(02)00013-3

Kim, D. W., Na, D. G., & Chen, F. F. (2003). Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer-Integrated Manufacturing*, *19*(1–2), 173–181. https://doi.org/10.1016/S0736-5845(02)00077-7

Lee, Y. H., & Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, *100*(3), 464–474. https://doi.org/10.1016/S0377-2217(95)00376-2

Lian, Z., Gu, X., & Jiao, B. (2008). A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. *Chaos, Solitons and Fractals*, *35*(5), 851–861. https://doi.org/10.1016/j.chaos.2006.05.082

Liao, C. J., Gen, M., Tiwari, M. K., & Chang, P. C. (2013). Meta-heuristics for manufacturing scheduling and logistics problems. *International Journal of Production Economics*, *141*(1), 1–3. https://doi.org/10.1016/j.ijpe.2012.09.004

Lin, S. W., Lu, C. C., & Ying, K. C. (2011). Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints. *International Journal of Advanced Manufacturing Technology*, *53*(1–4), 353–361. https://doi.org/10.1007/s00170-010-2824-y

Lin, Y. K. (2013). Particle swarm optimization algorithm for unrelated parallel machine scheduling with release dates. *Mathematical Problems in Engineering*, *2013*. https://doi.org/10.1155/2013/409486

Madureira, A., & Pereira, I. (2010). Self-optimization for dynamic scheduling in manufacturing systems. In *Technological Developments in Networking, Education and Automation* (pp.

421-426). Springer, Dordrecht.

Madureira, A., Pereira, I., & Sousa, N. (2011). Self-organization for scheduling in agile manufacturing. *Proceedings of 2011, 10th IEEE International Conference on Cybernetic Intelligent Systems, CIS 2011*, (May 2015), 38–43. https://doi.org/10.1109/CIS.2011.6169132

Madureira, A., Ramos, C., & Do Carmo Silva, S. D. (2002, May). A coordination mechanism for real world scheduling problems using genetic algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02* (Cat. No. 02TH8600) (Vol. 1, pp. 175-180). IEEE. https://doi.org/10.1109/CEC.2002.1006229

Martí, R., & Reinelt, G. (2011). *The linear ordering problem: exact and heuristic methods in combinatorial optimization* (Vol. 175). Springer Science & Business Media.

McNaughton, R. (1959). Scheduling with Deadlines and Loss Functions. *Management Science*, *6*(1), 1–12. https://doi.org/10.1287/mnsc.6.1.1

Morton, T. E., & Pentico, D. W. (1993). Heuristic Scheduling Systems. *Proceedings of the IEEE*, Vol. 84, p. 324. https://doi.org/10.1109/jproc.1996.482234

Pfund, M., Fowler, J. W., & Gupta, J. N. D. (2004). A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. *Journal of the Chinese Institute of Industrial Engineers*, *21*(3), 230–241. https://doi.org/10.1080/10170660409509404

Pinedo, M. L. (2016). Scheduling: Theory, algorithms, and systems Fifth ed. Cham: *Springer International Publishing*.

Rabadi, G., Arnaout, J. P., & Musa, R. (2010). A two-stage Ant Colony Optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *21*(6), 693–701. https://doi.org/10.1007/s10845-009-0246-1

Rabadi, G., Arnaout, J. P., & Musa, R. (2014). A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines - Part II: Enhancements and experimentations. *Journal of Intelligent Manufacturing*, *25*(1), 43–53. https://doi.org/10.1007/s10845-012-0672-3

Rabadi, G., Moraga, R. J., & Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, *17*(1), 85–97. https://doi.org/10.1007/s10845-005-5514-0

Rahmani, K., Mahdavi, I., Moradi, H., Khorshidian, H., & Solimanpur, M. (2011). A nondominated ranked genetic algorithm for bi-objective single machine preemptive scheduling in just-in-time environment. The International Journal of Advanced Manufacturing Technology, 55(9-12), 1135-1147. https://doi.org/10.1007/s00170-010-3126-0

Salehi Mir, M. S., & Rezaeian, J. (2016). A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines. *Applied Soft Computing Journal*, *41*, 488–504. https://doi.org/10.1016/j.asoc.2015.12.035

Schuh, G., Reuter, C., Prote, J. P., Brambring, F., & Ays, J. (2017). Increasing data integrity for improving decision making in production planning and control. *CIRP Annals - Manufacturing Technology*, *66*(1), 425–428. https://doi.org/10.1016/j.cirp.2017.04.003

Singh, A., Saxena, A., & Soni, B. (2015). Comparison of Population Based Intelligent Techniques to Solve Load Dispatch Problem. *European Journal of Advances in Engineering and Technology*, *2015*(3), 84–89.

Su, L., Cheng, T. C. E., & Chou, F. (2013). A minimum-cost network flow approach to preemptive parallel-machine scheduling. *Computers & Industrial Engineering*, *64*(1), 453–458. https://doi.org/10.1016/j.cie.2012.04.020

Tan, Z., Chen, Y., & Zhang, A. (2011). Parallel machines scheduling with machine maintenance for minsum criteria. *European Journal of Operational Research*, *212*(2), 287–292. https://doi.org/10.1016/j.ejor.2011.02.006

Tang, L., & Wang, X. (2009). Simultaneously scheduling multiple turns for steel color-coating production. *European Journal of Operational Research*, *198*(3), 715–725. https://doi.org/10.1016/j.ejor.2008.09.025

Torabi, S. A., Sahebjamnia, N., Mansouri, S. A., & Bajestani, M. A. (2013). A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. *Applied Soft Computing Journal*, *13*(12), 4750–4762. https://doi.org/10.1016/j.asoc.2013.07.029

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, *211*(3), 612–622. https://doi.org/10.1016/j.ejor.2011.01.011

Varela, Ma. L. R. (2007). *Uma Contribuição para o Escalonamento da Produção baseado em Métodos Globalmente Distribuídos*. Tese de doutoramento (270 pgs.), Universidade do Minho

Wan, G., & Yen, B. P. C. (2009). Single machine scheduling to minimize total weighted earliness subject to minimal number of tardy jobs. *European Journal of Operational Research*, *195*(1), 89–97. https://doi.org/10.1016/j.ejor.2008.01.029

Woo, Y. Bin, Jung, S., & Kim, B. S. (2017). A rule-based genetic algorithm with an improvement heuristic for unrelated parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. *Computers and Industrial Engineering*, *109*, 179–190. https://doi.org/10.1016/j.cie.2017.05.007

Xhafa, F., & Abraham, A. (2008). *Metaheuristics for Scheduling in Industrial and Manufacturing Applications Studies in Computational Intelligence , Volume 128*.

Yazdani, M., Khalili, S. M., Babagolzadeh, M., & Jolai, F. (2017). A single-machine scheduling problem with multiple unavailability constraints: A mathematical model and an enhanced variable neighborhood search approach. *Journal of Computational Design and Engineering*, *4*(1), 46–59. https://doi.org/10.1016/j.jcde.2016.08.001

Zeidi, J. R., & MohammadHosseini, S. (2015). Scheduling unrelated parallel machines with sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology*, *81*(9–12), 1487–1496. https://doi.org/10.1007/s00170-015-7215-y

Zheng, X. L., & Wang, L. (2018). A Collaborative Multiobjective Fruit Fly Optimization Algorithm for the Resource Constrained Unrelated Parallel Machine Green Scheduling Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*(5), 790–800. https://doi.org/10.1109/TSMC.2016.2616347

APÊNDICE 1 – MODELO MATLAB

```
function [z] = MyCost(q,model)

    [~,I]=sort(q);
    q=mat2cell(I,1);
    sol = ParseSolution(q,model);
    z = sol.Cmax;

end
```

Figura 39 - Função MyCost

```
function PlotSolution (x, model)

I = model.I;
J = model.J;

sol = ParseSolution(x, model);
L = sol.L;
ST = sol.ST;
FT = sol.FT;

H=1;
h=0.75;

Colors = hsv(I);
for j=1:J
    y1=(j-1)*H;
    y2=y1+h;
    for i=L{j}

        x1=ST(i);
        x2=FT(i);

        X=[x1 x2 x2 x1];
        Y=[y1 y1 y2 y2];

        C=Colors(i,:);
        C=(C+[1 1 1])/2;

        fill(X,Y,C);
        hold on;

        xm=(x1+x2)/2;
        ym=(y1+y2)/2;

      text(xm, ym, num2str(i),...
          'FontWeight', 'bold',...
          'HorizontalAlignment', 'center',...
          'VerticalAlignment', 'middle');
    end
end

Cmax = sol.Cmax;
plot([Cmax Cmax], [0 J*H], 'r:', 'LineWidth',2);
text(Cmax, J*H, ['C_{max}=' num2str(Cmax)],...
    'FontWeight','bold',...
    'HorizontalAlignment', 'right',...
    'VerticalAlignment', 'top',...
    'Color', 'red');
grid on;
hold off;
end
```

Figura 40 - Função PlotSolution

```
function sol = ParseSolution (q, model)

I = model.I;
J = model.J;
p = model.p;
s = model.s;
q=q{1};

%determine the delimiter position
DelPos = find(q>I);
%determine start and end of machines job sequence
From = [0 DelPos]+1;
To = [DelPos I+J]-1;

% create jobs list
L = cell(J,1);  %create cell array with J rows and 1 column
for j=1:J
    L{j}=q(From(j): To(j));
end
% starting time
ST = zeros(I,1);
% Finishing time
FT = zeros(I,1);
%Processing time
PT = zeros(I,1);
%mct time
MCT = zeros(J,1);
for j=1:J
    for i=L{j}
        %find position of the job on this machine
        k = find(L{j}==i);
        %find the index of the job that has the first position on L
        if k==1
           %ST(i)=0;
          ST(i) = s(i,i,j);
        else
            PreviousJob = L{j}(k-1);
            ST(i) = FT(PreviousJob)+ s(PreviousJob, i, j);
        end

        PT(i) = p(i,j);
        FT(i) = ST(i) + PT(i);
    end
    %if Lj is not empty
    if ~isempty(L{j})
        MCT(j) = FT(L{j}(end));
    end
end

Cmax = max(MCT);

sol.L = L;
sol.ST = ST;
sol.PT = PT;
sol.FT = FT;
sol.MCT = MCT;
sol.Cmax = Cmax;

end
```

Figura 41 - Função ParseSolution

```
nome_modelo='2_6_';
numruns = 20;
for run=1:numruns
    for op=1:15
        model=eval(['CreateModel_' nome_modelo int2str(op) '()']);

   CostFunction=@(q) MyCost(q,model);
   nVar=model.nVar;
   folha=[nome_modelo,'_',num2str(op)];
   A = {'run','time','feval','Cmax','work_order'};
   xlswrite(['solutions/sol_' nome_modelo '.xls'],A,folha,'A1');

%% PSO parameters
options = optimoptions(@particleswarm,'SwarmSize', 200);
options = optimoptions(options, ...
                    'MaxIterations', 100, ...
                    'PlotFcn', @pswplotbestf);

%% PSO
LB=zeros(nVar,1)';
UB=ones(nVar,1)';

tstart=tic;
[x,fval,exitflag,output] = particleswarm(CostFunction,nVar,LB,UB,options);

    [~,I]=sort(x);
    x=mat2cell(I,1);
telapsed = toc(tstart);

figure
PlotSolution(x,model)
BestCost=fval;
BestSol.Position=cell2mat(x);

 %write header of results file
       xx=num2str(cell2mat(x));
     A = {run,telapsed,output.funccount,fval,xx};
        xlswrite(['solutions/sol_' nome_modelo '.xls'],A,folha,['A',num2str(run+1)])

%Plot Solution work order
    %PlotSolution(x,model);
    savefig(['solutions/worder_',folha,'_run_',num2str(run)]);
    close(gcf);

 %Plot Cost Solution along iterations
    %plot(BestCost);
    xlabel('Iteration');
    ylabel('Best Cost');
    grid on;
    savefig(['solutions/cost_it_',folha,'_run_',num2str(run)]);
    close(gcf);
    end
end
```

Figura 42 – *Script* para correr cada modelo

APÊNDICE 2 – RESULTADOS DOS ALGORITMOS PARA GRANDES PROBLEMAS E TEMPOS DE PROCESSAMENTO E SETUP EQUILIBRADOS

Tabela 15 - Média do *makespan* para grandes problemas, Pij, Sij equilibrados

Máquinas	Trabalhos	ACOI	ACOII	SA	GA	PSO
2	20	1237,80	1235,27	1253,13	1301,47	1277,67
	40	2397,80	2394,93	2504,87	2677,20	2589,53
	60	3574,60	3565,13	3799,47	4097,13	3959,93
	80	4730,40	4722,87	5100,87	5520,20	5328,27
	100	5897,60	5881,93	6428,73	6943,60	6696,33
	120	7082,60	7072,67	7752,73	8417,67	8122,13
4	20	617,13	609,47	630,13	671,00	663,67
	40	1179,87	1166,93	1258,80	1370,53	1336,33
	60	1737,93	1725,67	1914,07	2098,07	2027,00
	80	2298,53	2288,93	2584,00	2828,07	2728,53
	100	2849,93	2837,80	3250,67	3560,27	3407,47
	120	3405,13	3389,87	3927,00	4295,87	4134,93
6	20	452,73	445,87	455,87	489,67	486,67
	40	805,40	791,93	869,93	957,47	926,00
	60	1163,47	1147,80	1304,13	1457,60	1388,73
	80	1545,33	1530,47	1757,53	1954,40	1862,20
	100	1897,47	1882,47	2215,20	2442,47	2326,53
	120	2253,93	2234,20	2673,60	2936,53	2800,13
8	20	347,60	340,27	358,07	387,47	387,53
	40	599,27	580,73	659,53	751,67	715,93
	60	893,80	880,47	1009,93	1132,00	1071,07
	80	1142,40	1131,13	1343,00	1512,40	1427,87
	100	1439,07	1422,00	1682,67	1895,33	1792,80
	120	1686,07	1670,33	2038,07	2273,87	2139,60
10	20	252,53	245,53	265,67	302,33	301,67
	40	485,53	476,13	536,60	635,93	593,47
	60	708,27	688,67	812,07	944,60	883,93
	80	925,87	903,13	1095,13	1265,40	1175,40
	100	1141,53	1126,47	1378,00	1585,13	1468,67
	120	1351,67	1336,33	1655,80	1903,60	1756,40
12	20	241,87	234,20	255,60	279,00	278,27
	40	448,13	436,93	484,47	559,00	534,80
	60	597,33	573,47	694,73	835,47	769,73
	80	790,07	773,13	938,80	1098,53	1005,87
	100	988,67	968,73	1170,33	1398,13	1260,20
	120	1138,73	1113,40	1405,27	1662,47	1499,47

APÊNDICE 3 – RESULTADOS DOS ALGORITMOS PARA GRANDES PROBLEMAS E COM TEMPOS DE PROCESSAMENTO DOMINANTES

Tabela 16 - Média do *makespan* para grandes problemas, Pij dominante

Máquinas	Trabalhos	ACOI	ACOII	SA	GA	PSO
2	20	1983,40	1982,00	2000,27	2047,60	2019,00
	40	3902,87	3895,93	4002,67	4173,07	4098,67
	60	5829,73	5822,27	6058,33	6346,13	6229,40
	80	7741,13	7733,07	8116,40	8514,27	8336,93
	100	9822,13	9720,13	10163,00	10724,80	10493,60
	120	11681,27	11662,60	12262,93	12916,73	12637,60
4	20	984,87	980,67	999,47	1038,53	1037,13
	40	1932,93	1923,47	2017,07	2138,60	2104,40
	60	2859,80	2852,60	3040,33	3222,53	3165,80
	80	3796,80	3790,73	4086,47	4341,60	4242,27
	100	4736,93	4724,67	5133,73	5450,60	5319,73
	120	5666,93	5653,33	6180,27	6574,00	6394,20
6	20	749,53	745,53	752,73	784,80	782,67
	40	1324,20	1310,53	1385,13	1481,93	1452,60
	60	1916,07	1906,73	2051,00	2216,27	2144,73
	80	2596,93	2559,20	2769,33	2978,13	2884,20
	100	3164,60	3154,40	3465,20	3721,27	3593,00
	120	3756,60	3744,67	2676,60	4474,07	4309,73
8	20	571,27	568,73	582,93	613,13	611,67
	40	973,40	956,93	1031,27	1130,40	1092,33
	60	1497,00	1476,40	1594,00	1726,73	1670,73
	80	1906,47	1883,67	2090,60	2313,53	2189,13
	100	2414,67	2401,87	2638,13	2863,07	2744,40
	120	2812,07	2803,27	3168,60	3467,67	3298,53
10	20	405,60	400,20	416,67	450,87	450,07
	40	782,53	777,73	837,53	960,87	891,60
	60	1160,13	1136,07	1261,93	1460,00	1338,87
	80	1526,93	1498,33	1689,47	1945,13	1803,60
	100	1888,47	1879,40	2128,00	2389,13	2228,67
	120	2259,67	2247,87	2562,60	2890,87	2685,67
12	20	391,33	384,67	404,67	426,93	431,47
	40	749,80	742,27	785,47	857,13	833,60
	60	971,33	954,60	1065,73	1270,73	1183,20
	80	1318,33	1299,33	1458,60	1701,80	1529,27
	100	1667,87	1642,33	1830,00	2120,93	1924,13
	120	1895,67	1868,93	2153,40	2497,73	2300,87

APÊNDICE 4 – RESULTADOS DOS ALGORITMOS PARA GRANDES PROBLEMAS E COM TEMPOS DE *SETUP* DOMINANTES

Tabela 17 - Média do *makespan* para grandes problemas, Sij dominante

Máquinas	Trabalhos	ACOI	ACOII	SA	GA	PSO
2	20	1986,20	1982,73	2002,53	2054,73	2023,87
	40	3897,67	3888,47	4008,13	4177,73	4089,53
	60	5823,80	5814,67	6042,00	6338,93	6203,53
	80	7715,40	7707,47	8089,80	8502,80	8327,40
	100	9725,93	9712,13	10180,40	10705,27	10479,20
	120	11661,13	11647,73	12260,87	12925,93	12637,13
4	20	991,33	984,20	1004,40	1045,73	1041,67
	40	1928,73	1921,33	2008,00	2127,40	2090,00
	60	2872,00	2862,40	2974,80	3164,47	3099,40
	80	3800,80	3792,87	4081,07	4353,87	4238,87
	100	4730,07	4717,93	5129,47	5443,47	5312,80
	120	5652,93	5646,20	6179,27	6572,87	6386,00
6	20	749,27	745,27	751,27	782,87	783,33
	40	1327,73	1319,40	1388,60	1481,00	1458,07
	60	1913,33	1904,47	2054,27	2205,73	2146,60
	80	2590,80	2564,73	2770,67	2977,20	2890,93
	100	3169,40	3154,53	3473,47	3720,20	3595,53
	120	3769,47	3741,07	4173,47	4490,67	4317,33
8	20	573,33	568,07	580,33	615,33	611,67
	40	974,67	960,87	1031,33	1133,40	1087,40
	60	1499,33	1479,00	1592,93	1720,40	1671,73
	80	1898,13	1888,53	2086,33	2320,67	2187,27
	100	2414,87	2401,87	2647,67	2878,47	2751,00
	120	2815,13	2807,00	3166,40	3459,60	3286,20
10	20	405,20	400,20	417,13	449,20	449,60
	40	783,33	778,20	837,40	974,13	889,40
	60	1157,13	1132,47	1260,00	1460,33	1340,73
	80	1530,20	1500,67	1693,47	1934,93	1789,60
	100	1890,53	1876,87	2124,73	2404,27	2232,13
	120	2256,80	2244,47	2561,07	2875,40	2687,60
12	20	390,27	383,13	406,33	429,27	432,33
	40	749,60	737,60	785,33	857,33	837,07
	60	973,47	950,07	1064,20	1262,33	1187,33
	80	1317,00	1297,00	1459,87	1692,87	1534,60
	100	1663,00	1636,20	1823,67	2064,53	1924,00
	120	1893,53	1860,73	2157,67	2509,53	2301,33

APÊNDICE 5 – GRÁFICOS PARA 20 TRABALHOS DE TEMPO DE *SETUP* DOMINANTE

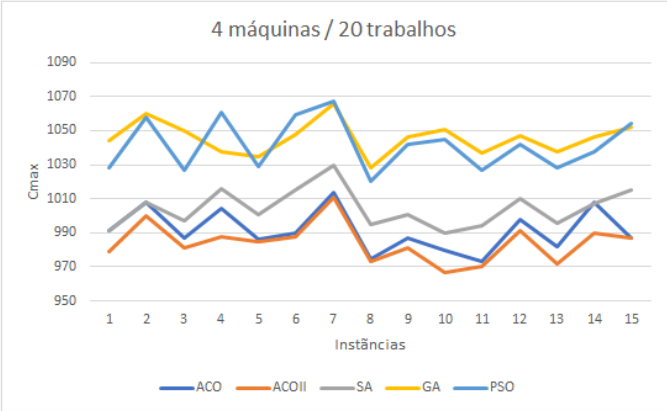

Figura 43 – Gráfico para quatro máquinas e 20 trabalhos

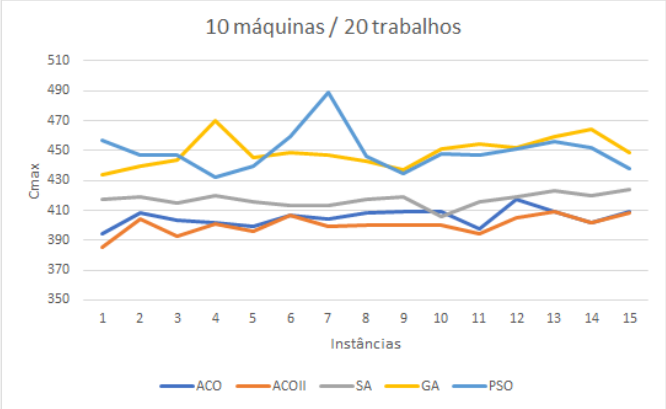

Figura 44 – Gráfico para dez máquinas e 20 trabalhos

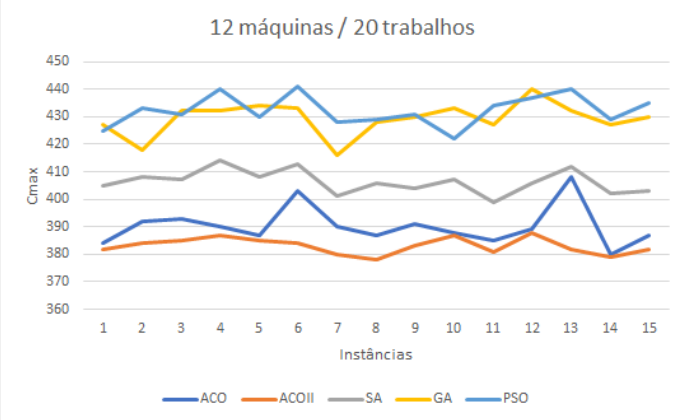

Figura 45 – Gráfico para doze máquinas e 20 trabalhos

APÊNDICE 6 – GRÁFICOS DA VARIAÇÃO PERCENTUAL DOS TEMPOS DE PROCESSAMENTO E *SETUP* EQUILIBRADOS

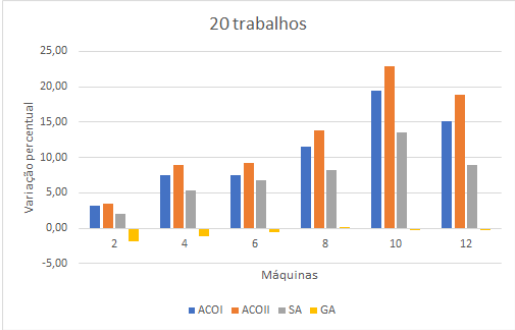

Figura 46 – Variação percentual para 20 trabalhos

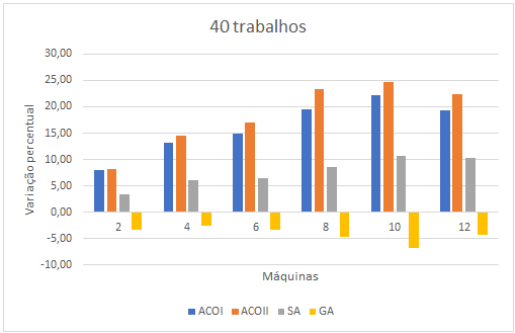

Figura 47 - Variação percentual para 40 trabalhos

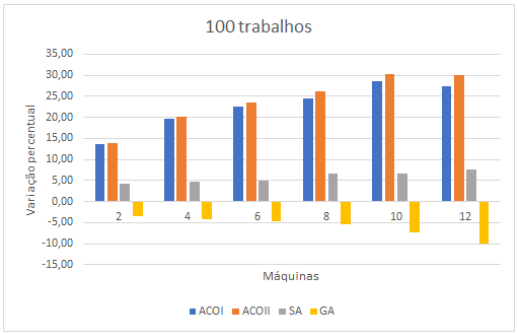

Figura 48 - Variação percentual para 100 trabalhos

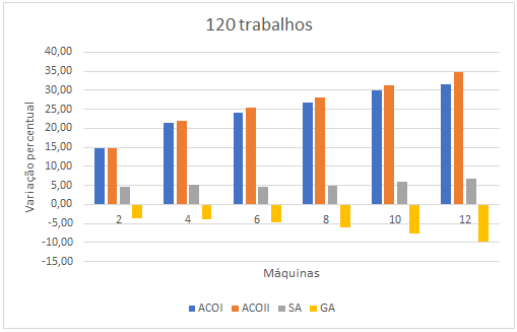

Figura 49 – Variação percentual para 120 trabalhos

APÊNDICE 7 – GRÁFICOS DA VARIAÇÃO PERCENTUAL DOS TEMPOS DE PROCESSAMENTO DOMINANTES

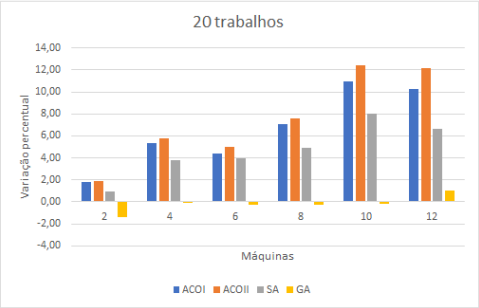

Figura 50 - Variação percentual para 20 trabalhos

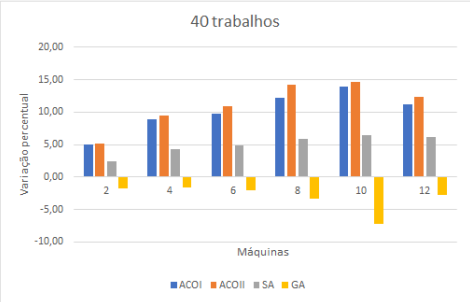

Figura 51 - Variação percentual para 40 trabalhos

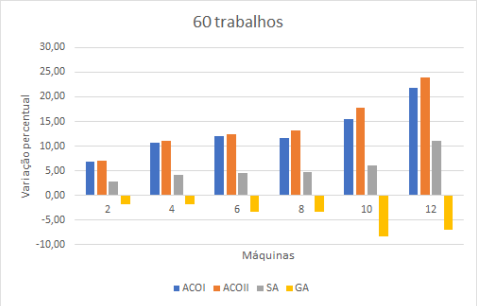

Figura 52 - Variação percentual para 60 trabalhos

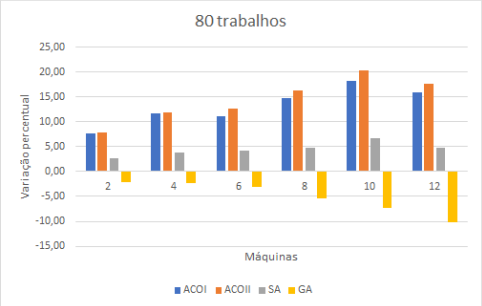

Figura 53 - Variação percentual para 80 trabalhos

APÊNDICE 8 – GRÁFICOS DA VARIAÇÃO PERCENTUAL DOS TEMPOS DE *SETUP* DOMINANTES

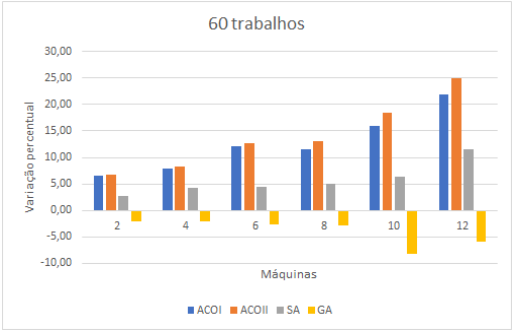

Figura 54 - Variação percentual para 60 trabalhos

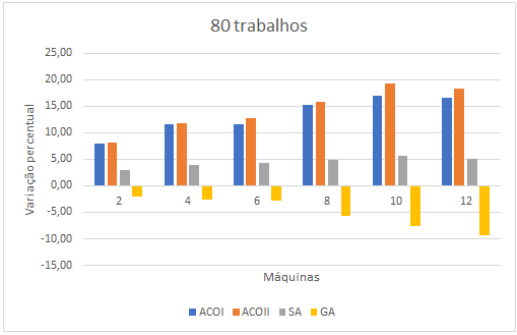

Figura 55 - Variação percentual para 80 trabalhos

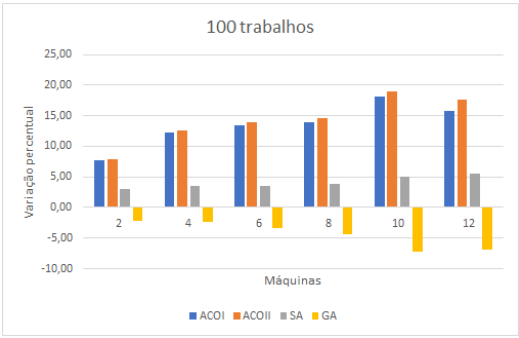

Figura 56 - Variação percentual para 100 trabalhos

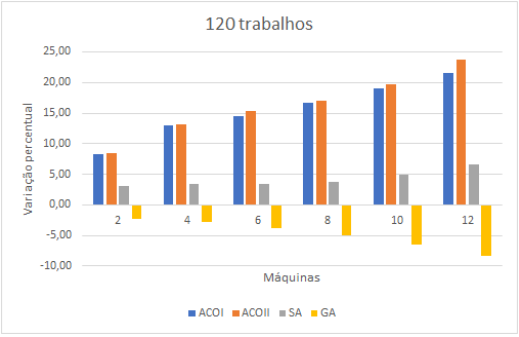

Figura 57 - Variação percentual para 120 trabalhos

APÊNDICE 9 – ORDEM DOS TRABALHOS PARA TEMPOS DE PROCESSAMENTO E SETUP EQUILIBRADOS

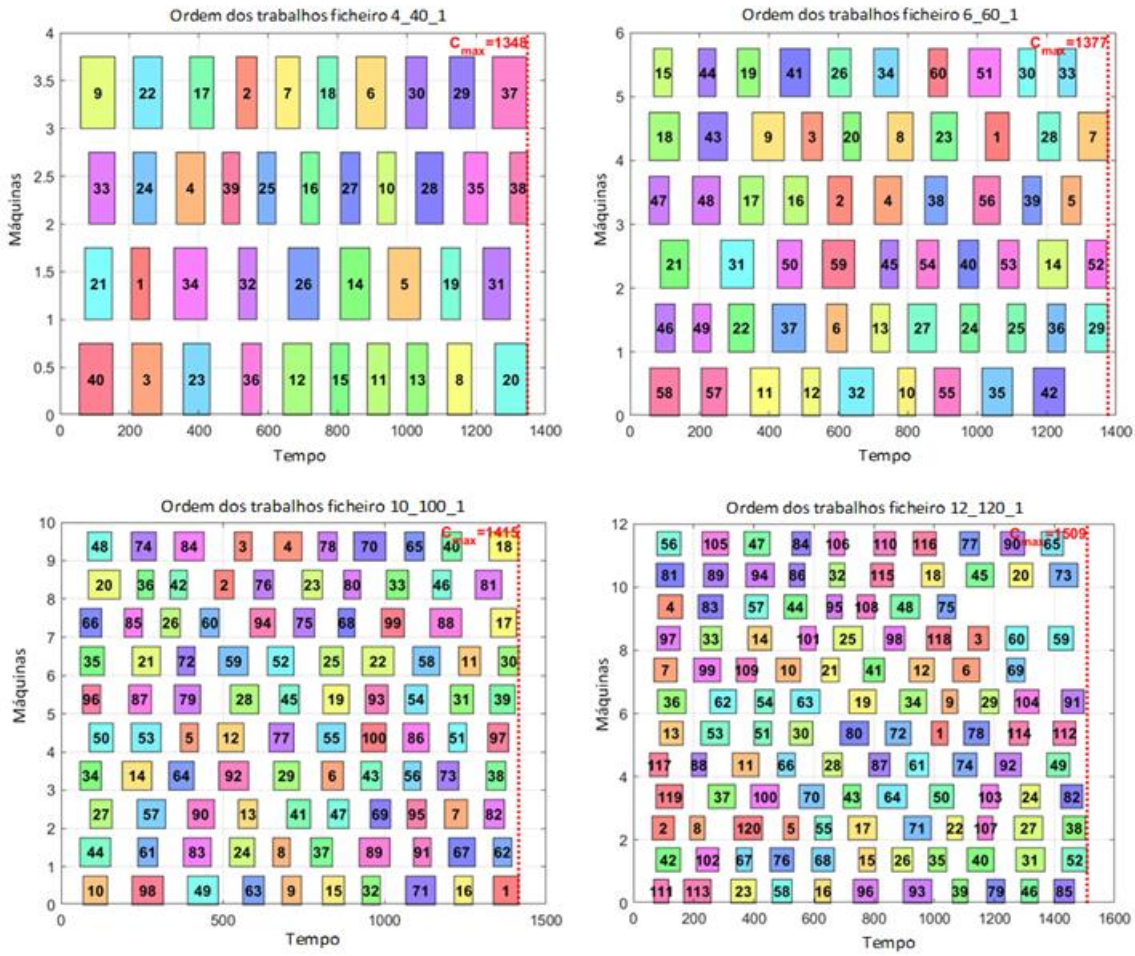

Figura 58 – Ordem de trabalhos para vários exemplos, Pij, Sij equilibrados

APÊNDICE 10 – ORDEM DOS TRABALHOS PARA TEMPOS DE PROCESSAMENTO DOMINANTES

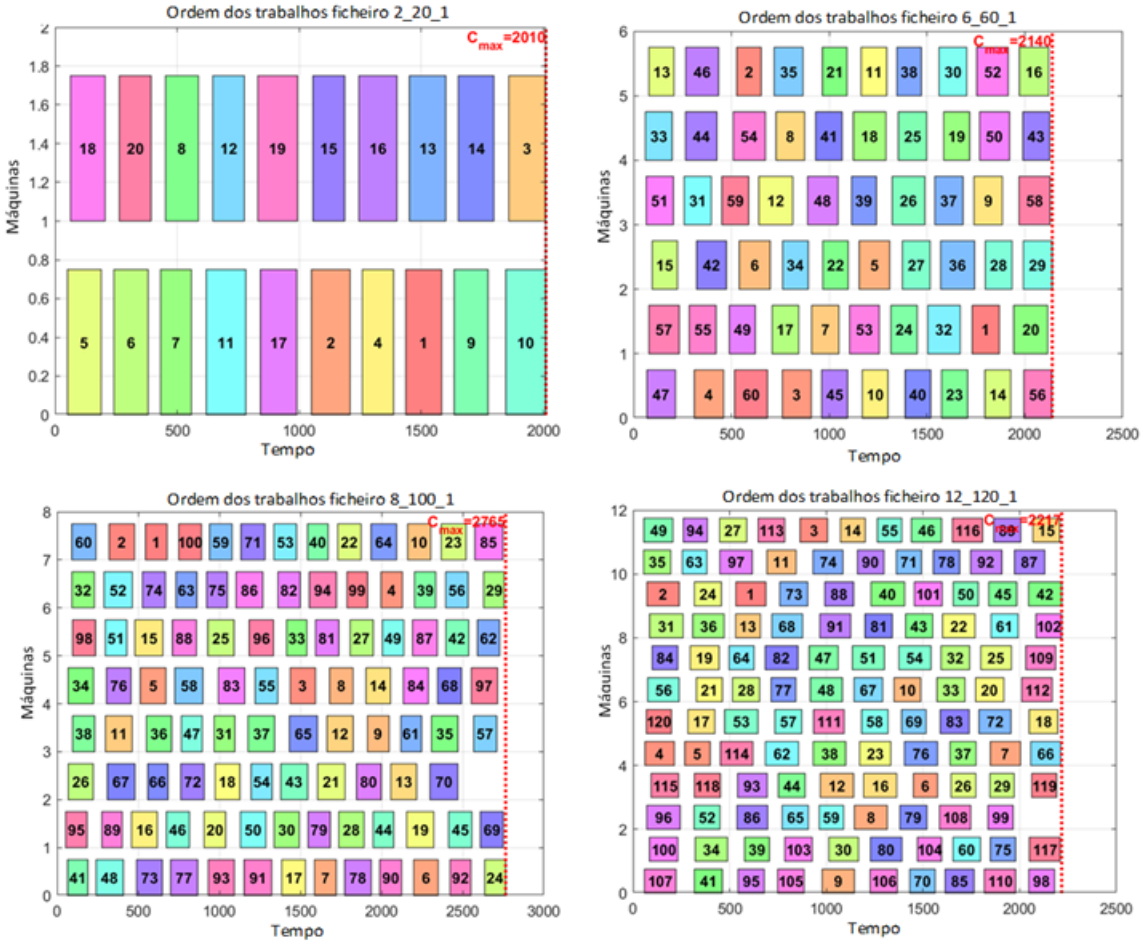

Figura 59 – Ordem de trabalhos para vários exemplos, Pij dominantes

APÊNDICE 11 – ORDEM DOS TRABALHOS PARA TEMPOS DE *SETUP* DOMINANTES

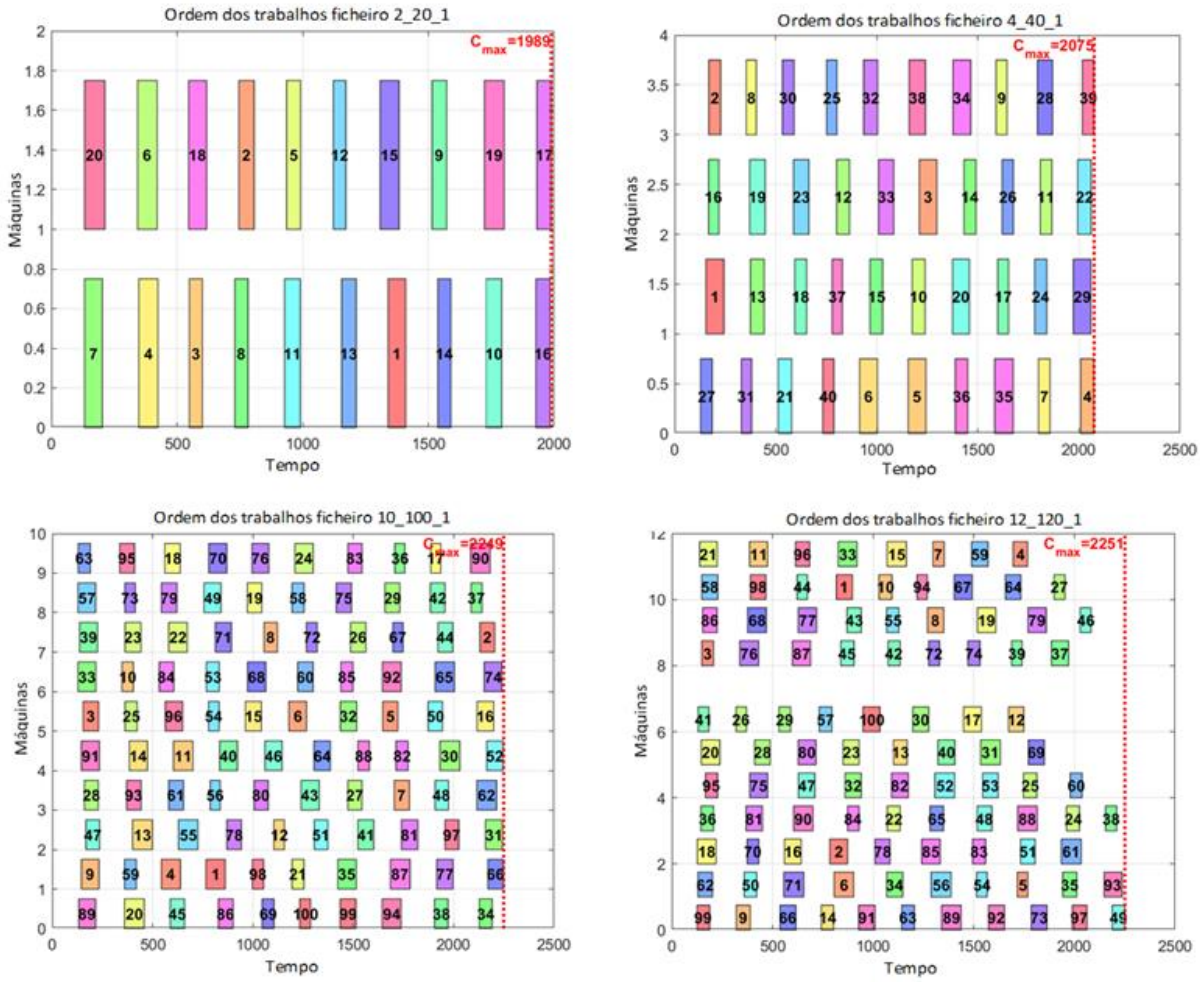

Figura 60 - Ordem de trabalhos para vários exemplos, Sij dominantes