INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Heurística auxiliada por Aprendizagem Automática aplicada a problemas de Escalonamento

SAMUEL DOMINGUES SANTOS COSTA CARVALHO
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Heurística auxiliada por Aprendizagem Automática aplicada a problemas de Escalonamento

Samuel Domingues Santos Costa Carvalho

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento

Orientador: Doutora Ana Maria Madureira
Co-Orientador: Doutor Ivo André Pereira

Porto, 22 de Outubro de 2017

Dedicatória

Dedicado à Dona Maria do Céu.

Resumo

A seleção dos parâmetros de (Meta-) Heurísticas é um processo complexo e por vezes trabalhoso, havendo casos em que abordagens baseadas em modelos de Aprendizagem Automática permitem melhorar esse processo por forma a obter combinações de parâmetros adequadas de modo automático, eficaz e eficiente.

Neste trabalho é proposta uma abordagem que integra uma heurística, *Elitist Particle Swarm Algorithm* (EPSA), com uma camada de *Clustering* e outra de Classificação, com o intuito de prever as combinações de parâmetros mais promissoras para a heurística, tendo em conta a instância do problema que se pretende resolver.

A heurística, EPSA, baseada no conceito das Meta-Heurísticas *Particle Swarm Optimization* (PSO) aplicadas a problemas de otimização discreta, é também apresentada. Esta heurística exibe duas características que, em conjunto, a diferenciam de grande parte das abordagens PSO: a existência de um grupo especial de partículas, Elite, com um comportamento distinto das restantes partículas e uma velocidade que depende apenas da qualidade das soluções.

À abordagem integrada da heurística EPSA com as camadas de *Clustering* e Classificação foi dado o nome *Auto Tuned Elitist Particle Swarm Algorithm* (ATEPSA). Esta abordagem baseia-se no uso de histórico de resolução de determinadas instâncias de um problema, com diferentes combinações de parâmetros, para proceder a uma afinação de cariz *off-line*, servindo-se da experiência anterior de resolução de instâncias semelhantes à que se quer resolver. A abordagem é aplicada ao problema de escalonamento *Single Machine Total Weighted Tardiness* que é um NP-Difícil bem conhecido.

A abordagem ATEPSA apresenta resultados promissores, conseguindo prever boas combinações de parâmetros para uma parte considerável dos casos estudados. A heurística EPSA mostra resultados bastante satisfatórios em tempos de resolução baixos e exibe uma variabilidade baixa numa grande gama de instâncias, o que indica que tende a ter resultados consistentes.

Palavras-chave: Afinação de parâmetros, *Particle Swarm Optimization*, Classificação, *Clustering*, Escalonamento

Abstract

The Metaheuristics' parameter tuning is a complex and hardworking process that, in some cases, is approached with Machine Learning based models which grant improvements to this process so that suitable parameters' combinations are obtained in an automatic, effective and efficient manner.

In this work, an approach that integrates a heuristic, Elitist Particle Swarm Algorithm (EPSA), with a Clustering layer and a Classification layer, aiming to predict the most promising parameter combinations for the heuristic, considering the instance of the problem to be solved is proposed.

The EPSA heuristic, based on the concept of Particle Swarm Optimization (PSO) for discrete optimization problems, is also presented. This heuristic has two characteristics that, together, make it different from most of the PSO approaches: the existence of a special group of particles, Elite, with a behavior that is different from the rest of the particles and a velocity that only depends on the quality of the solutions.

The integration of EPSA, Clustering layer and Classification layer was given the name Auto Tuned Elitist Particle Swarm Algorithm (ATEPSA). This approach is based on the usage of the history of solving certain instances of the problem, with different parameter combinations, to proceed with the tuning process in an off-line mode, taking advantage of the experience in previously solved instances that are similar to the one to be solved. This approach is applied to the Single Machine Total Weighted Tardiness scheduling problem which is a well-known NP-Hard.

The ATEPSA approach shows promising results, being able to predict good parameter combinations for a substantial portion of the studied cases. The EPSA heuristics shows good results with low running times and exhibits a low variability in a wide range of instances, which indicates that it tends to have consistent results.

Keywords: Parameter tuning, Particle Swarm Optimization, Classification, Clustering, Scheduling

Agradecimentos

Quero agradecer aos orientadores pela disponibilidade, colaboração e contribuições essenciais para a realização deste trabalho.

A todos os meus amigos e colegas que me deram força, não só durante esta fase, mas também durante os dois anos do Mestrado em Engenharia Informática.

Um agradecimento especial à Luísa Aguiar pelo apoio incondicional.

Agradeço à minha família que é uma constante fonte de força na minha vida.

Conteúdo

Lista de Figuras

Lista de Algorítmos

Lista de Acrónimos

ACO	*Ant Colony Optimization*.
ARP	*Arc Routing Problem*.
ATEPSA	*Auto Tuned Elitist Particle Swarm Algorithm*.
CV	Coeficiente de Variação.
EPSA	*Elitist Particle Swarm Algorithm*.
GA	*Genetic Algorithms*.
K-NN	*K-Nearest Neighbors*.
MA	*Memetic Algorithms*.
MCLA	Módulo de Classificação.
MCLU	Módulo de Clustering.
MILP	*Mixed Integer Linear Programming*.
MO	Módulo de Otimização.
OPSO	*Opposite-Based Particle Swarm Optimization*.
PSO	*Particle Swarm Optimization*.
SA	*Simulated Annealing*.
SMTWT	*Single Machine Total Weighted Tardiness*.
TS	*Tabu Search*.
WPSO	*Particle Swarm Optimization with a Linearly Decreasing Inertia Weight*.

Capítulo 1

Introdução

1.1 Motivação e Contexto

A otimização nas aplicações atuais assume um carácter fortemente interdisciplinar, estando relacionada com a necessidade de integração de diferentes técnicas e paradigmas na resolução de problemas reais complexos, sendo que a computação de soluções ótimas em muitos destes problemas é intratável. Alguns dos métodos de otimização mais usados para a resolução de problemas do mundo real designam-se por Meta-Heurísticas, que resultam da adaptação de ideias de várias áreas, principalmente com inspiração na natureza. Este tipo de abordagens contem um conjunto de parâmetros que são, tipicamente, de índole contínua ou categórica, e cuja afinação tem um papel essencial no processo de procura e criação de soluções.

O uso de metodologias para selecionar ou gerar, automaticamente, heurísticas (ou Meta-Heurísticas) para resolver problemas de otimização particularmente difíceis torna-se cada vez mais comum. Da mesma forma, a utilização de metodologias que permitem afinar os parâmetros de uma Meta-Heurística tem sido recorrente na comunidade cientifica, devido à sua importância no próprio desempenho das Meta-Heurísticas. A Aprendizagem Automática revela potencial para tornar essas metodologias mais robustas, de modo a ser possível introduzir comportamentos inteligentes na escolha das técnicas e respetiva afinação, com o objetivo de resolver problemas de otimização. Na realidade, a escolha e afinação dos parâmetros de uma Meta-Heurística para a resolução de um dado problema apresenta características semelhantes às de problemas de Aprendizagem Automática (Birattari 2009).

Os problemas de escalonamento fazem parte desse grupo de problemas com uma complexidade tal que, tipicamente, requerem a utilização de abordagens como as Meta-Heurísticas. Este tipo de problemas surge no contexto de linhas de produção em que um conjunto de tarefas deve ser distribuído pelos recursos disponíveis. Em particular, o problema *Single Machine Total Weighted Tardiness* (SMTWT) é um conhecido NP-Difícil que consiste em minimizar os atrasos ponderados de um conjunto de tarefas a serem processadas por uma única máquina, de forma sequencial e sem interrupções. Este problema tem recebido bastante atenção em termos académicos e o seu estudo é muito importante para a compreensão e simplificação de problemas mais complexos (Pinedo 2016).

Por outro lado, o contexto atual dos negócios industriais requer abordagens sofisticadas e inteligentes para que uma organização se possa estabelecer num determinado mercado com uma oferta de produtos e/ou serviços capazes de responder às necessidades dos clientes de forma rápida e eficaz. Um problema acrescido do contexto atual é o aumento da variedade

e personalização dos produtos, por exemplo na indústria do calçado, vestuário, metalomecânica, microprocessadores e outros. A natureza destes problemas, no contexto real, é propensa a perturbações de funcionamento e apresenta uma elevada dinâmica (Madureira 2003), o que torna o seu tratamento ainda mais complexo. Uma resposta eficiente a estas questões, que aumentam a incerteza na previsão dos planos de produção, obriga a utilização de métodos de resolução inteligentes e capazes de se adaptarem a diferentes contextos do ambiente de produção.

Nesta dissertação é apresentada uma heurística para a resolução do problema SMTWT e é proposta uma abordagem de integração dessa heurística com técnicas de Aprendizagem Automática que permitem a escolha dos parâmetros a usar, *a priori*, mediante as características das instâncias do problema a resolver.

1.2 Método e Contribuições

A heurística apresentada neste trabalho é baseada na técnica *Particle Swarm Optimization* (PSO) e é direcionada para a resolução de problemas de otimização combinatória. Neste caso é aplicada ao problema de escalonamento SMTWT, mas as suas características possibilitam que seja aplicada em outros contextos. Esta heurística apresenta uma estrutura muito simples, consegue resultados bastante satisfatórios, em tempos de resolução baixos, para o problema ao qual foi aplicada e, com uma afinação apropriada, atinge valores de variabilidade baixos nas soluções finais. Esses fatores são preponderantes quando se pretende medir a robustez de um método de resolução de problemas de otimização uma vez que transmitem a complexidade da abordagem, a sua eficácia, a sua eficiência e a sua consistência no que toca aos resultados que produz.

A proposta de integração dessa heurística com técnicas de Aprendizagem Automática consiste numa camada de *Clustering* e outra de Classificação para auxiliar a seleção dos parâmetros a usar na heurística, quando se pretende resolver uma determinada instância de um problema. Apesar de a utilização de Aprendizagem Automática na afinação de parâmetros de Meta-Heurísticas ser cada vez mais comum, a utilização deste conjunto de técnicas trata-se de uma forma pouco explorada de abordar esta problemática. No caso estudado, foi capaz de desempenhar o seu papel a um nível interessante, abrindo caminho para a possibilidade de um estudo mais aprofundado.

1.3 Estrutura do Documento

Após este capítulo introdutório, o documento está organizado em mais 6 capítulos.

No Capítulo 2 é feita uma revisão de métodos de resolução heurísticos para problemas de otimização, com especial atenção para o grupo de métodos *Swarm Intelligence*. São também identificados casos de utilização dessas abordagens em diferentes contextos e a problemática da afinação de parâmetros.

O Capítulo 3 aborda a Aprendizagem Automática e técnicas de *Clustering* e Classificação. Para cada tipo de técnica abordada são identificadas as vantagens e desvantagens da sua utilização, bem como o contexto em que se aplicam.

No Capítulo 4 é feita a descrição do problema de escalonamento, SMTWT, apresentando as suas características e identificando diferentes abordagens usadas, na literatura, para a sua resolução.

No Capítulo 5 é apresentada a heurística e a proposta da sua integração com as camadas de *Clustering* e Classificação, sendo detalhada a estrutura e as técnicas usadas.

O Capítulo 6 é dedicado à análise dos resultados da aplicação das abordagens propostas a um conjunto de instâncias do problema SMTWT. Numa primeira parte é estudado o comportamento da heurística para diferentes combinações de parâmetros onde é feita uma comparação com valores de referência. Numa segunda parte é estudado e avaliado o comportamento da camada de aprendizagem.

Por fim, no Capítulo 7, são apresentadas as principais conclusões do trabalho realizado, bem como as suas contribuições e as perspetivas de trabalho futuro.

Capítulo 2

Heurísticas e Meta-Heurísticas

2.1 Introdução

A palavra "heurística" tem origem no termo grego, *heuriskein*, que significa a arte de encontrar novas estratégias de resolver problemas (Talbi 2009). No contexto de problemas de otimização, as abordagens heurísticas são métodos de resolução cujo funcionamento é de caráter não exato, no sentido em que as soluções apresentadas não estão acompanhadas de uma prova de otimalidade. Usualmente, resultam da inspiração de experiências passadas e da observação de comportamentos na natureza, cujo mecanismo de funcionamento é transladado para a resolução de um determinado problema.

Este tipo de técnicas tem especial relevância na resolução de problemas com complexidade de cariz intratável, uma vez que, abdicando da prova de otimalidade, permitem encontrar soluções que podem ser consideradas aceitáveis em tempos de resolução baixos, quando comparados com o tempo necessário para chegar a soluções ótimas (Talbi 2009; Zanakis e Evans 1981). Este tipo de comportamento, que geralmente se baseia na pesquisa de ótimos locais, é particularmente importante em termos industriais e científicos (Pereira 2014), pelo que tem recebido muita atenção tanto ao nível empresarial como académico ao longo das últimas décadas.

Em meados do século passado, o norte americano Herbert Simon (Nobel da Economia 1978 e prémio Turing 1975) e Allen Newell (prémio Turing 1975) identificavam a resolução de problemas de forma heurística como o próximo passo na área de investigação operacional (Simon e Newell 1958). E, efetivamente, o interesse nas heurísticas intensificou-se cada vez mais, surgindo abordagens heurísticas para os mais variados tipos de problemas. No contexto de problemas de escalonamento, William Gere, vai para além das clássicas regras de despacho e propõe algumas heurísticas simples. Estas têm em consideração, entre outros, a influência que possíveis trocas na ordem das tarefas têm no custo total do atraso das tarefas (Gere 1966). Um clássico da investigação operacional, o algoritmo de *Clark and Wright*, surgiu também, há mais de meio século, no contexto do roteamento de veículos, com uma abordagem heurística que mede os "ganhos" de diferentes combinações na sequência pela qual os diferentes destinos são servidos pelos veículos (Clarke e Wright 1964).

Uma elevada quantidade de abordagens heurísticas surgiram nos anos subsequentes, nos mais variados contextos. Como fruto dessa corrente, surgiu o termo "Meta-Heurística", que teve das suas primeiras aparições, em termos científicos, por intermédio de F. Glover no artigo (Glover 1986). Este termo surge da junção dos dois termos gregos, *heuriskein* e *Meta* - metodologia de nível superior - (Talbi 2009). As Meta-Heurísticas são abordagens heurísticas em que a forma de pesquisa por novas solução não está ligada, conceptualmente,

a um problema em específico e, como tal, podem ser aplicadas a diferentes problemas. Este tipo de abordagens têm a característica de, usualmente, ser mais fácil de implementar uma versão *quick-and-dirty* das mesmas para um determinado conjunto de problemas e, ainda assim, apresentarem desempenhos razoáveis (Pereira 2014). No entanto, se o objetivo for a obtenção de resultados muito próximos do ótimo, será necessário, por vezes, proceder a afinações mais precisas (Pereira 2014). Entre as várias vantagens deste tipo de abordagens está também o facto de, ao contrário do que é comum nas heurísticas, as Meta-Heurísticas estão usualmente dotadas de mecanismos que promovem também a pesquisa global, tornando possível superar determinadas soluções ótimas locais (Talbi 2009). No entanto, também existem algumas limitações. Desde logo, é transversal entre as Meta-Heurísticas, a existência de parâmetros que têm de ser estabelecidos *a priori* e que têm grande influência no seu desempenho, sendo que este processo de *tuning* é complexo e muitas vezes moroso, principalmente quando é feito de forma completamente manual (Pereira 2014). Por outro lado, a própria escolha da Meta-Heurística, a sua implementação de forma correta e a definição apropriada do problema a resolver não são processos triviais e requerem, por vezes, abordagens complexas e multi metodológicas (Ferreira 2013).

Há uma grande variedade de Meta-Heurísticas que diferem em muitos aspetos como a natureza da vizinhança e do mecanismo de procura (Talbi 2009; Birattari et al. 2001). No entanto, estas apresentam muitos conceitos comuns. Desde logo, a função objetivo é dependente do problema a resolver e a sua estrutura tem grande impacto em qualquer Meta-Heurística (Pereira 2014). Por outro lado, as características intrínsecas a diferentes Meta-Heurísticas permitem que estas sejam agrupadas. Há várias abordagens, na literatura, para esse agrupamento, no entanto, é comum a divisão em três categorias (Pereira 2014; Talbi 2009):

- **Pesquisa Local:** Consistem na procura a partir de uma solução que, iterativamente, vai sendo melhorada através da pesquisa pela vizinhança. Exemplos deste tipo de Meta-Heurísticas são *Tabu Search* (TS) e *Simulated Annealing* (SA).
- **Evolucionárias:** Baseiam-se na noção de competição e simulam comportamentos semelhantes ao da evolução das espécies (Talbi 2009). Exemplos deste tipo de Meta-Heurísticas são os *Genetic Algorithms* (GA) e os *Memetic Algorithms* (MA).
- **Swarm Intelligence:** São inspirados pelo comportamento coletivo de populações de animais. Exemplos deste tipo de Meta-Heurísticas são *Particle Swarm Optimization* (PSO) e *Ant Colony Optimization* (ACO).

Uma vez que a quantidade de abordagens distintas entre estes três grupos é bastante elevada, concentramos o nosso estudo neste último grupo de Meta-Heurísticas, *Swarm Intelligence*, no qual vai incidir o restante deste capítulo. Com particular atenção às abordagens que, dentro deste grupo, apresentam maior sucesso na literatura.

2.2 Swarm Intelligence

Este tipo de Meta-Heurísticas, inspiradas no comportamento de determinadas espécies de animais, são métodos de resolução de problemas que se caracterizam pela existência de uma população de indivíduos que, iterativamente, vão partilhando informação entre sí (Bonabeau, Dorigo e Theraulaz 1999). Essa partilha de informação traduz-se numa procura pelo espaço de soluções em que as soluções encontradas por um indivíduo são resultantes não só

da evolução das soluções desse mesmo indivíduo, mas também da evolução dos restantes elementos do grupo. Cada indivíduo do grupo é, tipicamente, de natureza pouco sofisticada pelo que a essência deste tipo de algoritmos está na cooperação através da partilha de informação dos diferentes elementos (Talbi 2009).

Das abordagens que se baseiam nestes pressupostos, as que têm sido mais bem sucedidas são *Ant Colony Optimization* e *Particle Swarm Optimization* (Talbi 2009).

2.2.1 Ant Colony Optimization

Esta técnica de otimização foi inicialmente proposta por Marco Dorigo, na sua tese de doutoramento, como um novo método estocástico para otimização combinatória, sob o nome de *Ant System* (Madureira, Falcão e Pereira 2012; Dorigo e Maniezzo 1996). Posteriormente, com base nesta técnica, surgiu a Meta-Heurística ACO.

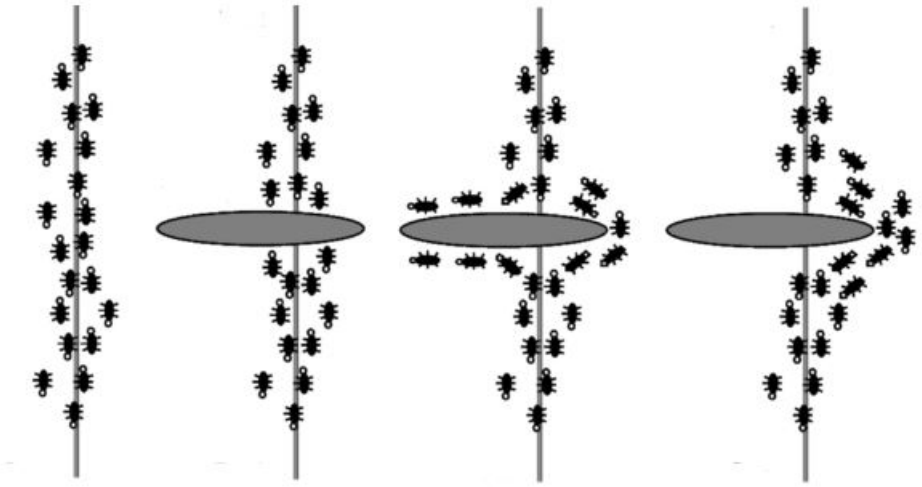

Figura 2.1: Inspiração da Meta-Heurística ACO. (Retirado de Talbi 2009).

Esta abordagem é baseada numa analogia ao comportamento das colónias de formigas e da sua interação mediante a deposição de feromonas (Dorigo e Maniezzo 1996). Na construção de novas soluções, as formigas são influenciadas por informação relacionada com a qualidade de soluções anteriores, sendo essa informação transmitida através da feromona. As soluções criadas a cada iteração são avaliadas e a quantidade de feromona a depositar é calculada consoante a qualidade da solução encontrada. Tipicamente, quanto mais promissora for a solução, maior será a quantidade de feromona a colocar.

Algorithm 2.1 Ant Colony Optimization

```
1: procedure ACO(Parâmetros)
2:     Inicialização dos rastos das feromonas
3:     while Critérios de paragem não satisfeitos do
4:         Constrói soluções para cada formiga
5:         Atualiza feromonas
6:     end while
7: end procedure
```

Em termos macroscópicos esta técnica é relativamente simples, como é mostrado no algoritmo 2.1. No entanto, as diferentes técnicas de construção de soluções e de atualização das feromonas são vastas (Dorigo e Stutzle 2006) e podem atingir graus de complexidade apreciáveis. Por vezes é, também, utilizada uma rotina de pesquisa local ou a recolha de informação global adicional para enviesar o processo de pesquisa de uma perspetiva não local (Dorigo e Stutzle 2006). Em todo o caso, a técnica ACO tem um conjunto de parâmetros iniciais relacionados com as quantidades de feromona a depositar, a taxa de evaporação da feromona, o número de formigas, entre outros, dependendo da abordagem (Dorigo e Stutzle 2006), que necessitam de ser fornecidos *a priori*.

Na sua origem, esta metodologia foi aplicada ao problema do caixeiro viajante, mas rapidamente se percebeu o seu potencial e passou a ser usada como uma Meta-Heurística na resolução de problemas de vários domínios.

Com uma abordagem baseada em ACO e funções de custo *fuzzy* (Garcia et al. 2009) apresenta um método de movimentação autónoma para robots. Para um problema de recolha de resíduos urbanos, (Carvalho, Rodrigues e Soeiro 2014) usam um modelo *Mixed Integer Linear Programming* (MILP) de *Arc Routing Problem* (ARP) em conjunto com uma abordagem ACO. ACO é também utilizada no campo da biologia, mais concretamente no sequenciamento do genoma (Rekaya et al. 2013) e em estratégias paralelas para unidades de processamento gráfico (Angelo, Augusto e H. J. C. Barbosa 2013).

No que diz respeito a aplicações em problemas de escalonamento, com um método híbrido de ACO e um algoritmo de pós processamento, (Heinonen e Pettersson 2007) resolvem um problema de *Job-Shop*. Com recurso a um método ACO (Rossi e Dini 2007) resolve um problema de *Job-Shop* com rotas flexíveis e tempos de *setup* separáveis. Para um problema de máquina única, (Liao e Juan 2007) usam uma abordagem ACO em que lidam com tempos de *setup* dependentes da sequência. Com recurso a uma abordagem *Knowledge-Based Ant Colony Optimization* (Xing et al. 2010) resolvem problemas *Job-Shop*. Para um problema de *Flow-Shop* multi objetivo (Yagmahan e Yenisey 2008) também usam uma abordagem ACO.

2.2.2 Particle Swarm Optimization

A técnica *Particle Swarm Optimization* é um método estocástico e populacional criado por James Kennedy e Russel Eberhart em 1995 (Kennedy e Eberhart 1995). A PSO é inspirada pelo comportamento social de populações de seres vivos como bandos de pássaros ou cardumes de peixes que se movem em conjunto, de forma coordenada e sincronizada com o objetivo de maximizar as suas capacidades de sobrevivência em processos como a descoberta de alimento e a defesa contra agressões externas (Kennedy e Eberhart 1995).

Nesta técnica os indivíduos da população são denominados como partículas e são, tipicamente, sempre os mesmos no sentido em que esta técnica não produz novas populações (Pereira 2014) nem descarta elementos da população. A posição de cada partícula representa uma solução no espaço das soluções do problema. Cada partícula parte de uma posição inicial que, a cada iteração, evolui mediante um vetor de velocidade (Kennedy e Eberhart 1995), analogamente ao movimento de uma partícula na mecânica de Newton.

$$\overrightarrow{X}_{p,t} = \overrightarrow{X}_{p,t-1} + \overrightarrow{V}_{p,t} \tag{2.1}$$

Em que $\overrightarrow{X}_{p,t-1}$ representa o vetor posição da partícula p na iteração $t-1$ e $\overrightarrow{X}_{p,t}$ representa o vetor posição da partícula p na iteração t, resultante da aplicação do vetor de velocidade $\overrightarrow{V}_{p,t}$ sobre a posição em $t-1$. Este comportamento traduz-se numa procura de soluções com o intuito de melhorar o valor de uma determinada função objetivo que, tipicamente, se pretende maximizar ou minimizar (Pereira 2014; Talbi 2009).

Algorithm 2.2 Particle Swarm Optimization

```
1:  procedure PSO(Parâmetros)
2:      Inicialização das Partículas
3:      while Critérios de paragem não satisfeitos do
4:          for all partículas do
5:              Avalia a função objetivo
6:          end for
7:          Determina gBEST
8:          Atualiza as velocidades
9:          Atualiza as posições
10:     end while
11:     Retorna gBEST
12: end procedure
```

Neste algoritmo, a posição e velocidade de cada partícula é iniciada, podendo ser de forma aleatória ou com outra abordagem. Após essa inicialização o algoritmo entra num ciclo, no qual se irá manter até que algum dos critérios de paragem pré-definidos seja satisfeito. Nesse ciclo são avaliadas as posições das partículas, sendo atualizados os valores da melhor posição obtida por cada partícula, *pBest*, e a melhor posição entre todas as partículas, *gBest*. De seguida são recalculados os valores das velocidades de cada partícula e aplicados à posição da mesma, originando novas soluções que voltarão a ser avaliadas, continuando o ciclo. Quando o ciclo termina é retornada a informação da melhor solução encontrada, *gBest* (Pereira 2014). Tipicamente, o vetor velocidade assume uma estrutura do tipo:

$$\overrightarrow{V}_{p,t} = W \times \overrightarrow{V}_{p,t-1} + C1 \times R1 \times (\overrightarrow{pBest}_p - \overrightarrow{X}_{p,t-1}) + C2 \times R2 \times (\overrightarrow{gBest} - \overrightarrow{X}_{p,t-1}) \quad (2.2)$$

Em que R1 e R2 são valores aleatórios pertencentes ao intervalo [0,1], C1 e C2 são, respetivamente, o fator cognitivo e o fator social, e W é, usualmente, o fator de inércia. Quanto maior for este fator, W, maior será a capacidade de exploração global das partículas, pelo que é necessário ajustar este valor para que as partículas tenham o equilíbrio entre a procura local e global pretendido (Pereira 2014). Em determinadas abordagens, o próprio fator de inercia é variável ao longo do tempo (Lin et al. 2016).

Ao contrário do método ACO, a abordagem PSO, surgiu inicialmente no contexto da resolução de problemas de otimização em funções contínuas e tem continuado a mostrar excelentes resultados nesse domínio (Talbi 2009). Exemplo disso é a heurística DEEPSO (Miranda e Alves 2014) que usa uma combinação de várias técnicas, entre as quais PSO, e mostra resultados interessantes na área dos sistemas energéticos.

No entanto, têm surgido cada vez mais abordagens PSO aplicadas também a problemas discretos e de natureza combinatória. Para um problema de múltiplos caixeiros viajantes, (Pang et al. 2013) aplicam uma heurística PSO clássica. Em (Marinakis, Iordanidou e Marinaki 2013), no contexto de um problema de roteamento de veículos com procura estocástica, são abordadas múltiplas variantes de PSO, nomeadamente no que diz respeito a variações na equação de atualização da velocidade. Para um problema de escalonamento

multi-facility (Lin et al. 2016) usam um conjunto de abordagens PSO, nomeadamente: *Opposite-Based Particle Swarm Optimization* (OPSO) e *Particle Swarm Optimization with a Linearly Decreasing Inertia Weight* (WPSO).

Em (MAH, Hossain e Akter 2016) é feito um estudo comparativo de várias Heurísticas baseadas em PSO, aplicadas ao problema do cacheiro viajante. Para além disso, são abordadas, explicitamente, as diferenças fundamentais entre a aplicação de PSO para problemas de combinatória e a original aplicação a problemas contínuos. As mesmas questões são abordadas em (Shi et al. 2007) e, ambos os casos, definem um operador de permutação do tipo $P^{i,j}$ que, aplicado a uma solução, faz trocas na sequências da seguinte forma: seja $S = [a, b, c, d, e]$ uma solução e S' a solução resultante da aplicação do operador $P^{1,3}$ a S, S' tomará a seguinte forma:

$$S' = P^{1,3}[a, b, c, d, e] = [c, b, a, d, e] \tag{2.3}$$

Este operador, permite redefinir o operador "+" (MAH, Hossain e Akter 2016) como uma permutação e a velocidade V=k como uma sequência de k permutações. No entanto, (Shi et al. 2007) vai mais longe e define outros operadores que não vamos pormenorizar no contexto deste trabalho.

2.3 Afinação de parâmetros

As Meta-Heurísticas apresentam um conjunto de parâmetros que desempenham um papel fundamental na procura e construção de soluções. Esses parâmetros têm de ser instanciados de forma apropriada para que se possa tirar o melhor partido das capacidades das Meta-Heurísticas (Birattari 2009).

Tipicamente, o número de combinações possíveis dos diferentes parâmetros pode atingir valores muito elevados o que faz com que a tarefa de determinação de um bom conjunto de parâmetros a usar seja reconhecida como um processo que consome muito tempo (Hutter et al. 2006). Por outro lado a qualidade de uma combinação de parâmetros é dependente não só do problema a abordar mas também da instância do problema, pelo que não existem combinações de parâmetros que sejam ótimas de forma universal, para uma dada Meta-Heurística (Talbi 2009). De facto os teoremas *No free lunch* (Wolpert e Macready 1997) mostram isso mesmo. Além disso, há ainda possibilidade de haver parâmetros que sejam mais relevantes do que outros, no sentido em que as suas variações têm maior peso no desempenho da Meta-Heurística (Pereira 2014) ou ainda a possibilidade de interdependências de parâmetros que dificultam o estudo de cada parâmetro de forma isolada.

Estes factos justificam o interesse no desenvolvimento de técnicas que permitam automatizar esse processo de escolha, por forma a diminuir o tempo necessário e melhorar a qualidade dos parâmetros usados (Birattari 2009).

Existem duas principais estratégias na afinação de parâmetros (Talbi 2009):

- *Off-line:* A combinação de parâmetros é estabelecida antes da execução da Meta-Heurística.
- *On-line:* Os parâmetros são atualizados ao longo da execução da Meta-Heurística.

Cada uma destas estratégias engloba diferentes conjuntos de técnicas que abordam o problema da afinação de parâmetros de formas distintas.

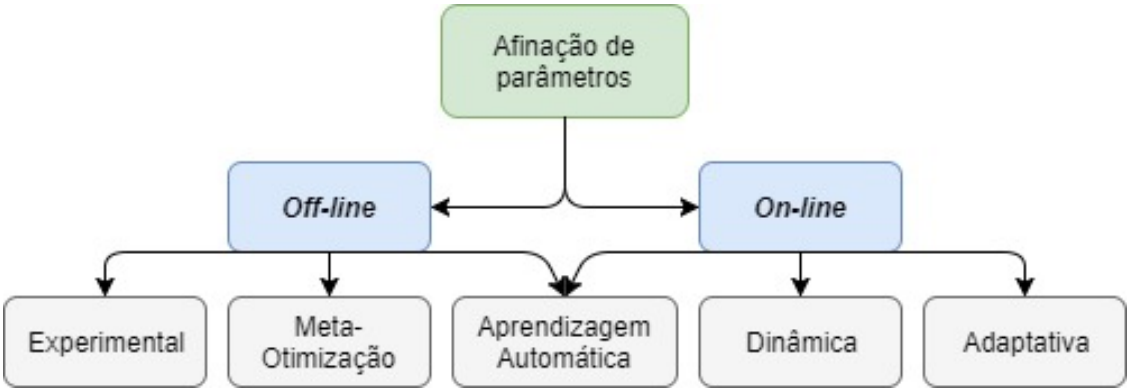

Figura 2.2: Diferentes técnicas para abordar a afinação de parâmetros. (Adaptado de Pereira 2014).

2.3.1 Afinação de Parâmetros Off-line

É comum, na utilização de Meta-Heurísticas, que a afinação de parâmetros seja feita de forma completamente empírica, sendo afinado um parâmetro de cada vez e não sendo estudadas as dependências entre parâmetros (Talbi 2009). As técnicas de afinação de parâmetros *off-line* têm o objetivo de determinar, *a priori*, combinações de parâmetros para usar na execução das Meta-Heurísticas, permitindo que esse processo de escolha seja automatizado.

A **Meta-Otimização** consiste na utilização de uma camada superior de otimização cuja procura incide sobre os parâmetros da Meta-Heurística que se quer afinar. Tipicamente, esta técnica é composta por dois níveis (Pereira 2014):

- **Nível Meta:** Neste nível o método de otimização retorna soluções que correspondem a parâmetros de uma Meta-Heurística, como por exemplo a taxa de evaporação para uma Meta-Heurística ACO. Cada resultado deste nível corresponde a uma Meta-Heurística com uma determinada configuração dos parâmetros.
- **Nível Base:** A este nível está a Meta-Heurística a ser usada para os parâmetros determinados no Nível Meta e corresponde à resolução do problema de otimização base que se pretende resolver. Tipicamente, retorna uma solução do problema de otimização e o seu valor mediante uma determinada função objetivo.

Esta abordagem é bastante usada na área de computação evolucionária onde algoritmos evolucionários funcionam como camada de nível meta (Pereira 2014). (Stephenson et al. 2003), usa um método de Meta-Otimização com aprendizagem automática e algoritmos evolucionários para a escolha de heurísticas de compiladores de código, conseguindo ganhos médios de 23% na velocidade.

Por outro lado, é comum o recurso a uma **abordagem experimental** onde são definidos (Talbi 2009):

- **Fatores** que representam os parâmetros a variar nas experiências a realizar.
- **Níveis** que representam os diferentes valores dos parâmetros, podendo estes ser quantitativos ou qualitativos.

Assim, são realizadas experiências para todas as combinações de níveis, sendo os melhores resultados para cada fator identificados. O principal problema deste tipo de métodos prende-se com o facto de, quando estamos na presença de muitos fatores e níveis, o custo computacional é demasiado elevado para realizar todas as experiências (Talbi 2009). Se tivermos f fatores com n níveis cada, o número de experiências a fazer, para explorar todas as combinações, será n^f.

Este tipo de abordagens é conhecido na literatura como *Design of Experiments* e é usualmente utilizado para a afinação de parâmetros, nomeadamente em conjugação com outros métodos. (E. Barbosa e Senne 2017) apresentam um método de *Design of Experiments* com métodos estatísticos e de Inteligência Artificial que aplicam à afinação de parâmetros de uma Meta-Heurística SA e outra GA. Numa combinação de *Design of Experiments* e *Artificial Neural Networks*, (Dobslaw 2010), apresenta uma *framework* para afinação de parâmetros de Meta-Heurísticas.

De facto, não só *Artificial Neural Networks*, mas também outras técnicas relacionadas com **aprendizagem automática** são usadas de forma recorrente neste contexto. (Hutter et al. 2006) mostra como certos modelos de aprendizagem automática podem ser usados para a determinação e ajuste de parâmetros, de forma *off-line*, bem como para fazer previsões precisas dos tempos de processamento.

A utilização de algoritmos de *racing*, tipicamente usados para parametrização e seleção de modelos no contexto de aprendizagem automática (Maron e Moore 1997), também mostra bons resultados quando aplicado à afinação de parâmetros *off-line* de Meta-Heurísticas, em particular o algoritmo *F-Race* (Birattari 2009).

2.3.2 Afinação de Parâmetros On-line

As estratégias de afinação de parâmetros *On-line* consistem em alterações ao valores dos parâmetros de uma Meta-Heurística na altura em que está a ser executada. Estas estratégias apresentam vantagens como o facto de a eficácia de um conjunto de parâmetros poder variar ao longo do processo de procurar da Meta-Heurística (Talbi 2009). Assim, neste aspeto, este tipo de técnicas consegue se adaptar às alterações necessárias nos parâmetros, ao contrário do que acontece no caso da afinação *off-line*.

Este tipo de técnicas podem ser classificadas da seguinte forma (Talbi 2009):

- **Afinação dinâmica:** Os parâmetros vão sendo alterados de forma aleatória ou seguindo uma lógica determinista, mas sem ter em conta o processo de procura.
- **Afinação adaptativa:** Os parâmetros são ajustados de acordo com o processo de procura. Esses ajustes podem ser baseados em vários fatores como a memória ou regras pré-definidas (Pereira 2014).

Também neste tipo de abordagens, *On-line*, a aprendizagem automática é aplicada. (Lessmann, Caserta e Arango 2011), com uma abordagem *on-line* adaptativa, usa um método híbrido de aprendizagem automática que integra modelos de regressão capazes de ajustar os valores dos parâmetros de uma Heurística PSO através da informação recolhida dos melhores movimentos em cada iteração.

Como parte da afinação adaptativa surgem ainda algoritmos auto-adaptativos que controlam a forma como se adaptam às alterações da procura (Pereira 2014). Com uma abordagem

auto-adaptativa de pesquisa local, (Alabas-Uslu e Dengiz 2011), apresenta uma heurística aplicada a um problema de rotas de veículos.

Capítulo 3

Aprendizagem Automática

3.1 Introdução

A aprendizagem automática é uma área da Inteligência Artificial que se foca na conceção de técnicas e algoritmos que permitem que os sistemas computacionais sejam capazes de processar, aprender e extrair informação sobre os dados (Raschka 2015). Esta capacidade de aprendizagem é particularmente importante em casos em que não é possível escrever, diretamente, um programa para resolver um determinado problema que depende de uma determinada fonte de dados ou de experiência (Alpaydin 2004).

A utilização de técnicas de aprendizagem automática tem vindo a crescer de forma considerável nos últimos anos (Dietterich 2009). No entanto, já em 1959, num artigo publicado no IBM Journal, o norte-americano Arthur Lee Samuel apresentava o seu programa *Game of Checkers*, capaz de aprender a jogar damas, melhor do que a pessoa que o criou, num espaço de tempo de 8 a 10 horas baseando-se apenas em conceitos que não iam muito para além das regras básicas do jogo e do seu objetivo (A. L. Samuel 1959). Em 1967, o mesmo autor, publicava avanços significativos, em relação à versão inicial do seu programa, em (A. Samuel 1967). Ainda antes, em 1950, Alan Turing, introduzia o Jogo da Imitação que pretendia determinar se uma máquina podia ter comportamento inteligente, num dos artigos mais emblemáticos com enquadramento nesta área (Turing 1950; Sammut e Webb 2010).

Com o passar dos anos, a comunidade cientifica passou a estar cada vez mais interessada e empenhada no desenvolvimento de novas técnicas, mais eficazes e que nos permitem chegar mais longe no que diz respeito à capacidade de permitir aprendizagens cada vez mais complexas por parte dos computadores (Dietterich 2009; Raschka 2015). Em 2016 o AlphaGo da Google foi capaz de vencer um *Master* do Jogo de GO da era moderna (jogo este que atinge uma complexidade superior à do xadrez). Usando a técnica de *Monte Carlo Tree Search*, em conjunto com *Deep Neural Networks* (Silver, Schrittweiser e Simonyan 2017), o AlphaGo venceu o Sul Coreano Lee Sedol por 4-1 num grande feito da Inteligência Artificial em termos do "confronto" Máquina-Homem (PHYS.ORG 2016a). Com a nota interessante de que, o único jogo perdido pela máquina foi por desistência (PHYS.ORG 2016b).

No entanto, apesar da grande contribuição que representa o estudo e implementação das técnicas no contexto dos jogos, a aplicação das técnicas de aprendizagem automática é muito mais ampla e, hoje em dia, estamos rodeados de algoritmos dessa natureza (Dietterich 2009; Alpaydin 2004). Todos os dias interagimos com dispositivos com capacidade de reconhecer padrões, extrair conhecimento e aprender nos mais variados contextos, muitas vezes sem nos

darmos conta. Com uma simples navegação na internet ou uma pesquisa no google estamos a interagir com algoritmos dessa natureza e a fornecer-lhe uma quantidade apreciável de informação em poucos minutos.

No livro "A Física do Futuro" (Kaku 2011), o cientista Michio Kaku, conhecido principalmente pela sua enorme apetência para a divulgação científica, tem uma visão sobre como a ciência irá moldar o mundo nos próximos 100 anos e mostra que, em grande parte, serão as áreas da Inteligência Artificial que irão tornar possíveis coisa inimagináveis. Afirma que estaremos rodeados de dispositivos que estarão constantemente a monitorizar o nosso estado de saúde, a sugerir determinados tipos de comportamentos, consoante o nosso histórico clínico, e a satisfazer "todos" os nossos desejos, prevendo-os. Em tom de brincadeira, mas com seriedade à mistura, afirma que tecnologias da série televisiva, futurista, *Star Trek*, estão bem mais próximas do que o que muitos de nós possamos imaginar.

Neste capítulo não vamos abordar os conceitos adjacentes ao *Warp drive*. Inicialmente, vamos contextualizar algumas técnicas em diferentes tipos de aprendizagem existentes e, posteriormente, focar no *Clustering* e na Classificação, que são dois grandes grupos de técnicas amplamente usadas em Aprendizagem Automática.

3.2 Tipos de Aprendizagem

Entre os vários tipos de Aprendizagem Automática existentes, há dois grande grupos: Aprendizagem Supervisionada e Aprendizagem Não-Supervisionada. É nestes dois grupos que vamos focar nesta secção, no entanto é importante referir que existem muitos outros grupos: Aprendizagem por Reforço, Aprendizagem baseada em Instâncias, Aprendizagem Analítica entre outros (Pereira 2014; Dietterich 2009).

3.2.1 Aprendizagem Supervisionada

A Aprendizagem Supervisionada tem, como principal objetivo, a obtenção da capacidade de fazer previsões assertivas para novos dados, através do resultado do treino efetuado, previamente, em dados conhecidos (Raschka 2015; Pereira 2014). Este treino é efetuado sobre um conjunto de dados organizado em pares do tipo (entrada, saída). Tipicamente, a entrada pertence a um determinado conjunto de partida com estrutura de natureza vetorial, no sentido em que possui várias componentes, e a saída pertence a um conjunto de chegada em que cada elemento representa um valor (rótulo) (Sammut e Webb 2010; Pereira 2014). O objetivo é, então, a obtenção de funções/regras que sejam capazes de realizar um mapeamento entre elementos do conjunto de partida e do conjunto de chegada através dos dados de treino.

Dependendo do tipo de resultado que é obtido pela função/regra, podemos estar perante uma de duas situações distintas:

- **Regressão:** O conjunto de chegada é de natureza contínua.
- **Classificação:** O conjunto de chegada está subdividido em classes.

Dependendo do tipo de problema que se pretende resolver há a necessidade de escolher uma técnica adequada.

3.2.2 Aprendizagem Não-Supervisionada

A Aprendizagem Não-Supervisionada tem o objetivo de, explorando a estrutura dos dados, extrair informação com significado sem recurso a exemplos rotulados (Raschka 2015; Pereira 2014). Assim, este tipo de técnicas permite a descoberta/confirmação de padrões (eu da não existência deles) nos dados. Assim sendo, a criação de grupos de objetos, *Clustering*, e a deteção de *outliers* são dois exemplos de aprendizagem Não-Supervisionada.

3.3 Classificação

A partir do conhecimento prévio de determinadas classes de objetos e das características dos próprios objetos, a Classificação, em Aprendizagem Automática, permite enquadrar (classificar) um novo objeto numa dessas classes. Tipicamente, existem, pelo menos, duas fases na conceção de um classificador (Alpaydin 2004):

- **Fase de Treino:** Usando um conjunto de dados de treino conhecido, o classificador é treinado. Isto significa que, são usadas as características e os rótulos de cada objeto do conjunto de treino para que o classificador estabeleça relações do tipo (características -> rótulo).
- **Fase de Teste:** A partir de dados em que faltam os rótulos, o classificador treinado vai prever, a partir das características de um dado objeto, qual o rótulo correspondente. Nesta fase, é comum os rótulos serem conhecidos a priori, para que se faça uma validação, posterior, da performance do classificador através da comparação com os resultados da classificação.

Um exemplo deste tipo de abordagens é o classificador *K-Nearest Neighbors* (K-NN) (k vizinhos mais próximos). O K-NN é um dos métodos mais simples de classificação sendo, por isso, amplamente utilizado como uma primeira abordagem, principalmente quando se tem pouco conhecimento sobre os dados de treino (Peterson 2009). A fase de treino desta técnica consiste em, simplesmente, guardar os pares (características, rótulo) de cada objeto. Posteriormente, através da definição de uma métrica, de uma decisão e de uma valor de k, o classificador vai determinar os k vizinhos mais próximos (dada a métrica escolhida) e tomar uma decisão. Usualmente a decisão é incluir o novo objeto na classe que for dominante entre os k vizinhos mais próximos. No entanto pode ser de naturezas variadas (Peterson 2009).

3.4 Clustering

As técnicas desta natureza têm o objetivo de criar grupos de objetos com características semelhantes. Os grupos criados dependem, tipicamente, de uma função de similaridade e do próprio mecanismo de agrupamento da técnica usada. As características dos objetos usadas para o agrupamento e a respetiva normalização também têm, usualmente, impacto na formação dos grupos.

Nesta secção vamos abordar duas técnicas diferentes: *K-Means* e *Affinity Propagation*.

3.4.1 K-Means

K-Means é uma técnica de agrupamento de n objetos em k *clusters*, para $k \leq n$. Usando a distância Euclidiana, os *clusters* são criados através da minimização dos quadrados das diferenças em relação ao centroide de cada *cluster* (Macqueen 1967; Singh, Malik e Sharma 2011):

$$\sum_{j=1}^{k} \sum_{i \in C_j} ||x_i - \mu_j||^2 \tag{3.1}$$

Em que C_j é o *cluster* j, μ_j o seu centroide e i representa um objeto de C_j. Assim, num processo iterativo, os centroides vão sendo recalculados, a expressão 3.1 reavaliada e os *clusters* reajustados até o algoritmo convergir. Esta técnica dá resultados bastante satisfatórios em muitos contextos e tem a enorme vantagem de ser muito intuitiva e simples de perceber/implementar (Raschka 2015). No entanto, apresenta algumas limitações (Singh, Malik e Sharma 2011), desde logo a necessidade de se fornecer *a priori* o número de *clusters* a serem formados:

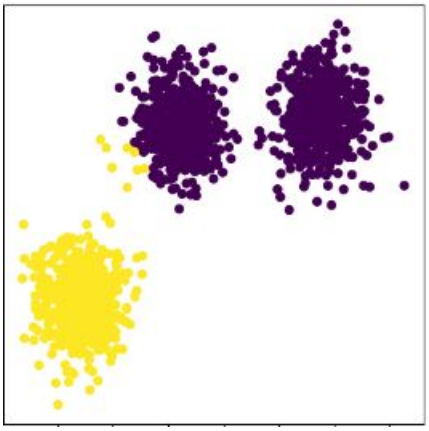

Figura 3.1: Escolha errada do número de *cluster*. (Retirado de Sklearn s.d.).

A Figura 3.1 é um exemplo da má definição do valor de k *clusters* a serem formados. Entre as limitações desta técnica estão também o facto de a inicialização dos centroides ter um grande penso na eficácia do método (Karimov e Ozbayoglu 2015) e de, tipicamente, ter um comportamento isotrópico, dificultando o reconhecimento de estruturas com formas não "circulares" (Figura 3.2).

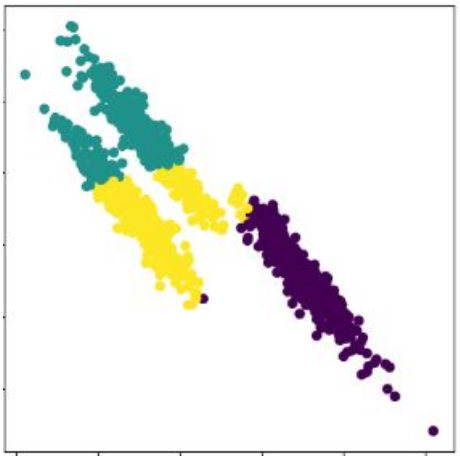

Figura 3.2: Escolha errada dos *clusters* devido à anisotropia. (Retirado de Sklearn s.d.).

Há abordagens que, conjugando o método *K-Means* com outras técnicas, permitem superar algumas das desvantagens, no entanto, aumentam a complexidade de implementação, o que leva a uma perda da simplicidade, que é um dos principais pontos fortes deste método de *clustering* (Karimov e Ozbayoglu 2015). No entanto, por vezes é necessário utilizar técnicas mais avançadas para que se tenha os resultados pretendidos, dependendo do âmbito.

3.4.2 Affinity Propagation

A técnica de *Affinity Propagation* é um algoritmo que, através de uma função de similaridade e da avaliação da força das ligações entre diferentes objetos, determina o grau de responsabilidade e o grau de disponibilidade entre pares de pontos (Dueck e Frey 2007). A responsabilidade $r(i,k)$ mede a tendência que haverá para o objeto k servir de representante para o objeto i. Por outro lado, a disponibilidade $d(i,k)$ mede a tendência para o objeto i escolher o objeto k como seu representante. Desta forma, iterativamente, vão sendo criados representantes de grupos de objetos até atingir um equilíbrio estável, em que nenhum representante de grupo é alterado, nem nenhum objeto altera o seu representante. A configuração final de representantes de grupos e respetivos apoiantes define os *clusters* (Dueck e Frey 2007).

Este algoritmo requer um parâmetro inicial, a preferência, que influencia a afinidade entre pontos de tal forma que, quanto menor for o seu valor, maior será a tendência para a criação de um número de *clusters* mais elevado. Desta forma, este algoritmo consegue encontrar, não só os elementos dos diferentes *clusters*, mas também o próprio número de *clusters*, o que representa uma diferença fundamental em relação a outras técnicas como o *K-Means*. Por outro lado, como se baseia em relações entre todos os pares de pontos, tem maior capacidade de ajuste a qualquer tipo de formas que os *clusters* possam revelar, não sofrendo da tendência para a isotropia, como no caso do *K-Means*. No entanto, este algoritmo tem uma complexidade de $O(N^2T)$ (Fujiwara, Irie e Kitahara 2011), em que N representa o número de objetos e T o número de iterações, tornando-o pouco recomendado para casos em que N é extremamente elevado.

Ainda assim, há técnicas que visam tornar este algoritmo mais rápido, mantendo quase intactas as suas restantes propriedades, como é o caso de (Fujiwara, Irie e Kitahara 2011),

em que o tempo computacional é transformado em $O(N^2 + MT)$ onde M é o número de arcos do grafo.

Capítulo 4

Problema de Escalonamento de máquina única

4.1 Introdução

Os problemas de escalonamento têm recebido bastante atenção em termos académicos devido à sua complexidade em termos teóricos e à sua relevância prática para a indústria. Estes problemas enquadram-se no grupo de problemas NP difíceis cuja complexidade é exponencial (Pinedo 2016).

O escalonamento é um processo de tomada de decisão que lida com a alocação de recursos a um conjunto de tarefas ao longo do tempo por forma a otimizar um conjunto de objetivos (Pinedo 2016). Consoante os objetivos e as restrições a que estão sujeitos, os problemas de escalonamento podem tomar variadas formas. Na literatura também é usado o termo "sequenciamento", que se refere à ordem de execução das diferentes tarefas (Madureira 1995).

Neste capítulo vamos focar a nossa atenção no problema de escalonamento de máquina única, mais concretamente para a variante que visa minimizar o custo total dos atrasos no processamento das diferentes tarefas, conhecido na literatura como *Single Machine Total Weighted Tardiness* (SMTWT) (Pinedo 2016).

4.1.1 Problema de máquina única: Total Weighted Tardiness

Os problemas de máquina única consistem em alocar um conjunto de tarefas, de forma sequencial, a uma só máquina. Estes problemas estão entre os mais elementares no que diz respeito a escalonamento (Pereira 2014) mas, apesar da sua aparente simplicidade, estes problemas podem ser bastante complexos e o seu estudo é muito importante para servir de base para a simplificação de problemas de escalonamento mais complexos e que envolvam um maior número de recursos disponíveis (Pinedo 2016). (Madureira 2003), divide um problema de escalonamento *Job-Shop* Alargado dinâmico em múltiplos subproblemas de máquina única onde aplica otimizações locais através de um conjunto de diferentes Meta-Heurísticas com o objetivo de construção de uma solução global.

Em particular, o problema SMTWT, é um NP difícil bem conhecido, e assume uma complexidade bastante elevada quando estamos perante datas de entrega variáveis, pesos positivos e distintos para os atrasos das tarefas (Kolliopoulos e Steiner 2006; Congram, Potts e Velde 2002).

Este problema, SMTWT, tem as seguintes características (OR-Library 1990):

- Um conjunto de n tarefas independentes (1,...,n) deve ser processado, sem interrupções, numa única máquina.
- A máquina processa as tarefas de forma sequencial e uma de cada vez.
- Cada tarefa, j (j=1,...,n), está disponível a ser processada no instante de tempo t=0 e requer um tempo de processamento p_j, sem interrupções, tem um peso positivo w_j e uma data de entrega d_j que, idealmente, deve ser respeitada.
- Para uma determinada ordem de processamento das tarefas C_j representa o instante em que a tarefa, j, foi completada e $T_j = \max\{C_j - d_j, 0\}$ representa o seu atraso.
- O problema consiste em minimizar o atraso total ponderado: $\sum_1^n w_j T_j$

Assim, da resolução deste tipo de problemas, para além da própria ordem pela qual as tarefas serão executadas, são obtidos indicadores como o atraso total ponderado (que se pretende minimizar) e a quantidade de tarefas com atraso. Este tipo de informação tem grande relevância em contextos industriais.

Estes problemas têm sido abordados das mais variadas formas, tanto na utilização de métodos exatos como não exatos. Aqui, vai-se focar em algumas abordagens e exemplos de abordagens não exatas que têm sido usadas ao longo do tempo, uma vez que é o conjunto de técnicas que se enquadram no trabalho que está a ser apresentado. No entanto, é importante ter em mente que a utilização de métodos exatos é útil na perceção da complexidade do problema e no estabelecimento de *upper bounds*, *lower bounds* e *benchmarks*.

As regras de despacho, ou regras de prioridade, são das abordagens mais antigas e consistem num conjunto de regras, bem definidas, para decidir qual será a tarefa (ou conjunto de tarefas) a ser processada de seguida numa máquina (Madureira 1995). Este tipo de abordagens permite, normalmente em conjunto com modelos de simulação, obter resultados de uma forma muito rápida uma vez que as regras definidas, usualmente, têm complexidades reduzidas. A quantidade e natureza deste tipo de regras é bastante elevada, (Blackstone, Phillips e Hogg 1982), pelo que aqui são apresentados dois desses casos a título de exemplo (Madureira 1995; Blackstone, Phillips e Hogg 1982):

- *First In First Out* (FIFO): consiste na realização das tarefas pela ordem em que estão disponíveis.
- *Earliest Due Date* (EDD): a prioridade é dada às tarefas cujas datas de entrega estão mais próximas.

Sendo muito simples e de cariz "guloso", este tipo de regras é, geralmente, de natureza míope, o que representa uma grande desvantagem para modelos que usem apenas este tipo de abordagens (Pereira 2014).

No que diz respeito a técnicas mais ricas, este problema também já foi abordado das mais variadas formas. Com uma abordagem baseada em colónia de formigas (Madureira, Falcão e Pereira 2012) estuda, para várias combinações de parâmetros da heurística, um conjunto de diferentes instâncias SMTWT. Para um problema SMTWT com datas de lançamento variáveis (Cakar e Koker 2015) apresenta uma abordagem Neuro-híbrida baseada em PSO, conjugada com GA e SA. Através de uma técnica de procura local baseada em programação dinâmica, intitulada de *Dynasearch Algorithm*, (Congram, Potts e Velde 2002) obtém excelentes resultados para as instâncias de teste (OR-Library 1990), que ainda hoje prevalece

como uma das mais bem sucedidas abordagens para o problema, servindo os seus resultados, em muitos casos, de *benchmark* para a comunidade cientifica. Também nas instâncias (OR-Library 1990), (Tasgetiren, Sevkli e Gencyilmaz 2004) apresentam uma heurística PSO que, conjugada com uma técnica chamada *Smallest Postition Value*, consegue resultados bastante interessantes em tempos de resolução bastante baixos.

Capítulo 5

ATEPSA: Auto Tuned Elitist Particle Swarm Algorithm

5.1 Introdução

Como já foi mencionado anteriormente, os parâmetros das (Meta-) Heurísticas têm uma influência significativa no seu desempenho. Por outro lado, a afinação de parâmetros é uma tarefa que pode ser extremamente complexa e demorada. Como foi identificado na Secção 2.3, diferentes técnicas de Aprendizagem Automática têm sido aplicadas à questão da afinação de parâmetros, com resultados interessantes. Este facto serve de incentivo para a exploração de outras técnicas dessa área, que ainda não têm recebido atenção no que diz respeito à sua potencial aplicação no problema da escolha de parâmetros de Meta-Heurísticas.

Assim, neste capítulo, propomos uma nova abordagem integrada de uma heurística com características PSO, uma camada de *Clustering* e outra de Classificação, para afinação de parâmetros de modo *off-line*, aplicada à resolução de problemas de escalonamento de máquina única. A escolha de basear a heurística nos conceitos da técnica PSO deveu-se à sua relativa simplicidade de implementação, aos bons resultados que este tipo de implementações mostra (quando usada da forma conveniente) na literatura, ao seu estudo aprofundado e à aceitação em termos científicos.

A heurística tem estrutura de Meta-Heurística para resolução de problemas de combinatória, no sentido em que só a função objetivo é que é estritamente dependente do tipo de problema. As camadas de *Clustering* e Classificação dependem da natureza do problema. A ideia base desta abordagem é a utilização do histórico de resolução de instâncias do problema, semelhantes à que se quer resolver, para fazer uma seleção inteligente dos parâmetros da heurística.

5.2 Arquitetura

Intitulada de *Auto Tuned Elitist Particle Swarm Algorithm* (ATEPSA), esta abordagem consiste numa estrutura dividida em três módulos:

- **Módulo de Classificação (MCLA):** Classifica a instância do problema a resolver num dos *clusters* de problemas já resolvidos anteriormente.

- **Módulo de Otimização (MO):** Resolve o instância do problema usando uma heurística baseada em PSO para as combinações de parâmetros mais promissoras do *cluster* em que a instância foi incluída.
- **Módulo de Clustering (MCLU):** Atualiza os *clusters* com os dados mais recentes dos resultados para as diferentes instâncias resolvidas.

Os módulos MCLA e MO atuam de forma sequencial em que os resultados do MCLA são usados como parâmetros do MO. Por outro lado o módulo MCLU atua de forma independente. Todos os módulos interagem também com uma base de dados central onde são guardadas informações das instâncias, tanto em termos de estrutura, como em termos de agrupamento.

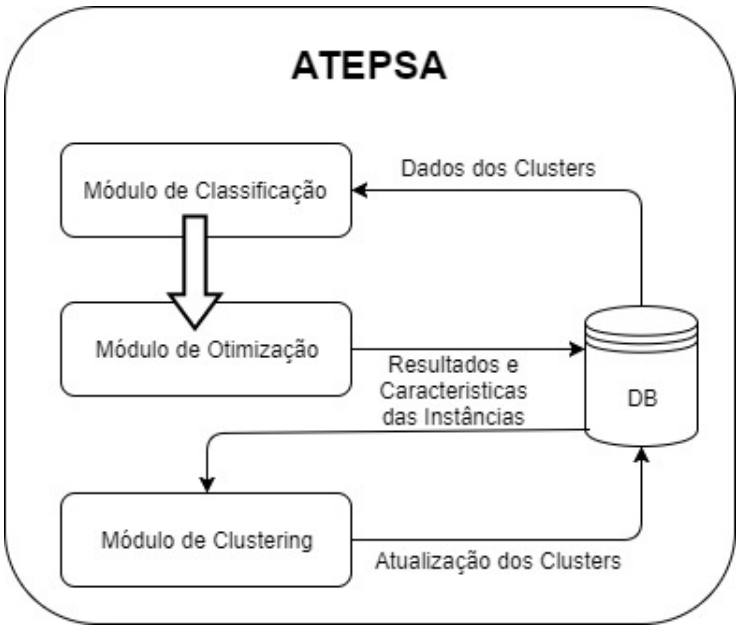

Figura 5.1: Os diferentes módulos do ATEPSA e a interação com a Base de Dados

5.3 Módulo de Classificação

Havendo um histórico de resolução de problemas com diferentes combinações de parâmetros da heurística, este módulo, ao receber uma nova instância, vai classifica-la. Para tal, obtém da base de dados a informação de diferentes *clusters* de instâncias e aplica o método K-NN, com K = 1, para os centroides de cada *cluster*, usando a distância Euclidiana. Determinando o centroide mais próximo, irá classificar a instância no correspondente *cluster*. De seguida, são escolhidas as combinações de parâmetros que, historicamente, obtiveram melhores resultados para as diferentes instâncias que fazem parte do *cluster* escolhido. Essas combinações de parâmetros serão usadas, posteriormente no Módulo de Otimização.

5.4 Módulo de Otimização

Este módulo consiste numa heurística com características PSO, ao qual foi dado o nome de *Elitist Particle Swarm Algorithm* (EPSA), que retorna um objeto, $gBEST$, representativo da melhor solução em termos de valor e de estrutura, recebendo como parâmetros:

- $iNUM\_PART$: Número de partículas a usar.
- $iVmax$: Velocidade máxima das partículas.
- $iITERmax$: Número máximo de iterações.
- $pELIT$: Percentagem de partículas elitistas.
- $fREF\_SOL$: Valor de uma solução de referência (caso exista).
- $oINST$: Objeto correspondente à instância a resolver.

Algorithm 5.1 EPSA

```
1:  Input: iNUM_PART, iVmax, iITERmax, pELIT, fREF_SOL e oINST
2:  Output: gBEST
3:
4:  procedure EPSA(Input)
5:      iITER = 0
6:
7:      particulas = inicializa_particulas(iNUM_PART)
8:      while iITER ≤ iITERmax e gBEST_VAL > fREF_SOL do
9:          for all particulas do
10:             avalia_funcao_obj()
11:         end for
12:         ordena_particulas()
13:         atualiza_velocidades()
14:         atualiza_posicoes()
15:         incrementa iITER
16:     end while
17:     return gBEST
18: end procedure
```

Assim, enquanto nenhum dos critérios de paragem (número máximo de iterações e valor de referência para a solução) for satisfeito, o valor da posição de cada partícula será avaliado consoante a função objetivo, que no caso do problema de escalonamento abordado neste trabalho tem a forma identificada na secção 4.1.1: $\sum_1^n w_j T_j$. Posteriormente, seguem-se a ordenação das partículas, atualização das velocidades e a atualização das posições.

A ordenação das partículas consiste, simplesmente, em dispor as mesmas numa estrutura de lista ordenada por ordem crescente do valor da solução, por forma a que a melhor solução seja o primeiro elemento da lista e a pior seja o último. $gBEST$, é a primeira posição dessa lista.

Na iteração t, a velocidade da partícula, p, na posição $X_{p,t}$ é determinada pela seguinte expressão:

$$V_{p,t} = upper\_int(iVmax \times \frac{val(X_{p,t})}{gWORST\_val}) \tag{5.1}$$

Em que $gWORST\_val$ é o valor da pior solução encontrada na corrente iteração, a função $val()$ retorna o valor da solução e a função $upper\_int()$ arredonda o valor para o primeiro

inteiro superior. Este valor de velocidade representa o número de permutações $P^{i,j}$, entre pares de posições na solução, que serão aplicados à posição da partícula.

Na atualização das posições das partículas é identificado um grupo de soluções que passarão a fazer parte de uma elite de partículas que irão explorar a vizinhança de $gBEST$. Esse grupo é constituído pelas $pELIT$ piores posições na corrente iteração. De seguida, para todas as partículas, exceto a que se encontra em $gBEST$, é aplicada a velocidade, na forma de $V_{p,t}$ permutações $P^{i,j}$. No caso das partículas que fazem parte da elite, para cada uma delas, as permutações são aplicadas sobre uma cópia de $gBEST$, fazendo assim com que a próxima posição destas partículas esteja a evoluir a partir de $gBEST$ e não da posição que estas ocupavam.

Para cada permutação, os pares (i, j) são escolhidos de forma aleatória. Seja $O_{p,t}$ um operador que representa as $V_{p,t}$ permutações $P^{i,j}$ a serem aplicadas na posição da partícula p na iteração t. A posição da partícula, p, é atualizada da seguinte forma:

$$X_{p,t+1} = O_{p,t} X_{p,t} \tag{5.2}$$

Para o caso das partículas pertencentes ao grupo de elite temos:

$$X_{p,t+1} = O_{p,t} gBEST \tag{5.3}$$

5.5 Módulo de Clustering

Como mencionado anteriormente, este módulo é completamente dependente das características do problema. Por isso, nesta secção, iremos focar o funcionamento deste módulo para o caso do problema SMTWT, sendo que serão usados vários temos específicos desse contexto. A partir do histórico de resolução dos problemas para diferentes combinações de parâmetros as instâncias são agrupadas usando o método *Affinity Propagation*. A utilização deste método, *Affinity Propagation*, deve-se ao facto de ser uma técnica que não recebe o número de *clusters* a serem formados, o que representa uma mais valia em casos em que não é clara a separação entre os objetos, como é o caso do problema que pretendemos abordar. Para este agrupamento é usada, como função de afinidade, o simétrico do quadrado da distância euclidiana para duas dimensões.

$$Af_{(i,j)} = -[(X_j - X_i)^2 + (Y_j - Y_i)^2] \tag{5.4}$$

Em que (i, j) representa um par de instâncias do problema, X_i é o valor normalizado do Coeficiente de Variação (CV) das datas de entrega da instância i e Y_i é o valor normalizado do logaritmo natural do quociente entre o valor total do processamento das tarefas e o valor médio das datas de entrega. A normalização foi feita usando a formula de *rescaling*:

$$Z_i = \frac{Z'_i - min(Z)}{max(Z) - min(Z)} \tag{5.5}$$

Em que Z_i representa o valor normalizado e Z'_i representa o valor não normalizado. Assim, sejam X'_i e Y'_i os valores, não normalizados, das quantidades X_i e Y_i referentes a uma instância i do problema SMTWT:

$$X_i = \frac{\sigma_i^d}{\overline{Dd_i}} \tag{5.6}$$

$$Y_i = log(\frac{Tpt_i}{\overline{Dd_i}}) \tag{5.7}$$

Em que σ_i^d e $\overline{Dd_i}$ são, respetivamente, o desvio padrão e o valor médio das datas de entrega das tarefas da instância i e Tpt_i é o tempo total de processamento das tarefas da instância i.

Após a criação dos *clusters* de instâncias com o método mencionado, para cada *cluster* são guardados o centroide e as combinações de parâmetros que obtiveram melhores resultados de cada instância desse *cluster*. Essa informação é guardada na base de dados para posteriormente ser usada no MCLA.

A utilização desta estratégia, guardando tuplos de combinações de parâmetros e estabelecendo relações de proximidade entre instâncias a resolver e as já usadas, para usar as combinações mais promissoras, permite manter as propriedades das combinações dos parâmetros como um todo. Ao passo que, por exemplo, uma estratégia que permita determinar os parâmetros através da utilização de regressões, para cada um, poderá estar a ignorar as dependências de parâmetros e o potencial dos parâmetros como um todo e não individualmente.

5.6 Implementação

A implementação foi feita na linguagem de programação Python v3.6 para todos os módulos com a arquitetura do esquema da Figura 5.1.

```
class Particle:
    def __init__(self, nJobs, shuffle = True):
        self.positionValue = 0
        self.bestPositionValue = None

        self.position = [x for x in range(nJobs)]
        self.bestPosition = None

        self.velocity = 0.0

        self.signedCosts = [0] * nJobs

        if(shuffle):
           random.shuffle(self.position)

    def get_positionValue(self):
        return self.positionValue
    def set_positionValue(self, value):
        self.positionValue = value

    def get_position(self):
        return self.position
    def set_position(self, position):
        self.position = position
```

Figura 5.2: Exemplo de implementação da classe partícula.

A Figura 5.2 exemplifica parte da implementação da classe "partícula", com vários atributos e métodos, usada na heurística EPSA.

O Módulo de Classificação e o Módulo de Otimização foram ambos implementados completamente de raiz, usando apenas bibliotecas standard de python. O Módulo de Clustering foi implementado com recurso à biblioteca (Sklearn s.d.). Esta biblioteca foi escolhida por conter as ferramentas que foram identificadas como necessárias e por ser amplamente usada na comunidade científica ao longo dos últimos anos (Feurer et al. 2015; Pedregosa et al. 2012).

A base de dados consistiu apenas em ficheiros do tipo .csv por dois motivos: porque os dados das instâncias de origem já se encontravam nesse formato e para agilizar a posterior análise dos resultados, uma vez que a incorporação de ficheiros do tipo .csv em excel é trivial.

Capítulo 6

Análise de Resultados

6.1 Introdução

Os resultados aqui apresentados referem-se à aplicação do ATEPSA a um conjunto de instâncias do problema SMTWT, comummente usadas para *benchmarking*, presentes em (OR-Library 1990).

Inicialmente o ATEPSA foi treinado, sem o uso do MCLA e, posteriormente, foi usado na sua íntegra. Neste contexto o treino consistiu na resolução de vários problemas para diferentes combinações de parâmetros e na criação de grupos de instâncias e respetivas combinações de parâmetros consideradas promissoras.

Uma vez que a heurística EPSA usada no MO apresenta uma combinação de características que não foi possível identificar na literatura, para além da análise dos resultados do ATEPSA como um todo, iremos também apresentar uma análise do desempenho do EPSA.

6.2 Instâncias ORLIB e gamas de parâmetros usadas

As instâncias usadas neste trabalho estão presentes em (OR-Library 1990) e, de forma sucinta, têm as seguintes características:

- As instâncias estão divididas em 3 grupos de 40, 50 e 100 tarefas.
- Em cada grupo existem 125 instâncias diferentes.
- Para cada instância, de cada grupo, estão disponíveis os tempos de processamento, p, as datas de entrega, d, e os pesos, w, de cada tarefa.
- Para todas as instâncias existe um valor de referência para a melhor solução.
- Estão disponíveis os valores ótimos de 124 instâncias de 40 tarefas e 115 de 50 tarefas. Os valores de referência das instâncias de 40 e 50 tarefas que não foram provados como ótimos não foram melhorados desde 1996 e, provavelmente, referem-se ao valor ótimo (OR-Library 1990).

No total foram usadas 125 instâncias, 80 de 40 tarefas e 45 de 50 tarefas, escolhidas de forma aleatória. Para este estudo decidiu-se limitar as combinações de parâmetros da seguinte forma:

- $iITERmax$: Fixo e igual a 500.

- $iNUM\_PART$: Variável em (5,6,7,8,9,10).
- $iVmax$: Variável em (1,2,3,4,5,6,7,8,9,10).
- $pELIT$: Variável em (0.0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0)

Isto representa um total de 420 combinações diferentes por instância. O intervalo de número de partículas e o valor fixo de máximo de 500 iterações foram escolhidos, após alguns testes à eficiência do EPSA, por forma a que a heurística resolvesse cada instância em tempos aproximadamente inferiores a 1 segundo, para que fosse possível a criação de um *data set* com uma quantidade significativa de dados em tempo útil. Também em testes iniciais a diferentes valores da velocidade máxima, $iVmax$, foi detetado que, sistematicamente, valores de $iVmax > 10$ apresentavam resultados consideravelmente piores do que os restantes. Daí o espectro escolhido para $iVmax$. Na verdade, este comportamento era expectável, uma vez que estando o valor da velocidade a representar o número de permutações aplicadas à posição de cada partícula, ao permitir valores de velocidade com ordens de grandeza comparáveis ao número de tarefas, estamos a baralhar as soluções de tal forma que não beneficia a convergência, uma vez que as características das soluções mais promissoras numa dada iteração são, em grande parte, perdidas na iteração seguinte devido ao elevado número de permutações. Os valores de percentagem de partículas elitistas, $pELIT$, foram escolhidos com o objetivo de percorrer a gama [0.0, 1.0] em intervalos aproximadamente uniformes.

Para cada combinação de parâmetros e para cada instância, foram feitas 20 repetições da heurística para também se poder inferir sobre a sua precisão e para estudar de uma forma mais consistente a validade dos resultados obtidos. Este valor foi escolhido por forma a ser ordens de grandeza superior a 1, tendo assim significado estatístico, e por já ter sido usado anteriormente, na literatura, para o mesmo grupo de instâncias (Madureira, Falcão e Pereira 2012). No total, representa mais de 1 milhão de corridas da heurística, o que permite criar um *dataset* com bastante informação para ser explorada nas gamas de valores dos parâmetros escolhidas.

Para cada corrida da heurística foi guardada a instância, a combinação de parâmetros usada, o tempo de resolução, o número de iterações e o resultado da melhor solução encontrada:

instance_code	maxVelocity	numIterations	numParticles	elitists	runningTime	bestResult
113_40	2	500	10	0.8	0.62	20097
113_40	2	500	10	1	0.64	20162
113_40	2	500	10	0.6	0.72	19934

Figura 6.1: Exemplo do output da heurística.

Para a obtenção dos resultados em tempo útil, as corridas da heurística foram efetuadas em duas máquinas diferentes:

- Intel(R) Core(TM) i7 2.7GHz, 8GB DDR3, Windows 10
- Intel(R) Core(TM) i5 2.5GHz, 4GB DDR3, Windows 10

Por isso, o estudo foi focado na qualidade das soluções e não nos tempos de resolução. No entanto foi tido o cuidado de escolher combinações de parâmetros que garantem tempos de resolução na casa do segundo, como explicado anteriormente.

6.3 Resultados EPSA

Após a obtenção dos resultados, estes foram comparados, em termos da melhor solução encontrada para cada instância, com as soluções de referência (OR-Library 1990). Essa comparação foi feita mediante o cálculo da percentagem de desvio relativo:

$$\Delta_{si} = \frac{|Sol_{si} - SolR_i|}{SolR_i} \times 100 \tag{6.1}$$

Em que Δ_{si} é o valor da percentagem do desvio relativo da solução s para a instância i, Sol_{si}, em relação à solução de referência da instância i, $SolR_i$.

Para as instâncias de 40 tarefas os resultados foram os seguintes:

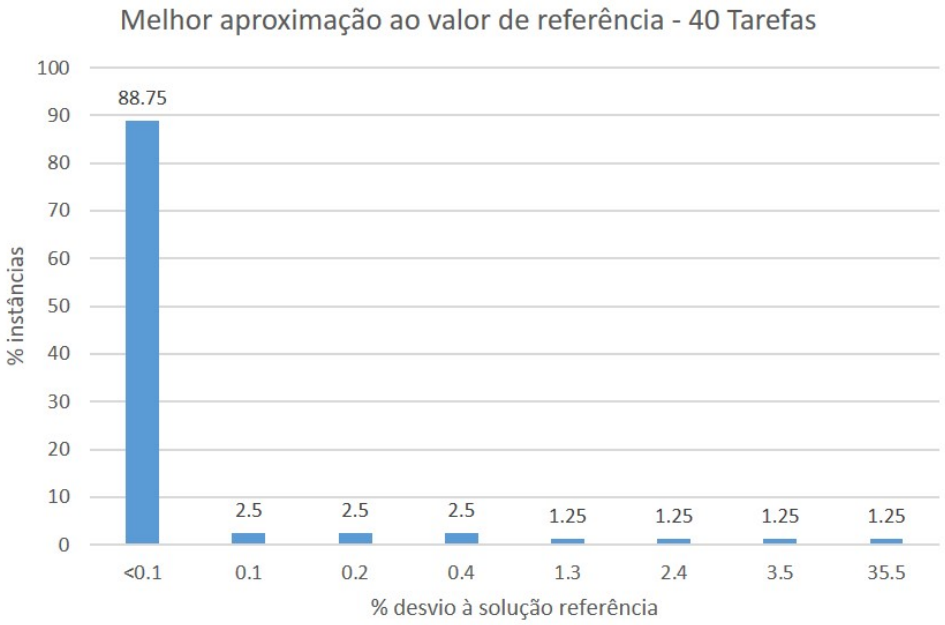

Figura 6.2: Melhor aproximação aos valores de referência - 40 tarefas.

Isto significa que, para 71 instâncias (88,75%) de 40 tarefas, foi possível obter uma solução com um desvio inferior a 0.1% da solução de referência. Destas, 64 igualaram a solução de referência (ver tabela em A.2). No entanto, foi verificada uma instância (1,25%) em que o melhor resultado ficou bastante longe da solução de referência (35,5%).

No caso das instâncias de 50 tarefas, os resultados foram os seguintes:

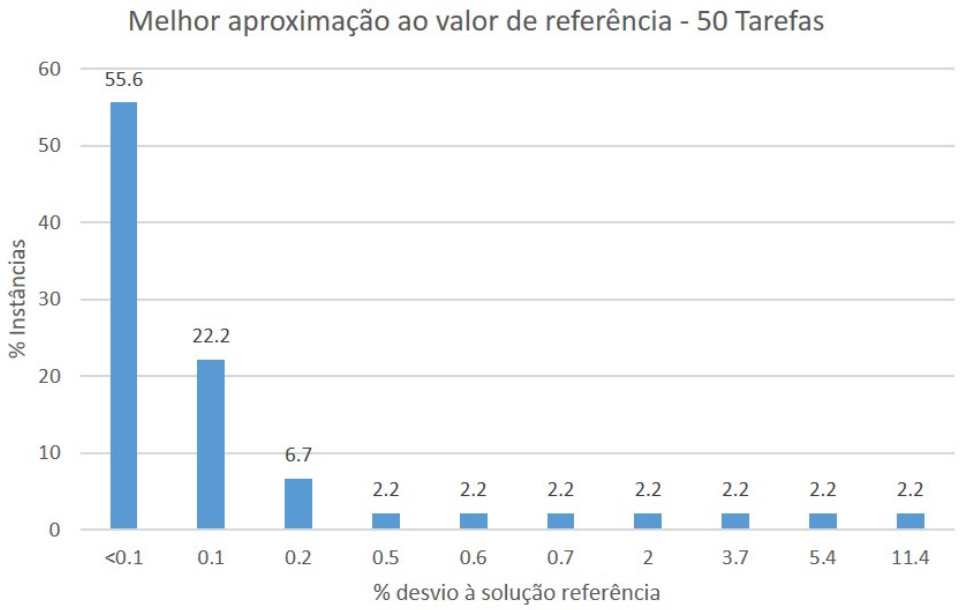

Figura 6.3: Melhor aproximação aos valores de referência - 50 tarefas.

Neste caso, para 25 instâncias (aprox 55,6%) foi possível chegar a um desvio inferior a 0.1% da solução de referência. Destas, 19 igualaram a melhor solução de referência (ver tabela em A.1). Analogamente ao caso de 40 tarefas também foi possível identificar uma instância (2,2%) em que o desvio relativamente à solução de referência foi significativo (11,4%).

A comparação destes resultados sugere uma maior dificuldade da heurística EPSA em se aproximar da solução de referência no caso de 50 tarefas. Esse comportamento é expectável, uma vez que um maior número de tarefas reflete-se num aumento do número de combinações possíveis e, consequentemente, numa maior dificuldade de encontrar as melhores soluções para um número de iterações fixo. No entanto, este indicador, por sí só, não permite tirar ilações significativas.

Os valores refletidos nos gráficos anteriores referem-se apenas às melhores soluções encontradas pela heurística. O facto de uma corrida a heurística ter chegado ao valor de referência, para uma determinada combinação de parâmetros, não significa que em todas as corridas para a mesma combinação de parâmetros também chegue. Daí que se tenha feito não só uma, mas sim 20 corridas da heurística por combinação de parâmetros e instância. Assim, considerando as 20 corridas para cada instância e combinação de parâmetros, foi determinada a média dos desvios em relação à solução de referência e o coeficiente de variação das soluções obtidas.

Identificadas as combinações de parâmetros que obtiveram melhor média de desvios e fazendo a média das médias dos desvios, para 40 e 50 tarefas obtém-se o seguinte resultado:

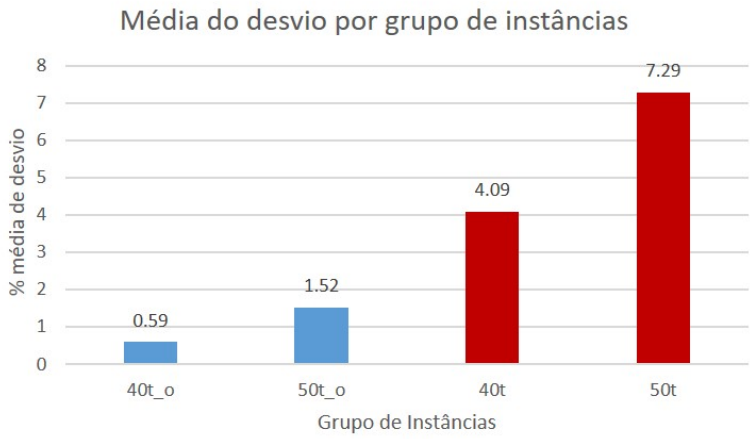

Figura 6.4: Valores médios das médias dos desvios para 40 e 50 tarefas

As barras a azul representam a média das médias dos desvios, para 40 e 50 tarefas sem considerar *outliers*, ao passo que as barras vermelhas representam os valores considerando os *outliers*.

No caso das instâncias de 40 tarefas foram identificados 11 *outliers* (13,75% das instâncias) e no caso das instâncias de 50 tarefas foram identificados 5 *outliers* (12,5% das instâncias). Os *outliers* foram identificados usando o método de Turkey para k=3, ou seja, aqueles fora do intervalo $[Q_1 - 3IIQ, Q_3 + 3IIQ]$ em que Q_1 é o primeiro quartil, Q_3 é o terceiro quartil e $IIQ = Q_3 - Q_1$ é o intervalo interquartil. Assim, podemos considerar que, em média, a heurística se aproximou bastante dos valores de referência (<2%) em ambos os casos de 40 e 50 tarefas. No entanto, há instâncias que se mostram consideravelmente mais difíceis de resolver, que justificam a discrepância entre as barras azuis e as vermelhas, e que neste caso identificamos como *outliers*. Para as mesmas combinações de parâmetros que obtiveram melhores médias de desvios, determinando as médias dos coeficientes de variação, obtém-se o seguinte gráfico.

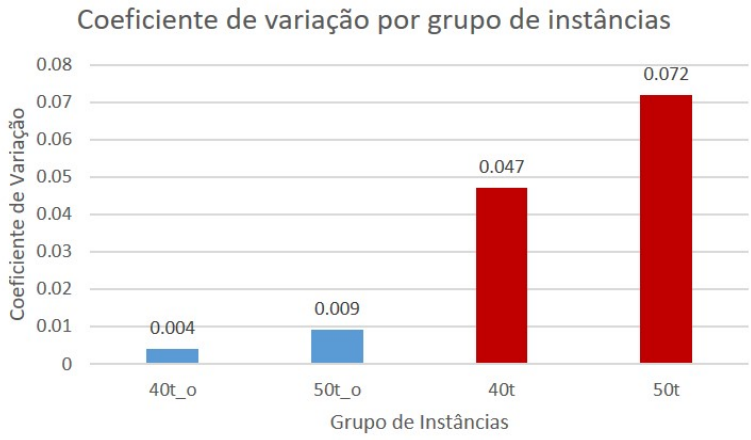

Figura 6.5: Médias dos coeficientes de variação para 40 e 50 tarefas

Na elaboração deste gráfico seguiu-se o mesmo raciocínio do anterior em que, para as barras

a azul, foram excluídos os mesmos *outliers* identificados para o caso anterior. É possível concluir que, em média, a heurística consegue atingir uma variabilidade baixa (coeficiente de variação < 0,01) para ambos os casos de 40 e 50 tarefas, com a ressalva dos *outliers*.

Um comportamento deste tipo é importante para garantir a consistência dos resultados obtidos pelo método de resolução.

É importante notar que a heurística demorou um tempo médio de 0,641s em cada corrida, considerando todas as corridas, independentemente dos parâmetros. Ao considerar apenas as corridas que obtiveram os melhores resultados para cada instância, o tempo médio de resolução ficou-se pelos 0,389s. No entanto, é importante referir que o tempo de execução, neste caso, deve ser visto como mera indicação de ordens de grandeza, uma vez que o facto de terem sido usadas duas máquinas diferentes para realizar as corridas da heurística impossibilita uma análise rigorosa destes tempos.

Como referido anteriormente, o espectro do número de partículas e o número máximo de iterações foram ambos escolhidos precisamente com o propósito de a heurística correr em menos de 1s. No entanto, essa rapidez tem impacto na qualidade das soluções. Os estudos efetuados nesta secção foram focados nos melhores resultados para as melhores combinações de parâmetros encontradas (entre todas as testadas) para cada instância. Na secção seguinte vamos apresentar os resultados da estratégia proposta para permitir a identificação prévia de combinações de parâmetros promissoras.

6.4 Resultados ATEPSA

Do total das 125 instâncias foram escolhidas 30 instâncias de 40 tarefas e 25 instâncias de 30 tarefas, de forma aleatória, para criar um conjunto de treino composto por 55 instâncias. Assim, foram estudadas várias características destas instâncias e analisadas as potenciais relações com a seleção de parâmetros que obteve melhores resultados para cada umas delas. Três características importantes foram identificadas e, posteriormente, analisadas:

1. Para todos casos testados, os valores de 9 e 10 partículas estão sempre entre o top 3 de combinações de parâmetros que obtiveram melhores resultados, para todas as instâncias.
2. Há um conjunto de instâncias (neste caso foram identificadas 7) para as quais há um elevado número de combinações diferentes que conseguiram atingir a melhor solução de referência.
3. 70% das instâncias obteve as melhores soluções para pares de velocidade máxima e percentagem de partículas elitistas $(2, 1.0)$ ou $(2, 0.8)$.

Em relação ao ponto 1, este comportamento era expectável no sentido em que um maior número de partículas significa um maior número de soluções a serem avaliadas a cada iteração da heurística e, consequentemente, uma maior probabilidade de melhorias das soluções. Assim, no estudo que se segue, deixou-se cair a variável número de partículas e concentrou-se apenas nos pares de velocidade máxima e percentagem de partículas elitistas ($iVmax$, $pELIT$). No entanto, para um caso de estudo em que o tempo computacional esteja a ser avaliado, o parâmetro de número de partículas é importante. Essa importância advém do facto de este algoritmo ordenar as partículas por posição a cada iteração, e essa ordenação

não tem tempo linear. No caso implementado foi usada a técnica de ordenação *QuickSort* que tem uma complexidade média de $O(nlogn)$.

O comportamento identificado no ponto 2 deve-se ao facto de haver instâncias que apresentam um conjunto de tarefas para as quais a data de entrega excede o tempo total de processamento de todas as tarefas. Assim, como no problema que estamos a tratar, SMTWT, as tarefas são todas efetuadas, sem interrupções, a partir do instante $t = 0$. A existência de uma tarefa, i, para a qual a data de entrega, d_i, é tal que $d_i \geq \sum_{j=1}^{n} p_j$, em que n é o número de j tarefas, significa que, independentemente da ordem em que i será processada, o instante em que a tarefa é completada, C_i, é tal que $d_i \geq C_i$, logo, pela expressão do atraso, $T_i = \max\{C_i - d_i, 0\}$, obtemos $T_i = 0$. Ou seja, cada tarefa i, para a qual a data de entrega é superior ao tempo total de processamento das tarefas, terá um atraso nulo independentemente da ordem em que for processada, contribuindo com um custo nulo para a função objetivo $\sum_1^n w_j T_j$.

Seja, novamente, i, uma tarefa para a qual $T_i = 0$, independentemente da ordem em que é processada, e $S = (j_1, ..., i, ..., j_n)$ uma solução com custo k_s. Movendo apenas i para a última posição, o custo da solução $S' = (j_1, ..., j_{n-1}, i)$, será tal que $k_s \geq k_{s'}$, porque em S', i não só tem custo nulo, mas também não contribui para o aumento do custo de nenhuma das outras tarefas. Logo, uma tarefa com data de entrega igual ou superior à soma dos tempos de processamento de todas as tarefas, pode ficar sempre no fim, independentemente de todo o resto. Por outro lado, se temos m tarefas, com $T_i = 0$ independentemente da ordem, podemos afirmar que haverá soluções ótimas com a forma $S_o = (j_1, ..., j_{n-m}, i_1, ..., i_m)$ em que cada i representa uma tarefa com $T_i = 0$ independentemente da ordem. Mais do que isso, podemos ainda afirmar que o valor ótimo terá uma degenerescência que é, no mínimo, $m!$ que corresponde a todas as trocas das m tarefas do tipo i nas últimas m posições. Ou seja, a existência de m tarefas deste tipo, pode diminuir o número de soluções possíveis do problema de $n!$ para $(n - m)!$, se houver um posicionamento prévio das m tarefas sem custo, independentemente da ordem, no fim. Esta multiplicidade de soluções ótimas (ou soluções próximas da ótima) faz com que a heurística tenha maior facilidade (em termos de probabilidade) de as encontrar e, consequentemente, dependa menos de uma afinação de parâmetros adequada.

Assim, todas as instâncias que apresentaram estas características foram excluídas porque não contribuem para a aprendizagem, uma vez que não servem para diferenciar as combinações de parâmetros quanto à sua eficácia. No entanto, num estudo focado em tempos de resolução, a identificação deste tipo de características nas instâncias, em termos de aprendizagem, poderá ser importante para, por exemplo, definir critérios de paragem adequados.

Focando agora no terceiro ponto, foi possível encontrar a relação presente no gráfico seguinte:

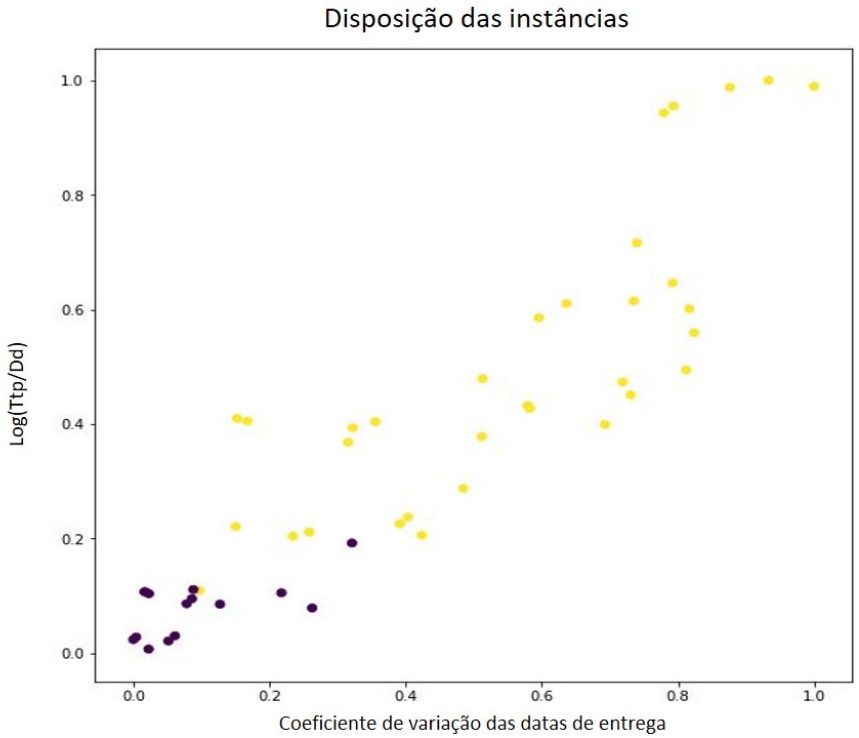

Figura 6.6: Disposição das instâncias consoante a relação entre CV_{Dd} e $log(\frac{T_{tp}}{Dd})$

Este gráfico mostra a relação entre o valor normalizado do CV das datas de entrega e o valor normalizado do logaritmo natural do quociente entre o valor total do processamento das tarefas e o valor médio das datas de entrega. A amarelo estão as instâncias que tiveram $(2, 1.0)$ ou $(2, 0.8)$ como par $(iVmax, pELIT)$ que obteve os melhores resultados. Assim, é possível visualizar uma clara diferença na disposição entre as instâncias que pertencem a esse grupo e as que não pertencem. Por outro lado e analisando não só a melhor solução mas as 3 melhores soluções de cada instância foi possível verificar que, quanto mais perto da origem do gráfico, menor o número de vezes que um dos pares $(iVmax, pELIT)$ mencionados anteriormente faz parte desse grupo. Por outro lado, quanto mais afastado da origem do gráfico, maior a tendência para esses pares serem dominantes.

Assim, não sendo possível estabelecer uma fronteira, mas sendo clara a existência de uma relação entre a proximidade das instâncias (na figura 6.6) e as combinações de parâmetros que obtiveram melhores resultados, foi utilizado o método de *clustering*, *Affinity Propagation*, da forma descrita na secção 5.5 para agrupar as instâncias, com o seguinte resultado.

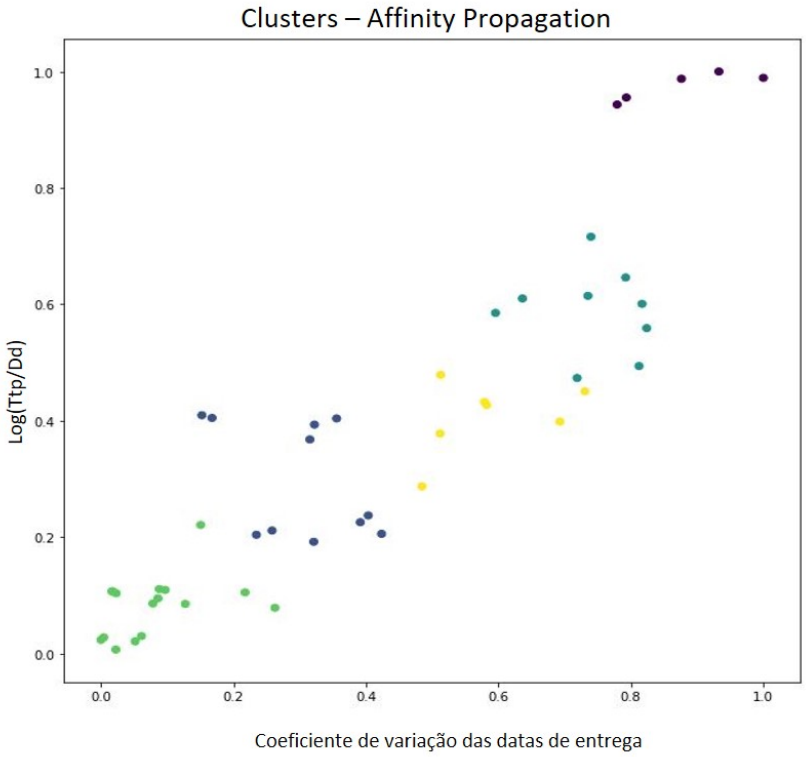

Figura 6.7: *Clusters* criados pelo método *Affinity Propagation*

Este método originou 5 *clusters* em que é clara a separação por distância à origem, como já sugeria a Figura 6.6. Por outro lado, o grupo de instâncias que não têm $(2, 1.0)$ ou $(2, 0.8)$ na melhor combinação de parâmetros foi incluído quase na totalidade em um só *cluster*.

Para cada *cluster* foram guardados o seu centroide e as melhores 3 combinações de parâmetros de cada instância pertencente ao *cluster*.

As restantes instâncias, que não foram utilizadas para a formação dos *clusters*, foram usadas como grupo de teste com o método de classificação descrito na secção 5.3.

Ao correr o modelo de Classificação, após a determinação do *cluster* de uma determinada instância, escolheram-se 5 combinações de parâmetros das mais promissoras para esse *cluster*. Os resultados foram cruzados com as corridas feitas previamente para todas as combinações de parâmetros, tendo-se verificado que em aproximadamente 89% das instâncias, pelo menos uma das 5 combinações escolhidas fazia parte do top 3 das combinações que tinham obtido melhores resultados. Por outro lado, em aproximadamente 79% das instâncias, todas as combinações do top 3 estavam entre as 5 selecionadas pelo modelo de classificação.

Para este grupo de teste também foi estudada a disposição das instâncias, tal como para o grupo de instâncias de treino, tendo-se obtido um gráfico em muito semelhante ao da figura 6.6.

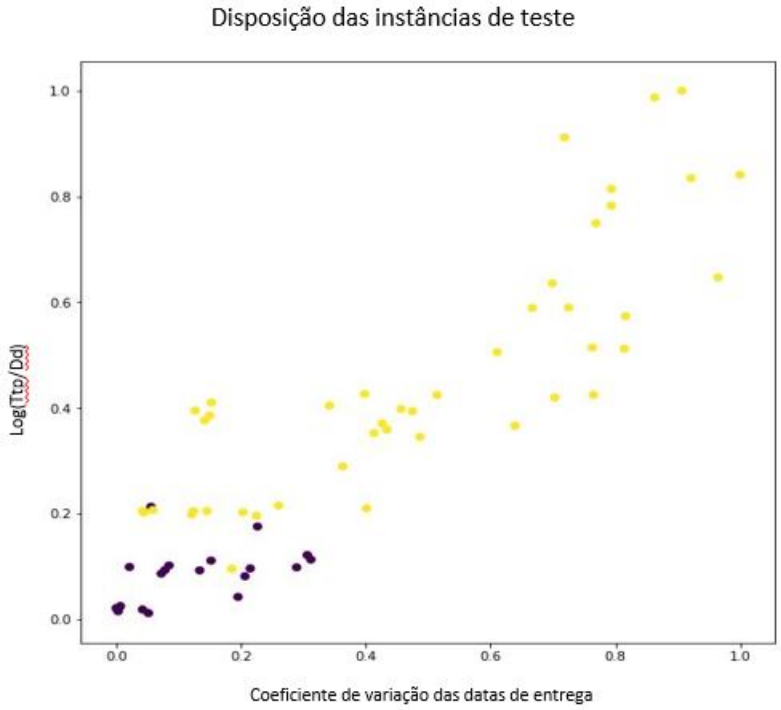

Figura 6.8: Disposição das instâncias de teste consoante a relação entre CV_{Dd} e $log(\frac{T_{tp}}{Dd})$

Cruzando os resultados da classificação com este gráfico, foi possível verificar que todas as instâncias para as quais o modelo de classificação falhou na escolha dos parâmetros, são instâncias da zona perto da origem (a roxo). Por outro lado e olhando um pouco mais em detalhe para este comportamento das instâncias, esta zona perto da origem do gráfico, expressa duas características em simultâneo:

- Se $log(\frac{T_{tp}}{Dd})$ está próximo de 0 por valores superiores então $\frac{T_{tp}}{Dd}$ estará próximo de 1 por valores superiores. O que quer dizer que, em média, as datas de entrega se aproximam do tempo total de processamento das tarefas por valores inferiores.
- Se o coeficiente de variação das datas de entrega,CV_{Dd}, é próximo de zero, então a variabilidade das datas de entrega é baixa.

Estas duas características sugerem que, nestes casos, tenhamos uma elevada percentagem de entrega próximas do tempo total de processamento, por valores inferiores. Assim, podemos estar uma situação especial, tal como a verificada anteriormente para os casos em que havia datas de entrega que eram superiores ao tempo total de processamento das tarefas. No entanto, não foi possível aprofundar mais o estudo para estes casos.

A abordagem tomada e a heurística implementada permitiram chegar a resultados bastante satisfatórios no que diz respeito à aproximação dos valores de referência para as instâncias do problema resolvidas. Este facto ganha ainda mais peso quando a média de tempos de resolução da heurística foi 0,389s para os melhores casos de cada instância e 0,641s considerando todas as corridas com todas as combinações de parâmetros. No entanto, como referido anteriormente, estes valores dos tempos devem ser vistos apenas como indicadores de ordem de grandeza, uma vez que foram utilizadas duas máquinas com capacidades de processamento diferentes para obtenção dos resultados.

Em relação à abordagem integrada com aprendizagem, o trabalho realizado permitiu identificar as combinações de parâmetros mais promissoras para a grande maioria dos casos, tendo ficado ainda questões em aberto relacionadas com as instâncias que se encontram perto da origem dos gráficos das figuras 6.6 e 6.8.

Assim, podemos concluir que, tanto a heurística EPSA como o conjunto ATEPSA apresentam resultados promissores.

Capítulo 7

Conclusão

Deste trabalho, motivado pela complexidade da afinação de parâmetros das Meta-Heurísticas resultou, não só um método novo de abordar o problema, mas também uma nova heurística com uma combinação de características própria.

A heurística implementada, EPSA, baseada em PSO, não só apresenta grande simplicidade em termos conceptuais como consegue atingir resultados bastante satisfatórios, em tempos de resolução relativamente baixos, quando aplicada ao problema *Single Machine Total Weighted Tardiness*. Em 66,4% das instâncias resolvidas a heurística conseguiu chegar à melhor solução conhecida, sendo que a média dos tempos de resolução se ficou pelos 0,641s, independentemente dos parâmetros usados, e 0,389s para os melhores casos de cada instância. Por outro lado, também apresentou níveis de variabilidade bastante baixos para a maioria dos casos testados, fator este que, de certa forma, transmite o nível de consistência dos resultados apresentados pela heurística EPSA.

Em particular, no que diz respeito aos tempos mencionados, estes devem ser vistos apenas como indicadores de ordem de grandeza, uma vez que foram utilizadas duas máquinas com capacidades de processamento diferentes para obtenção dos resultados. Facto este que não permitiu um estudo rigoroso da eficiência.

A simplicidade desta heurística e o facto de apenas a sua função objetivo ser estritamente dependente do problema a resolver, abre caminho para que possa ser facilmente explorado o seu comportamento na resolução de outros problemas de otimização combinatória como por exemplo o problema do caixeiro viajante ou até problemas de uma complexidade superior. Estas características motivam também a exploração de variantes desta heurística com o intuito de melhorar a sua performance e de a tornar mais robusta.

No que diz respeito ao método ATEPSA de integração da heurística com as camadas de *Clustering* e Classificação, os resultados também são promissores. Para um conjunto de casos, 89%, este método foi capaz de identificar corretamente uma das melhores 3 combinações de parâmetros a usar. No entanto, nos restantes casos falhou de forma categórica. Esses casos foram identificados como tendo características semelhantes em termos de estrutura das instâncias.

Não tendo sido identificada na literatura nenhuma abordagem ao problema de afinação de parâmetros que use esta combinação de técnicas, este método representa também um primeiro estudo sobre a viabilidade destas abordagens para o problema em questão. Sendo de salientar que, uma das suas principais características é facto de tratar as combinações de parâmetros como um todo e não parâmetro a parâmetro, ao contrário de várias abordagens identificadas na literatura. Isto permite manter as propriedades dos parâmetros como um

todo, não perdendo assim a informação inerente a questões como a dependência entre parâmetros.

Os resultados positivos servem de motivação para futuramente serem testadas outras abordagens desta natureza, bem como para dotar o ATEPSA de maior sofisticação por forma a preencher as lacunas identificadas na previsão de determinados casos. Por outro lado, a aplicação a outros tipos de problemas e a própria integração de várias heurísticas distintas com a camada de aprendizagem é também algo que será interessante explorar.

Bibliografia

Alabas-Uslu, Cigdem e Berna Dengiz (2011). «A self-adaptive local search algorithm for the classical vehicle routing problem». Em: *Expert Systems with Applications* 38.7, pp. 8990–8998. issn: 09574174. doi: 10.1016/j.eswa.2011.01.116. url: http://dx.doi.org/10.1016/j.eswa.2011.01.116.

Alpaydin, Ethen (2004). *Introduction to Machine Learning Second Edition*.

Angelo, Jaqueline S, Douglas A Augusto e Helio J C Barbosa (2013). «Strategies for Parallel Ant Colony Optimization on Graphics Processing Units». Em: *Intech*.

Barbosa, Eduardo e Edson Senne (2017). «Improving the Fine-Tuning of Metaheuristics: An Approach Combining Design of Experiments and Racing Algorithms». Em: *Journal of Optimization* 2017, pp. 1–7. issn: 2356-752X. doi: 10.1155/2017/8042436. url: https://www.hindawi.com/journals/jopti/2017/8042436/.

Birattari, Mauro (2009). *Tuning Metaheuristics - A Machine Learning Perspective*.

Birattari, Mauro et al. (2001). «Classification of metaheuristics and design of experiments for the analysis of components». Em: *Techn. Rep., N AIDA-01-05, Techn. Univ. Darmstadt, Darmstadt* March.

Blackstone, John H., Don T. Phillips e Gary L. Hogg (1982). «A state-of-the-art survey of dispatching rules for manufacturing job shop operations». Em: *International Journal of Production Research* 20.1, pp. 27–45. issn: 0020-7543. doi: 10.1080/00207548208947745. url: http://www.tandfonline.com/doi/abs/10.1080/00207548208947745.

Bonabeau, Eric, Marco Dorigo e Guy Theraulaz (1999). *Swarm Intelligence From Natural to Artificial Systems*. Vol. 64. 6. Oxford: Oxford University Press. arXiv: arXiv:1011.1669v3.

Cakar, Tarik e Rasit Koker (2015). «Problem with Unequal Release Date Using Neurohybrid Particle Swarm Optimization Approach». Em: 2015. issn: 1687-5265. doi: 10.1155/2015/838925.

Carvalho, Samuel, Ana Rodrigues e José Soeiro (2014). «Combination of ant colony optimization and exact methods applied to routing problems». Em: *Optimization 2014 Conference BOOK OF ABSTRACTS*, p. 46.

Clarke, G e J W Wright (1964). «SCHEDULING OF VEHICLES FROM A CENTRAL DEPOT TO A NUMBER OF DELIVERY POINTS». Em: *Operations Research* 12.4, pp. 568–581.

Congram, Richard K., Chris N. Potts e Steef L. van de Velde (2002). «An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem». Em: *INFORMS Journal on Computing* 14.1, pp. 52–67. issn: 1091-9856. doi: 10.1287/ijoc.14.1.52.7712. url: http://pubsonline.informs.org/doi/abs/10.1287/ijoc.14.1.52.7712.

Dietterich, Thomas G. (2009). *Machine Learning*, pp. 8–13. isbn: 9781577354260. doi: 10.1145/242224.242229. arXiv: 0-387-31073-8.

Dobslaw, Felix (2010). «A Parameter Tuning Framework for Metaheuristics Based on Design of Experiments and Artificial Neural Networks». Em: *Proceeding of the International Conference on Computer Mathematics and Natural Computing*, pp. 1–4. url: http://swepub.kb.se/bib/swepub:oai:DiVA.org:miun-11420?tab2=abs%7B%5C&%7Dlanguage=en.

Dorigo, Marco e Vittorio Maniezzo (1996). «Ant System: Optimization by a Colony of Cooperating Agents». Em: *IEEE Transactions on Systems, Man, and Cybernetics–Part B, Vol.26, No.1, 1996, pp.1-13* 26.1, pp. 1–13. url: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=484436%7B%5C&%7Dtag=1.

Dorigo, Marco e Thomas Stutzle (2006). *Ant colony optimization*. Ed. por MIT Press. Massachusetts. isbn: 1556-603X. doi: 10.1109/MCI.2006.329691.

Dueck, D e B J Frey (2007). «Clustering by Passing Messages Between Data Points». Em: *Science* 315.5814, pp. 972–976. issn: 1095-9203. doi: 10.1126/science.1136800. arXiv: 1401.2548.

Ferreira, J. (2013). «Multimethodology in Metaheuristics». Em: *Journal of the Operational Research Society* 64.September, pp. 873–883. issn: 0160-5682. doi: 10.1057/jors.2012.88. url: http://dx.doi.org/10.1057/jors.2012.88.

Feurer, Matthias et al. (2015). «Efficient and Robust Automated Machine Learning». Em: *Advances in Neural Information Processing Systems 28*, pp. 2944–2952. issn: 10495258. url: http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf.

Fujiwara, Yasuhiro, Go Irie e Tomoe Kitahara (2011). «Fast Algorithm for Affinity Propagation». Em: *Ijcai*, pp. 2238–2243. doi: 10.5591/978-1-57735-516-8/IJCAI11-373.

Garcia, M.a. Porta et al. (2009). «Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation». Em: *Applied Soft Computing* 9.3, pp. 1102–1110. issn: 15684946. doi: 10.1016/j.asoc.2009.02.014.

Gere, W. S. (1966). «Heuristics in Job Shop Scheduling». Em: *Management Science* 13.3, pp. 167–190. issn: 0025-1909. doi: 10.1287/mnsc.13.3.167.

Glover, Fred (1986). «Future paths for integer programming and links to artificial intelligence». Em: *Computers and Operations Research* 13.5, pp. 533–549. issn: 03050548. doi: 10.1016/0305-0548(86)90048-1.

Heinonen, J. e F. Pettersson (2007). «Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem». Em: *Applied Mathematics and Computation* 187.2, pp. 989–998. issn: 00963003. doi: 10.1016/j.amc.2006.09.023.

Hutter, Frank et al. (2006). «Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms». Em: *Principles and Practice of Constraint Programming* 4204, pp. 213–228. issn: 03029743. doi: 10.1007/11889205_17. url: http://dx.doi.org/10.1007/11889205%7B%5C_%7D17.

Kaku, Michio (2011). «A Física Do Futuro». Em:

Karimov, Jeyhun e Murat Ozbayoglu (2015). «Clustering Quality Improvement of k-means Using a Hybrid Evolutionary Model». Em: *Procedia Computer Science* 61, pp. 38–45. issn: 18770509. doi: 10.1016/j.procs.2015.09.143. url: http://dx.doi.org/10.1016/j.procs.2015.09.143.

Kennedy, J e R Eberhart (1995). «Particle swarm optimization». Em: *Neural Networks, 1995. Proceedings., IEEE International Conference on* 4, 1942–1948 vol.4. issn: 19353812. doi: 10.1109/ICNN.1995.488968.

Kolliopoulos, Stavros G. e George Steiner (2006). «Approximation algorithms for minimizing the total weighted tardiness on a single machine». Em: *Theoretical Computer Science* 355.3, pp. 261–273. issn: 03043975. doi: 10.1016/j.tcs.2005.11.039.

Lessmann, Stefan, Marco Caserta e Idel Montalvo Arango (2011). «Tuning metaheuristics: A data mining based approach for particle swarm optimization». Em: *Expert Systems with Applications* 38.10, pp. 12826–12838. issn: 09574174. doi: 10.1016/j.eswa.2011.04.075. url: http://dx.doi.org/10.1016/j.eswa.2011.04.075.

Liao, Ching Jong e Hsiao Chien Juan (2007). «An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups». Em: *Computers and Operations Research* 34.7, pp. 1899–1909. issn: 03050548. doi: 10.1016/j.cor.2005.07.020.

OR-Library (1990). *OR-Library*. url: http://people.brunel.ac.uk/%7B~%7Dmastjjb/jeb/info.html.

Lin, Win-Chin et al. (2016). «Particle swarm optimization and opposite-based particle swarm optimization for two-agent multi-facility customer order scheduling with ready times». Em: *Applied Soft Computing* 52, pp. 877–884. issn: 15684946. doi: 10.1016/j.asoc.2016.09.038. url: http://dx.doi.org/10.1016/j.asoc.2016.09.038.

Macqueen, J (1967). «Some methods for classification and analysis of multivariate observations». Em: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1.233, pp. 281–297. issn: 00970433. doi: citeulike-article-id:6083430.

Madureira, Ana (1995). *Aplicações de Meta-Heurísticas em Problemas de Sequenciamento*. Rel. téc. Faculdade de Engenharia Da Universidade Do Porto.

– (2003). «Aplicação de Meta-Heurísticas ao Problema de Escalonamento em ambiente Dinâmico de Produção Discreta». Tese de doutoramento. Universidade do Minho.

Madureira, Ana, Diamantino Falcão e Ivo Pereira (2012). «Ant Colony System based approach to single machine scheduling problems: Weighted tardiness scheduling problem». Em: *2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 86–91. doi: 10.1109/NaBIC.2012.6402244.

MAH, Akhand, Shaik Imran Hossain e Shahina Akter (2016). «A Comparative Study of Prominent Particle Swarm Optimization Based Methods to Solve Traveling Salesman Problem». Em: *International Journal of Swarm Intelligence and Evolutionary Computation* 05.03. issn: 20904908. doi: 10.4172/2090-4908.1000139. url: https://www.omicsonline.com/open-access/a-comparative-study-of-prominent-particle-swarm-optimization-basedmethods-to-solve-traveling-salesman-problem-2090-4908-1000139.php?aid=80966.

Marinakis, Yannis, Georgia-Roumbini Iordanidou e Magdalene Marinaki (2013). «Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands». Em: *Applied Soft Computing* 13.4, pp. 1693–1704. issn: 15684946. doi: 10.1016/j.asoc.2013.01.007. url: http://linkinghub.elsevier.com/retrieve/pii/S1568494613000240.

Maron, Oden e Andrew W. Moore (1997). «The Racing Algorithm: Model Selection for Lazy Learners». Em: *Artificial Intelligence Review* 11.1-5, pp. 193–225. issn: 0269-2821. doi: 10.1023/A:1006556606079. url: http://citeseer.ist.psu.edu/maron97racing.html%7B%5C%%7D5Cnhttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.954.

Miranda, Vladimiro e Rui Alves (2014). «PAR/PST location and sizing in power grids with wind power uncertainty». Em: *2014 International Conference on Probabilistic Methods Applied to Power Systems, PMAPS 2014 - Conference Proceedings*. doi: 10.1109/PMAPS.2014.6960679.

Pang, Shanchen et al. (2013). «Particle Swarm Optimization Algorithm for Multisalesman Problem with Time and Capacity Constraints». Em: *Applied Mathematics & Information Sciences* 7.6, pp. 2439–2444. issn: 1935-0090. doi: 10.12785/amis/070637. url: http://www.naturalspublishing.com/Article.asp?ArtcID=4118.

Pedregosa, Fabian et al. (2012). «Scikit-learn: Machine Learning in Python». Em: 12, pp. 2825–2830. issn: 15324435. doi: 10.1007/s13398-014-0173-7.2. arXiv: 1201.0490. url: http://arxiv.org/abs/1201.0490.

Pereira, Ivo (2014). «Sistema Inteligente para Escalonamento Assistido por Aprendizagem». PhD Thesis. Universidade de Trás-os-Montes e Alto Douro.

Peterson, L. E. (2009). «K-nearest neighbor». Em: *Scholarpedia* 4. url: http://scholarpedia.org/article/K-nearest%7B%5C_%7Dneighbor.

PHYS.ORG (2016a). *Google computer wins final game against S. Korean Go master*. url: https://phys.org/news/2016-03-google-game-korean-master.html.

– (2016b). «Human Go champ scores surprise victory over supercomputer». Em: url: https://phys.org/news/2016-03-human-champ-scores-victory-supercomputer.html%7B%5C#%7DnRlv.

Pinedo, Michael L (2016). *Scheduling: Theory, Algorithms, and Systems*. Fifth. isbn: 978-3-319-26578-0. doi: 10.1007/978-3-319-26580-3.

Raschka, Sebastian (2015). *Python Machine Learning*. Packt Publishing Ltd Birmingham B3 2PB, UK. url: Packt%20Publishing%20Birmingham%20-%20Mumbai.%20Open%20Source%20Community.

Rekaya, R. et al. (2013). «Ant Colony Algorithm with Applications in the Field of Genomics». Em: *Intech*. issn: 9789533070865. doi: 10.5772/52807.

Rossi, Andrea e Gino Dini (2007). «Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method». Em: *Robotics and Computer-Integrated Manufacturing* 23.5, pp. 503–516. issn: 07365845. doi: 10.1016/j.rcim.2006.06.004.

Sammut, Claude e Geoffrey Webb (2010). *Encyclopedia of Machine Learning*. Ed. por Springer. isbn: 978-0-387-30768-8. doi: 10.1007/978-0-387-30164-8. arXiv: arXiv:1011.1669v3. url: http://link.springer.com/10.1007/978-0-387-30164-8.

Samuel, A.L. (1967). «Some studies in machine learning using the game of checkers. II - Recent Progress». Em: *Ibm Journal* 3.3, p. 210. issn: 00664138. doi: 10.1016/0066-4138(69)90004-4. url: http://pages.cs.wisc.edu/%7B~%7Ddyer/cs540/handouts/samuel-checkers.pdf.

Samuel, Artur L (1959). «Some studies in machine learning using the game of checkers». Em: *IBM Journal of research and development* 3.3, pp. 210–229. issn: 0018-8646. doi: 10.1147/rd.33.0210.

Shi, X. H. et al. (2007). «Particle swarm optimization-based algorithms for TSP and generalized TSP». Em: *Information Processing Letters* 103.5, pp. 169–176. issn: 00200190. doi: 10.1016/j.ipl.2007.03.010.

Silver, David, Julian Schrittweiser e Karen Simonyan (2017). «Mastering the game of Go without human knowledge». Em: *Nature* 550, pp. 354–359. url: https://www.nature.com/nature/journal/v550/n7676/full/nature24270.html.

Simon, H. a. e A. Newell (1958). «Heuristic Problem Solving: The Next Advance in Operations Research». Em: *Operations Research* 6.1, pp. 1–10. issn: 0030-364X. doi: 10.1287/opre.6.1.1.

Singh, Kehar, Dimple Malik e Naveen Sharma (2011). «Evolving limitations in K-means algorithm in data mining and their removal». Em: *IJCEM International Journal of Computational Engineering & Management ISSN* 12.April, pp. 2230–7893. url: www.IJCEM.org%7B%5C%%7D5Cnwww.ijcem.org.

Sklearn (s.d.). «scikit learn». Em: url: http://scikit-learn.org/stable/.

Stephenson, Mark et al. (2003). «Meta Optimization: Improving Compiler Heuristics with Machine Learning». Em: *ACM Conference on Programming Language Design and Implementation*, pp. 77–90. issn: 03621340. doi: 10.1145/781131.781141. url: http://portal.acm.org/citation.cfm?doid=781131.781141.

Talbi, El Ghazali (2009). *Metaheuristics: From Design to Implementation*. isbn: 9780470278581. doi: 10.1002/9780470496916. arXiv: 1011.1669.

Tasgetiren, M.F., M. Sevkli e G. Gencyilmaz (2004). «Particle swarm optimization algorithm for single machine total weighted tardiness problem». Em: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, pp. 1412–1419. doi: 10.1109/CEC.2004.1331062. url: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1331062.

Turing, Alan (1950). «Turing. Computing machinery and intelligence». Em: *Mind* 59.236, pp. 433–460. issn: 0026-4423. doi: http://dx.doi.org/10.1007/978-1-4020-6710-5_3. arXiv: 2251299 [10.2307]. url: papers2://publication/uuid/E74CAAC6-F3DD-47E7-AEA6-5FB511730877.

Wolpert, D e W Macready (1997). «No free lunch theorems for optimization». Em: *IEEE Transactions on Evolutionary Computation* 1.1, pp. 67–82.

Xing, Li-Ning et al. (2010). «A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems». Em: *Applied Soft Computing* 10.3, pp. 888–896. issn: 15684946. doi: 10.1016/j.asoc.2009.10.006. url: http://www.sciencedirect.com/science/article/pii/S156849460900194X.

Yagmahan, Betul e Mehmet Mutlu Yenisey (2008). «Ant colony optimization for multi-objective flow shop scheduling problem». Em: *Computers and Industrial Engineering* 54.3, pp. 411–420. issn: 03608352. doi: 10.1016/j.cie.2007.08.003.

Zanakis, Stelios H e James R Evans (1981). «HEURISTIC "OPTIMIZATION ": WHY , WHEN , AND HOW TO USE IT». Em: *INTERFACES* 11.5, pp. 84–92.

Apêndice A

Tabelas de resultados da heurística

Resultados da heurística em que "v ref"representa o valor de referência e "v melhor"representa o valor do melhor resultado encontrado.

A.1 Instâncias de 50 tarefas

Instância	%desvio	v ref	v melhor
0_50	0	2134	2134
101_50	0	0	0
103_50	0	0	0
104_50	0	0	0
108_50	3.67	6185	6412
118_50	0.093	106043	106142
12_50	0	45383	45383
120_50	0.083	78315	78380
122_50	0.21	101157	101369
124_50	0.055	110392	110453
17_50	0.034	104795	104831
2_50	0	2583	2583
22_50	0.009	224025	224045
23_50	0.028	116015	116047
24_50	0.008	240179	240199
25_50	0	2	2
26_50	0	4	4
3_50	0	2691	2691
30_50	0	9934	9934
31_50	0.223	7178	7194
32_50	0	4674	4674
33_50	0	4017	4017
38_50	0.225	49352	49463

Instância	%desvio	v ref	v melhor
41_50	0.125	90163	90276
42_50	0.07	84126	84185
47_50	0.007	191099	191113
5_50	0	26276	26276
50_50	0	0	0
52_50	0	0	0
54_50	0	0	0
55_50	0	1258	1258
56_50	5.355	3679	3876
60_50	0.571	25212	25356
62_50	0.729	30729	30953
65_50	0.074	76878	76935
68_50	0.131	77930	78032
72_50	0.057	98494	98550
74_50	0.005	135677	135684
79_50	0	0	0
80_50	11.397	816	909
83_50	1.969	508	518
88_50	0.52	28846	28996
9_50	0	10655	10655
97_50	0.066	148906	149005
99_50	0.082	120108	120207

Figura A.1: Melhores resultados - 50 tarefas.

A.2 Instâncias de 40 tarefas

Instância	%desvio	v ref	v melhor
0_40	0	913	913
1_40	0	1225	1225
10_40	0	17465	17465
100_40	0	0	0
101_40	0	0	0
103_40	0	0	0
104_40	0	0	0
105_40	0	0	0
107_40	2.415	3354	3435
108_40	0	0	0
11_40	0	19312	19312
113_40	0.122	19648	19672
116_40	0	50364	50364
118_40	0	66707	66707
119_40	0.033	69019	69042
121_40	0	82456	82456
122_40	0	75118	75118
123_40	0	73041	73041
124_40	0	104531	104531
13_40	0	14377	14377
14_40	0	26914	26914
15_40	0	72317	72317
16_40	0	78623	78623
17_40	0	74310	74310
19_40	0.017	63229	63240
2_40	0	537	537
20_40	0	77774	77774
21_40	0	100484	100484
22_40	0	135618	135618
24_40	0	128747	128747
25_40	0	108	108
28_40	0	47	47
29_40	0	98	98
3_40	0	2094	2094
31_40	0	4098	4098
34_40	0	5290	5290
35_40	0	19732	19732
37_40	0	24499	24499
39_40	0	19611	19611
41_40	0	81462	81462

Instância	%desvio	v ref	v melhor
42_40	0	65134	65134
44_40	0	66579	66579
45_40	0	64451	64451
46_40	0	113999	113999
47_40	0	74323	74323
48_40	0	110295	110295
49_40	0	95616	95616
53_40	0	0	0
54_40	0	0	0
56_40	3.451	2260	2338
60_40	0.158	20281	20313
61_40	0.403	13403	13457
62_40	0	19771	19771
65_40	0	65386	65386
66_40	0.015	65756	65766
67_40	0	78451	78451
68_40	0	81627	81627
69_40	0.012	68242	68250
7_40	0	6865	6865
70_40	0	90486	90486
71_40	0	115249	115249
72_40	0	68529	68529
73_40	0	79006	79006
74_40	0	98110	98110
76_40	0	0	0
77_40	0	0	0
78_40	0	0	0
79_40	0	0	0
82_40	1.253	798	808
83_40	35.494	617	836
85_40	0.195	10262	10282
86_40	0.011	18646	18648
89_40	0.429	8159	8194
9_40	0	9737	9737
91_40	0	43004	43004
92_40	0	55730	55730
93_40	0.017	59494	59504
94_40	0.056	42688	42712
96_40	0	114686	114686
99_40	0	157296	157296

Figura A.2: Melhores resultados - 40 tarefas.